Oracle® TimesTen In-Memory Database Kubernetes Operator User's Guide



Release 22.1 F35390-11 June 2025

ORACLE

Oracle TimesTen In-Memory Database Kubernetes Operator User's Guide, Release 22.1

F35390-11

Copyright © 2020, 2025, Oracle and/or its affiliates.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software, software documentation, data (as defined in the Federal Acquisition Regulation), or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs) and Oracle computer documentation or other Oracle data delivered to or accessed by U.S. Government end users are "commercial computer software," "commercial computer software documentation," or "limited rights data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, reproduction, duplication, release, display, disclosure, modification, preparation of derivative works, and/or adaptation of i) Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs), ii) Oracle computer documentation and/or iii) other Oracle data, is subject to the rights and limitations specified in the license contained in the applicable contract. The terms governing the U.S. Government's use of Oracle cloud services are defined by the applicable contract for such services. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle®, Java, MySQL, and NetSuite are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Inside are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Epyc, and the AMD logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

Contents

What's New

New Features in Release 22.1.1.34.0	XV
New Features in Release 22.1.1.30.0	xvi
New Features in Release 22.1.1.27.0	xvi
New Features in Release 22.1.1.19.0	xviii
New features in Release 22.1.1.9.0	xviii
New features in Release 22.1.1.3.0	xix
New features in Release 22.1.1.1.0	xix

1 Overview of the Oracle TimesTen Kubernetes Operator

Overview of Containers and Kubernetes	1-1
Custom Resource Definition	1-2
Kubernetes Operator	1-2
About the TimesTen Kubernetes Operator	1-2
About TimesTenClassic and TimesTenScaleout Objects	1-3
About Provisioning Active Standby Pairs	1-3
About Deploying a Replicated TimesTenClassic Object	1-4
About Objects Created by the TimesTen Operator	1-5
StatefulSet	1-6
Service	1-6
Secret	1-6
Pods	1-6
Events	1-7
About the TimesTen Containers and the TimesTen Agent	1-7
Simple Deployment	1-7
About Deploying a TimesTen Scaleout Grid and Database	1-8
StatefulSets	1-8
Services	1-9
Secret	1-9
Persistent Volume Claims and Pods	1-9
Password-less ssh	1-9



2 Prepare to Use the TimesTen Kubernetes Operator

About TimesTen Container Images and Container Registry Options	2-1
Before You Begin	2-2
Choose a TimesTen Container Image	2-4
Option 1: Use a Container Image from Oracle Container Registry	2-4
Create Auth Token	2-4
Accept the Oracle License Agreement	2-4
Configure Your Development Host and Kubernetes Cluster	2-5
Option 2: Use a Container Image from Oracle Cloud Marketplace	2-6
Option 3: Build a Container Image and Push It to a Container Registry of Your Choice	2-6
Configure Your Development Host and Kubernetes Cluster	2-7
Download and Unpack a TimesTen Distribution	2-7
Build a TimesTen Container Image	2-9
Tag and Push the Container Image to Your Container Registry	2-9
Obtain TimesTen YAML Manifest Files and Helm Charts	2-10

3 Install TimesTen Custom Resource Definitions

About TimesTen CRDs	3-1
Install TimesTen CRDs	3-1

4 Learn About and Install the TimesTen Kubernetes Operator

About Kubernetes Operators	4-1
About the TimesTen Operator	4-1
About Deploying in a Multi-Architecture Kubernetes Cluster	4-2
About the Default Kubernetes Security Context for the TimesTen Operator	4-4
About Readiness and Liveness Probes	4-4
About Privileges	4-4
About Installing the TimesTen Operator	4-5
About Customizations for a TimesTen Operator Deployment	4-7
Install the TimesTen Operator	4-7
Install the TimesTen Operator at Namespace-Scope	4-8
Before You Begin	4-8
Install in Namespace One	4-10
Install in Namespace Two	4-14
Verify Installation	4-18
Install the TimesTen Operator at Cluster-Scope	4-19



5 Use Configuration Metadata

Overview of Configuration Metadata and Kubernetes Facilities	5-1
List of Configuration Metadata	5-1
About Configuration Metadata Details	5-3
adminUser	5-3
cachegroups.sql	5-3
cacheUser	5-4
csWallet	5-5
db.ini	5-5
epilog.sql	5-6
replicationWallet	5-6
schema.sql	5-6
sqlnet.ora	5-7
testUser	5-7
tnsnames.ora	5-7
*.connect	5-8
*.csconnect	5-8
Populate the /ttconfig Directory	5-9
Using ConfigMaps and Secrets	5-9
Example Using One ConfigMap	5-10
Example Using One ConfigMap and One Secret	5-12
Example Using One ConfigMap for a TimesTenScaleout Object	5-15
Using an init container	5-18
Additional Configuration Options	5-19
Persistent Storage	5-19
Additional Resource Specifications	5-21
About the Default Affinity and Anti-Affinity Settings for TimesTenClassic Objects	5-22
About the Default Kubernetes Security Context for TimesTen Pods	5-23
About Readiness Probes for TimesTen Containers	5-23
About Readiness Probes for Replicated Configurations	5-24
About the /tmp/active Readiness Probe	5-24
About the /tmp/readiness Readiness Probe	5-25
About Disabling Readiness Probes	5-26
About Readiness Probes for Non-Replicated Configurations	5-26

6 Specify CPU and Memory Requests and Limits

About Resource Requests and Limits	6-1
About TimesTen Containerized Deployments	6-2
About Specifying Requests and Limits for TimesTen Containers	6-4
Approach 1: Use Specific Datum for Requests and Limits	6-5
	• .

ORACLE

Approach 2: Use Templates for Requests and Limits	6-8
About Specifying Requests and Limits to Kubernetes	6-9
About Verifying databaseMemorySize	6-10
About Runtime Memory Monitoring	6-10

7 Create TimesTen Classic Databases

About Defining TimesTenClassic Objects	7-1
About the Deployment Process	7-3
About the Examples	7-4
Create Replicated TimesTen Classic Databases	7-5
Create Non-Replicated TimesTen Classic Databases	7-9
Modify the Number of Replicas in Non-Replicated Environments	7-15

8 Deploy TimesTen Scaleout Databases

About Deploying a Grid	8-1
About Planning a Grid	8-2
About Configuring a Grid	8-2
About Provisioning a Grid	8-3
About ssh	8-4
About Creating a Grid	8-4
Deploy a Grid	8-5
Create Configuration Metadata and a Kubernetes ConfigMap for a Grid	8-5
Define and Deploy a TimesTenScaleout Object	8-8
Monitor the High Level State of a TimesTenScaleout Object	8-10
Verify Underlying Objects	8-11
Connect to the Database	8-14
Manage a Grid and Its Database	8-15

9 Use Helm in Your TimesTen Kubernetes Operator Environment

Overview of Helm and TimesTen Helm Charts	9-1
About Helm	9-1
About a Helm Chart	9-2
About Helm Charts for TimesTen	9-2
About Installing and Testing a Release	9-3
About Versions in a Chart.yaml File	9-3
About the Helm Substitution Engine and Language	9-4
Install the TimesTen CRDs and the TimesTen Operator	9-4
Install the TimesTen CRDs	9-4
Install the TimesTen Operator	9-5



Test the TimesTen Operator	9-8
Create TimesTen Databases and Test TimesTen	9-9
About Creating TimesTen Databases	9-10
About Using the ttclassic Helm Chart	9-11
Create Replicated TimesTen Databases	9-13
Test TimesTen for a Replicated Configuration	9-17
Create Non-Replicated TimesTen Databases	9-18
Test TimesTen for a Non-Replicated Configuration	9-22
Upgrade	9-23
About Upgrading	9-23
Upgrade the TimesTen CRDs	9-24
Upgrade the TimesTen Operator	9-25
Upgrade Replicated TimesTen Databases	9-29
Upgrade Non-Replicated TimesTen Databases	9-36
Roll Back an Upgrade	9-42
Roll Back a Replicated TimesTen Upgrade	9-43
Roll back a Non-Replicated TimesTen Upgrade	9-48
Roll Back a TimesTen Operator Upgrade	9-54
Clean Up	9-57
About Uninstalling a Release	9-57
Delete TimesTen Databases	9-57
Delete the TimesTen Operator	9-60
Delete the TimesTen CRDs	9-61

10 Use TimesTen Databases

About Using Direct Mode Applications	10-1
About Using Client/Server Drivers	10-3

11 Manage and Monitor TimesTen Classic Databases

About the High Level State of TimesTenClassic Objects	11-1
ActiveDown	11-2
ActiveTakeover	11-2
AllReplicasReady	11-2
BothDown	11-2
ConfiguringActive	11-2
Failed	11-3
Initializing	11-3
ManualInterventionRequired	11-3
NoReplicasReady	11-3
Normal	11-3



	_ ·	
	Reexamine	11-3
	SomeReplicasReady	11-4
	StandbyCatchup	11-4
	StandbyDown	11-4
	StandbyStarting	11-4
	WaitingForActive	11-4
A	bout the High Level State of TimesTen Pods	11-4
	CatchingUp	11-5
	Down	11-5
	Healthy	11-5
	HealthyActive	11-5
	HealthyStandby	11-5
	Initializing	11-6
	ManualInterventionRequired	11-6
	Normal	11-6
	OtherDown	11-6
	Reexamine	11-7
	Terminal	11-7
	Unknown	11-7
	UpgradeFailed	11-7
A	bout the BothDown State	11-7
А	bout the ManualInterventionRequired State for Replicated Objects	11-9
А	bout Bringing Up One Database	11-10
	Verify Conditions Are Met for the Database	11-11
	Set the reexamine Value	11-12
А	bout Suspending Management of a TimesTenClassic Object	11-17
	Suspend Management	11-18
	Resume Management	11-21
А	bout Manual Operations	11-22
	Manually Invoke TimesTen Utilities	11-23
	Revert to Manual Control	11-23
	Delete TimesTen Databases	11-25
	Locate the TimesTen Operator	11-26
	Modify TimesTen Connection Attributes	11-26
	Manually Edit the db.ini File	11-27
	Modify First Connection Attributes	11-29
	Modify General Connection Attributes	11-31

12 Manage TimesTen Scaleout

About Managing TimesTen Scaleout	12-1
About Single Data Instance Failure	12-2

About Management Instance Failure	12-2
About the waiting for seed State	12-2
About Failure of All Data Instances	12-3
About High Level States	12-3
DatabaseDown	12-3
DatabaseImpeded	12-3
DatabasePartial	12-4
DatabaseRestarting	12-4
DatabaseRestartRequired	12-4
Failed	12-4
Initializing	12-4
ManualInterventionRequired	12-4
Normal	12-4
Reexamine	12-4
Unmanaged	12-5
About Management States	12-5
ActiveAgentUp	12-5
ActiveDaemonUp	12-5
ActiveDown	12-5
Error	12-6
Normal	12-6
Unknown	12-6
About Database and Element States	12-6
About the ManualInterventionRequired State	12-7
About Suspending Management	12-7
Simulate Single Data Instance Failure	12-8
Simulate Management Instance Failure	12-9
Simulate Replica Set Failure with Restart	12-10
Simulate Replica Set Failure with Manual	12-12
Suspend Management	12-14
Set reexamine Datum	12-16

13 Optimize Client/Server Performance

About Client/Server Challenges	13-1
About NodePort Services	13-1
About the Connection Manager	13-2
How to Use the Connection Manager	13-3
About Accessing the Endpoint from Inside the Cluster	13-4
About Accessing the Endpoint from Outside the Cluster	13-4
About Accessing TimesTen	13-5
About Handling Failures	13-5

ORACLE

About the NodePort Service for the Connection Manager	13-5
About the TimesTen Operator Configuration	13-5

14 Expose TimesTen Metrics with the TimesTen Kubernetes Operator

Overview of TimesTen Metrics	14-1
Overview of the TimesTen Kubernetes Operator and the TimesTen Exporter	14-2
About the Prometheus Operator	14-2
About Exposing TimesTen Metrics	14-3
About Using http or https for TimesTen Metrics	14-5
About Transport Layer Security (mutual TLS) Certificates for TimesTen Metrics	14-6
About Creating PodMonitor Objects	14-7
About the TimesTen Metrics Service	14-8
About Choosing to Expose TimesTen Metrics	14-8
Expose TimesTen Metrics Automatically	14-9

15 Expose Metrics from the TimesTen Kubernetes Operator

About Exposing Metrics from the TimesTen Kubernetes Operator	15-1
About Using http or https	15-3
About Transport Layer Security (mutual TLS) Certificates	15-3
About Creating ServiceMonitor Objects	15-4
About the TimesTen Kubernetes Operator's Metrics Service	15-4
About TimesTen Operator Metrics	15-4
Demonstrate How to Expose TimesTen Kubernetes Operator Metrics	15-5

16 Work with TimesTen Cache

About Using TimesTen Cache	16-1
----------------------------	------

17 Use Encryption for Data Transmission

Create TLS Certificates for Replication and Client/Server	17-1
Configure TLS for Replication	17-3
Create Metadata Files and Kubernetes Facilities	17-3
Create a Kubernetes Secret	17-4
Create a ConfigMap	17-5
Create a TimesTenClassic Object	17-6
Monitor Deployment of a TimesTenClassic Object	17-7
Verify TLS Is Being Used for Replication	17-9
Automatically Configure Client/Server TLS	17-10
About Configuring a TimesTenClassic Object for Automatic Client/Server TLS Encryption	17-11



About the Automation Process	17-12
How-to Example	17-14
Before You Begin	17-14
Configure and Deploy the TimesTenClassic Object	17-16
Configure TLS for Client/Server	17-21
Configuration Requirements for the Server	17-21
Overview of Metadata Files and Kubernetes Facilities	17-21
Create a Kubernetes Secret for the csWallet Metadata File	17-22
Create a ConfigMap for the Server-Side Attributes	17-23
Create a TimesTenClassic Object	17-25
Monitor Deployment of the TimesTenClassic Object	17-26
Configuration Requirements for the Client	17-27
Copy a Client Wallet	17-27
Configure Client-Side Attributes	17-28

18 Handle Failover and Recovery in TimesTen Classic

About Node Failure in Kubernetes	18-1
How Kubernetes Reports Node Status	18-1
How the TimesTen Kubernetes Operator Handles Node Failure	18-2
About Specifying the .spec.ttspec.deleteDbOnNotReadyNode Datum	18-3
About Kubernetes Events and TimesTen Operator Metrics	18-3
About Handling Failover and Recovery	
Illustrate the Failover and Recovery Process	18-4

19 Perform Upgrades

About Obtaining Container Images for the Upgrade	19-1
About Upgrading from Previous Releases	19-2
Upgrade the TimesTen CRDs	19-2
About Upgrading the TimesTen Operator	19-3
Upgrade the TimesTen Operator	19-4
Upgrade the TimesTen Operator at Namespace-Scope	19-4
Before You Begin	19-4
Upgrade in Namespace One	19-5
Upgrade in Namespace Two	19-10
Upgrade the TimesTen Operator at Cluster-Scope	19-14
About Upgrading TimesTen Classic Databases	19-17
About the Upgrade Process for Replicated Configurations	19-17
About the Upgrade Process for Non-Replicated Configurations	19-18
Perform an Automated Upgrade of a Replicated TimesTenClassic Object	19-19
Perform a Manual Upgrade of a Replicated TimesTenClassic Object	19-24

ORACLE

Modify a Replicated TimesTenClassic Object	19-24
Upgrade the Standby Database	19-25
Fail Over	19-27
Perform an Automated Upgrade of a Non-Replicated TimesTenClassic Object	
Perform a Manual Upgrade of a Non-Replicated TimesTenClassic Object	
About Upgrading Direct Mode Applications	
About Failures During an Upgrade	19-42

20 TimesTen Kubernetes Operator Object Types

Overview of the TimesTen Kubernetes Operator Object Types	20-1
About the TimesTenClassic Object Type	20-1
TimesTenClassic	20-2
TimesTenClassicSpec	20-2
TimesTenClassicSpecSpec	20-3
TimesTenClassicSpecClientTLS	20-23
TimesTenClassicSpecSpecPrometheus	20-27
TimesTenClassicStatus	20-29
About the TimesTenScaleout Object Type	20-30
TimesTen Scaleout	20-30
TimesTenScaleoutSpec	20-31
TimesTenScaleoutSpecSpec	20-32
TimesTenScaleoutSpecSpecPrometheus	20-42
TimesTenScaleoutStatus	20-44

21 Helm Charts for the TimesTen Kubernetes Operator

The ttoperator Chart	21-1
The ttclusteroperator Chart	21-11
The ttclassic Chart	21-21

22 TimesTen Kubernetes Operator Metrics

23 TimesTen Kubernetes Operator Environment Variables

24 Dockerfile ARGs

A Active Standby Pair Example

Before You Begin	A-1
Create a ConfigMap Object	A-1
Create a TimesTenClassic Object	A-3
Monitor Deployment	A-4
Verify Existence of Underlying Objects	A-9
Verify Connection to the Active TimesTen Database	A-9
Recover from Failure	A-10
Clean Up	A-11

B TimesTen Cache in TimesTen Classic Example

Set Up the Oracle Database to Cache Data	B-1
Create the Oracle Database Users	B-1
Grant Privileges to the Cache Administration User	B-2
Create the Oracle Database Tables to Be Cached	B-3
Create Metadata Files and a Kubernetes Facility	B-5
Create a TimesTenClassic Object	B-10
Monitor Deployment of a TimesTenClassic Object	B-11
Verify TimesTen Cache Configuration	B-13
Perform Operations On Cache Group Tables	
Perform Operations on the oratt.readtab Table	B-15
Perform Operations on the oratt.writetab Table	B-16
Clean Up Cache Metadata on the Oracle Database	B-17

C Create Your Own Oracle Wallet, Certificates, and Secrets for Exposing TimesTen Metrics

About Creating Your Own Oracle Wallet, Certificates, and Kubernetes Secrets	C-1
Create Your Own Oracle Wallet, Certificates, and Kubernetes Secrets	C-2
Before You Begin	C-3
Create Certificates	C-4
Create a Kubernetes Secret Containing an Oracle Wallet	C-7
Define and Deploy a TimesTenClassic Object	C-7



About This Content

This document covers TimesTen support for the TimesTen Kubernetes Operator.

Audience

This document is intended for anyone who wants to use the TimesTen Kubernetes Operator in a Kubernetes environment.

Kubernetes is a portable, extensible, open-source platform for managing containerized workloads and services, that facilitates both declarative configuration and automation. The Oracle TimesTen Kubernetes Operator (TimesTen Operator) is a Kubernetes Operator that provides the ability to do the following:

- Create and deploy highly available replicated pairs of TimesTen Classic databases to a Kubernetes cluster with minimal effort. It also provides the ability to automate failure detection and recovery.
- Create and deploy TimesTen Scaleout grids and their associated databases to a Kubernetes cluster with minimal effort.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Conventions

The following text conventions are used in this document.

Convention	Meaning
boldface Boldface type indicates graphical user interface elements associated action, or terms defined in text or the glossary.	
<i>italic</i> Italic type indicates book titles, emphasis, or placeholder variables you supply particular values.	
monospace Monospace type indicates commands within a paragraph, URLs, examples, text that appears on the screen, or text that you enter.	

What's New

This section summarizes the new features of Oracle TimesTen In-Memory Database Release 22.1 that are documented in this guide. It provides links to more information.

New Features in Release 22.1.1.34.0

- In previous releases, you could install the TimesTen Operator in one or more namespaces in your Kubernetes cluster at namespace-scope. In this release, you also have the option to install the TimesTen Operator in the timesten-operator namespace at cluster-scope. Specifically, you now have two options for installing the TimesTen Operator. See About the TimesTen Operator.
- To optimize client/server performance, the TimesTen Operator creates NodePort Services and provides a new feature, called the Connection Manager. The Connection Manager functions as an https server. It adds several new endpoints to the /metrics server. These new endpoints allow client/server applications to retrieve a connection string that can then be used to access TimesTen databases optimally. See Optimize Client/Server Performance. The TimesTen Operator provides the following syntax for the Connection Manager:
 - If you are using YAML manifest files to install the TimesTen Operator, see the TT_CONNECTION_MANAGER, TT_CONNECTION_MANAGER_NODEPORT, and TT_OPERATOR_SAN environment variables in TimesTen Kubernetes Operator Environment Variables.
 - If you are using either the ttoperator or ttclusteroperator Helm chart to install the TimesTen Operator, see connectionManager, operatorNodePort, and operatorSAN in The ttoperator Chart or The ttclusteroperator Chart.
- The TimesTen Operator can automatically create self-signed certificates and configure TimesTen to use those certificates for client/server TLS encryption. See Automatically Configure Client/Server TLS. The TimesTen Operator provides the following syntax for this feature:
 - If you are using YAML manifest files, see clientILS in TimesTenClassicSpecSpec.
 - If you are using Helm charts, see clientTLS in The ttclassic Chart.
- You can set the number of TimesTenClassic objects that the TimesTen Operator processes concurrently:
 - If using YAML manifest files to install the TimesTen Operator, see the TT_MAX_RECONCILES environment variable in TimesTen Kubernetes Operator Environment Variables.
 - If you are using either the ttoperator or ttclusteroperator Helm chart to install the TimesTen Operator, see maxReconciles in The ttoperator Chart or The ttclusteroperator Chart.
- The TimesTen Operator adds a default security context, including the Kubernetes default seccomp profile to the Pods it creates. In addition, the TimesTen Operator runs with a default security context and includes the Kubernetes default seccomp profile. See About



the Default Kubernetes Security Context for TimesTen Pods and About the Default Kubernetes Security Context for the TimesTen Operator.

- By default, the TimesTen Operator adds affinity and anti-affinity settings to Pods that are created when deploying a TimesTenClassic object. See About the Default Affinity and Anti-Affinity Settings for TimesTenClassic Objects.
- The TimesTen Operator stores its security certificates in Java Keystores. In order to create and use Java Keystores, they must be secured with a known password:
 - If you are using YAML manifest files, see storePassword in TimesTenClassicSpecSpec.
 - If you are using Helm charts, see storePassword in The ttclassic Chart.
- There is support for the v4 schema version of the TimesTenClassic Custom Resource Definition (CRD). The v2 schema version of the TimesTenClassic CRD is deprecated. The v2 schema version is fully supported in this release, but will be removed in a future release. See About TimesTen CRDs.

New Features in Release 22.1.1.30.0

- For non-replicated configurations, you can increase or decrease the number of databases deployed in your Kubernetes namespace. To make this change, modify the number of replicas associated with one or more of your TimesTenClassic objects. See Modify the Number of Replicas in Non-Replicated Environments.
- In both replicated and non-replicated configurations, the TimesTen Operator can detect and take action in situations where a Kubernetes Node is not ready (or unknown) for a specific period of time. To direct the TimesTen Operator to detect and take action, specify the .spec.ttspec.deleteDbOnNotReadyNode datum in a TimesTenClassic object definition. See How the TimesTen Kubernetes Operator Handles Node Failure and About Specifying the .spec.ttspec.deleteDbOnNotReadyNode Datum.

In order to use this feature, the TimesTen Operator requires additional privileges and permissions. See Install the TimesTen Operator at Namespace-Scope.

- The TimesTen Operator generates the following new metrics:
 - timesten_pods_deleted_unschedulable
 - timesten_pods_unschedulable
 - timesten_pvcs_deleted_unschedulable

For more information, see About Kubernetes Events and TimesTen Operator Metrics.

• The v1 schema version of the TimesTenClassic Custom Resource Definition (CRD) is deprecated. The v1 schema version is fully supported in this release, but will be removed in a future release. For more information, see About TimesTen CRDs.

New Features in Release 22.1.1.27.0

- The TimesTen Kubernetes Operator (TimesTen Operator) provides full support for nonreplicated TimesTenClassic objects. Databases in this configuration operate independently and have no relationship to each other:
 - The TimesTen Operator creates, manages, and monitors TimesTenClassic objects and associated databases in this configuration. See Create TimesTen Classic Databases.



- There is full support for upgrading (and downgrading) non-replicated TimesTenClassic objects to a new patch or patchset. See About Upgrading TimesTen Classic Databases.
- There are TimesTen Operator metrics for monitoring the overall status of nonreplicated TimesTenClassic objects. See TimesTen Kubernetes Operator Metrics.
- The TimesTen Operator provides and maintains high level states for TimesTenClassic objects and TimesTen Pods in non-replicated configurations. These states describe the health of TimesTenClassic objects and the health of TimesTen databases in Pods. See About the High Level State of TimesTenClassic Objects and About the High Level State of TimesTenPods.
- There are metrics for monitoring the memory used by the TimesTen Operator and the TimesTen Agent. See TimesTen Kubernetes Operator Metrics.
- The TimesTen Operator uses Kubernetes Custom Resource Definition (CRD) versioning to provide two different schema versions for the TimesTenClassic CRD. The TimesTen Operator creates, monitors, and manages TimesTenClassic objects in both schema versions. See About TimesTen CRDs.
- The TimesTen Operator provides and enables by default readiness probes for TimesTen containers in replicated and non-replicated configurations. See About Readiness Probes for TimesTen Containers.
- The TimesTen Operator runs in single and multi-architecture Kubernetes clusters. Pods that are created by the TimesTen Operator run on nodes of the same node type as the TimesTen Operator. For example, if the TimesTen Operator runs on arm64 nodes, Pods created by this TimesTen Operator also run on arm64 nodes. If you are using a multi-architecture Kubernetes cluster, you have to make customizations to the provided operator.yaml YAML manifest file and the ttoperator Helm charts. See About Deploying in a Multi-Architecture Kubernetes Cluster.
- TimesTen provides an additional option for obtaining container images that you can use in your TimesTen Kubernetes Operator environment. You can use the Oracle TimesTen In-Memory Database for Kubernetes - BYOL on Oracle Cloud Marketplace container listing to export a TimesTen container image into a repository in Oracle Cloud Infrastructure Registry (Container Registry). See Prepare to Use the TimesTen Kubernetes Operator.
- To support the new features, the .spec.ttspec section of the TimesTenClassic object type includes the following datum:
 - createASReadinessProbe
 - readOnlyRootFilesystem
 - replicas
 - replicationTopology
 - rollingUpdatePartition
 - terminationGracePeriod

See TimesTenClassicSpecSpec.

• Helm and the associated ttcrd, ttoperator, and ttclassic Helm charts are extended to support the new features. See Use Helm in Your TimesTen Kubernetes Operator Environment and Helm Charts for the TimesTen Kubernetes Operator.



New Features in Release 22.1.1.19.0

- The TimesTen Kubernetes Operator (TimesTen Operator) provides TimesTen-specific Helm charts. These charts let you use Helm to install the TimesTen CRDs, TimesTen Operator, and TimesTenClassic objects in your Kubernetes cluster. See Use Helm in Your TimesTen Kubernetes Operator Environment.
- By default, the TimesTen Operator automatically exports and exposes TimesTen metrics to Prometheus. See Expose TimesTen Metrics with the TimesTen Kubernetes Operator.

You can control how TimesTen metrics are exported and exposed by using specific datum in your TimesTenClassic and TimesTenScaleout object YAML manifest files. See About Exposing TimesTen Metrics and TimesTenClassicSpecSpecPrometheus and TimesTenScaleoutSpecSpecPrometheus.

 The TimesTen Operator automatically exposes metrics about its own functionality as well as the status of TimesTenClassic and TimesTenScaleout objects that it manages to Prometheus. See Expose Metrics from the TimesTen Kubernetes Operator.

You can control how these metrics are exposed by unsetting or setting TimesTen Operator environment variables located in the <code>operator.yaml YAML</code> manifest file of your TimesTen Operator deployment. See About Exposing Metrics from the TimesTen Kubernetes Operator and TimesTen Kubernetes Operator Environment Variables.

- The TimesTen Operator provides readiness and liveness probes so that Kubernetes can determine the health of the TimesTen Operator. See About Readiness and Liveness Probes.
- The TimesTen Operator creates and manages readiness probes for TimesTen containers. See About Readiness Probes for TimesTen Containers.

New features in Release 22.1.1.9.0

- The TimesTen Operator monitors and manages TimesTen Scaleout objects that are deployed in your Kubernetes cluster. It also detects, repairs, and recovers from failures in your grid and associated database. See Manage TimesTen Scaleout.
- The Operator supports TimesTen Cache in TimesTen Scaleout. See Work with TimesTen Cache.
- It is essential to specify memory requests and limits to Kubernetes. TimesTen recommends that CPU requests and limits be specified as well. See Specify CPU and Memory Requests and Limits. To support this functionality, there are new datum added to the .spec.ttspec fields of the TimesTenClassic and TimesTenScaleout object custom resource definitions. See TimesTenClassicSpecSpec and TimesTenScaleoutSpecSpec.
- TimesTen container images use the Oracle Linux base image. Oracle Java is installed into the TimesTen images using JDK script friendly URLs and Dockerfile techniques. For details about setting up your environment, see Prepare to Use the TimesTen Kubernetes Operator. For specific information about TimesTen container images, see About TimesTen Container Images and Container Registry Options. For information about Dockerfile ARGS, see Dockerfile ARGS.

New features in Release 22.1.1.3.0

 The TimesTen Operator can deploy TimesTen Scaleout grids and their associated TimesTen databases in your Kubernetes cluster. See Deploy TimesTen Scaleout Databases.

The TimesTen Operator supports the TimesTenScaleout object type. This object type provides the syntax you need to deploy a TimesTen Scaleout grid and database. See About the TimesTenScaleout Object Type.

- In previous releases, the TimesTen Operator required the creation of two container images, one for the Operator and one for TimesTen. In this release, one container image is used for both the Operator and TimesTen. You can create this container image or pull it from the Oracle Container Registry at container-registry.oracle.com. See Prepare to Use the TimesTen Kubernetes Operator.
- When using the TimesTen Operator, the name of the Linux user that is created in the container image is timesten with a numeric UID of 3429. The name of the Linux group that is created in the container image is timesten with a GID of 3429. The timesten user is the user who runs TimesTen and the timesten group is the TimesTen users group. You can override these defaults. This lets you tailor attributes of the image to meet your requirements. See Dockerfile ARGS.
- The TimesTen Operator supports the TimesTen Prometheus Exporter. You can configure your TimesTenClassic and your TimesTenScaleout objects to use the Exporter. The Exporter can then collect metrics from the TimesTen databases that are running in your Kubernetes cluster, and expose these metrics to Prometheus. See Create Your Own Oracle Wallet, Certificates, and Secrets for Exposing TimesTen Metrics.

The TimesTen Operator provides the prometheus object type as part of the TimesTenClassic and TimesTenScaleout object type definitions. Use the prometheus object type to configure the Exporter to meet your Prometheus requirements. See TimesTenClassicSpecSpecPrometheus and TimesTenScaleoutSpecSpecPrometheus.

New features in Release 22.1.1.1.0

You can define a readiness probe to tell Kubernetes that a TimesTen (tt) container is ready. See About Readiness Probes for TimesTen Containers.



1

Overview of the Oracle TimesTen Kubernetes Operator

This chapter provides an overview of containers and Kubernetes. It also gives an overview of the TimesTen Kubernetes Operator. It discusses the TimesTenClassic and TimesTenScaleout *Custom Resource Definitions* (CRDs) and the TimesTen Operator. The chapter details the role the TimesTen Operator plays in deploying, managing, and monitoring active standby pairs of TimesTen Classic databases as well as in deploying TimesTen Scaleout grids and their associated databases in your Kubernetes cluster.

Topics:

- Overview of Containers and Kubernetes
- About the TimesTen Kubernetes Operator
- About TimesTenClassic and TimesTenScaleout Objects
- About Provisioning Active Standby Pairs
- About Deploying a Replicated TimesTenClassic Object
- About Deploying a TimesTen Scaleout Grid and Database

Overview of Containers and Kubernetes

A container is a lightweight virtual machine, running the Linux operating system. A container usually runs one application that is started from an image. Files that are created and modified are usually not persistent. However, persistent storage is available. Containers are a key component of cloud computing environments.

Kubernetes is a portable, extensible, open-source platform for managing containerized workloads and services, that facilitates both declarative configuration and automation. Kubernetes has the ability to manage the resources of multiple hosts (called *Nodes*) in a cluster, and to run containers as required across these nodes. It can automatically spawn containers to react to various failures. Kubernetes also manages the networking among the containers and to the outside world. Kubernetes is portable across many cloud and on-premises environments.

Key Kubernetes concepts include:

Pod: One or more containers that share an IP address. For more information on Pods, see:

https://kubernetes.io/docs/concepts/workloads/pods/pod/

• *Deployment*: A named collection of *n* identical Pods (where *n* is the number of Pods). Kubernetes ensures that *n* identical Pods are running. For more information on Deployments, see:

https://kubernetes.io/docs/concepts/workloads/controllers/deployment/

• *PersistentVolume*: Storage that can be mounted to a Pod and is persistent beyond the lifetime of Pod. For more information on Persistent Volumes, see:

https://kubernetes.io/docs/concepts/storage/persistent-volumes/



• *StatefulSet*: Similar to a Deployment, but each Pod has an associated PersistentVolume. For more information on StatefulSets, see:

https://kubernetes.io/docs/concepts/workloads/controllers/statefulset/

 Service: A network endpoint for a Deployment or StatefulSet. It defines the set of addresses and ports that should be exposed to applications in the Kubernetes cluster. For more information on a Service, see:

https://kubernetes.io/docs/concepts/services-networking/service/

Kubernetes provides the facilities for the provisioning of Pods and other Kubernetes resources that are required to deploy applications. Once deployed, the objects must be monitored and managed.

Kubernetes does some monitoring and managing of applications, but not all. It does handle problems at the Pod level automatically. For example, if a container fails, Kubernetes restarts it automatically. If an entire Node fails, Kubernetes starts replacement Pods on the other Nodes. However, Kubernetes has no knowledge about problems inside a container. This is not problematic for stateless applications, but for databases (which are stateful), Kubernetes needs help managing what is inside the containers.

This help comes in the form of:

- Custom Resource Definition
- Kubernetes Operator

Custom Resource Definition

A *Custom Resource Definition* (commonly known as a CRD) extends the Kubernetes' object model. It adds a new object type to the Kubernetes cluster, similar to the Pod, the StatefulSet, and the Service object types that it natively supports.

Kubernetes Operator

A *Kubernetes Operator* (also called *Operator*) is the brains behind a CRD. An Operator is an application that performs the functions of a human computer operator. It starts, stops, monitors, and manages other applications.

An Operator runs in one or more Pods, one active and the others idle. The active Operator performs the work. The remaining Operators are idle and remain idle until the active Operator fails. The active Operator manages all objects of a particular type and when combined with a CRD enables you to add custom facilities to Kubernetes.

About the TimesTen Kubernetes Operator

The TimesTen Kubernetes Operator provides the ability for you to deploy both active standby pairs of TimesTen Classic databases as well as TimesTen Scaleout grids and their associated databases in your Kubernetes cluster.

The TimesTen Kubernetes Operator consists of these interrelated components:

Custom Resource Definitions (CRDs): There are two CRDs. The TimesTenClassic CRD
defines an object of type TimesTenClassic to Kubernetes. This TimesTenClassic object
type provides the metadata for deploying active standby pairs of TimesTenClassic
databases. The TimesTenScaleout CRD defines an object of type TimesTenScaleout to
Kubernetes. This TimesTenScaleout object type provides the metadata for deploying
TimesTen Scaleout grids and their associated databases.



- TimesTen Operator: There is one TimesTen Operator. The Operator monitors the TimesTenClassic and TimesTenScaleout objects and properly handles both. It deploys, manages, and monitors active standby pairs of TimesTen Classic database. This same Operator deploys TimesTen Scaleout grids and their associated databases.
- TimesTen Agent: There is one TimesTen agent. This agent runs inside each container that runs TimesTen. The TimesTen Operator communicates with these agents both to determine the state of TimesTen insides of the container as well as to create, start, stop, and control TimesTen instances. The agent does not know how to manage TimesTen. It gives information to the Operator and the Operator provides the instructions for the agent. This agent knows how to work with TimesTen Classic and TimesTen Scaleout.

About TimesTenClassic and TimesTenScaleout Objects

The TimesTen Operator distribution provides the file you need to deploy the TimesTenClassic and TimesTenScaleout CRDs in the Kubernetes cluster. Once deployed, Kubernetes understands the TimesTenClassic and TimesTenScaleout object types, just as it understands Pods, Secrets, and Services.

You can define objects of type TimesTenClassic or of type TimesTenScaleout or both. This lets you define the specific attributes for your TimesTen configuration and TimesTen database.

Objects in Kubernetes are named and typed. You can define a TimesTenClassic object named sample and another TimesTenClassic object named sample2. Similarly, you can define a TimesTenScaleout object named sample and another TimesTenScaleout object named sample2. You can have as many of these Kubernetes objects as you want, limited only by the available resources in your Kubernetes cluster.

Objects of different types have different meanings. For example, an object of type TimesTenClassic has a different meaning than an object of type ConfigMap. Therefore, you can define a sample TimesTenClassic object and a sample ConfigMap. The same is true for TimesTenScaleout objects.

Kubernetes supports *namespaces*. Namespaces split a cluster into multiple independent ones. Each namespace has a completely independent set of names. There can be an object of type a called x in namespace1 and a different object of type a called x in namespace2. For example, you can define an object of type TimesTenClassic called sample in the namespace1 namespace, and a different object of type TimesTenClassic called sample in the namespace2 namespace. The same is true for TimesTenScaleout objects.

Note:

CRDs are cluster-scoped, not namespace-scoped.

Kubernetes object definitions are expressed in JSON or YAML. The examples in this book use YAML.

About Provisioning Active Standby Pairs

TimesTen Classic databases almost always run in active standby pairs. Figure 1-1 illustrates an active standby pair replication scheme. There are two databases. One database is the active, and the second database is the standby. Applications update the active database. The standby database is read-only and receives replicated updates from the active database. Only one of the two databases function as the active database at any one time. If the active

database fails, the standby database is promoted to be the active. After the failed (active) database is recovered, it becomes the standby database. See Types of Replication Schemes in the *Oracle TimesTen In-Memory Database Replication Guide* for more information on the active standby pair replication scheme.



Figure 1-1 Active standby pair of TimesTen databases

An active standby pair replication scheme is a good fit for Kubernetes. Specifically, consider a pair of Pods, each with persistent storage, that are running an active standby pair of TimesTen databases. If the Pod containing the active database fails, Kubernetes automatically spawns another Pod to take its place, and attaches the appropriate storage to it.

But, since Kubernetes doesn't know anything about TimesTen, it will not automatically perform the necessary operations to cause the standby database on the surviving Pod to become the active database. This is where the TimesTen Operator comes in.

TimesTen provides a CRD that adds the *TimesTenClassic* object type to Kubernetes as well as an Operator for managing TimesTen databases. The Operator automates setup and initial configuration, manages databases and replication, and automates failover and recovery.

When you define a TimesTenClassic object, you can specify the configuration of your TimesTen deployment using Kubernetes facilities. When you create a TimesTenClassic object in a Kubernetes cluster, a pair of Pods are created, each running TimesTen. Each Pod contains a TimesTen instance. Each instance provides one TimesTen database. Database replication, through active standby pairs, is automatically configured. In short, you can deploy highly available replicated pairs of TimesTen databases and manage them by issuing a small number of commands.

A Kubernetes Operator manages objects of a particular type. TimesTen provides an Operator that manages Kubernetes objects of type TimesTenClassic. In so doing, TimesTen can be deployed, monitored, managed, and controlled in an automated manner with no required human intervention.

About Deploying a Replicated TimesTenClassic Object

When you create a replicated TimesTenClassic object in the Kubernetes cluster, the process to create and configure your active standby pair of databases begins. The Operator is invoked

and creates several Kubernetes objects that are required to run TimesTen. After the objects are created and linked together, the TimesTen containers run a script to configure and start the TimesTen agent. The Operator communicates with the TimesTen agent that is running in each Pod in order to monitor and control TimesTen. The Operator configures one database as the active database, copies the active database to the standby, and then configures the active standby pair replication scheme. The process is described in detail in these sections:

- About Objects Created by the TimesTen Operator
- About the TimesTen Containers and the TimesTen Agent
- Simple Deployment

About Objects Created by the TimesTen Operator

The Operator creates a number of Kubernetes objects that are required to run TimesTen, including a StatefulSet, a Service, and a Secret. These objects in turn create other objects. All of these objects are linked together by Kubernetes and are associated with the TimesTenClassic object you created. Figure 1-2 shows the objects that are created and how they are linked together.



Figure 1-2 Creating the TimesTenClassic object

The objects that are created are described in the following sections:

- StatefulSet
- Service
- Secret
- Pods



Events

StatefulSet

The Operator creates a StatefulSet consisting of two Pods to run TimesTen. Each Pod has one or more PersistentVolumes (persistent storage), that are mounted in the TimesTen containers. This is where your TimesTen databases are stored. Applications running in the containers with PersistentVolumes mounted can create files that live beyond the lifetime of the container. (By default, all files that containers create and modify automatically vanish when the container exits. Containers are ephemeral.)

One attribute of a StatefulSet is the number of replicas that can be provisioned. Each TimesTenClassic object has an associated StatefulSet with two replicas. If one Pod fails, Kubernetes automatically creates a new one to replace it, and automatically mounts the appropriate PersistentVolume(s) to it.

For example, for a TimesTenClassic object called sample, the Operator creates a StatefulSet called sample in the same Kubernetes namespace. The StatefulSet, in turn, creates two Pods in the namespace, called sample-0 and sample-1.

Service

A Kubernetes Service defines the set of network addresses and ports that should be exposed to applications in the cluster.

The Operator automatically creates a *headless* Service when you create the TimesTenClassic object. It automatically associates this Service with the StatefulSet. This causes Kubernetes to define entries in the Kubernetes cluster's DNS for the Pods in the StatefulSet, and to keep those DNS entries up to date.

A headless Service is used such that the DNS name/address entry for the active database is different than the DNS name/address entry for the standby database. This enables incoming client connections to be routed to the database that is active. For more information on a headless Service, see:

https://kubernetes.io/docs/concepts/services-networking/service/#headlessservices/

For a TimesTenClassic object called sample, a headless Service called sample is also created in the same Kubernetes namespace. This results in entries in the cluster's DNS for sample-0.sample.namespace.svc.cluster.local and sample-1.sample.namespace.svc.cluster.local.

Secret

The TimesTen Operator creates a Secret to inject an SSL certificate into the TimesTen containers. This secures the communication between the TimesTen Operator and the TimesTen Agent.

Pods

The StatefulSet creates two pods. Each Pod contains three containers:

• The tt container. This TimesTen container is always present in the Pods. It executes the TimesTen agent and runs TimesTen.



- The daemonlog container: This container copies the contents of the TimesTen ttmesg.log file to stdout, resulting in Kubernetes logging the file. This enables the daemon log of the TimesTen instances to be recorded by the Kubernetes infrastructure.
- The exporter container: This container runs the TimesTen exporter. The TimesTen exporter is a utility that exports TimesTen metrics to Prometheus or some other scraping mechanism.

Events

The Operator creates a Kubernetes event whenever important changes occur.

About the TimesTen Containers and the TimesTen Agent

After the objects are created, the TimesTen containers run a script that configures and starts the TimesTen agent. The Operator communicates with the TimesTen agent running in each Pod, in order to configure, manage, and monitor TimesTen in that Pod. The agent provides an HTTPS endpoint in the Pod that the Operator uses to query and control the tt container in the Pod. If the TimesTen agent fails, the tt container automatically terminates and is re-spawned by Kubernetes. Figure 1-3 illustrates the two way communication between the Operator and the TimesTen agent.



Figure 1-3 The Operator and the TimesTen agent

The TimesTen agent starts TimesTen and thus runs as the instance administrator user. It has full control over TimesTen.

Simple Deployment

The TimesTen Operator is designed for simple deployment of your active standby pairs of TimesTen Classic databases and for automated failure detection and recovery. For example,

You decide you want to deploy a new replicated pair of TimesTen databases.

- You decide the attributes of those databases.
- You create the configuration files for those attributes.
- You use the kubectl create command to create a TimesTenClassic object to represent the replicated pair.
- You use the kubectl get and kubectl describe commands to observe the provisioning of the active standby pair.
- Applications that run in other Pods use TimesTen's standard client/server drivers to access TimesTen databases.

You do not have to monitor the TimesTen databases continually, configure replication, perform failover, or re-duplicate a database after failure. The TimesTen Operator performs all these functions and works to keep the databases up and running with minimal effort on your part.

About Deploying a TimesTen Scaleout Grid and Database

When you create a TimesTenScaleout object in the Kubernetes cluster, the process to create and configure the TimesTen Scaleout grid and its associated database begins. Kubernetes informs the TimesTen Operator that a TimesTenScaleout object has been created. The Operator begins the process of creating several Kubernetes objects that are required to deploy the grid.

Topics:

- StatefulSets
- Services
- Secret
- Persistent Volume Claims and Pods
- Password-less ssh
- Quick Rollout

StatefulSets

The TimesTen Operator creates the following StatefulSets:

- One StatefulSet that provides the management instance for the grid. The underlying Pod for this management instance is also created.
- One StatefulSet that provides one or more ZooKeeper instances. There is one StatefulSet replica for each Zookeeper instance. For example, if there are three ZooKeeper instances, there is one StatefulSet and this one StatefulSet has three replicas.
- One or more additional StatefulSets, the number of which is determined by your K-Safety
 (k) value. For example, a k value of 2 means that there are two copies of your TimesTen
 database. One copy of your database resides in data space one, and the second copy in
 data space two. In the Kubernetes environment, the TimesTen Operator creates k
 StatefulSets. Using the previous example, if k is set to 2, the Operator creates two
 StatefulSets. Each of the k StatefulSets provides Pods to implement a single data space in
 the resultant grid.

The number of replicas for each StatefulSet is determined by the number of replica sets you want in your grid. A replica set contains k elements, where each element in the replica set is an exact copy of the other elements in the replica set. For example, if you want three replica sets in your grid, each StatefulSet has three replicas. Each replica contains a Pod



for one data instance. Therefore, in this example, one StatefulSet has three Pods each of which contain one data instance. The second StatefulSet (assuming k=2) also has three replicas and each replica also contains a Pod for one data instance. Therefore, the second StatefulSet has three Pods each of which contain one data instance.

In summary, in a case where k is set to 2 and the number of replica sets is 3, the Operator creates two StatefulSets, each with three replicas. Each StatefulSet provides the Pods to implement a single data space. There are a total of six Pods created for the six data instances.

Services

The TimesTen Operator creates the following headless Services:

- One headless Service that provides DNS names for the Pods that contain the management and data instances. This Service enables client/server access to the Pods using the TimesTen client/server port 6625.
- One headless Service that provides DNS names for the Pods that contain the ZooKeeper instances. This Service enables access to the Zookeeper internal ports 2888 and 3888, as well as the external port 2181.

Secret

The TimesTen Operator creates a Secret to inject an SSL certificate into the TimesTen containers. This secures the communication between the TimesTen Operator and the TimesTen Agent.

Persistent Volume Claims and Pods

The TimesTen Operator creates Persistent Volume Claims (PVCs) for the TimesTen and ZooKeeper Pods. These PVCs cause persistent volumes to be allocated by Kubernetes and attached to the TimesTen and ZooKeeper Pods. See Persistent Storage.

Each Pod that runs a ZooKeeper instance consists of a single container called zookeeper.

Each Pod that runs TimesTen consists of at least two containers:

- tt container: Runs the TimesTen Agent and TimesTen.
- daemonlog container: Writes the TimesTen daemonlog to stdout. This enables the daemon log of the TimesTen instances to be recorded by the Kubernetes infrastructure.

As the ${\tt tt}$ containers in the TimesTen Pods start, the Operator assembles them into a working grid.

Password-less ssh

A TimesTen Scaleout grid relies on password-less ssh among the instances of the grid. The TimesTen Operator automatically configures password-less ssh among the tt containers in the grid in your Kubernetes environment. There is no intervention that you need to do.

Quick Rollout

The TimesTen Kubernetes Operator provides the functionality for deploying TimesTen Scaleout grids and their associated databases. The TimesTen Operator rolls out the grid quickly and proficiently.



The Operator creates the StatefulSets and Services that are required to deploy the grid. It creates the ZooKeeper Pods. When the TimesTen Agent in the management Pod starts up, the Operator instructs the agent to create a TimesTen instance and grid. The Operator waits for all of the TimesTen agents in all of the Pods to start. Once all of the agents have started, the Operator instructs the agent in the management instance to create the hosts, the installations, and the instances in the grid's model for the data instances in the grid. The DbDef is created as are any direct and client/server connectables. The model is applied and the data instances of the grid are created.

The Operator instructs the management instance to create the database and to create the initial distribution map. The Operator then instructs the TimesTen agent in one data instance to run the TimesTen ttisql utility to create initial database users and objects.

The grid and databases are created. The TimesTen agent opens the database for connections.

The grid rollout is quick and proficient with little or no intervention from you. Once the grid is rolled out, the TimesTen Operator does not manage the grid or database or perform maintenance operations.

Here is additional information:

- For information about TimesTen Scaleout: See Overview of TimesTen Scaleout in the Oracle TimesTen In-Memory Database Scaleout User's Guide.
- For information about deploying a TimesTen Scaleout grid and database in Kubernetes: See Deploy TimesTen Scaleout Databases in this book.

Prepare to Use the TimesTen Kubernetes Operator

Before you can use the TimesTen Kubernetes Operator (TimesTen Operator), you must choose a container image that contains TimesTen and a container registry that contains TimesTen container images. You must also obtain TimesTen YAML manifest files and Helm charts. These files and charts are necessary to deploy the TimesTen Operator.

This chapter discusses your options and then shows you how to perform the tasks for each option.

Topics:

- About TimesTen Container Images and Container Registry Options
- Before You Begin
- Choose a TimesTen Container Image
- Obtain TimesTen YAML Manifest Files and Helm Charts

About TimesTen Container Images and Container Registry Options

A TimesTen container image contains everything that is needed to run the TimesTen Operator and TimesTen containers in the TimesTen Pods created by this Operator. You can acquire these container images from a number of container registries. In addition, TimesTen provides a Dockerfile that lets you build your own TimesTen container image that you can then store in a container registry of your choice.

A TimesTen container image stored in a container registry is necessary to run TimesTen and the TimesTen Operator in your Kubernetes cluster.

TimesTen provides several options for obtaining a TimesTen container image for use in your Kubernetes environment:

- Container images on Oracle Container Registry (container-registry.oracle.com): TimesTen publishes official TimesTen container images to two TimesTen repositories on Oracle Container Registry:
 - timesten: Contains TimesTen container images for Oracle TimesTen In-Memory Database (TimesTen). This repository contains container images that run a licensed copy of TimesTen. Using container images in this repository require that you accept the Oracle license agreement.

For information about TimesTen, see Overview for the Oracle TimesTen In-Memory Database in Oracle TimesTen In-Memory Database Introduction.

 timesten-xe: Contains TimesTen container images for Oracle TimesTen In-Memory Database Express Edition (TimesTen XE). The TimesTen XE license enables free of charge usage of TimesTen and is subject to Oracle Free Use Terms and Conditions. You do not need to accept an Oracle license agreement to use TimesTen container images in this repository. For information about TimesTen XE, see Overview in Oracle TimesTen In-Memory Database Quick Start Guide for Express Edition.

A TimesTen container image on Oracle Container Registry corresponds to a specific TimesTen release. For example, the container-registry.oracle.com/timesten/timesten/22.1.1.34.0 container image contains TimesTen release 22.1.1.34.0. Simply scroll through the list of container images on Oracle Container Registry to identify a container image that you want to use when deploying the TimesTen Operator and TimesTen databases in your Kubernetes environment.

- Container images on Oracle Cloud Marketplace: TimesTen provides a container listing called Oracle TimesTen In-Memory Database for Kubernetes BYOL. You use this listing to export a TimesTen container image into a repository in Oracle Cloud Infrastructure Registry (Container Registry). See Introduction in the Oracle TimesTen In-Memory Database Using Oracle Cloud Marketplace to Obtain a TimesTen Container Image (BYOL) guide.
- Build your own TimesTen container image: If you have a TimesTen distribution, you can use the Dockerfile file provided in this distribution to build a TimesTen container image. You can then tag and push this container image to a container registry of your choice.

Before You Begin

Ensure to set up the following:

- A working Kubernetes cluster:
 - Your cluster must provide a StorageClass that is used to request Persistent Volumes.
 You must know the name of this storage class. For example, in Oracle Cloud
 Infrastructure Container Engine for Kubernetes (OKE), you can use the oci-bv storage
 class. The examples in this book use this oci-bv storage class.
 - The nodes in your cluster must have their clocks synchronized through NTP, or equivalent.

Note:

For a list of supported Kubernetes releases, see the Oracle TimesTen In-Memory Database Release Notes.

- A development host to access the Kubernetes cluster. For this host, ensure the following:
 - The host resides outside the Kubernetes cluster.
 - You can control and access the Kubernetes cluster from this host.
 - You install the following command line tools on the host:
 - * **kubectl:** You use kubect1 to communicate with the Kubernetes cluster.
 - * docker or podman: You use docker or podman when working with TimesTen container images on your local development host.



Note: The podman tool has a command that mimics docker. Examples in this guide use the docker command. See podman in the Podman documentation.

* helm: You can use TimesTen Helm charts to deploy the TimesTen Kubernetes Operator and create TimesTen databases, If you plan on using TimesTen Helm charts, install the helm command line utility.

Let's run through a quick test to verify you have docker and kubectl pre-installed and configured on your development host.

1. On your development host, run docker hello-world.

docker run hello-world

The output is similar to the following:

```
!... Hello Podman World ...!
         .--"--.
      / - - \
/ (0) (0) \
   \sim \sim \sim \mid -=(, Y,) = - \mid
    .---. /` \ |~~
 ~/ 0 0 \~~~~.---. ~~
  | = (X) = | ~ / (O (O) )
   ~~~~~~ ~ | = (Y ) =- |
  ~~~~ ~~~ U |~~
Project: https://github.com/containers/podman
Website: https://podman.io
Desktop: https://podman-desktop.io
Documents: https://docs.podman.io
YouTube: https://youtube.com/@Podman
X/Twitter: @Podman io
Mastodon: @Podman io@fosstodon.org
```

2. Confirm kubectl returns the nodes in your Kubernetes cluster.

kubectl get nodes

The output is similar to the following:

NAME	STATUS	ROLES	AGE	VERSION
192.0.2.1	Ready	node	32d	v1.27.2
192.0.2.2	Ready	node	31d	v1.27.2
192.0.2.4	Ready	node	101d	v1.27.2
192.0.2.6	Ready	node	101d	v1.27.2
192.0.2.10	Ready	node	32d	v1.27.2



Choose a TimesTen Container Image

As discussed in About TimesTen Container Images and Container Registry Options, you have several options for obtaining TimesTen container images for use in your Kubernetes environment. Choose one of the following options and complete the tasks for the option.

- Option 1: Use a Container Image from Oracle Container Registry
- Option 2: Use a Container Image from Oracle Cloud Marketplace
- Option 3: Build a Container Image and Push It to a Container Registry of Your Choice

Option 1: Use a Container Image from Oracle Container Registry

Here are the tasks to use a TimesTen container image from Oracle Container Registry:

- Create Auth Token
- Accept the Oracle License Agreement
- Configure Your Development Host and Kubernetes Cluster

Create Auth Token

- 1. From a web browser, log in to the Oracle Container Registry web interface using your Oracle account.
- 2. In the upper right hand corner next to your **Oracle login**, expand the **down arrow**, and choose **Auth Token**.
- 3. On the Auth Token page, choose Generate Key.
- 4. Copy and save the generated key (auth token) in a safe location. You need it later.

Accept the Oracle License Agreement

- 1. If you are not already on Oracle Container Registry, log in to the Oracle Container Registry web interface using your Oracle account.
- 2. From the TimesTen Repositories page, in the Repository column, choose either the timesten or timesten-xe repository. The timesten repository contains container images for running TimesTen, whereas the timesten-xe repository contains container images for running TimesTen XE.
- 3. Accept the Oracle license agreement. Complete this step only if you choose container images in the timesten repository. Since TimesTen is a licensed Oracle product, you must accept the Oracle license agreement to use a container image in this repository. You only need to complete this step once.
 - a. On the Official container images for the Oracle TimesTen In-Memory Database page, to the right of the Quick Reference Description, locate the Select Language drop down list. In the Select Language drop down list, choose your language. Then, review the text before the Continue button and click Continue.

The text that is displayed before the **Continue** button is similar to the following: "You must agree to and accept the *Oracle Standard Terms and Restrictions* prior to downloading from the Oracle Container Registry. Please read the license agreement on the following page carefully."



b. On the **Oracle Standard Terms and Restrictions** page, review the information on the page, then at the bottom of the page, click **Accept**.

The Official container images for the Oracle TimesTen In-Memory Database page displays for a second time. To the right of the Quick Reference Description, look for a green check mark with text similar to the following: "You last accepted the Oracle Standard Terms and Restrictions on 01/08/2025 at 01:28 PM Coordinated Universal Time (UTC)."

4. Choose a TimesTen container image.

On either the timesten or timesten-xe repository page, scroll through the list of container images to identify a TimesTen container image that you want to use for deploying the TimesTen Kubernetes Operator in your Kubernetes cluster.

For example, let's choose container-registry.oracle.com/timesten/ timesten:22.1.1.34.0.

Here's a summary sheet of the information obtained. Save it for future reference.

Item	Example
Your Oracle Container Registry user name	john.smith@example.com
Generated auth token	z1SbL04JgwqzLEn1ZxJ
Container image	<pre>container-registry.oracle.com/timesten/ timesten:22.1.1.34.0</pre>

Configure Your Development Host and Kubernetes Cluster

1. On your development host, use the docker login command to log in to Oracle Container Registry.

docker login container-registry.oracle.com

At the prompt, enter the following:

- username: Enter the username you use to sign in to the Oracle Container Registry web interface. This example uses john.smith@example.com.
- password: Enter the auth token you previously generated and saved.

```
Username: john.smith@example.com
Password: auth token
Login Succeeded!
```

If you are using docker, the docker login operation creates or updates the \$HOME/.docker/config.json file with the auth token you provided when you were prompted for your password. You now can use docker to access and pull TimesTen container images from Oracle Container Registry to your local development host.

If you are using podman, the credentials may be stored in a different location. See https:// docs.podman.io/en/v4.3/markdown/podman-login.1.html.



 (Optional): Confirm the credentials are stored in the config.json file. This example assumes the credentials are located in \$HOME/.docker/config.json.

```
cat $HOME/.docker/config.json
```

The output contains the auth token for container-registry.oracle.com.

```
{
    "auths": {
        "container-registry.oracle.com": {
                "auth": "z1SbL04JgwqzLEn1ZxJ"}
     }
}
```

3. Pull the TimesTen container image from Oracle Container Registry.

docker pull container-registry.oracle.com/timesten/timesten:22.1.1.34.0

 Create the image pull secret. This example creates the sekret secret. You can use any name.

```
kubectl create secret generic sekret \
--from-file=.dockerconfigjson=$HOME/.docker/config.json \
--type=kubernetes.io/dockerconfigjson
```

5. (Optional): Confirm the Secret is created in your Kubernetes cluster.

```
kubectl get secretsTYPEDATAAGEsekretkubernetes.io/dockerconfigjson11d111
```

For more information, see Pulling Images from Registry during Deployment in the Kubernetes documentation.

6. Save the name of the image pull secret. You need it later.

Congratulations! You obtained a TimesTen container image from Oracle Container Registry.

Next Steps

Proceed to Obtain TimesTen YAML Manifest Files and Helm Charts.

Option 2: Use a Container Image from Oracle Cloud Marketplace

See Introduction in the Oracle TimesTen In-Memory Database Using Oracle Cloud Marketplace to Obtain a TimesTen Container Image (BYOL) guide.

Option 3: Build a Container Image and Push It to a Container Registry of Your Choice

Here are the tasks to build a TimesTen container image and push it to a registry of your choice:



- Configure Your Development Host and Kubernetes Cluster
- Download and Unpack a TimesTen Distribution
- Build a TimesTen Container Image
- Tag and Push the Container Image to Your Container Registry

Configure Your Development Host and Kubernetes Cluster

Let's choose container-registry.visioncorp.com/repo for the registry and repository of your choice.

1. On your development host, use the docker login command to log in to your containerregistry.visioncorp.com container registry. At the prompt, enter your credentials for your container registry.

docker login container-registry.visioncorp.com

If you are using docker, the docker login operation creates or updates the \$HOME/.docker/config.json file with the auth token you provided when you were prompted for your password.

If you are using podman, the credentials may be stored in a different location. See https:// docs.podman.io/en/v4.3/markdown/podman-login.1.html

2. On your development host, create an image pull secret.

A Kubernetes Secret takes a copy of your credentials from the docker login command and makes them available to your Kubernetes cluster.

This example creates a Secret called sekret. You can use any name.

```
kubectl create secret generic sekret \
--from-file=.dockerconfigjson=$HOME/.docker/config.json \
--type=kubernetes.io/dockerconfigjson
```

For more information, see Pulling Images from Registry during Deployment in the Kubernetes documentation.

3. Save the name of the image pull secret. You need it later.

Download and Unpack a TimesTen Distribution

- 1. Download a TimesTen distribution to a download location.
- 2. On your development host, from the directory of your choice:
 - Create one subdirectory for the TimesTen distribution. This example creates the *installation dir* subdirectory.
 - Create a second subdirectory for the TimesTen Operator distribution. This example creates the *kube_files* subdirectory.

```
mkdir -p installation_dir
mkdir -p kube files
```


Change to the TimesTen distribution subdirectory. Next, copy the TimesTen distribution that you previously downloaded into this subdirectory.

```
cd installation_dir
cp download_location/timesten2211340.server.linux8664.zip .
```

Unpack the TimesTen distribution.

unzip timesten2211340.server.linux8664.zip

The output is similar to the following:

```
Archive: timesten2211340.server.linux8664.zip
    creating: tt22.1.1.34.0/
    ...
    creating: tt22.1.1.34.0/kubernetes/
    extracting: tt22.1.1.34.0/kubernetes/operator.zip
...
```

The TimesTen Kubernetes Operator distribution is *installation_dir/tt22.1.1.34.0/* kubernetes/operator.zip.

Note:

Do not delete the TimesTen distribution. You need it later.

 Change to the TimesTen Kubernetes Operator subdirectory you created in a previous step. (In this example, *kube_files*.) Next, unpack the TimesTen Kubernetes Operator distribution.

```
cd kube_files
unzip installation dir/tt22.1.1.34.0/kubernetes/operator.zip
```

The output is similar to the following:

Archive: installation dir/tt22.1.1.34.0/kubernetes/operator.zip

```
...
creating: helm/
creating: helm/ttclassic/
...
creating: helm/ttcrd/
...
creating: helm/ttoperator/
...
creating: deploy/
inflating: deploy/crd.yaml
inflating: deploy/service_account.yaml
inflating: deploy/service_account_cluster.yaml
inflating: deploy/operator.yaml
creating: image/
inflating: image/Dockerfile
```



```
creating: operator/
...
```

Build a TimesTen Container Image

TimesTen provides the files you need to build a TimesTen container image in its TimesTen Kubernetes Operator distribution. In this example, the directory that contains the unzipped TimesTen Kubernetes Operator distribution is *kube files*.

1. On your development host, change to the image directory of the unzipped TimesTen Kubernetes Operator distribution (*kube files*/image, in this example).

cd kube files/image

2. Copy the TimesTen distribution into the directory. In a previous example, you created the *installation dir* directory. This directory contains the TimesTen distribution.

cp installation dir/timesten2211340.server.linux8664.zip .

3. On your development host, change to the *kube_files/image* directory if you are not in that directory. Next, use the docker build command to build a TimesTen container image.

This example specifies the following Dockerfile ARGs on the command line.

- TT_DISTRO: The name of the file containing the TimesTen distribution (timesten2211340.server.linux8664.zip, in this example).
- TT_RELEASE: The name of the TimesTen release in dotted format (22.1.1.34.0, in this example).

```
docker build -t ttimage:22.1.1.34.0 \
--build-arg TT_DISTRO=timesten2211250.server.linux8664.zip \
--build-arg TT RELEASE=22.1.1.34.0 .
```

The build process starts. When completed, output similar to the following displays:

Successfully tagged ttimage:22.1.1.34.0

The name of the container image is ttimage:22.1.1.34.0.

4. Save the name of the container image. You need it later.

Congratulations! You successfully built a TimesTen container image.

Tag and Push the Container Image to Your Container Registry

Let's use the container image you built in Build a TimesTen Container Image. In the example, the name of the image is ttimage:22.1.1.34.0.

1. On your development host, tag the TimesTen container image.

docker tag ttimage:22.1.1.34.0 container-registry.visioncorp.com/
repo:ttimage22.1.1.34.0



2. Push the TimesTen container image to your container registry.

```
docker push container-registry.visioncorp.com/repo:ttimage22.1.1.34.0
```

Note:

To reduce the size of the final TimesTen container image, the Dockerfile uses a multi-stage build. This results in a dangling image left behind. To locate dangling images, use the docker command with the -f filter flag with a value of dangling=true. Once you locate the dangling image, you can use the docker image prune command to remove it. This example removes all dangling images.

docker images -f dangling=true
docker image prune

Congratulations! You successfully tagged and pushed a TimesTen container image to a registry of your choice.

Next Steps

Proceed to Obtain TimesTen YAML Manifest Files and Helm Charts.

Obtain TimesTen YAML Manifest Files and Helm Charts

TimesTen provides YAML manifest files and Helm charts in TimesTen container images. You use these files and charts to install the TimesTen Operator and create TimesTen databases in your Kubernetes cluster.

Now that you have completed the tasks to obtain and use a TimesTen container image in your Kubernetes environment, you can use this container image to obtain TimesTen YAML manifest files and Helm charts. To do so, you create a container from the container image and copy the TimesTen YAML manifest files and Helm charts from the container to a location on your development host.

In this example, let's use a container image located on Oracle Container Registry. If you are using a container image in another location, use it instead. Let's walk through the steps to copy the TimesTen YAML manifest files and Helm charts from this container image to a directory on your development host. This example uses <code>container-registry.oracle.com/timesten/timesten:22.1.1.34.0</code>.

1. On your development host, from a directory of your choice, create a subdirectory for the TimesTen YAML manifest files and Helm charts. This example creates the kube_files subdirectory.

mkdir -p kube files

2. Create a new container from the container-registry.oracle.com/timesten/ timesten:22.1.1.34.0 image, supplying a name for the new container. In this example, the name of the container is ttoper.

```
docker create --name ttoper container-registry.oracle.com/timesten/
timesten:22.1.1.34.0
```



3. Copy the YAML manifest files and Helm charts from the ttoper container to the kube_files directory on your development host. The ttoper:/timesten/operator/deploy directory contains the YAML manifest files and the ttoper:/timesten/operator/helm directory contains the Helm charts.

```
docker cp ttoper:/timesten/operator/deploy kube_files
docker cp ttoper:/timesten/operator/helm kube files
```

Verify the directories exist on your development host.

ls kube files/*

The output is the following:

ttclassic ttcrd ttoperator

kube_files/deploy: crd.yaml operator.yaml service_account.yaml service_account_cluster.yaml kube_files/helm:

The deploy directory contains the TimesTen YAML manifest files and the helm directory contains the TimesTen Helm charts. You use these files later to deploy the TimesTen Operator and create TimesTen databases.

4. Remove the ttoper container.

docker rm ttoper

5. Remove the TimesTen container image.

docker image rm container-registry.oracle.com/timesten/timesten:22.1.1.34.0

Congratulations! You successfully obtained the TimesTen YAML manifest files and Helm charts.

Next Steps

You are now ready to install the TimesTen Custom Resource Definitions (CRDs). See the next chapter Install TimesTen Custom Resource Definitions.



3 Install TimesTen Custom Resource Definitions

Learn about TimesTen Custom Resource Definitions (CRDs) and how to install them.

Topics:

- About TimesTen CRDs
- Install TimesTen CRDs

About TimesTen CRDs

Kubernetes operates on objects of various built-in object types, including Pods, Services, and StatefulSets. One way to extend Kubernetes is with CRDs. CRDs define new object types to Kubernetes. Once installed, Kubernetes understands these object types just like it understands Pods, Services, StatefulSets, and all other built-in object types.

The TimesTen Kubernetes Operator (TimesTen Operator) provides two CRDs, TimesTenClassic and TimesTenScaleout. The TimesTenClassic CRD defines the attributes and metadata for TimesTen Classic databases while the TimesTenScaleout CRD defines the attributes and metadata for TimesTenScaleout grids and associated databases.

The CRD definition, including the object type and its set of attributes, is referred to as a schema.

Kubernetes supports CRD versioning whereby different versions of the same CRD are supported simultaneously and each version can provide a different schema. This enables the attributes of a CRD to be extended from release to release.

The TimesTen Operator provides different versions of the TimesTenClassic CRD. The v4 schema version is the most current. It provides the attributes of the v2 schema version plus additional attributes specific to the latest release of the TimesTen Operator. When defining new TimesTenClassic objects, use the v4 schema version. The v3 and v2 schema versions are available to facilitate compatibility with previous releases of the TimesTen Operator. No additional changes are made to these schema versions.

Note:

The v2 schema version is deprecated. It is fully supported in this release. However, it will be removed in a future release.

For more information about the TimesTenClassic CRD, see About the TimesTenClassic Object Type.

Install TimesTen CRDs

CRDs are cluster-scoped. When you install CRDs in your Kubernetes cluster, the installation affects all namespaces within the cluster.



Let's walk-through an example showing you how to install the TimesTen CRDs using a TimesTen YAML manifest file. If you want to use TimesTen Helm charts, see Use Helm in Your TimesTen Kubernetes Operator Environment.

Before you begin the following steps, ensure you complete the tasks in Prepare to Use the TimesTen Kubernetes Operator. After completing the tasks, the TimesTen YAML manifest files and Helm charts reside in a location on your development host. In this example, the location is kube files/deploy.

1. On your development host, change to the *kube_files/deploy* directory.

cd kube files/deploy

2. Install the TimesTen CRDs.

kubectl create -f crd.yaml

The output is the following:

customresourcedefinition.apiextensions.k8s.io/ timestenclassics.timesten.oracle.com created

3. Verify installation.

kubectl get crds | grep timesten

The output is similar to the following:

timestenclassics.timesten.oracle.com

2025-01-17T14:41:03Z

Congratulations! You successfully installed the TimesTen CRDs. You are now ready to install the TimesTen Kubernetes Operator. Proceed to Learn About and Install the TimesTen Kubernetes Operator.

Learn About and Install the TimesTen Kubernetes Operator

The TimesTen Kubernetes Operator (TimesTen Operator) creates, manages, and monitors TimesTenClassic objects (and associated databases) in your Kubernetes environment. Let's learn about the TimesTen Operator and walk through the steps to install it.

Topics:

- About Kubernetes Operators
- About the TimesTen Operator
- About Deploying in a Multi-Architecture Kubernetes Cluster
- About the Default Kubernetes Security Context for the TimesTen Operator
- About Readiness and Liveness Probes
- About Privileges
- About Installing the TimesTen Operator
- About Customizations for a TimesTen Operator Deployment
- Install the TimesTen Operator

About Kubernetes Operators

Kubernetes Operators monitor a Kubernetes cluster for the creation of objects of specified types. When you create an object of a supported type, a Kubernetes Operator takes action to implement the object. Kubernetes Operators can be namespace-scoped or cluster-scoped. Namespace-scoped Operators monitor a Kubernetes cluster looking for supported objects in the namespace in which the Operator is running whereas cluster-scoped Operators monitor a Kubernetes cluster looking for supported objects in the namespace in which the Operator is running whereas cluster-scoped Operators monitor a Kubernetes cluster looking for supported objects in any namespace in the cluster.

TimesTen provides a customized Kubernetes Operator called the TimesTen Kubernetes Operator.

About the TimesTen Operator

The TimesTen Operator is an application that functions like a human operator. It creates, monitors, and manages TimesTen databases in a Kubernetes cluster. It configures a database, configures the database's users and schema, and if applicable, configures cache and replication. The TimesTen Operator automatically manages and monitors the health of TimesTen databases and takes action to fix problems. For example, in an active standby pair replication scheme, if the active database fails, the TimesTen Operator takes automatic action by promoting the standby to the new active. Kubernetes replaces the failed Pod that contains the active database. The TimesTen Operator brings TimesTen back up as the new standby in this replaced Pod. The TimesTen Operator performs these operations automatically and there is no human intervention needed. This TimesTen Operator can create and manage many TimesTen databases simultaneously.



You can install the TimesTen Operator in either of the following ways:

- In one or more namespaces in your Kubernetes cluster at namespace-scope (namespacescoped).
- In the timesten-operator namespace in your Kubernetes cluster at cluster-scope (clusterscoped). The timesten-operator namespace is non-configurable and is reserved for internal use. This namespace is explained in later sections.

You have the option of deciding which is best. Let's look at these options in further detail.

- If you install the TimesTen Operator in a namespace in your Kubernetes cluster at namespace-scope, you can then create TimesTenClassic objects in that same namespace. The TimesTen Operator provisions Pods, Services, and other Kubernetes resources necessary to deploy TimesTen databases in this same namespace. If you want to deploy TimesTen databases in multiple namespaces, you must install multiple copies of the TimesTen Operator, one in each namespace. While this adds complexity, it also adds flexibility. For example, since it is possible to run different versions of the TimesTen Operator in each namespace, test and production environments can share a single cluster.
- If you install the TimesTen Operator in a Kubernetes cluster at cluster-scope, a new namespace called timesten-operator is created in the cluster. The TimesTen Operator runs in this timesten-operator namespace. All other namespaces in the Kubernetes cluster can create TimesTenClassic objects. These objects are handled by this TimesTen Operator. The Pods, Services, and other Kubernetes resources that are used to provision these databases reside in the namespace of the associated TimesTenClassic objects. Since there is only one TimesTen Operator that manages all TimesTenClassic objects, this option adds a less complex environment. However, this environment is less flexible. For example, there can only be one version of the TimesTen Operator, so test and production environments may not be able to share a single cluster if these environments require different versions of the TimesTen Operator.

Note:

The timesten-operator namespace is reserved for internal use. You must not run anything in the timesten-operator namespace. In addition, you must create TimesTenClassic objects in your own namespaces.

Regardless of which scope you use, when you create a TimesTenClassic object in a namespace (x, for example), any references to other Kubernetes objects included in that object also refer to objects in namespace x. This includes dbConfigMap, dbSecret, imagePullSecret, and all other object references. A TimesTenClassic object in namespace x cannot refer to such objects in other namespaces.

About Deploying in a Multi-Architecture Kubernetes Cluster

Kubernetes supports single architecture and multi-architecture clusters. A single architecture cluster is a cluster in which nodes within the cluster use the same computer instruction set. For example, a single-architecture cluster could consist of all arm64 nodes or all amd64 nodes. A multi-architecture cluster is a Kubernetes cluster in which nodes within the cluster may be of different computer instruction sets. For example, a multi-architecture cluster could consist of both amd64 and arm64 nodes.

The TimesTen Operator runs in single and multi-architecture Kubernetes clusters. Pods that are created by the TimesTen Operator run on nodes of the same node type as the TimesTen



Operator. For example, if the TimesTen Operator runs on amd64 nodes, Pods created by this TimesTen Operator also run on amd64 nodes.

If you are running the TimesTen Operator in a namespace-scoped environment, you can deploy the TimesTen Operator in one namespace on arm64 nodes (and have it provision and manage TimesTen on arm64 nodes) and you can deploy a second TimesTen Operator in a different namespace on amd64 nodes (and have it provision and manage TimesTen on amd64 nodes). However, if you are running the TimesTen Operator in a cluster-scoped environment, you must use the same node type. For more information about namespace-scoped and cluster-scoped Operators, see About Kubernetes Operators and About the TimesTen Operator.

The affinity settings of both the TimesTen Operator YAML manifest files and the TimesTen Operator helm charts control whether the Operator and the objects it manages uses arm64 or amd64 nodes. For example, here is a snippet of the affinity section of a TimesTen Operator YAML manifest file:

```
# An example affinity definition; this pod will only be assigned to a node
# running on amd64 (the default)
#
#
     affinity:
#
        nodeAffinity:
#
         requiredDuringSchedulingIgnoredDuringExecution:
#
            nodeSelectorTerms:
#
              - matchExpressions:
#
                - key: "kubernetes.io/arch"
#
                  operator: In
                  values: ["amd64"]
#
. . .
```

If your Kubernetes cluster consists of a single architecture, you do not need to change the affinity section. However, if you are using a multi-architecture cluster, you must instruct Kubernetes to deploy the TimesTen Operator on a specific architecture. Here's how:

- Uncomment the .affinity.nodeAffinity section.
- In the .affinity.nodeAffinity.nodeSelectorTerms section where key has the value "kubernetes-io/arch", specify either "amd64" or "arm64" for values.

For example, to instruct Kubernetes to deploy the TimesTen Operator and the objects it manages on amd64 nodes, do the following:

Valid values are as follows:



- "amd64": Use this for TimesTen Operators that you want to run on amd64 nodes.
- "arm64": Use this for TimesTen Operators that you want to run on arm64 nodes.

About the Default Kubernetes Security Context for the TimesTen Operator

A Kubernetes security context defines privilege and access control settings for a Pod or Container. There are several security context settings. See Configure a Security Context for a Pod or Container in the Kubernetes documentation.

The TimesTen Operator runs with a default security context and includes the Kubernetes default seccomp profile.

The following snippet shows the default security context for the TimesTen Operator.

```
securityContext:
  allowPrivilegeEscalation: false
  capabilities:
    drop:
    - ALL
  privileged: false
  readOnlyRootFilesystem: true
  runAsNonRoot: true
  seccompProfile:
    type: RuntimeDefault
```

About Readiness and Liveness Probes

The TimesTen Operator provides readiness and liveness probes so that Kubernetes can determine the health of the TimesTen Operator.

The TimesTen Operator exposes these probes to applications in the Kubernetes cluster by creating a Kubernetes Service called timesten-operator. Although we do not recommend it, you can choose to not expose these probes by setting the TimesTen Operator EXPOSE_PROBES environment variable to "0" or by modifying the TimesTen Operator helm charts.

For information about the TimesTen Operator environment variables, see TimesTen Kubernetes Operator Environment Variables. For information about helm charts, see Use Helm in Your TimesTen Kubernetes Operator Environment.

About Privileges

•

The TimesTen Operator needs different privileges depending on whether you install the TimesTen Operator at namespace-scoped or at cluster-scoped.

- When the TimesTen Operator runs in a namespace in your cluster at namespace-scope, it only needs privileges within the namespace where you installed it. The TimesTen Operator does not require privileges in any other namespace in the cluster.
- When the TimesTen Operator runs in the timesten-operator namespace in your cluster at cluster-scope, it needs privileges within the namespace it runs in as well as in every namespace in the cluster.



Kubernetes uses Roles and RoleBindings to specify privileges within a single namespace and uses ClusterRoles and ClusterRoleBindings to specify privileges across all namespaces. For more information about Kubernetes Roles, RoleBindings, ClusterRoles, and ClusterRoleBindings, see https://kubernetes.io/docs/reference/access-authn-authz/rbac/ in the Kubernetes documentation.

When the TimesTen Operator runs at namespace-scope, it requires Roles and RoleBindings to operate. Specifically, when you install the TimesTen Operator in a namespace at namespace-scope, a ServiceAccount called timesten-operator is created in this namespace and a RoleBinding is created and used to give the timesten-operator service account the necessary privileges. These privileges are defined in the timesten-operator Role.

When the TimesTen Operator runs at cluster-scope, it requires Roles, RoleBindings, ClusterRoles, and ClusterRoleBindings to operate. Since the TimesTen Operator needs privileges in every namespace and needs additional privileges just in its namespace, the following objects are provisioned:

- ClusterRole called timesten-operator: Defines the privileges that the TimesTen Operator requires in each namespace in the cluster.
- ClusterRoleBinding called timesten-operator: Gives the timesten-operator service account in the timesten-operator namespace the privileges in the timesten-operator cluster role.
- Namespace timesten-operator: Is the namespace in which the TimesTen Operator runs. This namespace is non-configurable. In this namespace, the following objects are created:
 - ServiceAccount called timesten-operator: The TimesTen Operator runs with the privileges of this service account.
 - Role called timesten-operator: Defines the privileges the TimesTen Operator requires in the timesten-operator namespace.
 - RoleBinding called timesten-operator: Gives the timesten-operator service account in the timesten-operator namespace the privileges in the timesten-operator role.

In summary, when you install the TimesTen Operator in a namespace in your cluster at namespace-scope, a ServiceAccount called timesten-operator is created in that namespace and a RoleBinding is used to give the service account the necessary privileges. In contrast, when you install the TimesTen Operator in your cluster at cluster-scope, the timesten-operator namespace is created. In addition, a ServiceAccount is created in this timesten-operator namespace and RoleBindings and ClusterRoleBindings are used to give the service account the necessary privileges. There are examples later in the chapter that illustrate the creation of these objects.

About Installing the TimesTen Operator

There are YAML manifest files and Helm charts for installing the TimesTen Operator. A TimesTen container image contains these files and charts. To find out how to obtain these files and charts, see Obtain TimesTen YAML Manifest Files and Helm Charts.

The YAML manifest files and Helm charts for installing the TimesTen Operator in a namespace in your Kubernetes cluster at namespace-scope are different than the files for installing the TimesTen Operator in the timesten-operator namespace in your Kubernetes cluster at cluster-scope. For information about these scopes, see About the TimesTen Operator.

Note:

You must install the TimesTen Operator in either one or more namespaces in your cluster at namespace-scope or in the timesten-operator namespace in your cluster at cluster-scope. You cannot install in both scopes.

To install the TimesTen Operator in a namespace in your Kubernetes cluster at namespacescope, you can use either YAML manifest files or a Helm chart:

- To install using YAML manifest files, use the following files:
 - service_account.yaml: Defines the RoleBinding, Role, and ServiceAccount objects that are necessary for the TimesTen Operator to run properly in a namespace in your cluster at namespace-scope. The service account requires a set of defined privileges and permissions, which are granted through a role. A role binding assigns a role to a service account. See About Privileges.
 - service_account_cluster.yaml: Defines additional privileges and permissions for the TimesTen Operator. Although not mandatory, we recommend that you install this file. It defines the privileges and permissions that are required if you want the TimesTen Operator to delete TimesTen Pods and PVCs when detecting a node failure. For more information about this feature, see How the TimesTen Kubernetes Operator Handles Node Failure.
 - operator.yaml: Defines the Deployment that is used to run the TimesTen Operator in a namespace in your cluster at namespace-scope. There are customizations for this Deployment. See About Customizations for a TimesTen Operator Deployment.
- To install using a Helm chart, use the ttoperator Helm chart. See Use Helm in Your TimesTen Kubernetes Operator Environment and The ttoperator Chart.

To install the TimesTen Operator in the timesten-operator namespace in your Kubernetes cluster at cluster-scope, you can use either YAML manifest files or a Helm chart:

- To install using YAML manifest files, use the following files:
 - cluster_config.yaml: Defines the timesten-operator namespace as well as the ServiceAccount, Role, RoleBinding, ClusterRole, and ClusterRoleBinding objects necessary to run the TimesTen Operator in the timesten-operator namespace at cluster-scope. For more information about these objects, see About Privileges.
 - cluster_operator.yaml: Defines the Deployment that is used to run the TimesTen Operator in the timesten-operator namespace in your cluster at cluster-scope. There are customizations for this Deployment. See About Customizations for a TimesTen Operator Deployment.
- To install using a Helm chart, use the ttclusteroperator Helm chart. See Use Helm in Your TimesTen Kubernetes Operator Environment and The ttclusteroperator Chart.

There are examples in this chapter that show you how to install the TimesTen Operator in a namespace in your Kubernetes cluster at namespace-scope and in the timesten-operator namespace in your Kubernetes cluster at cluster-scope using YAML manifest files. To install using Helm charts, see Use Helm in Your TimesTen Kubernetes Operator Environment.



Note:

Most of the examples in the book assume you have installed the TimesTen Operator in a namespace in your Kubernetes cluster at namespace-scope. If there are cases where the TimesTen Operator is installed in the timesten-operator namespace in your Kubernetes cluster at cluster-scope, the documentation points this out.

About Customizations for a TimesTen Operator Deployment

The operator.yaml YAML file defines the Deployment to run the TimesTen Operator in a namespace-scoped environment and the cluster_operator.yaml YAML file defines the Deployment to run the TimesTen Operator in a cluster-scoped environment. These files provide customizations that you can make for your specific environment. The customizations that you can make are the same for namespace-scoped and cluster-scoped environments.

The ttoperator and ttclusteroperator Helm charts define the TimesTen Operator deployments using Helm. If you are using Helm, see Use Helm in Your TimesTen Kubernetes Operator Environment.

The following information is specific to YAML manifest files:

To customize the TimesTen Operator, modify either the <code>operator.yaml</code> or <code>cluster_operator.yaml</code> file (modify <code>operator.yaml</code> for namespace-scoped environments or <code>cluster_operator.yaml</code> for cluster-scoped environments). The modifications are as follows:

- image: Container image to run.
- imagePullSecrets: Image pull secret that contains the credentials to access the image registry for the image.
- replicas: Number of TimesTen Operator Pods to run.
- affinity: Section that allows you to define whether the TimesTen Operator runs on amd64 or arm64 Nodes. These Node settings are specific to running in a multi-architecture Kubernetes environment. See About Deploying in a Multi-Architecture Kubernetes Cluster.
- Environment variables for the TimesTen Operator. Even though we recommend that you do not change the default settings, these are the environment variables that are customizable if you so choose:
 - CREATE_SERVICEMONITOR: See Expose Metrics from the TimesTen Kubernetes
 Operator.
 - EXPOSE METRICS: See Expose Metrics from the TimesTen Kubernetes Operator.
 - EXPOSE PROBES: See About Readiness and Liveness Probes.
 - METRICS SCHEME: See Expose Metrics from the TimesTen Kubernetes Operator.
 - TT MAX RECONCILES: See TimesTen Kubernetes Operator Environment Variables.

For more information about the TimesTen Operator environment variables, see TimesTen Kubernetes Operator Environment Variables.

Install the TimesTen Operator

Let's walk through some examples that show you how to install the TimesTen Operator in two namespaces in a Kubernetes cluster at namespace-scope and in the timesten-operator



namespace in a Kubernetes cluster at cluster-scope. These examples assume you are using YAML manifest files. If you are using Helm charts, see Use Helm in Your TimesTen Kubernetes Operator Environment.

- Install the TimesTen Operator at Namespace-Scope
- Install the TimesTen Operator at Cluster-Scope

Install the TimesTen Operator at Namespace-Scope

The examples in this section assume you have obtained the YAML manifest files. See Obtain TimesTen YAML Manifest Files and Helm Charts.

There is one example for installing the TimesTen Operator in one namespace in your cluster at namespace-scope. There is a second example for installing the TimesTen Operator into a second namespace in your cluster at namespace-scope. The examples illustrate that you must install the TimesTen Operator in each namespace. Once installed, the TimesTen Operator that you installed in namespace one services TimesTenClassic objects in namespace one while the TimesTen Operator that you installed in namespace two services TimesTenClassic objects in namespace two.

To complete the installation, perform the steps in the following sections:

- Before You Begin
- Install in Namespace One
- Install in Namespace Two
- Verify Installation

Before You Begin

Let's confirm our current namespace in the Kubernetes cluster. Next, let's create a new namespace and a new Kubernetes Secret in that namespace. The examples use these two namespaces.

- 1. Confirm the current namespace and review the Secrets in the namespace.
 - a. Confirm the current namespace.

kubectl config view | grep namespace

The output is similar to the following:

namespace: mynamespace

The current namespace is mynamespace.

b. Confirm the Secret in the namespace.

kubectl get secrets

NAME		
TYPE	DATA	AGE



```
sekret
kubernetes.io/dockerconfigjson 1 15d
...
```

The sekret Secret exists in the mynamespace namespace.

- 2. Create a new namespace and create a Kubernetes Secret in the namespace.
 - a. Create a new namespace.

kubectl create namespace mynamespace2

The output is similar to the following:

namespace/mynamespace2 created

b. Switch to the new namespace.

kubectl config set-context --current --namespace=mynamespace2

The output is similar to the following:

Context "default" modified.

c. Confirm the current namespace.

kubectl config view | grep namespace

The output is similar to the following:

namespace: mynamespace2

d. Create a Kubernetes Secret in this namespace.

```
kubectl create secret generic sekret --from-
file=.dockerconfigjson=$HOME/.docker/config.json --type=kubernetes.io/
dockerconfigjson
```

The output is the following:

secret/sekret created

3. Confirm the namespaces.

kubectl get namespaces

NAME	STATUS	AGE
mynamespace	Active	15d
mynamespace2	Active	49m



You confirmed the namespaces in your Kubernetes cluster.

Install in Namespace One

Let's install the TimesTen Operator in a namespace called mynamespace in your Kubernetes cluster at namespace-scoped.

1. Switch to namespace one (mynamespace, in this example).

kubectl config set-context --current --namespace=mynamespace

The output is similar to the following:

Context "default" modified.

 Change to the directory that contains the YAML manifest files. In this example, the kube_files/deploy contains the files.

cd kube files/deploy

Install the required service account, role, and role binding.

kubectl create -f service account.yaml

The output is similar to the following:

```
role.rbac.authorization.k8s.io/timesten-operator created
serviceaccount/timesten-operator created
rolebinding.rbac.authorization.k8s.io/timesten-operator created
```

 Make a copy of the service_account_cluster.yaml file for the first namespace (service account cluster n1.yaml, in this example).

cp service account cluster.yaml service account cluster n1.yaml

- 5. Install the service account cluster n1.yaml YAML file by doing the following:
 - a. (Optional): Display the contents of the service_account_cluster_n1.yaml file.

```
cat service_account_cluster_n1.yaml
```

```
# Copyright (c) 2025, Oracle and/or its affiliates.
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
   name: timesten-operator
   # If running multiple operators on the same cluster:
   #name: timesten-operator-<NAMESPACE>
rules:
   - apiGroups:
   - ""
   resources:
```



```
- nodes
 verbs:
  - get
  - list
  - watch
- apiGroups:
  _ ""
 resources:
  - persistentvolumeclaims
 verbs:
  - get
  - list
  - watch
  - delete
___
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
 name: timesten-operator
  # If running multiple operators on the same cluster:
  #name: timesten-operator-<NAMESPACE>
subjects:
- kind: ServiceAccount
  name: timesten-operator
  #namespace: <NAMESPACE>
roleRef:
  kind: ClusterRole
  name: timesten-operator
  # If running multiple operators on the same cluster:
  #name: timesten-operator-<NAMESPACE>
  apiGroup: rbac.authorization.k8s.io
```

b. Use a text editor to modify the service_account_cluster_n1.yaml file.

Make the following changes:

- Locate #namespace, remove #, and replace <NAMESPACE> with the name of your namespace (mynamespace, in this example).
- Locate the three occurrences of #name, remove #, and replace <NAMESPACE> with the name of your namespace (mynamespace, in this example).

```
vi service_account_cluster_n1.yaml
# Copyright (c) 2025, Oracle and/or its affiliates.
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
   name: timesten-operator
   # If running multiple operators on the same cluster:
   name: timesten-operator-mynamespace
rules:
   - apiGroups:
    - ""
   resources:
    - nodes
   verbs:
```

```
- get
  - list
  - watch
- apiGroups:
  _ ""
 resources:
  - persistentvolumeclaims
 verbs:
  - get
  - list
  - watch
  - delete
___
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
 name: timesten-operator
  # If running multiple operators on the same cluster:
 name: timesten-operator-mynamespace
subjects:
- kind: ServiceAccount
  name: timesten-operator
 namespace: mynamespace
roleRef:
 kind: ClusterRole
  name: timesten-operator
  # If running multiple operators on the same cluster:
  name: timesten-operator-mynamespace
  apiGroup: rbac.authorization.k8s.io
```

- c. Save and close the service account cluster n1.yaml file.
- d. Install the service_account_cluster_n1.yaml file.

kubectl create -f service account cluster n1.yaml

The output is similar to the following:

clusterrole.rbac.authorization.k8s.io/timesten-operator created clusterrolebinding.rbac.authorization.k8s.io/timesten-operator created

- 6. Modify the operator.yaml file by doing the following:
 - a. Use a text editor to modify the operator.yaml file.

Replace the following:

- image: **Replace** container-registry.oracle.com/timesten/timesten:latest with the name of your image. In this example, the name of the image is container-registry.oracle.com/timesten/timesten:22.1.1.34.0.
- imagePullSecrets: Replace sekret with the name of your image pull secret. In this example, the name of the image pull secret is sekret.

• If you are running in a multi-architecture environment, modify the affinity section, and specify either amd64 or arm64. This example assumes you are running in a multi-architecture environment and sets nodes to amd64.

```
vi operator.yaml
# Copyright (c) 2019 - 2025, Oracle and/or its affiliates.
apiVersion: apps/v1
kind: Deployment
metadata:
 name: timesten-operator
spec:
 replicas: 1
. . .
    spec:
      serviceAccountName: timesten-operator
      imagePullSecrets:
      - name: sekret
      containers:
        - name: timesten-operator
          image: container-registry.oracle.com/timesten/
timesten:22.1.1.34.0
. . .
# An example affinity definition; this pod will only be assigned to a
node
# running on amd64 (the default)
#
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
              - matchExpressions:
                - key: "kubernetes.io/arch"
                  operator: In
                  values: ["amd64"]
```

- **b.** Save and close the operator.yaml file.
- 7. Install the TimesTen Operator.

kubectl create -f operator.yaml

The output is the following:

deployment.apps/timesten-operator created

8. Verify the TimesTen Operator is running.

kubectl get pods

NAME	READY	STATUS	RESTARTS	AGE
timesten-operator-577f7fbc6f-tbr7m	1/1	Running	0	61s



Congratulations! You installed the TimesTen Operator into a namespace (called mynamespace) in your Kubernetes cluster at namespace-scope.

Install in Namespace Two

Let's install the TimesTen Operator in a namespace called mynamespace2 in your Kubernetes cluster at namespace-scoped.

1. Confirm the namespaces.

kubectl get namespaces

The output is similar to the following:

NAME	STATUS	AGE
mynamespace	Active	16d
••••		
mynamespace2	Active	19h

2. Switch to namespace two (mynamespace2, in this example).

kubectl config set-context --current --namespace=mynamespace2

The output is similar to the following:

Context "default" modified.

3. Change to the directory that contains the YAML manifest files. In this example, the kube_files/deploy contains the files.

cd kube files/deploy

Install the required service account, role, and role binding.

kubectl create -f service_account.yaml

The output is similar to the following:

```
role.rbac.authorization.k8s.io/timesten-operator created
serviceaccount/timesten-operator created
rolebinding.rbac.authorization.k8s.io/timesten-operator created
```

5. Make a copy of the service_account_cluster.yaml file for the second namespace (service account cluster n2.yaml, in this example).

cp service_account_cluster.yaml service_account_cluster_n2.yaml

- 6. Install the service account cluster n2.yaml YAML file by doing the following:
 - a. (Optional): Display the contents of the service account cluster n2.yaml file.

cat service_account_cluster_n2.yaml



The output is similar to the following:

```
# Copyright (c) 2025, Oracle and/or its affiliates.
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
 name: timesten-operator
  # If running multiple operators on the same cluster:
  #name: timesten-operator-<NAMESPACE>
rules:
- apiGroups:
  _ ""
 resources:
  - nodes
 verbs:
  - get
  - list
  - watch
- apiGroups:
  _ ""
 resources:
  - persistentvolumeclaims
 verbs:
  - get
  - list
  - watch
  - delete
___
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
 name: timesten-operator
  # If running multiple operators on the same cluster:
  #name: timesten-operator-<NAMESPACE>
subjects:
- kind: ServiceAccount
  name: timesten-operator
 #namespace: <NAMESPACE>
roleRef:
  kind: ClusterRole
  name: timesten-operator
  # If running multiple operators on the same cluster:
  #name: timesten-operator-<NAMESPACE>
  apiGroup: rbac.authorization.k8s.io
```

b. Use a text editor to modify the service_account_cluster_n2.yaml file.

Make the following changes:

- Locate #namespace, remove #, and replace <NAMESPACE> with the name of your namespace (mynamespace2, in this example).
- Locate the three occurrences of #name, remove #, and replace <NAMESPACE> with the name of your namespace (mynamespace2, in this example).

vi service account cluster n2.yaml



```
# Copyright (c) 2025, Oracle and/or its affiliates.
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: timesten-operator
  # If running multiple operators on the same cluster:
 name: timesten-operator-mynamespace2
rules:
- apiGroups:
  _ ""
  resources:
  - nodes
 verbs:
  - get
  - list
  - watch
- apiGroups:
  _ ""
  resources:
  - persistentvolumeclaims
 verbs:
 - get
  - list
  - watch
  - delete
___
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
 name: timesten-operator
  # If running multiple operators on the same cluster:
 name: timesten-operator-mynamespace2
subjects:
- kind: ServiceAccount
  name: timesten-operator
 namespace: mynamespace2
roleRef:
  kind: ClusterRole
 name: timesten-operator
  # If running multiple operators on the same cluster:
 name: timesten-operator-mynamespace2
  apiGroup: rbac.authorization.k8s.io
```

- c. Save and close the service_account_cluster_n2.yaml file.
- d. Install the service_account_cluster_n2.yaml file.

kubectl create -f service account cluster n2.yaml

```
clusterrole.rbac.authorization.k8s.io/timesten-operator created clusterrolebinding.rbac.authorization.k8s.io/timesten-operator created
```



- 7. Modify the operator.yaml file. In this example, the modifications are the same as the operator.yaml file in namespace one. However, these modifications do not need to be the same as the modifications for namespace one. For example, you can use a different container image:
 - a. Use a text editor to modify the operator.yaml file.

Replace the following:

- image: Replace container-registry.oracle.com/timesten/timesten:latest with the name of your image. In this example, the name of the image is container-registry.oracle.com/timesten/timesten:22.1.1.34.0.
- imagePullSecrets: Replace sekret with the name of your image pull secret. In this example, the name of the image pull secret is sekret.
- If you are running in a multi-architecture environment, modify the affinity section, and specify either amd64 or arm64. This example assumes you are running in a multi-architecture environment and sets nodes to amd64.

```
vi operator.yaml
```

```
# Copyright (c) 2019 - 2025, Oracle and/or its affiliates.
apiVersion: apps/v1
kind: Deployment
metadata:
  name: timesten-operator
spec:
 replicas: 1
. . .
    spec:
      serviceAccountName: timesten-operator
     imagePullSecrets:
      - name: sekret
      containers:
        - name: timesten-operator
          image: container-registry.oracle.com/timesten/
timesten:22.1.1.34.0
. . .
# An example affinity definition; this pod will only be assigned to a
node
# running on amd64 (the default)
#
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
              - matchExpressions:
                - key: "kubernetes.io/arch"
                  operator: In
                  values: ["amd64"]
```

- **b.** Save and close the operator.yaml file.
- 8. Install the TimesTen Operator.

kubectl create -f operator.yaml



The output is the following:

deployment.apps/timesten-operator created

9. Verify the TimesTen Operator is running.

kubectl get pods

The output is similar to the following:

NAME	READY	STATUS	RESTARTS	AGE
timesten-operator-577f7fbc6f-h8hj8	1/1	Running	0	61s

Congratulations! You installed the TimesTen Operator into a namespace (called mynamespace2) in your Kubernetes cluster at namespace-scope.

Verify Installation

Let's verify there is one TimesTen Operator running in mynamespace and a second TimesTen Operator running in mynamespace2 in a Kubernetes cluster in a namespace-scoped environment. The TimesTen Operator in mynamespace services TimesTenClassic objects in mynamespace while the TimesTen Operator in mynamespace2 services TimesTenClassic objects in mynamespace2.

1. Review the namespaces.

kubectl get namespaces

The output is similar to the following:

NAME	STATUS	AGE
mynamespace	Active	16d
mynamespace2	Active	19h

2. Confirm there is a TimesTen Operator running in the mynamespace namespace.

kubectl get pods -n mynamespace

The output is similar to the following:

NAME			READY
STATUS	RESTARTS	AGE	
•••			
timesten-	operator-57	7f7fbc6f-tbr7m	1/1
Running	0	53m	

3. Confirm there is a TimesTen Operator running in the mynamespace2 namespace.

kubectl get pods -n mynamespace2



The output is similar to the following:

NAME	READY	STATUS	RESTARTS	AGE
timesten-operator-577f7fbc6f-h8hj8	1/1	Running	0	8m27s

Congratulations! You successfully installed the TimesTen Operator in two namespaces in your Kubernetes cluster in a namespace-scoped environment. The TimesTen Operator in the mynamespace namespace services TimesTenClassic objects in the mynamespace namespace while the TimesTen Operator in the mynamespace2 namespace services TimesTenClassic objects in the mynamespace namespace2 namespace services TimesTenClassic objects in the mynamespace namespace.

You are now ready to create TimesTen databases in either namespace. See Create TimesTen Classic Databases.

Install the TimesTen Operator at Cluster-Scope

The examples in this section assume you have obtained the YAML manifest files. See Obtain TimesTen YAML Manifest Files and Helm Charts.

Let's install the TimesTen Operator at cluster-scope. As part of the installation, the timestenoperator namespace is created. The TimesTen Operator runs in this namespace in your Kubernetes cluster at cluster-scope. Once installed, this TimesTen Operator services TimesTenClassic objects in all namespaces in your cluster.

1. Change to the directory that contains the YAML manifest files. In this example, the kube_files/deploy contains the files.

cd kube files/deploy

2. Create the necessary privileges to run the TimesTen Operator.

```
kubectl create -f cluster config.yaml
```

The output is similar to the following

```
namespace/timesten-operator created
clusterrole.rbac.authorization.k8s.io/timesten-operator created
serviceaccount/timesten-operator created
clusterrolebinding.rbac.authorization.k8s.io/timesten-operator created
```

Installing this file results in the creation of the timesten-operator namespace.

3. Create a Kubernetes Secret in the timesten-operator namespace. This Kubernetes Secret is used as an image pull secret and is required so that the TimesTen Operator can pull a TimesTen container image from an image registry. This example creates the sekret Secret.

```
kubectl create secret generic sekret --from-
file=.dockerconfigjson=$HOME/.docker/config.json --type=kubernetes.io/
dockerconfigjson -n timesten-operator
```

The output is the following:

secret/sekret created



For more information about creating Kubernetes Secrets, see Pulling Images from Registry during Deployment in the Kubernetes documentation.

- 4. Modify the cluster operator.yaml file by doing the following:
 - a. Use a text editor to modify the cluster operator.yaml file.

Replace the following:

- image: Replace container-registry.oracle.com/timesten/timesten:latest with the name of your image. In this example, the name of the image is container-registry.oracle.com/timesten/timesten:22.1.1.34.0.
- imagePullSecrets: Replace sekret with the name of your image pull secret you created in the prior step. In this example, the name of the image pull secret is sekret.
- If you are running in a multi-architecture environment, modify the affinity section, and specify either amd64 or arm64. This example assumes you are running in a multi-architecture environment and sets nodes to amd64.

```
vi cluster operator.yaml
# Copyright (c) 2019 - 2025, Oracle and/or its affiliates.
apiVersion: apps/v1
kind: Deployment
metadata:
 name: timesten-operator
spec:
 replicas: 1
. . .
    spec:
     serviceAccountName: timesten-operator
     imagePullSecrets:
      - name: sekret
      containers:
        - name: timesten-operator
          image: container-registry.oracle.com/timesten/
timesten:22.1.1.34.0
. . .
# An example affinity definition; this pod will only be assigned to a
node
# running on amd64 (the default)
#
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
              - matchExpressions:
                - key: "kubernetes.io/arch"
                  operator: In
                  values: ["amd64"]
```

- b. Save and close the cluster operator.yaml file.
- 5. Install the TimesTen Operator.

kubectl create -f cluster_operator.yaml

The output is the following:

deployment.apps/timesten-operator created

6. Verify the TimesTen Operator is running in the timesten-operator namespace .

kubectl get pods -n timesten-operator

The output is similar to the following:

NAME	READY	STATUS	RESTARTS	AGE
timesten-operator-6bf76dd84b-c8j59	1/1	Running	0	51s

Congratulations! You installed the TimesTen Operator. It is running in the timesten-operator namespace in your Kubernetes cluster at cluster-scope. The TimesTen Operator services TimesTenClassic objects in all namespaces in your Kubernetes cluster.

You are now ready to create TimesTen databases in your namespaces. See Create TimesTen Classic Databases.



5 Use Configuration Metadata

This chapter gives an overview of the configuration metadata that is supported in the TimesTen Operator. It also discusses the Kubernetes facilities that you can use to get the configuration metadata into your TimesTen containers. The chapter then discusses additional configuration options. There are examples throughout.

Topics:

- Overview of Configuration Metadata and Kubernetes Facilities
- List of Configuration Metadata
- About Configuration Metadata Details
- Populate the /ttconfig Directory
- Additional Configuration Options

Overview of Configuration Metadata and Kubernetes Facilities

Configuration metadata lets you define the attributes of your TimesTen database and how that database is to interact with other applications and components. The TimesTen Operator supports several metadata files that contain the configuration metadata. Each metadata file has a specific name. You use a text editor to create the metadata file with the specific name and then add the appropriate metadata to it. For example, the TimesTen Operator supports the db.ini metadata file. You use your editor to create the db.ini file and in it you define attributes for you database.

Kubernetes supports various facilities that places the metadata files into the /ttconfig directory of the TimesTen containers. See Populate the /ttconfig Directory.

List of Configuration Metadata

Table 5-1 lists the metadata files that are supported by the TimesTen Operator. The table provides a description for each of the metadata files and indicates if the metadata file is supported in TimesTen Classic, in TimesTen Scaleout, or in both.

Name	Description	TimesTen Classic support	TimesTen Scaleout support
adminUser	Defines an initial user in the database and assigns this user ADMIN privileges.	Yes	Yes
	Optional.		
cachegroups.sql	Defines the cache groups in the database. This file is specific to TimesTen Cache.	Yes	No
	Required if using TimesTen Cache.		

Table 5-1 TimesTen Operator metadata files



Name	Description	TimesTen Classic support	TimesTen Scaleout support
cacheUser	Defines the cache administration user in the database. This file is specific to TimesTen Cache.	Yes	No
	Required if using TimesTen Cache.		
csWallet	Defines the credentials that are used for Transport Layer Security (TLS) encryption of client/server communications.	Yes	Yes
	Required if using TLS.		
db.ini	Defines the connection attributes of the database. See List of Connection Attributes in the Oracle TimesTen In-Memory Database Reference.	Yes	Yes
	Required if using TimesTen Cache. Otherwise, optional.		
epilog.sql	Performs operations after the replication scheme is created.	Yes	No
	Optional.		
replicationWallet	Defines the credentials that are used for Transport Layer Security (TLS) encryption of replication traffic between the TimesTen databases.	Yes	No
	Required if using TLS.		
schema.sql	Defines an initial schema for the database.	Yes	Yes
	Optional.		
sqlnet.ora	Defines how client applications communicate with an Oracle database. This file is specific to TimesTen Cache.	Yes	No
	Optional.		
testUser	Defines a test user in the database. This user is used for testing TimesTen using Helm. The user is assigned CONNECT privileges.	Yes	No
	Optional.		
tnsnames.ora	Defines the Oracle Database service that TimesTen Cache uses to connect to Oracle Database.	Yes	No
	Required if using TimesTen Cache.		
*.connect	Defines one or more direct connectables for the database in TimesTen Scaleout.	No	Yes
	Optional.		
*.csconnect	Defines one or more client/server connectables for the database in TimesTen Scaleout.	No	Yes
	Optional.		

 Table 5-1 (Cont.) TimesTen Operator metadata files

About Configuration Metadata Details

Metadata files let you specify the attributes and the metadata for your TimesTen database. After you create these files, and you choose a facility to get these files in your TimesTen containers, TimesTen accesses them to determine the attributes and the metadata that is specific to your database.

adminUser

The adminUser file creates an initial user with ADMIN privileges in the TimesTen database. If you provide this file, this user is created after the database is created. This file must contain one line of the form:

```
user/password
```

cachegroups.sql

The cachegroups.sql file contains the create cache group definitions and the cache group operations for your database. You can specify the following cache group definitions and cache operations in this file:

- (Required): CREATE CACHE GROUP statements to create TimesTen cache groups
- (Optional): LOAD CACHE GROUP statements to load data from the Oracle database into your cache groups
- (Optional): ttOptUpdateStats or ttOptEstimateStats TimesTen built-in procedures to update statistics on the cache tables

The <code>cachegroups.sql</code> file is required if you are using TimesTen Cache in your TimesTenClassic deployment. This requirement ensures cache groups are created before replication is configured. Note: The instance administrator uses the <code>ttlsql</code> utility to run the <code>cachegroups.sql</code> file.

See:

- CREATE CACHE GROUP and LOAD CACHE GROUP in the Oracle TimesTen In-Memory
 Database SQL Reference
- Cache Group Types in the Oracle TimesTen In-Memory Database Cache Guide
- ttOptUpdateStats and ttOptEstimateStats in the Oracle TimesTen In-Memory Database Reference

Here is an example of a <code>cachegroups.sql</code> file. The file defines two cache groups and loads data into one cache group.

```
CREATE DYNAMIC ASYNCHRONOUS WRITETHROUGH CACHE GROUP writecache
FROM oratt.writetab (
    pk NUMBER NOT NULL PRIMARY KEY,
    attr VARCHAR2(40)
);
CREATE READONLY CACHE GROUP readcache
AUTOREFRESH
    INTERVAL 5 SECONDS
FROM oratt.readtab (
    keyval NUMBER NOT NULL PRIMARY KEY,
    str VARCHAR2(32)
```



);

LOAD CACHE GROUP readcache COMMIT EVERY 256 ROWS;

cacheUser

The cacheUser file lets you create the TimesTen cache manager user. This user must have the same name as the cache administration user in the Oracle database, and must already exist in the Oracle database. See Create the Oracle Database Users and Default Tablespace in the Oracle TimesTen In-Memory Database Cache Guide.

This file must contain one line of the form,

cacheUser/ttPassword/oraPassword

where cacheUser is the TimesTen cache manager user, ttPassword is the TimesTen password for the TimesTen cacheUser user, and oraPassword is the Oracle database password you specified when you created the cacheUser user in the Oracle database.

For example, assume you have created the cacheuser2 cache administration user in the Oracle Database with password oraclepwd. Assume you designate this cacheuser2 user as the TimesTen cache manager user with a TimesTen password of ttpwd. In this example, the cacheUser metadata file contains this one line:

cacheuser2/ttpwd/oraclepwd

The TimesTen Operator creates the cacheuser2 user with the ttpwd in the TimesTen database. This cacheuser2 user then serves as the cache manager user in your TimesTen database. You do not need to create this TimesTen user. The Operator does it for you.

See Create the TimesTen Users in the Oracle TimesTen In-Memory Database Cache Guide.

The Operator grants privileges to the TimesTen cacheUser user (cacheuser2, in this example) that are appropriate for this user's role as the cache manager. These privileges are:

- CREATE SESSION
- CACHE MANAGER
- CREATE ANY TABLE
- LOAD ANY CACHE GROUP
- REFRESH ANY CACHE GROUP
- FLUSH ANY CACHE GROUP
- DROP ANY CACHE GROUP
- ALTER ANY CACHE GROUP
- UNLOAD ANY CACHE GROUP
- SELECT ANY TABLE
- INSERT ANY TABLE
- UPDATE ANY TABLE
- DELETE ANY TABLE



csWallet

In a TimesTen Client/Server environment, data is transmitted between your client applications and your TimesTen database unencrypted by default. However, you can configure TLS for Client/Server to ensure secure network communication between TimesTen clients and servers. To encrypt Client/Server traffic, specify the /ttconfig/csWallet file. This file contains the Oracle wallet for the server, which contains the credentials that are used for configuring TLS encryption between your TimesTen database and your Client/Server applications. The file will be available in the containers of your TimesTen databases in the directory /tt/home/ timesten/csWallet. You can reference this directory in your db.ini file (by specifying the wallet connection attribute). See Create TLS Certificates for Replication and Client/Server and Create TLS Certificates for Replication and Client/Server.

The client wallet must also be available to your client applications. See Create TLS Certificates for Replication and Client/Server and Configure TLS for Client/Server.

db.ini

The db.ini file contains the TimesTen DSN definition for your database.

In TimesTen Classic, the db.ini file contains the connection attributes for your database. This file is used to to generate the sys.odbc.ini file for the instances. You can specify data store attributes, first connection attributes, and general connection attributes in the db.ini file. The name of the DSN is the name of the TimesTenClassic object. For example, if your TimesTenClassic object is called sample, the name of your DSN is sample.

In TimesTenScaleout, the db.ini file contains the connection attributes for each element of your database in the grid. The database definition file (dbDef) and its contents are used to create a TimesTen Scaleout database definition. The name of the database definition is the name of the TimesTenScaleout object. For example, if the name of your TimesTenScaleout object is sample, the name of the database definition is sample. The TimesTen Scaleout database definition. For information about creating a database and creating a database definition in TimesTen Scaleout, see Creating a Database in the *Oracle TimesTen In-Memory Database Scaleout User's Guide*.

If you are using TimesTen Cache in your TimesTenClassic deployment, you must specify the OracleNetServiceName and the DatabaseCharacterSet connection attributes in the db.ini file. The DatabaseCharacterSet value must match the value of the database character set in the Oracle Database.

Do not specify the the DataStore or the LogDir connection attributes in the db.ini file. The Operator sets these attributes, placing the database files in Kubernetes Persistent Volumes.

See List of Connection Attributes in the *Oracle TimesTen In-Memory Database Reference* for information on the TimesTen connection attributes.

Note:

If the /ttconfig/db.ini file is not present in a TimesTen container, TimesTen creates a default sys.odbc.ini file. For this default sys.odbc.ini, the connection attributes are: Permsize=200 and DatabaseCharacterSet=AL32UTF8.



This example shows a sample db.ini file that contains various connection attributes for TimesTen databases in TimesTen Classic or TimesTen Scaleout.

PermSize=500 LogFileSize=1024 LogBufMB=1024 DatabaseCharacterSet=AL32UTF8

Here is an example that shows a sample db.ini file that contains the OracleNetServiceName for TimesTen databases that use TimesTen Cache in TimesTen Classic.

```
PermSize=500
LogFileSize=1024
LogBufMB=1024
DatabaseCharacterSet=AL32UTF8
OracleNetServiceName=OraCache
```

epilog.sql

In TimesTen Classic, the epilog.sql file includes operations that occur after the replication scheme has been created and the replication agent has been started. For example, if you want to create replicated bookmarks in XLA, you can include the ttXlaBookmarkCreate TimesTen built-in procedure in this file.

The Operator instructs the instance administrator to run the epilog.sql file using the ttIsql utility.

Here is an example of an epilog.sql file. The example calls the ttXlaBookmarkCreate TimesTen built-in procedure to create XLA bookmarks.

call ttXlaBookmarkCreate('mybookmark',0x01);

For information about replicated bookmarks, see the ttXlaBookmarkCreate TimesTen built-in procedure in the Oracle TimesTen In-Memory Database Reference.

replicationWallet

In TimesTen Classic, TimesTen replication transmits data between your TimesTen databases unencrypted by default. However, you can configure TLS for replication to ensure secure network communication between your replicated TimesTen databases. To do this, specify the / ttconfig/replicationWallet file. This file contains an Oracle wallet, which contains the credentials that are used by TimesTen replication for configuring TLS encryption between your active standby pair of TimesTen databases. See Create TLS Certificates for Replication and Client/Server and Configure TLS for Replication.

If you specify this file, you must include the <code>replicationCipherSuite</code> field and optionally include the <code>replicationSSLMandatory</code> field in your TimesTenClassic object definition. See the <code>replicationCipherSuite</code> entry and the <code>replicationSSLMandatory</code> entry in <code>TimesTenClassicSpecSpec</code> and Configure TLS for Replication.

schema.sql

The TimesTen Operator can automatically initialize your database with schema objects, such as users, tables, and sequences. To have the Operator do this, create the schema.sql file.



The Operator directs the instance administrator to use the ttIsql utility to run the schema.sql file immediately after the database is created. This operation occurs before the Operator configures replication or cache in your TimesTen database.

In TimesTen Cache, one or more cache table users own the cache tables. If this cache table user is not the cache manager user, then you must specify the schema.sql file and in it you must include the schema user and assign the appropriate privileges to this schema user. For example, if the oratt schema user was created in the Oracle database, and this user is not the TimesTen cache manager user, you must create the TimesTen oratt user in this file. See Create the Oracle Database Users.

Do not include cache definitions in this file. Instead, use the cachegroups.sql metadata file. See cachegroups.sql.

sqlnet.ora

The Oracle Database sqlnet.ora file defines the options for how client applications communicate with the Oracle Database. To use TimesTen Cache or to use tools like ttLoadFromOracle, define a sqlnet.ora file. This file describes how applications, including TimesTen, can connect to your Oracle database. Note: If you define a sqlnet.ora file, you must define a tnsnames.ora file. See tnsnames.ora.

This is an example of a sqlnet.ora file:

```
NAME.DIRECTORY_PATH= {TNSNAMES, EZCONNECT, HOSTNAME}
SQLNET.EXPIRE_TIME = 10
SSL VERSION = 1.2
```

testUser

The testUser file defines a test user. This user is used for testing TimesTen with the helm test command. The user is granted CONNECT privileges. The test connects to a TimesTen database as the test user. The test verifies your TimesTen Classic databases are up and running.

The testUser file must contain one line of the form:

testuser/testuserpassword

where testuser is the name of your test user and testuserpassword is the password for this test user.

For more information about using Helm and using the helm test command to test TimesTen, see Use Helm in Your TimesTen Kubernetes Operator Environment and Test TimesTen for a Replicated Configuration.

tnsnames.ora

The Oracle Database tnsnames.ora file defines Oracle Net Services to which applications connect. You need to use tnsnames.ora (and perhaps a sqlnet.ora file, described in sqlnet.ora) if you are using:

- TimesTen Cache
- SQL APIs, such as Pro*C, OCI, or ODPI-C



• The ttLoadFromOracle feature

For information about the ttLoadFromOracle TimesTen built-in procedure, see ttLoadFromOracle in the Oracle TimesTen In-Memory Database Reference.

Here is an example of a tnsnames.ora file:

```
OraTest =
 (DESCRIPTION =
  (ADDRESS = (PROTOCOL = TCP)(HOST = database.mynamespace.svc.cluster.local)
  (PORT = 1521))
 (CONNECT_DATA =
  (SERVER = DEDICATED)
  (SERVICE_NAME = OraTest.my.sample.com)))
OraCache =
 (DESCRIPTION =
  (ADDRESS = (PROTOCOL = TCP)(HOST = database.mynamespace.svc.cluster.local)
  (PORT = 1521))
  (CONNECT_DATA =
   (SERVER = DEDICATED)
   (SERVICE NAME = OraCache.my.sample.com)))
```

*.connect

Files with names that end with the .connect extension define one or more direct connectables for direct mode access to a database in TimesTen Scaleout. You can create as many direct connectables as you like. A direct connectable specifies a set of general connection attribute settings for the database. You can choose any name for the direct connectable as long as it is a valid DSN name. The TimesTen Operator creates one direct connectable for each direct connectable file you create. The .connect extension denotes a direct connectable. Ensure the .connect extension is in lowercase.

The following example creates one direct connectable. The name of the file is sample.connect. The TimesTen Operator creates the sample direct connectable based on the contents of the sample.connect file.

ConnectionCharacterSet=AL32UTF8

For more information about direct connectables, see Connectable Operations in the Oracle TimesTen In-Memory Database Reference and Connecting to a Database in the Oracle TimesTen In-Memory Database Scaleout User's Guide.

*.csconnect

Files with names that end with the .csconnect extension define one or more client/server connectables for client/server access to a database in TimesTen Scaleout. You can create as many client/server connectables as you like. A client/server connectable specifies a set of general connection attribute settings for the database. You can choose any name for the client/server connectable as long as it is a valid DSN name. The TimesTen Operator creates one client/server connectable for each client/server connectable file you create. The .csconnect extension denotes a client/server connectable. Ensure the .csconnect extension is in lowercase.



The following example creates one client/server connectable. The name of the file is samplecs.csconnect. The TimesTen Operator creates the samplecs client/server connectable based on the contents of the samplecs.csconnect file.

```
ConnectionCharacterSet=AL32UTF8
TTC_Timeout=30
```

For more information about client/server connectables, see Connectable Operations in the *Oracle TimesTen In-Memory Database Reference* and Connecting to a Database in the *Oracle TimesTen In-Memory Database Scaleout User's Guide*.

Populate the /ttconfig Directory

You can use different methods to ensure metadata files are placed in the /ttconfig directory of TimesTen containers. There is no requirement as to which method to use. Kubernetes provides such facilities as ConfigMaps, Secrets, and init containers for you to consider.

- Using ConfigMaps and Secrets
- Using an init container

Using ConfigMaps and Secrets

You can use one or more ConfigMaps and one or more Secrets to incorporate metadata files into the TimesTen containers. This lets you specify different TimesTen metadata for different deployments. In addition, you can use Secrets for metadata that contains sensitive data, like passwords and certificates.

The use of a ConfigMap to populate the metadata into Pods is a standard Kubernetes technique. One benefit is that you can modify the ConfigMap after it is created, which results in the immediate update of the files that are in the Pod.

Note:

TimesTen may not immediately notice and act on the changed content of the files.

When you use ConfigMaps and Secrets to hold your metadata and then reference them in the TimesTenClassic object definition, the TimesTen Operator creates a Projected Volume called tt-config. This tt-config volume contains the contents of all the ConfigMaps and all the Secrets specified in the dbConfigMap and the dbSecret fields of your TimesTenClassic or your TimesTenScaleout object. This volume is mounted as /ttconfig in the TimesTen Containers.
Note:

You can specify one or more ConfigMaps and/or Secrets in your TimesTenClassic or TimesTenScaleout object using the dbConfigMap and dbSecret datum. The result is that these ConfigMaps and/or Secrets are mounted read-only at /ttconfig. Since such a mount is read-only, you cannot write into it from an init container. Alternatively, you can use an emptydir volume and use an init container to write files into it. However, you cannot combine ConfigMaps and Secrets with an init container. For information about using an init container, see Using an init container.

To use ConfigMaps and Secrets, follow this process:

- Decide what facilities will contain what metadata files. For example, you can use one ConfigMap for all the metadata files. Or, for example, you can use one ConfigMap for the db.ini metadata file and one Secret for the adminUser and the schema.sql metadata files. There is no specific requirement.
- Create the directory (or directories) that will contain the metadata files.
- Use the kubectl create command to create the ConfigMap and the Secrets in the Kubernetes cluster.
- Include the ConfigMaps and Secrets in your TimesTenClassic or your TimesTenScaleout object definition.

The following examples illustrate how to use ConfigMaps and Secrets for a TimesTenClassic or a TimesTenScaleout object.

- Example Using One ConfigMap
- Example Using One ConfigMap and One Secret
- Example Using One ConfigMap for a TimesTenScaleout Object

Example Using One ConfigMap

This example uses one ConfigMap (called sample) for the db.ini, the adminUser, and the schema.sql metadata files.

You can use this ConfigMap for a TimesTenClassic or a TimesTenScaleout object.

1. On your development host, from the directory of your choice, create an empty subdirectory for the metadata files. This example creates the cm_sample subdirectory. (The cm_sample directory is used in the remainder of this example to denote this directory.)

```
mkdir -p cm_sample
```

2. Change to the ConfigMap directory.

cd cm_sample

3. Create the db.ini file. In this db.ini file, define the PermSize and DatabaseCharacterSet connection attributes.

vi db.ini

```
PermSize=200
DatabaseCharacterSet=AL32UTF8
ConnectionCharacterSet=AL32UTF8
```



4. Create the adminUser file. In this adminUser file, create the sampleuser user with the samplepw password.

vi adminUser

sampleuser/samplepw

5. Create the schema.sql file. In this schema.sql file, define the s sequence and the emp table for the sampleuser user. The Operator automatically initializes your database with these object definitions.

```
vi schema.sql
create sequence sampleuser.s;
create table sampleuser.emp (
   id number not null primary key,
   name char(32)
);
```

6. Create the ConfigMap. The files in the cm_sample directory are included in the ConfigMap. These files are later available in the TimesTen containers.

In this example:

- The name of the ConfigMap is sample. Replace sample with a name of your choosing.
- This example uses cm_sample as the directory where the files that will be copied into the ConfigMap reside. If you use a different directory, replace cm_sample with the name of your directory.

kubectl create configmap sample --from-file=cm_sample

The output is the following:

configmap/sample created

7. Verify the contents of the ConfigMap.

kubectl describe configmap sample

The output is the following:

Name:	sample		
Namespace:	mynamespace		
Labels:	<none></none>		
Annotations:	<none></none>		
Data ==== adminUser: sampleuser/sam	nplepw		
db.ini:			
PermSize=200 DatabaseCharacterSet=AL32UTF8 ConnectionCharacterSet=AL32UTF8			
schema.sql:			



```
create sequence sampleuser.s;
create table sampleuser.emp (
  id number not null primary key,
  name char(32)
);
```

- Events: <none>
- 8. Include the ConfigMap in the object definition. In the dbConfigMap field, specify the name of the your ConfigMap (sample, in this example).

Note this example uses a storageSize of 250Gi (suitable for a production environment). For demonstration purposes, a storageSize of 50Gi is adequate.

This is an example of using the ConfigMap for a TimesTenClassic object.

```
apiVersion: timesten.oracle.com/v4
kind: TimesTenClassic
metadata:
   name: sample
spec:
   ttspec:
    storageClassName: oci-bv
    storageSize: 250Gi
   image: container-registry.oracle.com/timesten/timesten:22.1.1.34.0
   imagePullSecret: sekret
   dbConfigMap:
    - sample
```

This is an example of using the ConfigMap for a TimesTenScaleout object.

```
apiVersion: timesten.oracle.com/v4
kind: TimesTenScaleout
metadata:
   name: sample
spec:
   ttspec:
    storageClassName: oci-bv
    storageSize: 250Gi
   image: container-registry.oracle.com/timesten/timesten:22.1.1.34.0
   imagePullSecret: sekret
   k: 2
   nReplicaSets: 3
   nZookeeper: 3
   dbConfigMap:
   - sample
```

The sample ConfigMap holds the metadata files. The tt-config volume contains the contents of the sample ConfigMap.

Example Using One ConfigMap and One Secret

This example uses one ConfigMap (called myconfig) for the db.ini metadata file and one Secret (called mysecret) for the adminUser and the schema.sql metadata files.

You can use this ConfigMap and Secret for a TimesTenClassic or a TimesTenScaleout object.

1. On your development host, from the directory of your choice:



- Create one empty subdirectory for the ConfigMap. This example creates the cm_myconfig subdirectory. (The cm_myconfig directory is used in the remainder of this example to denote this directory.) This directory will contain the db.ini metadata file.
- Create a second empty subdirectory for the Secret. This example creates the secret_mysecret subdirectory. (The secret_mysecret directory is used in the remainder of this example to denote this directory.) This directory will contain the adminUser and the schema.sql metadata files.

```
mkdir -p cm_myconfig
mkdir -p secret_mysecret
```

2. Change to the ConfigMap directory.

```
cd cm_myconfig
```

3. Create the db.ini file in this ConfigMap directory (cm_myconf, in this example). In this db.ini file, define the PermSize and DatabaseCharacterSet connection attributes.

```
vi db.ini
```

```
PermSize=200
DatabaseCharacterSet=AL32UTF8
ConnectionCharacterSet=AL32UTF8
```

4. Change to the Secret directory.

cd secret_mysecret

5. Create the adminUser file in this Secret directory (secret_mysecret in this example). In this adminUser file, create the sampleuser user with the samplepw password.

vi adminUser

sampleuser/samplepw

6. Create the schema.sql file in this Secret directory (secret_mysecret in this example). In this schema.sql file, define the s sequence and the emp table for the sampleuser user. The Operator automatically initializes your database with these object definitions.

```
vi schema.sql
create sequence sampleuser.s;
create table sampleuser.emp (
   id number not null primary key,
   name char(32)
);
```

7. Create the ConfigMap. The files in the cm_myconfig directory are included in the ConfigMap and, later, will be available in the TimesTen containers.

In this example:

- The name of the ConfigMap is myconfig. Replace myconfig with a name of your choosing.
- This example uses cm_myconfig as the directory where the files that will be copied into the ConfigMap reside. If you use a different directory, replace cm_myconfig with the name of your directory.

Create the ConfigMap.

```
kubectl create configmap myconfig --from-file=cm_myconfig
```



The output is the following:

configmap/myconfig created

8. Verify the contents of the ConfigMap.

kubectl describe configmap myconf

The output is the following:

```
Name: myconfig
Namespace: mynamespace
Labels: <none>
Annotations: <none>
Data
====
db.ini:
-----
PermSize=200
DatabaseCharacterSet=AL32UTF8
ConnectionCharacterSet=AL32UTF8
```

Events: <none>

9. Create the Secret. The files in the secret_mysecret directory are included in the Secret and, later, will be available in the TimesTen containers.

In this example:

- The name of the Secret is mysecret. Replace mysecret with a name of your choosing.
- This example uses secret_mysecret as the directory where the files that will be copied into the Secret reside. If you use a different directory, replace secret_mysecret with the name of your directory.

kubectl create secret generic mysecret --from-file=secret_mysecret

The output is the following:

secret/mysecret created

 Verify the Secret. Note the contents of the adminUser and the schema.sql files are not displayed.

kubectl describe secret mysecret

The output is the following:

Name:		mysecret
Namespa	ace:	mynamespace
Labels	:	<none></none>
Annota	tions:	<none></none>
Type:	Opaque	
Data		



```
adminUser: 12 bytes
schema.sql: 98 bytes
```

====

- **11.** Include the ConfigMap and the Secret in the object definition.
 - In the dbConfigMap field, specify the name of the your ConfigMap.
 - In the dbSecret field, specify the name of the your Secret.

This is an example of using the ConfigMap and the Secret for a TimesTenClassic object.

```
apiVersion: timesten.oracle.com/v4
kind: TimesTenClassic
metadata:
   name: sample
spec:
   ttspec:
   storageClassName: oci-bv
   storageSize: 250Gi
   image: container-registry.oracle.com/timesten/timesten:22.1.1.34.0
   imagePullSecret: sekret
   dbConfigMap:
    - myconfig
   dbSecret:
    - mysecret
```

This is an example of using the ConfigMap and the Secret for a TimesTenScaleout object.

```
apiVersion: timesten.oracle.com/v4
kind: TimesTenScaleout
metadata:
 name: sample
spec:
  ttspec:
    storageClassName: oci-bv
    storageSize: 250Gi
    image: container-registry.oracle.com/timesten/timesten:22.1.1.34.0
    imagePullSecret: sekret
    k: 2
    nReplicaSets: 3
    nZookeeper: 3
    dbConfigMap:
    - myconfig
    dbSecret:
    - mysecret
```

The myconfig ConfigMap and the mysecret Secret holds the metadata files. The ttconfig volume contains the contents of the myconfig ConfigMap and the mysecret Secret.

Example Using One ConfigMap for a TimesTenScaleout Object

This example shows you how to create a metadata file for a direct connectable and a metadata file for a client/server connectable. It then shows you how to include these connectables in a ConfigMap for a TimesTenScaleout object. The ConfigMap also includes the db.ini, adminUser, and schema.sql metadata files.

 On your development host, from the directory of your choice, create an empty subdirectory for the metadata files. This example creates the cm_samplescaleout subdirectory. (The cm_samplescaleout directory is used in the remainder of this example to denote this directory.)

mkdir -p cm samplescaleout

2. Change to the ConfigMap directory.

cd cm_samplescaleout

- 3. Create the direct connectable.
 - vi samplescaleout.connect

ConnectionCharacterSet=AL32UTF8

4. Create the client/server connectable.

vi samplecsscaleout.csconnect

```
ConnectionCharacterSet=AL32UTF8
TTC Timeout=30
```

5. Create the db.ini file. In this db.ini file, define the PermSize and DatabaseCharacterSet connection attributes.

vi db.ini

```
PermSize=200
DatabaseCharacterSet=AL32UTF8
```

6. Create the adminUser file. In this adminUser file, create the sampleuser user with the samplepw password.

vi adminUser

sampleuser/samplepw

7. Create the schema.sql file. In this schema.sql file, define the s sequence and the emp table for the sampleuser user. The Operator automatically initializes your database with these object definitions.

```
vi schema.sql
create sequence sampleuser.s;
create table sampleuser.emp (
   id number not null primary key,
   name char(32)
);
```

8. Create the ConfigMap. The files in the cm_samplescaleout directory are included in the ConfigMap. These files are later available in the TimesTen containers.

In this example:

- The name of the ConfigMap is samplescaleout. Replace samplescaleout with a name of your choosing.
- This example uses cm_samplescaleout as the directory where the files that will be copied into the ConfigMap reside. If you use a different directory, replace cm samplescaleout with the name of your directory.



kubectl create configmap samplescaleout --from-file=cm_samplescaleout

The output is the following:

configmap/samplescaleout created

9. Verify the contents of the ConfigMap.

kubectl describe configmap samplescaleout

The output is the following:

```
Name:
              samplescaleout
Namespace: mynamespace
Labels:
              <none>
Annotations: <none>
Data
====
adminUser:
____
sampleuser/samplepw
db.ini:
____
PermSize=200
DatabaseCharacterSet=AL32UTF8
sampleconnectable.connect:
____
ConnectionCharacterSet=AL32UTF8
samplecsconnectable.csconnect:
____
ConnectionCharacterSet=AL32UTF8
TTC Timeout=30
schema.sql:
____
create sequence sampleuser.s;
create table sampleuser.emp (
  id number not null primary key,
  name char(32)
);
```

Events: <none>

10. Include the ConfigMap in the object definition.

Note this example uses a storageSize of 250Gi (suitable for a production environment). For demonstration purposes, a storageSize of 50Gi is adequate.

```
apiVersion: timesten.oracle.com/v4
kind: TimesTenScaleout
metadata:
   name: samplescaleout
spec:
```



```
ttspec:
  storageClassName: oci-bv
  storageSize: 250Gi
  image: container-registry.oracle.com/timesten/timesten:22.1.1.34.0
  imagePullSecret: sekret
  k: 2
  nReplicaSets: 3
  nZookeeper: 3
  dbConfigMap:
  - samplescaleout
```

Using an init container

You can use an init container to place your metadata files into the /ttconfig directory of the TimesTen containers. An init container lets you to create your own scripts to populate the / ttconfig directory. You can use an init container for TimesTenClassic and for TimesTenScaleout objects. For more information about init containers, see:

https://kubernetes.io/docs/concepts/workloads/pods/init-containers

Note:

You can specify one or more ConfigMaps and/or Secrets in your TimesTenClassic or TimesTenScaleout object using the dbConfigMap and dbSecret datum. The result is that these ConfigMaps and/or Secrets are mounted read-only at /ttconfig. Since such a mount is read-only, you cannot write into it from an init container. Alternatively, you can use an emptydir volume and use an init container to write files into it. However, you cannot combine ConfigMaps and Secrets with an init container. For information about using ConfigMaps and Secrets, see Using ConfigMaps and Secrets.

Here is an example that illustrates how to use an init container for a TimesTenClassic object. The template element is required. This element is applied to Pods that contain the TimesTen Classic instances. The example shows you where to specify the script that populates the / ttconfig directory. It also uses the tt-config volume name in the volumes field of the TimesTenClassic object. If you specify a volume with the tt-config name, it is automatically mounted at /ttconfig in your TimesTen containers.

```
apiVersion: timesten.oracle.com/v4
kind: TimesTenClassic
metadata:
 name: init1
spec:
 ttspec:
   storageClassName: oci-by
   storageSize: 250Gi
   image: container-registry.oracle.com/timesten/timesten:22.1.1.34.0
   imagePullSecret: sekret
  template:
    spec:
     imagePullSecrets:
      - name: sekret
     initContainers:
      - name: initclassic
        image: container-registry.oracle.com/timesten/timesten:22.1.1.34.0
```



```
command:
- sh
- "-c"
- |
/bin/bash <<'EOF'
Your script to populate /ttconfig goes here
EOF
volumeMounts:
- name: tt-config
mountPath: /ttconfig
volumes:
- name: tt-config
emptyDir: {}
```

When using an init container for a TimesTenScaleout object, the metadata files must be placed in both the dataTemplate and the mgmtTemplate elements.

Additional Configuration Options

This section discusses additional configuration options. These are optional configurations for your environment:

- Persistent Storage
- Additional Resource Specifications

Persistent Storage

When you create a TimesTenClassic object, the Operator automatically creates one or two Persistent Volume Claims (PVCs) per Pod. These PVCs cause Persistent Volumes (PVs) to be allocated by Kubernetes and to be attached to the TimesTen Pods. TimesTen uses the PVs to hold the TimesTen instance and the TimesTen database. If you specify two PVCs, one PV holds the instance and the checkpoint files and the second PV holds the transaction log files.

When you create a TimesTenScaleout object, the same mechanism is used to provision persistent storage for the data instances, the management instance, and the ZooKeeper instances:

- For the data instances: The Operator creates one or two PVCs per Pod. TimesTen uses the PVs to hold the TimesTen instance and the TimesTen database. If you specify two PVCs, one PV holds the instance and the checkpoint files and the second PV holds the transaction log files.
- For the management instance: The Operator creates one PVC for the Pod that contains the management instance. The PV holds the the TimesTen management instance and the grid database.
- For the ZooKeeper instances: The Operator creates one PVC for each Pod that runs a ZooKeeper instance. The PV holds ZooKeeper's persistent data.

When you create a TimesTenClassic object, you must specify storageClassName and you may specify storageSize. These attributes determine the characteristics of the Persistent Volumes. The storageClassName must be one that is provided in your Kubernetes environment. For example, in Oracle Kubernetes Environment (OKE), you may use oci-bv.

The default storage is 50Gi. Use the storageSize attribute to request a different size. A storage size of 50Gi may be adequate for demonstration purposes, but in production environments, you should consider greater storage.



TimesTen places the TimesTen installation, the instance, and the database in this storage. It is mounted in each container, in each Pod, as /tt. The TimesTen instance is located at /tt/ home/timesten/instances/instance1.

When you create a TimesTenScaleout object, the following attributes are supported:

- Storage class name:
 - dataStorageClassName: Name of the storage class that is used to request persistent volumes for the elements of the TimesTen database in the grid. If not specified, the default is the value of storageClassName.
 - mgmtStorageClassName: Name of the storage class that is used to request persistent volumes for the database of the management instance. If not specified, the default is the value of storageClassName.
 - zookeeperStorageClassName: Name of the storage class that is used to request persistent volumes for ZooKeeper's persistent data. If not specified, the default is the value of storageClassName.
 - storageClassName: If the data storage class name, the management storage class name, and the zookeeper storage class name are the same, you can just specify storageClassName.
- Storage size:
 - dataStorageSize: Amount of storage to be provisioned for each element of the TimesTen database in the grid. The default is 50Gi.
 - mgmtStorageSize: Amount of storage to be provisioned for the database of the management instance. The default is 50Gi.
 - zookeeperStorageSize: Amount of storage to be provisioned for ZooKeeper's persistent data. The default is 50Gi.
 - storageSize: If the data storage size, the management storage size, and the zookeeper storage size are the same, you can just specify storageSize. For example, if the dataStorageSize is 75Gi, and the mgmtStorageSize is 75Gi, and the zookeeperStorageSize is 75Gi, you can specify storageSize with a value of 75Gi. The value for dataStorageSize, for mgmtStorageSize, and for zookeeperStorageSize is set to the value of storageSize.

For the TimesTen databases (using TimesTen Classic) and for the TimesTen database (using TimesTen Scaleout):

- TimesTen best practices recommends that the transaction log files associated with a TimesTen database be located on a different storage volume than the checkpoint files for the database. This provides separate paths to storage for the checkpoint and the transaction log operations. For example, you can store the transaction log files in a high performance storage, while storing the checkpoint files in a slower storage. See Locate Checkpoint and Transaction Log Files on Separate Physical Device in the Oracle TimesTen In-Memory Database Operations Guide for more information.
- To locate the checkpoint files and the transaction log files on a separate path of storage, provide a value for a second persistent storage that is used only for the transaction log files. Use the logStorageSize attribute for this and control its placement by using the logStorageClassName attribute. This causes a second PVC to be created for each Pod, which will then be available in each container at /ttlog. (This second storage volume has a /ttlog mount point.)

Here is an example for a TimesTenClassic object. The same example can be used for a TimesTenScaleout object:

```
apiVersion: timesten.oracle.com/v4
kind: TimesTenClassic
metadata:
   name: sample
spec:
   ttspec:
   storageClassName: slower
   storageSize: 750Gi
   logStorageClassName: faster
   logStorageSize: 200G
```

Additional Resource Specifications

Kubernetes supports affinity and anti-affinity settings that let applications control their placement within the Kubernetes cluster. These settings can be used to ensure all replicas do not reside on a single physical host.

You can specify affinity settings, node selectors, additional containers, tolerations, resource requirements, and other Kubernetes attributes for the TimesTen Pods and the containers within these Pods that are created by the TimesTen Operator.

In a TimesTenClassic deployment, you specify these resource specifications in the TimesTenClassic object's .spec.template datum. The TimesTen Operator passes this template to the StatefulSet. For example, when you deploy a TimesTenClassic object, the Operator configures a replicated pair of TimesTen databases that provide high availability. However, since the Operator does not control the placement of Pods, you can achieve an even greater level of high availability by controlling the placement of the TimesTen Pods. TimesTen Pods can then be available in different availability zones or are on different Kubernetes nodes. To do this, you specify the affinity option in the .spec.template datum for the TimesTenClassic object.

Similar to a TimesTenClassic deployment, you can specify the same resource specifications for a TimesTenScaleout object. The TimesTenScaleout object supports the .spec.mgmtTemplate, .spec.dataTemplate and .spec.zookeeperTemplate attributes. You can use these attributes to pass affinity and other settings to Kubernetes. These are of type PodTemplateSpec:

- mgmtTemplate: Applied to the Pod that contains the TimesTen Scaleout management instance. Consists of a single PodTemplateSpec.
- dataTemplate: Applied to the Pods that contain the TimesTen Scaleout data instances. Consists of an array of PodTemplateSpec. If specified, there must be one entry in the array for each data space in the grid (k entries in the array). This lets you specify a different placement for each data space. For example, you can have data space one reside in availability zone 1 and data space two reside in availability zone 2.
- zookeeperTemplate: Applied to the Pods that contain the ZooKeeper instances used by Scaleout. Consists of a single PodTemplateSpec.

For information about PodTemplateSpec see, https://kubernetes.io/docs/reference/ kubernetes-api/

Here is an example of specifying the affinity setting for a TimesTenClassic object.



```
template:
affinity:
   podAntiAffinity:
    preferredDuringSchedulingIgnoredDuringExecution:
        - weight: 1
        podAffinityTerm:
        labelSelector:
        matchExpressions:
        - key: "app"
            operator: In
        values:
            - ds1
        topologyKey: "kubernetes.io/hostname"
```

About the Default Affinity and Anti-Affinity Settings for TimesTenClassic Objects

Kubernetes affinity and anti-affinity settings constrain how Pods are scheduled on a Node. See Affinity and anti-affinity in the Kubernetes documentation.

By default, the TimesTen Operator adds the following affinity and anti-affinity settings to Pods that are created when deploying a TimesTenClassic object:

- A nodeAffinity setting that instructs Kubernetes to provision TimesTen Pods on nodes with the proper architecture (either amd64 or arm64). For more information about these architectures, see About Deploying in a Multi-Architecture Kubernetes Cluster in this book. This architecture setting is required.
- A podAntiAffinity section that instructs Kubernetes to schedule Pods for a TimesTenClassic object on different nodes, if possible. This section is preferred. Because the anti-affinity setting is preferred (a lesser constraint), Kubernetes can schedule multiple Pods on the same Node if resource constraints make it necessary. For example, although not recommended, an active standby pair of TimesTen databases can be provisioned in a one Node cluster. However, if two or more Nodes are available, Kubernetes attempts to schedule TimesTen Pods on different Nodes.

If you provide your own affinity section in a TimesTenClassic object definition, the TimesTen Operator uses your affinity section and does not modify it.

Note:

If you specify a nodeAffinity.requiredDuringSchedulingIgnoredDuringExecution section, the TimesTen Operator adds the arch clause to it.

The following snippet shows the default affinity and anti-affinity settings for a TimesTenClassic object named sample.

```
affinity:
  nodeAffinity:
   requiredDuringSchedulingIgnoredDuringExecution:
    nodeSelectorTerms:
    - matchExpressions:
    - key: kubernetes.io/arch
        operator: In
        values:
```

About the Default Kubernetes Security Context for TimesTen Pods

A Kubernetes security context defines privilege and access control settings for a Pod or Container. There are several security context settings. See Configure a Security Context for a Pod or Container in the Kubernetes documentation.

The TimesTen Operator adds a default security context, including the Kubernetes default seccomp profile to the Pods it creates. This security context defines the privileges and capabilities in which TimesTen containers run. If you provide your own security context for TimesTen Pods, the TimesTen Operator uses your provided security context and does not make any changes to it.

The following snippet shows the default security context for TimesTen containers.

```
securityContext:
  allowPrivilegeEscalation: false
  capabilities:
    drop:
    - ALL
    privileged: false
    readOnlyRootFilesystem: true
    runAsNonRoot: true
    seccompProfile:
       type: RuntimeDefault
```

About Readiness Probes for TimesTen Containers

A Kubernetes readiness probe enables Kubernetes to determine whether a particular application is *ready*. For example, consider an application that has to perform a lengthy startup procedure. When the application is first started by Kubernetes, it is not immediately ready. It cannot handle requests or workloads until the startup procedure is complete.

The TimesTen Kubernetes Operator (TimesTen Operator) provides different definitions of ready for TimesTen (tt) containers and defines readiness probes based on these definitions. This applies to replicated and non-replicated configurations. The following sections discuss these readiness probes for replicated and non-replicated configurations:

- About Readiness Probes for Replicated Configurations
- About Readiness Probes for Non-Replicated Configurations

About Readiness Probes for Replicated Configurations

For active/standby pair replication configurations, the TimesTen Operator provides two different definitions of ready along with two associated readiness probes. They are as follows:

- A TimesTen database is considered ready if it is the active database in an active/standby pair replication scheme. The TimesTen Operator provides the /tmp/active readiness probe for this definition of ready.
- A TimesTen database is considered ready if it is loaded and open for connections. The TimesTen Operator provides the /tmp/readiness readiness probe for this definition of ready.

These two definitions of ready fulfill different and incompatible requirements. You must decide what definition of ready is best for your environment.

You also have the option of defining your own readiness probe. If you do so, the TimesTen Operator uses your readiness probe rather than one of the provided ones.

To learn about the readiness probes that the TimesTen Operator provides:

- About the /tmp/active Readiness Probe
- About the /tmp/readiness Readiness Probe

About the /tmp/active Readiness Probe

This section discusses the definition of ready whereby the tt container providing the active database is the one considered ready.

The TimesTen Operator creates and manages a file called /tmp/active in the tt container's file system to determines if the tt container is ready. If the /tmp/active file exists, the tt container is ready. If the file does not exist, the tt container is not ready.

The TimesTen Operator provides and defines the /tmp/active readiness probe for this definition of ready. The TimesTen Operator enables this readiness probe by default for replicated TimesTenClassic objects. The definition is in YAML format and is as follows:

```
readinessProbe:
exec:
    command:
    - cat
    - /tmp/active
failureThreshold: 1
periodSeconds: 10
successThreshold: 1
```

In this example, Kubernetes runs the cat command in the tt container every 10 seconds. If the command exits with a return code of 0, the container is ready. If the command returns any other value, the container is not ready.

Defining a readiness probe in this manner ensures that Kubernetes Services routes incoming client/server connections to databases that are working and that can be read and written. For more information about client/server connections, see About Using Client/Server Drivers.



About the /tmp/readiness Readiness Probe

This section discusses the definition of ready whereby a tt container providing a TimesTen database that is loaded and open for connections is considered ready.

The TimesTen Operator creates and manages a file called /tmp/readiness in the tt container's file system to determines if the container is ready. If the /tmp/readiness file exists, the tt container is considered ready. If the file does not exist, the tt container is considered not ready.

The TimesTen Operator provides and defines the /tmp/readiness readiness probe for this definition of ready. The definition is in YAML format and is as follows:

```
readinessProbe:
exec:
    command:
    - cat
    - /tmp/readiness
failureThreshold: 1
periodSeconds: 10
successThreshold: 1
```

In this example, Kubernetes runs the cat command in the tt container every 10 seconds. If the command exits with a return code of 0, the container is ready. If the command returns any other value, the container is not ready.

This readiness probe is useful if you want to replace one or more Nodes in your Kubernetes cluster. In this case, you can cause Kubernetes to drain the workload from the Node. This causes Kubernetes to evict any Pods that are running on that Node and create new Pods on other Nodes in the cluster to replace them. Kubernetes supports a *Pod disruption budget* whereby you specify a budget for your application. This budget tells Kubernetes how many evicted Pods in a given Deployment can be tolerated. For example, assume you configure a Deployment with 20 replicas of your application. You could tell Kubernetes to tolerate up to 5 of the replicas being down at a time. When moving a workload from one Node to another, Kubernetes is careful not to delete more than 5 replicas at a time, and waits for their replacements to become ready before deleting more.

In the case of the TimesTen Operator, you can use the /tmp/readiness readiness probe to prevent Kubernetes from terminating both the active and standby TimesTen Classic databases simultaneously while draining Kubernetes Nodes.

If you use a Pod disruption budget of 1 on TimesTen, you can drain the workload from one or more Nodes without creating a total TimesTen outage. When Kubernetes deletes a Pod that is running TimesTen in TimesTen Classic, Kubernetes does not know if the Pod contains an active or a standby database. Therefore, it may choose to delete the Pod that contains the active database. This causes a failover to the standby, which disrupts applications if performed during normal hours. There is no way to prevent this. However, Kubernetes does not proceed to delete the other database until the one that was deleted comes back up and is completely in the Healthy state.

For more information on the health of a Pod and the Healthy state, see About the High Level State of TimesTen Pods.



For information on Kubernetes Pod disruption budgets, see https://kubernetes.io/docs/concepts/workloads/pods/disruptions/ and https://kubernetes.io/docs/tasks/run-application/configure-pdb/.

About Disabling Readiness Probes

We recommend that you do not disable readiness probes. However, if do not want to use a readiness probe for replicated objects, you can instruct the TimesTen Operator not to provision one. Here's how:

In your TimesTenClassic object definition, specify a value of false for the .spec.ttspec.createASReadinessProbe datum. For example:

```
apiVersion: timesten.oracle.com/v4
kind: TimesTenClassic
metadata:
   name: sample
spec:
   ttspec:
   ...
        createASReadinessProbe: false
```

For information about the .spec.ttspec.createASReadinessProbe datum, see TimesTenClassicSpecSpec.

About Readiness Probes for Non-Replicated Configurations

For non-replicated configurations, the TimesTen Operator provides one definition of ready and one associated readiness probe. The definition is as follows: A TimesTen database is considered ready if it is loaded and open for connections. The TimesTen Operator provides the /tmp/readiness readiness probe for this definition of ready.

The TimesTen Operator creates and manages a file called /tmp/readiness in the tt container's file system to determines if the container is ready. If the /tmp/readiness file exists, the tt container is ready. If the file does not exist, the tt container is not ready.

By default, the TimesTen Operator provides and defines this /tmp/readiness readiness probe for non-replicated TimesTenClassic objects. The definition is in YAML format and is as follows:

```
readinessProbe:
exec:
    command:
    - cat
    - /tmp/readiness
failureThreshold: 1
periodSeconds: 10
successThreshold: 1
```

In this example, Kubernetes runs the cat command in the tt container every 10 seconds. If the command exits with a return code of 0, the container is ready. If the command returns any other value, the container is not ready.

Defining a readiness probe in this manner ensures that Kubernetes Services routes incoming client/server connections to databases that are working and that can be read and written. For more information about client/server connections, see About Using Client/Server Drivers.

You have the option of defining your own readiness probe. If you do so, the TimesTen Operator uses your readiness probe rather than the provided one.

6

Specify CPU and Memory Requests and Limits

This chapter discusses the importance of specifying CPU and memory requests and limits for TimesTen Classic and TimesTenScaleout objects. It also provides an understanding of Linux cgroups and gives background information about the Linux out of memory (OOM) killer.

Topics:

- About Resource Requests and Limits
- About TimesTen Containerized Deployments
- About Specifying Requests and Limits for TimesTen Containers
- Approach 1: Use Specific Datum for Requests and Limits
- Approach 2: Use Templates for Requests and Limits
- About Specifying Requests and Limits to Kubernetes
- About Verifying databaseMemorySize
- About Runtime Memory Monitoring

About Resource Requests and Limits

One of the core Linux technologies that is used to implement containers is cgroups. Cgroups can be used to enforce CPU and memory use limitations on a process or processes.

Kubernetes provides facilities that let you specify the amount of CPU and memory that a container consumes. When specified, Kubernetes uses your requests to do the following:

- Determine the node a particular Pod should be created on: Kubernetes determines which nodes have enough free resources to satisfy your request.
- Enforce these limits at runtime: Kubernetes ensures that applications do not exceed their requests.

Kubernetes chooses which node of the cluster to run a Pod on based on the Pod's resource requests, and the resources available on each node in the cluster. But once a Pod is scheduled onto a node, Kubernetes does not directly enforce limits at runtime. Rather, Kubernetes passes requests to Linux through cgroups. The node's Linux kernel then enforces the limits on Kubernetes behalf.

Let's look at an example of a Pod with the following definition:

```
apiVersion: v1
kind: Pod
metadata:
   name: samplePod
spec:
   containers:
   - image: container-registry.oracle.com/timesten/timesten:22.1.1.34.0
    name: sample
```



```
resources:
limits:
cpu: 20
memory: 200Gi
requests:
cpu: 20
memory: 200Gi
```

Kubernetes determines which node it will run the Pod on. It does this by examining the Pod's resources requests to determine which node has enough free resources to accommodate the Pod.

Once the node is determined, Kubernetes creates a cpu cgroup for the container (sample in this example) and configures the cgroup to have a limit of 20 CPUs. Kubernetes also creates a memory cgroup for the container and configures it to have a limit of 200 gigabytes. Kubernetes then forks off the lead process of the newly created container, and associates the initial process with the cpu and memory cgroups.

Since the lead process is associated with or running under these cgroups, that process and all its children are subject to the limits that the cgroups define. The Linux kernel on the node where the container is running automatically enforces these limits without any involvement from Kubernetes.

CPU limits are easily enforced by Linux. If an application wants to use more CPU than its limit, the kernel can choose not to dispatch the application for a period of time to keep its usage under control.

Memory limit enforcement is different than CPU enforcement. Linux has a component called the out of memory (OOM) killer. If processes exceed their intended memory usage, or if the system gets stressed, the OOM killer terminates processes abruptly.

Next, let's consider a Pod with this definition:

```
apiVersion: v1
kind: Pod
metadata:
    name: samplePod
spec:
    containers:
    - image: container-registry.oracle.com/timesten/timesten:22.1.1.34.0
    name: sample
```

In this Pod, there are no specified memory limits. The container runs in default cgroups with infinite limits. This cgroup is shared by all processes on the node that have no specified memory limit, whether the processes are running in containers or not. In this case, if the Linux node becomes memory constrained, the OOM killer chooses processes from the entire node to terminate. The victim might be TimesTen, it might be the Kubernetes kubelet, it might be some other process, or it might be all of them.

About TimesTen Containerized Deployments

As an in-memory database, TimesTen uses a large amount of memory by design. Processes that run TimesTen may be the largest memory users on a given node. If the operating system gets stressed, TimesTen is likely a prime candidate to get terminated by the Linux OOM killer. Therefore, it is crucial that memory requests and limits be specified for your TimesTen containers. TimesTen also recommends that CPU requests and limits be specified.



Note:

Ensure the value of a request and the value of a limit for a resource is the same. For example, if the memory request for the daemonlog container has a value of 30Mi ensure the memory limit for this daemonlog container also has a value of 30Mi. There are examples later in the chapter that illusrate this.

The memory request and memory limit for the tt container is the most essential and crucial to specify. The value is dependent on the memory required to hold the TimesTen database as well as the memory required for the TimesTen daemon, subdaemons, cache and replication agents, Client/Server server, and so on. The memory required to hold the database is dependent on the size of your database.

The TimesTen Operator provides functionality to accurately size your TimesTen database. This functionality is discussed later. The additional memory required for the TimesTen daemon, subdaemons, Client/Server server is dependent on your SQL and PL/SQL usage and the memory requirements vary with your workload. The TimesTen Operator provides a default of 2Gi for this additional memory. You can use this default or change it. How to change it is discussed later.

Let's take a look at the defaults for the TimesTen containers. In all cases, you have the option of changing the default.

- tt container:
 - memory: This value is discussed in detail throughout this chapter.
 - cpu: The value is dependent on how much CPU the tt container requires. This
 includes CPU used by the TimesTen daemon, subdaemons, replication agents, cache
 agents, and by the Client/Server server. The Client/Server server executes SQL on
 behalf of your applications, so the value depends on the workload. There is no default.
- daemonlog container:
 - memory: The default is 200Mi.
 - cpu: The default is 200m.
- exporter container (if provisioned):
 - memory: The default is 200Mi.
 - cpu: The default is 200m.

In addition, if you are using TimesTen Scaleout, there are additional TimesTen containers to consider:

- tt container of the management instance:
 - memory: The default is 3Gi.
 - cpu: The default is 1.
- zookeeper container:
 - memory: The default is 1Gi.
 - cpu: The default is 500m.



About Specifying Requests and Limits for TimesTen Containers

For TimesTen Classic and TimesTen Scaleout, the TimesTen Operator provides default values for memory and CPU requests for the daemonlog and exporter containers. For TimesTen Scaleout specifically, the TimesTen Operator provides default values for CPU and memory requests for the tt container of the management instance and for the zookeeper container. You can override these defaults by specifying specific datum in the .spec.ttspec section of your TimesTenClassic and TimesTenScaleout object definitions.

These datum are as follows:

- daemonLogMemoryRequest
- daemonLogCPURequest
- exporterMemoryRequest
- exporterCPURequest
- mgmtMemoryRequest: (TimesTen Scaleout)
- mgmtCPURequest: (TimesTenScaleout)
- zookeeperMemoryRequest: (TimesTen Scaleout)
- zookeeperCPURequest: (TimesTen Scaleout)

The TimesTen Operator does not provide a default value for the CPU request of a tt container. If you want to specify a value for this CPU request, use the .spec.ttspec.databaseCPURequest datum in your TimesTenClassic and TimesTenScaleout object definitions.

For details about these datum, see TimesTenClassicSpecSpec and TimesTenScaleoutSpecSpec.

Let's explore how the TimesTen Operator gathers information for memory and CPU requests and limits and passes the information to Kubernetes. There are two approaches:

- Use specific datum in .spec.ttspec for requests and limits: This is the default approach and the approach we recommend. The TimesTen Operator provides specific datum in the .spec.ttspec section of your TimesTenClassic and TimesTenScaleout object definitions and passes the information in these datum to Kubernetes. The TimesTen Operator also automatically determines an appropriate memory request and limit for your tt containers.
- Use templates for requests and limits: This approach uses memory and CPU requests and limits information in templates. In this approach, you specify values for memory and CPU requests for the TimesTen containers. The TimesTen Operator passes this information to Kubernetes.

The specific datum are discussed in detail in subsequent sections in this chapter.

Here is a summary list of the datum:

- automaticMemoryRequests
- databaseCPURequest
- databaseMemorySize
- additionalMemoryRequest



memoryWarningPercent

For details about these datum, see TimesTenClassicSpecSpec and TimesTenScaleoutSpecSpec.

Approach 1: Use Specific Datum for Requests and Limits

In this approach, the TimesTen Operator automatically determines the appropriate memory request and limit for the tt containers. This is the default behavior (.spec.ttspec.automaticMemoryRequests is set to true by default) for a TimesTenClassic or TimesTenScaleout object.

For the remaining TimesTen containers, the TimesTen Operator uses specific datum in the .spec.ttspec section of the TimesTenClassic and TimesTenScaleout object definitions to determine the memory request and limit and the CPU request and limit for each of the TimesTen containers. The TimesTen Operator either uses the default values for the datum or uses a value that you specify. The exception is the CPU request and limit for the tt container. Since there is no default, if you want to define a CPU request and limit, you must manually specify a value in the .spec.ttspec.databaseCPURequest datum of your TimesTenClassic and TimesTenScaleout objects. The TimesTen Operator then supplies all this information for memory and CPU requests and limits to Kubernetes.

For details about these datum, including defaults, see TimesTenClassicSpecSpec and TimesTenScaleoutSpecSpec.

It is essential that the value for the memory request for the tt container that holds the TimesTen database be accurate.

Recall that the memory request for the tt container is based on:

- Shared memory for the database: This is dependent on the size of the database.
- Additional memory: This is the memory required in addition to the database. It includes
 memory that is used for the TimesTen daemon, subdaemons, agents, Client/Server server.

The TimesTen Operator provides the .spec.ttspec.databaseMemorySize and .spec.ttspec.additionalMemoryRequest datum for these specific memory requirements. The .spec.ttspec.databaseMemorySize is used to specify the size of the database and the .spec.ttspec.additionalMemoryRequest is used for the additional memory.

The TimesTen Operator adds the value of .spec.ttspec.additionalMemoryRequest to the value of .spec.ttspec.databaseMemorySize. The sum is the memory request and memory limit to Kubernetes.

You do not have to specifically specify the <code>.spec.ttspec.databaseMemorySize</code> datum for a TimesTenClassic or TimesTenScaleout object. If not specified, the TimesTen Operator attempts to determine the appropriate value.

TimesTen provides the ttShmSize utility to determine the shared memory requirements of a database, given its sys.odbc.ini entry. For information about ttShmSize, see ttShmSize in the Oracle TimesTen In-Memory Database Reference.

The equivalent for a TimesTen sys.odbc.ini file is the TimesTen Operator db.ini metadata file. You can provide the db.ini file in several ways:

- Embed in a ConfigMap referenced in .spec.ttspec.dbConfigMap.
- Embed in a Secret referenced in .spec.ttspec.dbSecret.
- Use an init container.



For details about the facilities that you can use to provide metadata files, see Populate the / ttconfig Directory.

TimesTen recommends that you provide the db.ini metadata file in either a ConfigMap or a Secret. The TimesTen Operator examines the Configmaps and Secrets, if any, in your TimesTenClassic or TimesTenScaleout objects. If the db.ini is found in a Configmap or Secret, the TimesTen Operator uses the TimesTen ttShmSize utility to determine the appropriate amount of shared memory to request based on your database definition. This value is then used for the .spec.ttspec.databaseMemorySize value. With this approach, the TimesTen Operator does the database sizing for you.

Let's look at an example:

```
kind: ConfigMap
metadata:
  name: resource9
data.
  adminUser: |
    adminuser/adminuserpwd
  schema.sql: |
    create user sampleuser identified by sampleuserpwd;
    grant admin to sampleuser;
    create table sampleuser.a (b number not null primary key, c number, d
timestamp);
    insert into sampleuser.a values(-1, -1, sysdate);
  db.ini: |
    Permsize=32768
    TempSize=4096
    LogBufMB=1024
    Connections=2048
    DatabaseCharacterSet=AL32UTF8
apiVersion: timesten.oracle.com/v4
kind: TimesTenClassic
metadata:
  name: recommendedoption
spec:
  ttspec:
   dbConfigMap:
    - option1
    storageClassName: standard
    storageSize: 200Gi
    image: container-registry.oracle.com/timesten/timesten:22.1.1.34.0
    imagePullSecret: sekret
    prometheus:
      insecure: true
```

In this case, the TimesTen Operator runs ttShmSize against your provided db.ini file and determines the value for .spec.ttspec.databaseMemorySize automatically. The TimesTen Operator then adds this value to the value for .spec.ttspec.additionalMemoryRequest. The sum is the memory request and memory limit to Kubernetes.

If you provide the db.ini file by using an init container or other mechanism, the TimesTen Operator cannot determine the value for .spec.ttspec.databaseMemorySize. By the time the Pod is provisioned and the init container is executed, the Pod has already been created and its memory requirements defined. In such cases, you must manually provide the .spec.ttspec.databaseMemorySize as part of your YAML.

TimesTen recommends that you use the ttShmSize utility in a TimesTen instance outside of Kubernetes to determine the appropriate value for .spec.ttspec.databaseMemorySize. You do not need to create the database.

This example assumes you have created a TimesTen instance outside of Kubernetes and have created a DSN in your sys.odbc.ini file with the name database1. Use the ttShmSize utility based on provided values for the PermSize, TempSize, LogBufMB, and Connections connection attributes.

```
ttShmSize -connstr
"DSN=database1;PermSize=32768;TempSize=4096;LogBufMB=1024;Connections=2048"
The required shared memory size is 39991547720 bytes.
```

Specify this value in the .spec.ttspec.databaseMemorySize datum.

Let's look at an example that uses an init container and uses the calculated value for .spec.ttspec.databaseMemorySize.

```
apiVersion: timesten.oracle.com/v4
kind: TimesTenClassic
metadata:
  name: init1
spec:
  ttspec:
    databaseMemorySize: 41Gi
    storageClassName: standard
    storageSize: 200Gi
    image: container-registry.oracle.com/timesten/timesten:22.1.1.34.0
    imagePullSecret: sekret
    prometheus:
      insecure: true
  template:
    spec:
      initContainers:
      - name: init1a
        command:
        - sh
        - "-c"
        - |
          /bin/bash <<'EOF'
          echo adminuser/adminuserpwd > /ttconfig/adminUser
          echo PermSize=32768 > /ttconfig/db.ini
          echo TempSize=4096 > /ttconfig/db.ini
          echo LogBufMB=1024 > /ttconfig/db.ini
          echo Connections=2048 > /ttconfig/db.ini
          echo DatabaseCharacterSet=AL32UTF8 >> /ttconfig/db.ini
          ls -l /ttconfig
          EOF
        volumeMounts:
        - name: tt-config
          mountPath: /ttconfig
      volumes:
```



```
- name: tt-config
emptyDir: { }
```

In this case, the TimesTen Operator uses the value you specified

for .spec.ttspec.databaseMemorySize to determine the size of the shared memory segment to hold the TimesTen database. The TimesTen Operator then adds this value to the value for .spec.ttspec.additionalMemoryRequest. The sum is the memory request and memory limit to Kubernetes.

Note:

In TimesTen Scaleout, the sys.odbc.ini file (and corresponding db.ini file) define the size of a single element of the database, not the entire database. When provisioning a TimesTenScaleout object, the TimesTen Operator uses the provided data in the same manner as it does for a TimesTenClassic object.

Approach 2: Use Templates for Requests and Limits

This approach uses a template for specifying resource requests and limits. To enable this behavior, set the .spec.ttspec.automaticMemoryRequests datum to false for your TimesTenClassic or TimesTenScaleout object.

In your YAML for TimesTenClassic and TimesTenScaleout objects, you can specify a template for Pods. In this template, you specify attributes of various containers in the Pods, including the tt container. If you specify a template for one or more containers, the resource requests and limits for the containers are used by Kubernetes.

TimesTen recommends that you use the ttShmSize utility in a TimesTen instance outside of Kubernetes to determine the appropriate value for the memory request and limit for the tt container. You do not need to create the database.

This example assumes you have created a TimesTen instance outside of Kubernetes and have created a DSN in your sys.odbc.ini file with the name database1. Use the ttShmSize utility based on provided values for the PermSize, TempSize, LogBufMB, and Connections connection attributes.

```
ttShmSize -connstr
"DSN=database1;PermSize=32768;TempSize=4096;LogBufMB=1024;Connections=2048"
The required shared memory size is 39991547720 bytes.
```

For information about ttShmSize, see ttShmSize in the Oracle TimesTen In-Memory Database Reference.

Let's look at an example:

```
apiVersion: timesten.oracle.com/v4
kind: TimesTenClassic
metadata:
   name: resource1
spec:
   ttspec:
   storageClassName: standard
   storageSize: 100Gi
```



```
image: container-registry.oracle.com/timesten/timesten:22.1.1.34.0
  imagePullSecret: sekret
  prometheus:
    insecure: true
template:
  spec:
    containers:
    - name: tt
      resources:
        requests:
          memory: "41Gi"
                  "20"
          cpu:
        limits:
          memory: "41Gi"
                  "20"
          cpu:
    - name: daemonlog
      resources:
        requests:
          memory: "30Mi"
          cpu:
                  "210"
        limits:
          memory: "30Mi"
                  "210"
          cpu:
     name: exporter
      resources:
        requests:
          memory: "22Mi"
          cpu:
                  "310m"
        limits:
          memory: "22Mi"
          cpu:
                  "310m"
```

In this example, cpu and memory requests and limits for the tt, daemonlog, and exporter containers are included in the template. These resources are specified to Kubernetes.

About Specifying Requests and Limits to Kubernetes

The TimesTen Operator follows a specific order in determining the cpu and memory request and limits to Kubernetes:

- For the tt container that holds your TimesTen database, the TimesTen Operator looks for the following in this order:
 - If you specify a template, the TimesTen Operator uses the values in it.
 - If you specify .spec.ttspec.databaseMemorySize, the TimesTen Operator uses its value.
 - If there is a db.ini file, the TimesTen Operator uses the values in it.
 - If none of the above are true, the TimesTen Operator uses the default.

In addition, if you specified a value for .spec.ttspec.databaseCPURequest, that value is used as the cpu request and cpu limit to Kubernetes.

• If you specify resource requests or limits for the daemonlog container in your container templates in your TimesTenClassic or TimesTenScaleout object, the TimesTen Operator honors those requests. If you do not, the TimesTen Operator uses the values you supply in



your object's .spec.ttspec.daemonLogMemoryRequest
and .spec.ttspec.daemonLogCPURequest datum.

- If you specify resource requests or limits for the exporter container in your container templates in your TimesTenClassic or TimesTenScaleout object, the TimesTen Operator honors those requests. If you do not, the TimesTen Operator uses the values you supply in your object's .spec.ttspec.exporterMemoryRequest and .spec.ttspec.exporterCPURequest datum.
- If you specify a template for the tt container in the mgmtTemplate in a TimesTenScaleout object, the TimesTen Operator uses the resource data in that template. If you do not, the TimesTen Operator uses the values you supply in your object's .spec.ttspec.mgmtMemoryRequest and .spec.ttspec.mgmtCPURequest datum.
- If you specify a template for the <code>zookeeper</code> container in the <code>zookeeperTemplate</code> in a TimesTenScaleout object, the TimesTen Operator uses the resource data in that template. If you do not, the TimesTen Operator uses the values you supply in your object's <code>.spec.ttspec.zookeeperMemoryRequest</code> and <code>.spec.ttspec.zookeeperCPURequest</code> datum.

About Verifying databaseMemorySize

Whether specified by you or determined by the TimesTen Operator, before a database is created, the TimesTen Operator and the TimesTen agent checks that the tt containers in the relevant TimesTen Pods have the memory resources required to create the database. The TimesTen Operator accomplishes this by running the TimesTen ttShmSize utility and comparing it with the memory quotas in the tt container's cgroup.

If the required resources are not available, the TimesTen Operator returns an error message (as an Event) and moves the TimesTenClassic or TimesTenScaleout object to the Failed state.

This checking is performed even if the value of .spec.ttspec.automaticMemoryRequests is false.

About Runtime Memory Monitoring

The TimesTen Operator monitors the memory usage of TimesTen Pods at runtime. It informs you of the following:

- If any tt containers are approaching their specified memory limits.
- If any TimesTen containers have been terminated by the Linux OOM killer.

Kubernetes Events are generated to report on these conditions.

Every .spec.ttspec.pollingInterval seconds, the TimesTen agent in each tt container queries the container's underlying Linux cgroup to determine the cgroup's memory.limit_in_bytes and memory.usage_in_bytes and reports these values to the TimesTen Operator. The Kubernetes status of each container is similarly queried. The TimesTen Operator uses this data to generate appropriate Events as needed.

If the usage is greater than .spec.ttspec.memoryWarningPercent of the limit, an Event is generated to notify you. In addition, if the TimesTen Operator observes that one or more of the TimesTen related containers have been terminated or restarted (by the OOM killer or otherwise), the TimesTen Operator reports the observation by using appropriate Events.



7 Create TimesTen Classic Databases

The TimesTen Kubernetes Operator (TimesTen Operator) creates, manages, and monitors TimesTen Classic databases. It supports the following topologies:

- Replicated active standby pair configurations: The TimesTen Operator configures one TimesTen database as the active database, copies the active database to the standby, and configures an active standby pair replication scheme between them.
- Non-replicated configurations: The TimesTen Operator configures one or more TimesTen Classic databases. Each database is independent and has no relationship to each other.

The TimesTen Operator supports TimesTen cache for both replicated and non-replicated configurations. See Work with TimesTen Cache.

Let's learn how to define these configurations and how to deploy and create TimesTen databases in your Kubernetes namespace.

Topics:

- About Defining TimesTenClassic Objects
- About the Deployment Process
- About the Examples
- Create Replicated TimesTen Classic Databases
- Create Non-Replicated TimesTen Classic Databases
- Modify the Number of Replicas in Non-Replicated Environments

About Defining TimesTenClassic Objects

The TimesTenClassic object type provides the metadata and attributes to define and create TimesTen Classic databases in your Kubernetes namespace. The metadata and attributes are applicable to replicated and non-replicated configurations. To distinguish the two configurations, the TimesTenClassic object type provides

the .spec.ttspec.replicationTopology datum. The value of

the .spec.ttspec.replicationTopology datum determines whether the TimesTen Operator configures a replicated active standby pair configuration or a non-replicated configuration. Valid values for .spec.ttspec.replicationTopology are as follows:

- activeStandbyPair (or not specified): The TimesTen Operator configures a replicated active standby pair of TimesTen Classic databases.
- none: The TimesTen Operator configures one or more non-replicated TimesTen Classic databases. Each database operates independently and there is no TimesTen replication used.

For non-replicated configurations, with .spec.ttspec.replicationTopology set to none, the TimesTenClassic object provides additional datum to assist in your customization:

• replicas: Determines the number of Pods to be deployed. For example, if you set replicas to 3, there are three Pods, each containing a TimesTen database.



 rollingUpdatePartition: Determines how many databases are upgraded. See Perform Upgrades.

For more information about the TimesTenClassic object type, see About the TimesTenClassic Object Type.

Let's review two YAML files that contain definitions for replicated and non-replicated TimesTenClassic objects. These are simplistic in nature. There are many more customizations available to you:

Replicated:

```
apiVersion: timesten.oracle.com/v4
kind: TimesTenClassic
metadata:
    name: repsample
spec:
    ttspec:
    storageClassName: oci-bv
    storageSize: 250Gi
    image: container-registry.oracle.com/timesten/timesten:22.1.1.34.0
    imagePullSecret: sekret
    dbConfigMap:
    - repsample
```

Non-replicated:

The following customizations apply to both configurations:

- storageClassName and storageSize: Since TimesTen is a database and is persistent, you have to specify a storage class, which is used to request persistent volumes for the database. The storageSize datum determines the amount of storage to be provisioned for TimesTen and its database.
- image: You have to specify the container registry that contains the TimesTen container image. This image runs a specific release of TimesTen.
- imagePullSecret: You have to specify a Kubernetes Secret that authenticates
 Kubernetes to pull a TimesTen container image from the specified container registry.
- dbConfigMap: Optionally, you can specify one or more ConfigMaps that contain the metadata files for your TimesTen databases. See Use Configuration Metadata.

The following customizations are specific to the type of configuration:



 replicationTopology: A value of none indicates a non-replicated configuration. Since the default replicationTopology is activeStandbyPair, you do not need to specify replicationTopology for replicated configurations.

replicas: This datum is specific to non-replicated configurations.

In addition to the many options for customizing your TimesTenClassic object and your TimesTen configuration, the TimesTen Operator provides metadata files that enable you to customize and define the configuration for your TimesTen databases. These files include db.ini, adminUser, cacheUser, schema, sql, cachegroups.sql, and others. The TimesTen Operator provides several methods for you to make those files available to the TimesTen Operator and TimesTen. For details, see Use Configuration Metadata.

About the Deployment Process

After you define a TimesTenClassic object for either a replicated or non-replicated configuration, you deploy the object into your namespace using either the <code>kubectl create</code> or the <code>helm install</code> command. After the object is deployed in your namespace, the TimesTen Operator notices and takes action. The process to create and configure TimesTen and its databases begins:

- StatefulSet: The TimesTen Operator creates a StatefulSet to run TimesTen. The StatefulSet contains a specified number of replicas:
 - For replicated objects, the number of replicas is 2.
 - For non-replicated objects, the TimesTen Opertor uses the value you specified for .spec.ttspec.replicas. If you do not specify a value, the TimesTen Operator uses the default value of 1.
- Pods: Because the TimesTen Operator creates a StatefulSet, Kubernetes in turn provisions replicas number of Pods.

For both replicated and non-replicated configurations, each Pod contains several containers, including tt, daemonlog1, and exporter. If you define additional containers in .spec.template of your TimesTenClassic object, they are included in the Pod.

- PersistentVolumes: Kubernetes provisions one or more PersistentVolumes (persistent storage), which are mounted in the TimesTen containers. This is where your TimesTen databases are stored.
- Service: The TimesTen Operator creates a headless Service, which causes Kubernetes to issue unique DNS names for each Pod.

When Kubernetes creates and starts a new Pod as part of a TimesTenClassic object's StatefulSet, the TimesTen Operator interacts with the TimesTen Agent in each Pod to perform the actions to get TimesTen and its databases up and running.

The Operator performs the following actions:

- Creates a TimesTen installation.
- Creates a TimesTen instance that uses the installation just created.
- Creates TimesTen configuration files in the instance based on your provided metadata files (such as db.ini).
- Starts the TimesTen instance.
- Creates the TimesTen database.



- Create database users based on your provided metadata files (such as adminUser and cacheUser).
- Creates database schemas, tables, views, procedures, and other database objects based on your provided metadata files (such as schema.sql, cachegroups.sql, and epilog.sql).

Let's walk through some examples that show you how to define and create replicated and nonreplicated TimesTenClassic objects and how the TimesTen Operator uses the information to create TimesTen databases.

About the Examples

The following examples show you how to create YAML manifest files that define replicated and non-replicated configurations. Both examples use the same set of metadata files and both use Kubernetes ConfigMaps to get the metadata files into the tt containers. The examples show you how to deploy the TimesTenClassic objects in your Kubernetes namespace. The examples then show you how to monitor the progress of the TimesTen Operator as it creates TimesTen databases and gets them up and running and operational. The examples conclude with showing you how to connect to a TimesTen database and perform operations on it.

The examples use YAML manifest files. If you are using Helm to create TimesTen Classic databases, see Use Helm in Your TimesTen Kubernetes Operator Environment.

For simplicity, let's create the metadata files and ConfigMaps used in the examples. For complete details about metadata files, see Use Configuration Metadata and Populate the / ttconfig Directory.

 On your development host, from the directory of your choice, create an empty subdirectory for the metadata files and change to that directory. This example creates the kube files/cm subdirectory.

mkdir -p kube_files/cm
cd kube files/cm

2. Create a db.ini file.

vi db.ini

PermSize=200 DatabaseCharacterSet=AL32UTF8

3. Create an AdminUser file.

vi adminUser

adminuser/adminuserpwd

4. Create a schema.sql file.

vi schema.sql

create table adminuser.emp (id number not null primary key, name char
(32));

5. Create the ConfigMaps.



a. For the replicated object, create the repsample ConfigMap.

kubectl create configmap repsample --from-file .

The output is the following:

configmap/repsample created

b. For the non-replicated object, create the norepsample ConfigMap.

kubectl create configmap norepsample --from-file .

The output is the following:

configmap/norepsample created

The examples show how to specify the preceding ConfigMaps in the .spec.ttspec.dbConfigMap section of your YAML manifest files.

Create Replicated TimesTen Classic Databases

This example shows you how to create and deploy a replicated TimesTenClassic object in your Kubernetes namespace. It shows you how to monitor the TimesTen Operator as it uses the information in the object's definition to create an active standby pair of replicated TimesTen Classic databases.

1. Create the YAML manifest file for the TimesTenClassic object.

```
vi repsample.yaml
apiVersion: timesten.oracle.com/v4
kind: TimesTenClassic
metadata:
    name: repsample
spec:
    ttspec:
    storageClassName: oci-bv
    storageSize: 10Gi
    image: container-registry.oracle.com/timesten/timesten:22.1.1.34.0
    imagePullSecret: sekret
    dbConfigMap:
    - repsample
```

Note the following:

- The storageClassName is oci-bv. Replace oci-bv with the name of your storage class.
- The storageSize is 10Gi. Replace 10Gi with the amount of storage that needs to be requested for each Pod to hold TimesTen.
- The image is container-registry.oracle.com/timesten/timesten:22.1.1.34.0. Replace image with the name and location of your TimesTen container image.

- The imagePullSecret is sekret. Replace sekret with the image pull secret that Kubernetes needs to use to fetch the TimesTen container image.
- The dbConfigMap specifies the repsample ConfigMap created in About the Examples.
- 2. Deploy the TimesTenClassic object into your namespace.

kubectl create -f repsample.yaml

The output is the following:

timestenclassic.timesten.oracle.com/repsample created

- 3. Monitor deployment.
 - a. Check status of object.

kubectl get ttc repsample

The output is similar to the following:

NAME	STATE	ACTIVE	AGE
repsample	Initializing	None	9s

The provisioning starts, but is not completed as indicated by the Initializing state.

b. Wait a few minutes. Then, check status again.

kubectl get ttc repsample

The output is similar to the following:

NAME	STATE	ACTIVE	AGE
repsample	Normal	repsample-0	2m27s

The provisioning process completes. The active and standby databases are up and running and operational. The TimesTen Operator transitions the <code>repsample</code> object to the <code>Normal</code> state.

4. (Optional): Review the state transitions.

kubectl get events

The output is similar to the following:

LAST SEEN OBJECT	TYPE	REASON	MESSAGE
•••			
4m46s repsample	Normal	Create	timestenclassic/ Service repsample created
4m46s	Warning	Warning	timestenclassic/
repsample specified			Database CPU limit/request not



4m46s	Normal	Create	timestenclassic/
repsample	NOTINAL	Cleate	StatefulSet repsample created
4m46s	Warning	Create	timestenclassic/
	Warning	Cleale	PodMonitor repsample created
repsample 3m46s	Normal	Info	timestenclassic/
	NOTINAL	11110	
repsample 3m46s	N o com o l	Tufa	Pod repsample-0 Agent Up
	Normal	Info	timestenclassic/
repsample 3m46s	N o com o l	Tufa	Pod repsample-0 Release 22.1.1.34.0
	Normal	Info	timestenclassic/
repsample 3m46s	No um o l	Tofo	Pod repsample-1 Agent Up timestenclassic/
	Normal	Info	
repsample	N	т., б.,	Pod repsample-1 Release 22.1.1.34.0
3m46s	Normal	Info	timestenclassic/
repsample			Pod repsample-0 Daemon Up
3m45s	Normal	Info	timestenclassic/
repsample			Pod repsample-1 Daemon Up
3m30s	Normal	Info	timestenclassic/
repsample	_		Pod repsample-0 Database Loaded
3m30s	Normal	Info	timestenclassic/
repsample			Pod repsample-0 Database Updatable
3m29s	Normal	Info	timestenclassic/
repsample			Pod repsample-0 RepAgent Running
3m29s	Normal	Info	timestenclassic/
repsample			Pod repsample-0 RepScheme Exists
3m29s	Normal	StateChange	timestenclassic/
repsample			Pod repsample-0 RepState ACTIVE
2m38s	Normal	Info	timestenclassic/
repsample			Pod repsample-1 Database Loaded
2m38s	Normal	Info	timestenclassic/
repsample			Pod repsample-1 RepScheme Exists
2m38s	Normal	StateChange	timestenclassic/
repsample			Pod repsample-1 RepState IDLE
2m37s	Normal	Info	timestenclassic/
repsample			Pod repsample-1 Database Loaded
2m37s	Normal	Info	timestenclassic/
repsample			Pod repsample-1 RepScheme Exists
2m37s	Normal	StateChange	timestenclassic/
repsample			Pod repsample-1 RepState IDLE
2m32s	Normal	Info	timestenclassic/
repsample			Pod repsample-1 RepAgent Running
2m32s	Normal	StateChange	timestenclassic/
repsample			Pod repsample-1 RepState STANDBY
2m31s	Normal	StateChange	timestenclassic/
repsample			TimesTenClassic was Initializing, now
Normal			
2m26s	Normal	StateChange	timestenclassic/
repsample		2	Pod repsample-0 is Ready
2m26s	Normal	StateChange	timestenclassic/
repsample		2	Pod repsample-0 is Active Ready
2m26s	Normal	StateChange	timestenclassic/
repsample		2	Pod repsample-1 is Ready
·			<u> </u>

During the provisioning process, the TimesTenClassic object transitions from the Initializing to the Normal state.

5. Verify the existence of the underlying objects.

1. StatefulSet:

kubectl get statefulset repsample

The output is similar to the following:

NAME READY AGE repsample 1/2 7m22s

2. Service:

kubectl get service repsample

The output is similar to the following:

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
repsample	ClusterIP	None	<none></none>	6625/TCP , 8888/TCP	8m2s

3. Pods:

kubectl get pods

The output is similar to the following:

NAME	READY	STATUS	RESTARTS	AGE
repsample-0	3/3	Running	0	8m31s
repsample-1	2/3	Running	0	8m31s

4. PVCs:

kubectl get pvc

The output is similar to the following:

NAME	STATUS		
VOLUME		CAPACITY ACCES	S MODES
STORAGECLASS AGE			
tt-persistent-repsample-0	Bound	csi-f36b2402-9745-46	od-
a023-811839ab518e 250Gi	RWO	oci-bv	8m59s
tt-persistent-repsample-1	Bound	csi-0a0cfd59-b2bf-48	o7 -
bdef-6ee03794891b 250Gi	RWO	oci-bv	8m59s

6. Connect to the active database.

a. From your development host, establish a shell in the Pod.

kubectl exec -it repsample-0 -c tt -- /bin/bash


```
b. Connect to the active database. Insert a row in a table.
```

```
[timesten@repsample-0 ~]$ ttIsql repsample
   Copyright (c) 1996, 2025, Oracle and/or its affiliates. All rights
   reserved.
   Type ? or "help" for help, type "exit" to quit ttIsql.
   connect "DSN=repsample";
   Connection successful: DSN=repsample;UID=timesten;DataStore=/tt/home/
   timesten/datastore/
   repsample; DatabaseCharacterSet=AL32UTF8; ConnectionCharacterSet=US7ASCII;
   AutoCreate=0;PermSize=200;DDLReplicationLevel=3;ForceDisconnectEnabled=1
   ;
   (Default setting AutoCommit=1)
   Command> select * from adminuser.emp;
   0 rows found.
   Command> describe adminuser.emp;
   Table ADMINUSER.EMP:
     Columns:
      *ID
                                        NUMBER NOT NULL
       NAME
                                        CHAR (32)
     PRIMARY KEY (ID) RANGE INDEX
   1 table found.
   (primary key columns are indicated with *)
   Command> insert into adminuser.emp values (1, 'test');
   1 row inserted.
   Command> commit;
   Command> select * from adminuser.emp;
   < 1, test
   1 row found.
c. Exit from ttIsql and from the shell.
   Command> exit
   Disconnecting...
   Done.
   [timesten@repsample-0 ~]$ exit
   exit
```

Congratulations! You successfully created and deployed a replicated TimesTenClassic object. The replicated active standby pair of TimesTen Classic databases are up and running and fully operational.

Create Non-Replicated TimesTen Classic Databases

This example shows you how to create and deploy a non-replicated TimesTenClassic object in your Kubernetes namespace. It shows you how to monitor the TimesTen Operator as it uses the information in the object's definition to create independent TimesTen databases.

1. Create the YAML manifest file for the TimesTenClassic object.

```
vi norepsample.yaml
apiVersion: timesten.oracle.com/v4
kind: TimesTenClassic
metadata:
    name: norepsample
spec:
    ttspec:
    storageClassName: oci-bv
    storageSize: 10Gi
    image: container-registry.oracle.com/timesten/timesten:22.1.1.34.0
    imagePullSecret: sekret
    replicationTopology: none
    replicas: 3
    dbConfigMap:
    - norepsample
```

Note the following:

- The storageClassName is oci-bv. Replace oci-bv with the name of your storage class.
- The storageSize is 10Gi. Replace 10Gi with the amount of storage that needs to be requested for each Pod to hold TimesTen.
- The image is container-registry.oracle.com/timesten/timesten:22.1.1.34.0. Replace image with the name and location of your TimesTen container image.
- The imagePullSecret is sekret. Replace sekret with the image pull secret that Kubernetes needs to use to fetch the TimesTen container image.
- The replicationTopology is none. Do not change this setting for non-replicated configurations.
- The replicas value is 3. Replace 3 with the number of TimesTen Pods (each containing a TimesTen database) to provision. The value must be between 1 and 3. If you omit replicas, the default is 1.
- The dbConfigMap specifies the norepsample ConfigMap created in About the Examples.
- 2. Deploy the TimesTenClassic object into your namespace.

kubectl create -f norepsample.yaml

The output is the following:

timestenclassic.timesten.oracle.com/norepsample created

- 3. Monitor deployment.
 - a. Check status of object.

kubectl get ttc norepsample



The output is similar to the following:

NAMESTATEACTIVEAGEnorepsampleNoReplicasReadyN/A43s

The provisioning starts, but is not completed as indicated by the NoReplicasReady state.

b. Wait a few minutes. Then, check status again.

kubectl get ttc norepsample

The output is similar to the following:

NAME STATE ACTIVE AGE norepsample SomeReplicasReady N/A 74s

The provisioning continues. As evidenced by the SomeReplicasReady state, some of the databases are ready, but not all.

c. Check the status again.

kubectl get ttc norepsample

The output is similar to the following:

NAME	STATE	ACTIVE	AGE
norepsample	AllReplicasReady	N/A	2m54s

The provisioning process completes. Databases are up and running and operational. The TimesTen Operator transitions the norepsample object to the AllReplicasReady state.

4. (Optional) Review the state transitions.

kubectl get events

The output is similar to the following:

LAST SEEN OBJECT	TYPE	REASON	MESSAGE
 6m6s norepsample 6m6s	Normal Warning	Create Warning	timestenclassic/ Service norepsample created timestenclassic/
norepsample specified	2		Database CPU limit/request not
6m6s norepsample	Normal	Create	timestenclassic/ StatefulSet norepsample created
6m6s norepsample	Warning	Create	timestenclassic/ PodMonitor norepsample created
6m6s norepsample	Normal	StateChange	timestenclassic/ TimesTenClassic was Initializing, now



NoReplicasReady

NoReplicasR	eauy	
5m10s	Normal	Info
norepsample		
5ml0s	Normal	Info
norepsample 5m10s	Normal	Info
norepsample	NOIMAI	IIIO
5m10s	Normal	Info
norepsample		-
5m9s	Normal	Info
norepsample		
5m9s	Normal	Info
norepsample		
4m59s	Normal	Info
norepsample	NT]	T in G i
4m58s	Normal	Info
norepsample 4m58s	Normal	Info
norepsample	NOTINAL	11110
4m58s	Normal	StateChange
norepsample		2
4m58s	Normal	StateChange
norepsample		
Initializin	g, now Nori	mal
4m58s	Normal	StateChange
norepsample		
now SomeRep	licasReady	
4m48s	Normal	Info
	NOTINAL	
norepsample		-
norepsample 4m48s	Normal	Info
norepsample 4m48s norepsample	Normal	Info
norepsample 4m48s norepsample 4m48s		-
norepsample 4m48s norepsample	Normal	Info
norepsample 4m48s norepsample 4m48s norepsample 4m48s	Normal	Info Info
norepsample 4m48s norepsample 4m48s norepsample	Normal	Info Info
norepsample 4m48s norepsample 4m48s norepsample 4m48s norepsample	Normal Normal Normal	Info Info Info
norepsample 4m48s norepsample 4m48s norepsample 4m48s 4m48s	Normal Normal Normal	Info Info Info
norepsample 4m48s norepsample 4m48s norepsample 4m48s norepsample 4m48s norepsample	Normal Normal Normal Normal	Info Info Info Info
norepsample 4m48s norepsample 4m48s norepsample 4m48s norepsample 4m48s norepsample 3m37s	Normal Normal Normal Normal	Info Info Info Info
norepsample 4m48s norepsample 4m48s norepsample 4m48s norepsample 4m48s norepsample 4m48s norepsample 3m37s norepsample	Normal Normal Normal Normal Normal	Info Info Info Info Info
norepsample 4m48s norepsample 4m48s norepsample 4m48s norepsample 4m48s norepsample 4m48s norepsample 3m37s norepsample 3m37s	Normal Normal Normal Normal	Info Info Info Info
norepsample 4m48s norepsample 4m48s norepsample 4m48s norepsample 4m48s norepsample 3m37s norepsample 3m37s norepsample	Normal Normal Normal Normal Normal Normal	Info Info Info Info Info Info
norepsample 4m48s norepsample 4m48s norepsample 4m48s norepsample 4m48s norepsample 3m37s norepsample 3m37s norepsample 3m37s	Normal Normal Normal Normal Normal	Info Info Info Info Info
norepsample 4m48s norepsample 4m48s norepsample 4m48s norepsample 4m48s norepsample 3m37s norepsample 3m37s norepsample	Normal Normal Normal Normal Normal Normal	Info Info Info Info Info Info Info Info
norepsample 4m48s norepsample 4m48s norepsample 4m48s norepsample 4m48s norepsample 3m37s norepsample 3m37s norepsample 3m37s norepsample	Normal Normal Normal Normal Normal Normal Normal	Info Info Info Info Info Info
norepsample 4m48s norepsample 4m48s norepsample 4m48s norepsample 4m48s norepsample 3m37s norepsample 3m37s norepsample 3m37s norepsample 3m37s	Normal Normal Normal Normal Normal Normal Normal	Info Info Info Info Info Info Info Info
norepsample 4m48s norepsample 4m48s norepsample 4m48s norepsample 4m48s norepsample 3m37s norepsample 3m37s norepsample 3m37s norepsample 3m37s norepsample	Normal Normal Normal Normal Normal Normal Normal Normal	Info Info Info Info Info Info Info StateChange
norepsample 4m48s norepsample 4m48s norepsample 4m48s norepsample 4m48s norepsample 4m48s norepsample 3m37s norepsample 3m37s norepsample 3m37s norepsample 3m37s	Normal Normal Normal Normal Normal Normal Normal Normal Normal	Info Info Info Info Info Info Info StateChange StateChange
norepsample 4m48s norepsample 4m48s norepsample 4m48s norepsample 4m48s norepsample 4m48s norepsample 3m37s norepsample 3m37s norepsample 3m37s norepsample 3m37s norepsample 3m37s norepsample	Normal Normal Normal Normal Normal Normal Normal Normal Normal	Info Info Info Info Info Info Info StateChange StateChange
norepsample 4m48s norepsample 4m48s norepsample 4m48s norepsample 4m48s norepsample 4m48s norepsample 3m37s norepsample 3m37s norepsample 3m37s norepsample 3m37s norepsample 3m37s norepsample 3m37s norepsample	Normal Normal Normal Normal Normal Normal Normal Normal Normal Normal	Info Info Info Info Info Info Info StateChange StateChange mal Info
norepsample 4m48s norepsample 4m48s norepsample 4m48s norepsample 4m48s norepsample 4m48s norepsample 3m37s norepsample 3m37s norepsample 3m37s norepsample 3m37s norepsample 3m37s norepsample 3m37s norepsample 3m37s	Normal Normal Normal Normal Normal Normal Normal Normal Normal g, now Norm	Info Info Info Info Info Info Info StateChange StateChange
norepsample 4m48s norepsample 4m48s norepsample 4m48s norepsample 4m48s norepsample 4m48s norepsample 3m37s norepsample 3m37s norepsample 3m37s norepsample 3m37s norepsample 3m37s norepsample 3m37s norepsample	Normal Normal Normal Normal Normal Normal Normal Normal Normal Normal	Info Info Info Info Info Info Info StateChange StateChange mal Info

	timestenclassic/
Pod	norepsample-0 Agent Up
	timestenclassic/
Pod	norepsample-0 Release 22.1.1.34.0
	timestenclassic/
Pod	norepsample-0 Instance Exists
	timestenclassic/
Pod	norepsample-0 Daemon Down
	timestenclassic/
Pod	norepsample-0 Daemon Up
	timestenclassic/
Pod	norepsample-0 Database None
	timestenclassic/
Pod	norepsample-0 Database Loaded
	timestenclassic/
Pod	norepsample-0 Database Updatable
	timestenclassic/
Pod	norepsample-0 RepAgent Not Running
	timestenclassic/
Pod	norepsample-0 is Ready
	timestenclassic/
Pod	norepsample-0 state was

timestenclassic/ TimesTenClassic was NoReplicasReady,

```
timestenclassic/
Pod norepsample-2 Agent Up
           timestenclassic/
Pod norepsample-2 Release 22.1.1.34.0
           timestenclassic/
Pod norepsample-2 Instance Exists
           timestenclassic/
Pod norepsample-2 Daemon Down
           timestenclassic/
Pod norepsample-2 Daemon Up
           timestenclassic/
Pod norepsample-2 Database None
           timestenclassic/
Pod norepsample-2 Database Loaded
           timestenclassic/
Pod norepsample-2 Database Updatable
           timestenclassic/
Pod norepsample-2 RepAgent Not Running
           timestenclassic/
Pod norepsample-2 is Ready
           timestenclassic/
Pod norepsample-2 state was
           timestenclassic/
```

Pod norepsample-1 Agent Up timestenclassic/ Pod norepsample-1 Release 22.1.1.34.0 timestenclassic/ Pod norepsample-1 Instance Exists

norepsample

3m37s	Normal	Info	timestenclassic/
norepsample			Pod norepsample-1 Daemon Down
3m36s	Normal	Info	timestenclassic/
norepsample			Pod norepsample-1 Daemon Up
3m36s	Normal	Info	timestenclassic/
norepsample			Pod norepsample-1 Database None
3m30s	Normal	Info	timestenclassic/
norepsample			Pod norepsample-1 Database Loaded
3m30s	Normal	Info	timestenclassic/
norepsample			Pod norepsample-1 Database Updatable
3m30s	Normal	Info	timestenclassic/
norepsample			Pod norepsample-1 RepAgent Not Running
3m30s	Normal	StateChange	timestenclassic/
norepsample			Pod norepsample-1 is Ready
3m30s	Normal	StateChange	timestenclassic/
norepsample			Pod norepsample-1 state was
Initializin	g, now Nor	rmal	
3m30s	Normal	StateChange	timestenclassic/
norepsample			TimesTenClassic was SomeReplicasReady,
now AllRepl	icasReady		

During the provisioning process, the TimesTenClassic object and TimesTen Pods transition from and to various states.

5. Verify the existence of the underlying objects.

1. StatefulSet:

kubectl get statefulset norepsample

The output is similar to the following:

NAME	READY	AGE
norepsample	3/3	8m30s

2. Service:

kubectl get service norepsample

The output is similar to the following:

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)
AGE				
norepsample	ClusterIP	None	<none></none>	6625/TCP,8888/TCP
8m58s				

3. Pods:

kubectl get pods



The output is similar to the following:

NAME	READY	STATUS	RESTARTS	AGE
norepsample-0	3/3	Running	0	9m15s
norepsample-1	3/3	Running	0	9m15s
norepsample-2	3/3	Running	0	9m15s

4. PVCs:

kubectl get pvc

The output is similar to the following:

NAME	STATUS		
VOLUME		CAPACITY	ACCESS MODES
STORAGECLASS AGE			
tt-persistent-norepsample-0	Bound	csi-00bf2a2e-e	£70-4£41-
a3ef-66341f7404ac 250Gi	RWO	oci-bv	9m34s
tt-persistent-norepsample-1	Bound	csi-516d9ef4-9	492-4927-
b483-05f7d320cc73 250Gi	RWO	oci-bv	9m34s
tt-persistent-norepsample-2	Bound	csi-99c95d75-	
f9d9-4754-9378-0c5a7d288871	250Gi	RWO	oci-bv
9m34s			

6. Connect to one of the databases.

a. From your development host, establish a shell in the Pod.

kubectl exec -it norepsample-0 -c tt -- /bin/bash

b. Connect to the database. Insert a row in a table.

ttIsql norepsample

Copyright (c) 1996, 2025, Oracle and/or its affiliates. All rights reserved. Type ? or "help" for help, type "exit" to quit ttIsql.

```
connect "DSN=norepsample";
Connection successful: DSN=norepsample;UID=timesten;DataStore=/tt/home/
timesten/datastore/
norepsample;DatabaseCharacterSet=AL32UTF8;ConnectionCharacterSet=US7ASCI
I;AutoCreate=0;PermSize=200;DDLReplicationLevel=3;ForceDisconnectEnabled
=1;
(Default setting AutoCommit=1)
Command> select * from adminuser.emp;
0 rows found.
Command> describe adminuser.emp;
Table ADMINUSER.EMP:
Columns:
*ID NUMBER NOT NULL
```



```
NAME CHAR (32)

PRIMARY KEY (ID) RANGE INDEX

1 table found.

(primary key columns are indicated with *)

Command> insert into adminuser.emp values (1,'test');

1 row inserted.

Command> commit;

Command> commit;

Command> select * from adminuser.emp;

< 1, test >

1 row found.
```

c. Exit from ttIsql and from the shell.

```
Command> exit
Disconnecting...
Done.
[timesten@norepsample-0 ~]$ exit
exit
```

Congratulations! You successfully created and deployed a non-replicated TimesTenClassic object. The non-replicated TimesTen Classic databases are up and running and fully operational.

Modify the Number of Replicas in Non-Replicated Environments

There may be situations where you want to increase or decrease the number of databases in your non-replicated environment. For example, you may want to increase the resources during a peak holiday buying season and then reduce the resources later. You can accomplish this by modifying the number of replicas for a TimesTenClassic object.

When you increase the number of replicas, Kubernetes provisions TimesTen in the new Pods.

When you decrease the number of replicas:

- Kubernetes deletes the relevant Pods.
- The TimesTen Operator does not delete the PVCs.

Let's walk through an example. In this example, a TimesTenClassic object is deployed in your namespace. The number of replicas is 3. The example shows you how to decrease the number of replicas to 1 and then increase the number of replicas back to 3.

1. Confirm there is a TimesTenClassic object deployed in your namespace.

kubectl get ttc

The output is similar to the following:

NAME	STATE	ACTIVE	AGE
norepsample	AllReplicasReady	N/A	17m

The norepsample TimesTenClassic object is deployed and is in the AllReplicasReady state. All replicas are available and running and the databases are functioning properly.



2. Confirm the number of replicas.

kubectl describe ttc norepsample | grep 'Replicas'

The output is similar to the following:

Replicas:

The number of replicas is 3.

3. Confirm the number of Pods.

kubectl get pods

The output is similar to the following

NAME	READY	STATUS	RESTARTS	AGE
norepsample-0	3/3	Running	0	18m
norepsample-1	3/3	Running	0	18m
norepsample-2	3/3	Running	0	18m

3

There are three Pods running, corresponding to the number of replicas. Each replica contains a TimesTen database.

- 4. Insert data into databases.
 - a. Database running in the norepsample-0 Pod:

```
kubectl exec -it norepsample-0 -c tt -- /bin/bash
[timesten@norepsample-0 ~]$ ttIsql norepsample
```

```
Copyright (c) 1996, 2025, Oracle and/or its affiliates. All rights reserved.
Type ? or "help" for help, type "exit" to quit ttIsql.
```

```
connect "DSN=norepsample";
Connection successful: DSN=norepsample;UID=timesten;DataStore=/tt/home/
timesten/datastore/
norepsample; DatabaseCharacterSet=AL32UTF8; ConnectionCharacterSet=US7ASCI
I;AutoCreate=0;PermSize=200;DDLReplicationLevel=3;ForceDisconnectEnabled
=1;
(Default setting AutoCommit=1)
Command> CREATE TABLE testdata (col1 TT INTEGER);
Command> INSERT INTO testdata values (0);
1 row inserted.
Command> SELECT * FROM testdata;
< 0 >
1 row found.
Command> exit
Disconnecting...
Done.
```



```
[timesten@norepsample-0 ~]$ exit
exit
```

There is one row in the testdata table and the value for coll is 0.

b. Database running in the norepsample-1 Pod:

```
kubectl exec -it norepsample-1 -c tt -- /bin/bash
[timesten@norepsample-0 ~]$ ttIsql norepsample
Copyright (c) 1996, 2025, Oracle and/or its affiliates. All rights
reserved.
Type ? or "help" for help, type "exit" to quit ttIsql.
```

```
connect "DSN=norepsample";
Connection successful: DSN=norepsample;UID=timesten;DataStore=/tt/home/
timesten/datastore/
norepsample;DatabaseCharacterSet=AL32UTF8;ConnectionCharacterSet=US7ASCI
I;AutoCreate=0;PermSize=200;DDLReplicationLevel=3;ForceDisconnectEnabled
=1;
(Default setting AutoCommit=1)
Command> CREATE TABLE testdata (col1 TT INTEGER);
Command> INSERT INTO testdata values (1);
1 row inserted.
Command> SELECT * FROM testdata;
<1>
1 row found.
Command> exit
Disconnecting...
Done.
[timesten@norepsample-1 ~]$ exit
exit
```

There is one row in the testdata table and the value for coll is 1.

c. Database running in the norepsample-2 Pod:

```
kubectl exec -it norepsample-2 -c tt -- /bin/bash
[timesten@norepsample-0 ~]$ ttIsql norepsample
```

Copyright (c) 1996, 2025, Oracle and/or its affiliates. All rights reserved. Type ? or "help" for help, type "exit" to quit ttIsql.

```
connect "DSN=norepsample";
Connection successful: DSN=norepsample;UID=timesten;DataStore=/tt/home/
timesten/datastore/
norepsample;DatabaseCharacterSet=AL32UTF8;ConnectionCharacterSet=US7ASCI
I;AutoCreate=0;PermSize=200;DDLReplicationLevel=3;ForceDisconnectEnabled
=1;
(Default setting AutoCommit=1)
```



```
Command> CREATE TABLE testdata (coll TT_INTEGER);
Command> INSERT INTO testdata values (2);
1 row inserted.
Command> SELECT * FROM testdata;
< 2 >
1 row found.
Command> exit
Disconnecting...
Done.
[timesten@norepsample-2 ~]$ exit
exit
```

There is one row in the testdata table and the value for coll is 2.

- 5. Decrease the number of replicas.
 - a. Edit the norepsample TimesTenClassic object, changing replicas to 1.

```
kubectl edit ttc norepsample
# Please edit the object below. Lines beginning with a '#' will be
ignored,
# and an empty file will abort the edit. If an error occurs while
saving this file will be
# reopened with the relevant failures.
#
apiVersion: timesten.oracle.com/v4
kind: TimesTenClassic
...
spec:
    ttspec:
    ...
    replicas: 1
...
```

b. Save your edit.

The output is similar to the following:

timestenclassic.timesten.oracle.com/norepsample edited

6. Verify the TimesTenClassic object is in the AllReplicasReady state.

kubectl get ttc

The output is similar to the following:

NAME	STATE	ACTIVE	AGE
norepsample	AllReplicasReady	N/A	63m

7. Confirm the number of Pods.

kubectl get pods



The output is similar to the following:

NAME	READY	STATUS	RESTARTS	AGE
norepsample-0	3/3	Running	0	65m

There is one Pod. One TimesTen database is running in the Pod.

8. Confirm the number of PVCs.

kubectl get pvc

The output is similar to the following:

NAME	STATUS			
VOLUME		CAPACITY	ACCESS MODES	
STORAGECLASS AGE				
tt-persistent-norepsample-0	Bound	csi-836f3962-		
cbfa-4f7d-9271-6393167b00bd	50Gi	RWO	oci-bv	67m
tt-persistent-norepsample-1	Bound	csi-b2bfa486-6	5d09-463f-	
a8a3-25a760d1449d 50Gi	RWO	oci-bv	67m	
tt-persistent-norepsample-2	Bound	csi-820a206b-1	662-4c24-821c-	
a947572b618a 50Gi RV	10	oci-bv	67m	

There are three PVCs. When you modify the number of replicas, the TimesTen Operator does not delete the PVCs associated with a TimesTenClassic object.

- 9. Increase the number of replicas.
 - a. Edit the norepsample TimesTenClassic object, changing replicas to 3.

```
kubectl edit ttc norepsample
# Please edit the object below. Lines beginning with a '#' will be
ignored,
# and an empty file will abort the edit. If an error occurs while
saving this file will be
# reopened with the relevant failures.
#
apiVersion: timesten.oracle.com/v4
kind: TimesTenClassic
...
spec:
...
replicas: 3
...
```

b. Save your edit.

The output is similar to the following:

timestenclassic.timesten.oracle.com/norepsample edited



10. Verify the TimesTenClassic object is in the AllReplicasReady state.

kubectl get ttc

The output is similar to the following:

NAME	STATE	ACTIVE	AGE
norepsample	AllReplicasReady	N/A	76m

11. Confirm the number of Pods.

kubectl get pods

The output is similar to the following:

NAME	READY	STATUS	RESTARTS	AGE
norepsample-0	3/3	Running	0	77m
norepsample-1	3/3	Running	0	4m
norepsample-2	3/3	Running	0	4m

There are three Pods. Each Pod is running a TimesTen database.

12. Confirm the number of PVCs.

kubectl get pvc

The output is similar to the following:

```
NAME
                          STATUS
VOLUME
                                      CAPACITY ACCESS MODES
STORAGECLASS AGE
tt-persistent-norepsample-0 Bound csi-836f3962-
cbfa-4f7d-9271-6393167b00bd 50Gi
                                  RWO
                                                 oci-bv
                                                               79m
tt-persistent-norepsample-1 Bound csi-b2bfa486-6d09-463f-
                          RWO
                                                     79m
a8a3-25a760d1449d
                 50Gi
                                        oci-bv
tt-persistent-norepsample-2 Bound csi-820a206b-1662-4c24-821c-
a947572b618a 50Gi
                                  oci-bv
                                                 79m
                  RWO
```

There are three PVCs associated with the three TimesTen databases.

- **13.** Verify the data in the databases.
 - a. Database running in the norepsample-0 Pod:

```
kubectl exec -it norepsample-0 -c tt -- /bin/bash
[timesten@norepsample-0 ~]$ ttIsql norepsample
Copyright (c) 1996, 2025, Oracle and/or its affiliates. All rights
reserved.
Type ? or "help" for help, type "exit" to quit ttIsql.
```

connect "DSN=norepsample";



```
Connection successful: DSN=norepsample;UID=timesten;DataStore=/tt/home/
timesten/datastore/
norepsample;DatabaseCharacterSet=AL32UTF8;ConnectionCharacterSet=US7ASCI
I;AutoCreate=0;PermSize=200;DDLReplicationLevel=3;ForceDisconnectEnabled
=1;
(Default setting AutoCommit=1)
Command> tables;
TIMESTEN.TESTDATA
1 table found.
Command> SELECT * FROM testdata;
< 0 >
1 row found.
```

Since this Pod was not deleted when you decreased the number of replicas, the data in the database has not changed.

b. Database running in the norepsample-1 Pod:

kubectl exec -it norepsample-1 -c tt -- /bin/bash
[timesten@norepsample-1 ~]\$ ttIsql norepsample

Copyright (c) 1996, 2025, Oracle and/or its affiliates. All rights reserved. Type ? or "help" for help, type "exit" to quit ttIsql.

```
connect "DSN=norepsample";
Connection successful: DSN=norepsample;UID=timesten;DataStore=/tt/home/
timesten/datastore/
norepsample;DatabaseCharacterSet=AL32UTF8;ConnectionCharacterSet=US7ASCI
I;AutoCreate=0;PermSize=200;DDLReplicationLevel=3;ForceDisconnectEnabled
=1;
(Default setting AutoCommit=1)
Command> tables;
0 tables found.
Command> exit
Disconnecting...
Done.
[timesten@norepsample-1 ~]$ exit
exit
```

The TimesTen Operator correctly provisions TimesTen in the norepsample-1 Pod and has correctly overwritten the original database.

c. Database running in the norepsample-2 Pod:

```
kubectl exec -it norepsample-2 -c tt -- /bin/bash
[timesten@norepsample-2 ~]$ ttIsql norepsample
Copyright (c) 1996, 2025, Oracle and/or its affiliates. All rights
reserved.
Type ? or "help" for help, type "exit" to quit ttIsql.
```



```
connect "DSN=norepsample";
Connection successful: DSN=norepsample;UID=timesten;DataStore=/tt/home/
timesten/datastore/
norepsample;DatabaseCharacterSet=AL32UTF8;ConnectionCharacterSet=US7ASCI
I;AutoCreate=0;PermSize=200;DDLReplicationLevel=3;ForceDisconnectEnabled
=1;
(Default setting AutoCommit=1)
Command> tables;
0 tables found.
Command> tables;
0 tables found.
Command> exit
Disconnecting...
Done.
[timesten@norepsample-2 ~]$ exit
exit
```

The TimesTen Operator correctly provisions TimesTen in the norepsample-2 Pod and has correctly overwritten the original database.

Congratulations! You successfully modified the number of replicas for a TimesTenClassic object in a non-replicated environment.



8 Deploy TimesTen Scaleout Databases

The TimesTen Operator supports the deployment of a TimesTen Scaleout grid and associated databases in your Kubernetes cluster.

This chapter provides background information about the deployment process. It summarizes the planning process for configuring a grid and shows you how to apply that plan when configuring a grid in the Kubernetes environment. It describes the steps the TimesTen Operator takes to deploy and provision a grid based on the information you provide. Use this information to gain an understanding of how the Operator functions when deploying your TimesTen Scaleout grid.

The second part of the chapter provides you with an end-to-end example that shows you the steps to deploy your grid in the Kubernetes cluster.

If you want to advance directly to the example, see Deploy a Grid.

The TimesTen Operator also supports TimesTen Cache in TimesTen Scaleout. See Work with TimesTen Cache.

Note:

You cannot use Helm to deploy TimesTen Scaleout databases. Instead, use the procedures in this chapter.

Topics:

- About Deploying a Grid
- Deploy a Grid

About Deploying a Grid

The information in this section provides background information about configuring and deploying a grid in the Kubernetes environment.

Topics:

- About Planning a Grid
- About Configuring a Grid
- About Provisioning a Grid
- About ssh
- About Creating a Grid



About Planning a Grid

One of the features of the TimesTen Operator is the ability for it to provision a TimesTen Scaleout grid and its database in the Kubernetes cluster. Just like in any other TimesTen Scaleout environment, you must do some planning for your grid.

Here are some considerations:

- K-Safety (represented by k): How many copies of your TimesTen database? The K-Safety factor determines the number of data spaces in your grid. For example, if you set k to 2, there are two copies of your database: one copy resides in data space one and the second copy in data space two.
- Replica sets: How many replica sets in your grid? A replica set contains k elements, where each element in the replica set is an exact copy of the other elements in the replica set. The value of k, combined with the number of replica sets, determines the number of data instances in the grid. For example, if you set k to 2, and replica sets to 3, then there are six data instances in the grid.
- ZooKeeper instances: How many ZooKeeper instances for the grid?
- Database definition file (DbDef): What data store and first connection attributes are needed for the database in the grid?
- Direct connectable: What general connection attributes are needed for the database when using direct access?
- Client/server connectable: What general connection attributes are needed for the database when using client/server access?

After you define your configuration, you provide that information to the TimesTen Operator. The Operator takes over from there. It automatically does the provisioning and the deployment of the grid and database for you.

About Configuring a Grid

The TimesTen Operator provides the TimesTenScaleout object type and configuration metadata so that you can define and then deploy your TimesTen Scaleout grid and database. The TimesTenScaleout object type provides the syntax for configuring your grid. The configuration metadata lets you define the connection attributes for your database. Taken together, the Operator has the necessary information to provision your grid and database in the Kubernetes cluster.

Based on your planned configuration in About Planning a Grid, you can apply that configuration to the Kubernetes environment:

- K-Safety (k): The Operator provides the .spec.ttspec.k element as part of the syntax for the TimesTenScaleout object type. You specify k in the YAML manifest file for your TimesTenScaleout object.
- Replica sets: The Operator provides the .spec.ttspec.nReplicaSets element as part of the syntax for the TimesTenScaleout object type. You specify nReplicaSets in the YAML manifest file for your TimesTenScaleout object.
- ZooKeeper instances: The Operator provides the .spec.ttspec.nZookeeper element as part of the syntax for the TimesTenScaleout object type. You specify nZookeeper in the YAML manifest file for your TimesTenScaleout object.



- Database definition file (DbDef): The Operator creates a DbDef from the contents of the db.ini metadata file. You create this metadata file and then use a Kubernetes facility (or some other means) to place this file in the /ttconfig directory of the tt containers.
- One or more direct connectables: The Operator creates one or more direct connectables from the contents of the *.connect metadata files. You create one or more of these *.connect files and then use a Kubernetes facility (or some other means) to place the files in the /ttconfig directory of the tt containers.
- One or more client/server connectables: The Operator creates one or more client/server connectables from the contents of the *.csconnect metadata files. You create one or more of these *.csconnect files and then use a Kubernetes facility (or some other means) to place the files in the /ttconfig directory of the tt containers.

See TimesTenScaleoutSpecSpec, List of Configuration Metadata, and Populate the /ttconfig Directory.

After the metadata files are placed in the /ttconfig directory of the tt containers, and you configure and then deploy your TimesTenScaleout object in the Kubernetes cluster, Kubernetes informs the Operator that a TimesTenScaleout object has been created. The Operator begins the process of creating additional Kubernetes objects in order to implement your grid.

About Provisioning a Grid

The TimesTen Operator gathers the information from the TimesTenScaleout object and begins instantiating the TimesTen Scaleout grid and database. It creates the following StatefulSets:

- One StatefulSet that provides the management instance for the grid. The Operator supports one management instance. The underlying Pod for this management instance is also created.
- One StatefulSet that provides one or more ZooKeeper instances. The Operator determines the number of ZooKeeper instances by the value you specified for the nZookeeper field in your TimesTenScaleout object definition. For example, if you specified a value of 3 for nZookeeper, the Operator creates one StatefulSet with three replicas. The underlying Pods for these ZooKeeper instances are also created. There is one Pod per ZooKeeper instance.
- Additional StatefulSets, the number of which is determined by the value you specified for the k field in your TimesTenScaleout object definition. For example, if you specified a value of 2 for k, the Operator creates two StatefulSets. These StatefulSets provide data instances for the grid. Each of the k StatefulSets provides Pods to implement a single data space in the resultant grid. The StatefulSet has M replicaSets field in your TimesTenScaleout object definition. For example, if you set nReplicaSets to 3, each StatefulSet has three replicas. This number of replicas determines the number of replica sets in the resultant grid.

In the preceding example, one StatefulSet has three replicas. This one StatefulSet contains three data instances in data space one. A Pod is created for each data instance, so there are three Pods created. The second StatefulSet contains three data instances in data space two. A Pod is created for each data instance, so there are three Pods created for each data instance, so there are three Pods created. There are a total of six total Pods created for the six data instances.

In addition, the Operator creates the following Kubernetes headless Services:



- One headless Service that provides DNS names for the Pods that contain the management and data instances. This service enables client/server access to the Pods using the TimesTen client/server port 6625.
- One headless Service that provides DNS names for the Pods that contain the ZooKeeper instances. This service enables access to the ZooKeeper internal ports 2888 and 3888, as well as the external port 2181.

There is an example of these StatefulSets and headless Services in Verify Underlying Objects.

The Operator also creates Persistent Volume Claims (PVCs) for the Pods. These PVCs cause persistent volumes to be allocated by Kubernetes and attached to the TimesTen Pods and ZooKeeper Pods. See Persistent Storage.

Pods that run ZooKeeper consists of a single container called <code>zookeeper</code>. Each Pod that is running TimesTen consists of at least two containers. The <code>tt</code> container runs the TimesTen agent and TimesTen. The <code>daemonlog</code> container writes the TimesTen daemon log to stdout.

As the tt containers in the TimesTen Pods start, the Operator assembles them into a working grid. The grid's model is configured with several objects, including:

- Hosts for the data instances in each of the data space groups
- Each host configured with an installation of TimesTen
- Each host configured with a single TimesTen instance
- A DbDef (the contents of which are from the db.ini file)
- Direct mode connectables, if any, (the contents of which are from the *.connect files)
- Client/server connectables, if any, (the contents of which are from the *.csconnect files)

About ssh

A TimesTen Scaleout grid relies on password-less ssh among the instances of the grid. The TimesTen Operator automatically configures password-less ssh among the tt containers in the grid in your Kubernetes environment. There is no intervention that you need to do.

About Creating a Grid

When creating the grid, the TimeTen Operator transitions to and from various High Level states. Here is an explanation of these states:

After you create your TimesTenScaleout object in the Kubernetes cluster, the TimesTen Operator creates the StatefulSets and Services that are required to deploy your TimesTenScaleout grid and database. The Operator assigns a High Level state of Initializing to the TimesTenScaleout object.

The Operator periodically polls the status of the StatefulSets' objects and their underlying Pods. When the ZooKeeper Pods are ready, the TimesTenScaleout object transitions from the Initializing state to the ZooKeeperReady state.

When the TimesTen agent in the management Pod starts up, the Operator instructs the agent to create a TimesTen instance and grid. The TimesTenScaleout object transitions to the GridCreated state.

The Operator waits for all of the TimesTen agents in all of the Pods to start. Once all have started, the Operator instructs the TimesTen agent in the management instance to create the hosts, the installations, and the instances in the grid's model for the data instances in the grid.



The DbDef is then created from the contents of the db.ini file. The direct connectables are created from the contents of the *.connect files. The client/server connectables are created from the contents of the *.csconnect files.

The model is applied and the data instances of the grid are created. The TimesTenScaleout object transitions to the InstanceCreated state.

The Operator then instructs the management instance to create the database (by using the TimesTen ttGridAdmin utility with the dbCreate option) and to create the initial distribution map (by using the TimesTen ttGridAdmin utility with the dbDistribute -add all -apply options). The TimesTenScaleout object then transitions to the DbCreated state.

The Operator then instructs the TimesTen agent in one data instance to run the TimesTen ttIsql utility to create the user in the adminUser file and run the schema.sql file (if you provided these files). The TimesTenScaleout object transitions to the Normal state.

The grid and databases are created. The TimesTen agent then opens the database for connections.

The Operator manages and monitors the TimesTenScaleout objects after they are deployed in your Kubernetes cluster. The Operator also detects, repairs, and recovers from failures in your grid and associated databases. See Manage TimesTen Scaleout.

There is an example showing the Operator transitioning to and from these High Level states in Monitor the High Level State of a TimesTenScaleout Object.

Deploy a Grid

This section provides a step by step walk-through that shows you how to create and deploy a TimesTen Scaleout grid and database in your Kubernetes cluster. The walk-through starts with an example that shows you how to create metadata files and create a Kubernetes ConfigMap. It continues with an example that shows you how to create a YAML file that contains the definitions for your TimesTenScaleout object. It shows you how to deploy that YAML file in your Kubernetes cluster. You learn how to monitor the provisioning of the TimesTen grid and database, and verify the underlying Kubernetes objects were created by the TimesTen Operator. The walk-through completes with one example that shows you how to connect to the TimesTen database and run operations in it. The final example shows you how to connect to the management instance to verify the health of the database and its elements.

- Create Configuration Metadata and a Kubernetes ConfigMap for a Grid
- Define and Deploy a TimesTenScaleout Object
- Monitor the High Level State of a TimesTenScaleout Object
- Verify Underlying Objects
- Connect to the Database
- Manage a Grid and Its Database

Create Configuration Metadata and a Kubernetes ConfigMap for a Grid

The following example creates the db.ini, the adminUser, and the schema.sql metadata files for the TimesTen grid and the database. The example also creates a direct and a client/server connectable. The example creates a Kubernetes ConfigMap to place these metadata files into the /ttConfig directory of the TimesTen containers.

Note:

You can use any Kubernetes mechanism to place these metadata files into the / ttConfig directory of the TimesTen containers. See Populate the /ttconfig Directory.

On your development host, complete the following steps:

From the directory of your choice, create an empty subdirectory for the metadata files. This
example creates the cm_scaleout subdirectory. (The cm_scaleout directory is used in the
remainder of this example to denote this directory.)

```
mkdir -p cm_scaleout
```

2. Change to this ConfigMap directory.

```
cd cm scaleout
```

 Create the db.ini file in this ConfigMap directory. In this example, define the PermSize and the DatabaseCharacterSet connection attributes.

vi db.ini PermSize=200 DatabaseCharacterSet=AL32UTF8

4. Create the adminUser file.

vi adminUser

adminuser/adminuserpwd

5. Create the schema.sql file. In this schema.sql file, create the sampleuser user, create the s sequence for the sampleuser user, and the emp table for the sampleuser user. The Operator automatically initializes each element of the TimesTen database with these object definitions.

```
vi schema.sql
create user sampleuser identified by sampleuserpwd;
grant admin to sampleuser;
create sequence sampleuser.s;
create table sampleuser.emp (id number not null primary key, name char
(32));
```

6. Create the sampledirect direct connectable.

vi sampledirect.connect

ConnectionCharacterSet=AL32UTF8

7. Create the sampleclient client/server connectable.

```
vi sampleclient.csconnect
```



```
ConnectionCharacterSet=AL32UTF8
TTC Timeout=30
```

8. Optional: Verify the metadata files are present in the cm_scaleout ConfigMap directory.

ls

The output is the following:

```
adminUser
db.ini
sampleclient.csconnect
sampledirect.connect
schema.sql
```

9. From the cm_scaleout directory, create the samplescaleout ConfigMap. The files in the cm_scaleout directory are included in the ConfigMap. These files are later available in the TimesTen containers. Replace samplescaleout with a name of your choosing.

kubectl create configmap samplescaleout --from-file .

The output is the following:

configmap/samplescaleout created

10. Use the kubectl describe command to verify the contents of the samplescaleout ConfigMap.

kubectl describe configmap samplescaleout

The output is the following:

```
samplescaleout
Name:
Namespace: mynamespace
Labels: <none>
Annotations: <none>
Data
====
adminUser:
____
adminuser/adminuserpwd
db.ini:
____
PermSize=200
DatabaseCharacterSet=AL32UTF8
sampleclient.csconnect:
____
ConnectionCharacterSet=AL32UTF8
TTC Timeout=30
```



```
sampledirect.connect:
----
ConnectionCharacterSet=AL32UTF8
schema.sql:
----
create user sampleuser identified by sampleuserpwd;
grant admin to sampleuser;
create sequence sampleuser.s;
create table sampleuser.emp (id number not null primary key, name char
(32));
Events: <none>
```

You successfully created the metadata files and the ConfigMap.

Define and Deploy a TimesTenScaleout Object

Defining your environment involves creating your TimesTenScaleout object with attributes customized for your environment. These attributes are described in the steps below. For additional information on defining objects of type TimesTenScaleout, see TimesTen Scaleout.

To define and create the TimesTenScaleout object, complete the following steps:

- 1. Create a YAML file. You can choose any name for this YAML file, but you may want to use the same name you used for the name of the TimesTenScaleout object. (This example uses samplescaleout.) The YAML file contains the definitions for the TimesTenScaleout object. In this example, the fields that are specific to a TimesTenScaleout object are as follows:
 - k: Set the value of k to the number of copies of data for your TimesTen database. This value determines the number of StatefulSets that the TimesTen Operator creates. A StatefulSet provides the Pods that are used to implement a single data space in the grid. For example, if you set k to 2, the Operator creates two StatefulSets. One StatefulSet provides the Pods for the data instances in data space one. The second StatefulSet provides the Pods for the data instances in data space two.

This example sets k to 2.

For information on K-safety and determining an appropriate value for k, see K-Safety in the Oracle TimesTen In-Memory Database Scaleout User's Guide.

nReplicaSets: Set the value to the number of replica sets in the grid. A replica set contains k elements (where each element is an exact copy of the other elements in the replica set). The nReplicaSets value determines the number of replicas for each StatefulSet. For example, if you set k to 2, the TimesTen Operator creates two StatefulSets for the data instances. If you set nReplicaSets to 3, each StatefulSet contains three replicas, and the total number of replica sets in the database is three.

This example sets nReplicaSets to 3.

For information on replica sets, see Understanding Replica Sets in the Oracle TimesTen In-Memory Database Scaleout User's Guide.

• nZookeeper: Set the value to the number of ZooKeeper Pods to provision in a StatefulSet. Your options are 1 or 3 or 5.

This example sets nZookeeper to 3.

You also need to specify the following fields:



- name: Replace samplescaleout with the name of your TimesTenScaleout object.
- storageClassName: Replace oci-bv with the name of the storage class in your Kubernetes cluster that is used to allocate Persistent Volumes to hold the TimesTen database.
- storageSize: Replace 250Gi with the amount of storage that should be requested for each Pod to hold TimesTen. (This example assumes a production environment and uses 250Gi for storage. For demonstration purposes, you can use 50Gi of storage.) See the storageSize and the logStorageSize entries in Table 20-8.
- image: Replace container-registry.oracle.com/timesten/timesten:22.1.1.34.0 with the location of the image registry and the name of the image. If you are using the Oracle container-registry.oracle.com/timesten registry as the image registry and the timesten:22.1.1.34.0 image as the container image, no replacement is necessary.
- imagePullSecret: Replace sekret with the image pull secret that Kubernetes should use to fetch the TimesTen image.
- dbConfigMap: This example uses one ConfigMap (called samplescaleout) for the db.ini, the adminUser, the schema.sql, the sampledirect.connect, and the sampleclient.csconnect metadata files. This ConfigMap is included in the Projected Volume. The volume is mounted as /ttconfig in the TimesTen containers. See Using ConfigMaps and Secrets.

vi samplescaleout.yaml

```
kind: TimesTenScaleout
metadata:
name: samplescaleout
spec:
   ttspec:
   storageClassName: oci-bv
   storageSize: 250Gi
   image: container-registry.oracle.com/timesten/timesten:22.1.1.34.0
   imagePullSecret: sekret
   dbConfigMap:
      - samplescaleout
      k: 2
      nReplicaSets: 3
      nZookeeper: 3
```

 Create the TimesTenScaleout object from the contents of the YAML file. Doing so begins the process of creating and provisioning a TimesTen grid and database in your Kubernetes cluster.

```
kubectl create -f samplescaleout.yaml
```

The output is the following:

timestenscaleout.timesten.oracle.com/samplescaleout created

You successfully created the TimesTenScaleout object in the Kubernetes cluster. The process of provisioning your TimesTen grid and database begins, but is not yet complete.



Monitor the High Level State of a TimesTenScaleout Object

Use the kubectl get and the kubectl describe commands to monitor the progress of the grid and database creation.

Note:

For the <code>kubectl get timestenscaleout command</code>, you can alternatively specify <code>kubectl get tts</code>. When used in the <code>kubectl get command</code>, <code>timestenscaleout and tts are synonymous</code>, and return the same results. The first <code>kubectl get examples</code> in this chapter use <code>timestenscaleout</code>. For simplicity, the remaining examples in this book use <code>tts</code>.

 Review the High Level state of the TimesTenScaleout object. Use the kubectl get command and observe the STATE field. Notice that the value is Initializing. The Operator has created the Kubernetes StatefulSets and Services. The process to deploy and provision your grid and database has begun, but is not yet complete. As you issue additional kubectl get commands, observe how the TimesTenScaleout object transitions to different states. For more information on these states, see About Creating a Grid.

kubectl get tts samplescaleout

The output is similar to the following:

NAME samplescaleout	OVERALL Initializing	MGMT	CREATE	LOAD	OPEN	AGE 20s
kubectl get tts s	samplescaleout					
The output is simila	r to the following:					
NAME samplescaleout	OVERALL ZookeeperReady	MGMT	CREATE	LOAD	OPEN	AGE 2m48s
kubectl get tts s	samplescaleout					
The output is simila	r to the following:					
NAME samplescaleout	OVERALL GridCreated	MGMT (CREATE I	LOAD		AGE 3m58s

kubectl get tts samplescaleout



The output is similar to the following:

NAME samplescaleout	OVERALL InstancesCreated	MGMT	CREATE	LOAD	OPEN	AGE 6m4s	
kubectl get tts	samplescaleout						
The output is simil	ar to the following:						
NAME samplescaleout 6m10s	OVERALL MGMT DatabaseCreated	CREATE	LOAD			OPEN	AGE

2. Use the kubectl get command again to observe if the High Level state has transitioned from DatabaseCreated to Normal. A Normal state indicates the grid and database are provisioned and the process is complete.

kubectl get tts samplescaleout

The output is similar to the following:

NAME	OVERALL	MGMT	CREATE	LOAD	OPEN	AGE
samplescaleout	Normal	Normal	created	loaded-complete	open	
6m52s						

Your TimesTen Scaleout grid and database are successfully created and provisioned in your Kubernetes cluster. The database is open for connections.

Verify Underlying Objects

The TimesTen Operator creates the following objects for the samplescaleout TimesTenScaleout object.

1. StatefulSets:

```
kubectl get statefulsets
```

The output is the following:

NAME	READY	AGE
samplescaleout-data-1	3/3	26m
samplescaleout-data-2	3/3	26m
samplescaleout-mgmt	1/1	26m
samplescaleout-zk	3/3	26m

The Operator creates the samplescaleout-data-1 and the samplescaleout-data-2 StatefulSets. Two StatefulSets are created because the value of k is set to 2. Each StatefulSet provides the Pods for a single data space. There are two data spaces in the grid. Each of the StatefulSets has three replicas (nReplicaSets is set to 3). Therefore, the number of replica sets in the grid is three. There are three data instances in data space one and three data instances in data space two.



The samplescaleout-mgmt StatefulSet provides the Pod for the management instance. There is one management instance.

The samplscaleout-zk StatefulSet provides the Pods for the ZooKeeper instances. There are three ZooKeeper instances (nZookeeper is set to 3).

2. Pods:

kubectl get pods

The output is the following:

NAME	READY	STATUS	RESTARTS	AGE
samplescaleout-data-1-0	2/2	Running	0	27m
samplescaleout-data-1-1	2/2	Running	0	27m
samplescaleout-data-1-2	2/2	Running	0	27m
samplescaleout-data-2-0	2/2	Running	0	27m
samplescaleout-data-2-1	2/2	Running	0	27m
samplescaleout-data-2-2	2/2	Running	0	27m
samplescaleout-mgmt-0	2/2	Running	0	27m
samplescaleout-zk-0	1/1	Running	0	27m
samplescaleout-zk-1	1/1	Running	0	26m
samplescaleout-zk-2	1/1	Running	0	25m
timesten-operator-7677964df9	-sp2zp 1/1	Running	0	3d20h

The samplescaleout-data-1 StatefulSet contains the data instances in data space one. This StatefulSet creates the samplescaleout-data-1-0, samplescaleout-data-1-1, and samplescaleout-data-1-2 Pods. The samplescaleout-data-2 contains the data instances in data space two. This StatefulSet creates the samplescaleout-data-2-0, samplescaleout-data-2-1, and samplescaleout-data-2-2 Pods.

The samplescaleout-mgmt-0 StatefulSet contains the management instance for the grid. This StatefulSet creates the samplescaleout-mgmt-0 Pod.

Each of the Pods previously mentioned run TimesTen. Each Pod contains at least two containers. In this example, each Pod contains two containers. The tt container runs the TimesTen agent and TimesTen. The daemonlog container writes the TimesTen daemon log to stdout.

The samplescaleout-zk StatefulSet contains the three ZooKeepers instances. This StatefulSet creates the samplescaleout-zk-1, samplescaleout-zk-2, and samplescaleout-zk-3 Pods. Each of these Pods contain a single container called zookeeper.

The TimesTen Operator is running in the timesten-operator-554887b4c-48zwk Pod.

3. Headless Services:

kubectl get services

The output is the following:

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP
PORT(S)		AGE	
samplescaleout	ClusterIP	None	<none></none>
6625/TCP		28m	



samplescaleout-zk ClusterIP None <none> 2181/ TCP,2888/TCP,3888/TCP 28m

The Operator creates the samplescaleout Headless Service. This Service provides the DNS names for the Pods that contain the TimesTen management and data instances. The Service allows client/server access to the Pods that use the client/server port (6625). The DNS names are in the format: *podname*.samplescaleout.*namespace*.svc.cluster.local.

The following example shows the DNS names for the samplescaleout Headless Service:

```
samplescaleout-mgmt-0.samplescaleout.mynamespace.svc.cluster.local
samplescaleout-data-1-0.samplescaleout.mynamespace.svc.cluster.local
samplescaleout-data-1-1.samplescaleout.mynamespace.svc.cluster.local
samplescaleout-data-2-0.samplescaleout.mynamespace.svc.cluster.local
samplescaleout-data-2-1.samplescaleout.mynamespace.svc.cluster.local
samplescaleout-data-2-1.samplescaleout.mynamespace.svc.cluster.local
samplescaleout-data-2-2.samplescaleout.mynamespace.svc.cluster.local
```

The Operator creates a second Headless Service called samplescaleout-zk. This Service allows access to the ZooKeeper internal ports (2888 and 3888) as well as the external port (2181). The DNS names are in the format: samplescaleout-zk-n.samplescaleout-zk.namespace.svc.cluster.local.

The following example shows the DNS names for the samplescaleout-zk Headless Service:

```
samplescaleout-zk-0.samplescaleout-zk.mynamespace.svc.cluster.local
samplescaleout-zk-1.samplescaleout-zk.mynamespace.svc.cluster.local
samplescaleout-zk-2.samplescaleout-zk.mynamespace.svc.cluster.local
```

Persistent Volume Claims (PVCs):

kubectl get pvc

The output is the following:

NAME STATUS VOLUME ACCESS MODES STORAGECLASS AGE CAPACITY tt-persistent-samplescaleout-data-1-0 Bound ocid1.volume.oc1.phx.abyhqljtl63wgaxd3nvengilkaxs5h2b23kmtinpzmmqt7bkzjdpnf u2c2fa 53687091200 oci-bv RWO 14m tt-persistent-samplescaleout-data-1-1 Bound ocid1.volume.oc1.phx.abyhqljthxuwtpqx7rjtwvwsxjjkbgr25amjk7wtk26untbqrealcq he324q 53687091200 RWO oci-bv 14m tt-persistent-samplescaleout-data-1-2 Bound ocid1.volume.oc1.phx.abyhqljthtm6frcjtttp6ye7hq4w5vm3jxyay54f4xtcbolv2ercje ca5khq 53687091200 RWO oci-bv 14m tt-persistent-samplescaleout-data-2-0 Bound ocid1.volume.oc1.phx.abyhqljtxgzm3raxj5sfoe56aonlh2mqqjre4quva4k3q3zbbe7lft 53687091200 RWO wqk3xa oci-bv 14m tt-persistent-samplescaleout-data-2-1 Bound ocid1.volume.oc1.phx.abyhqljtk3htdair4akll5dfrwpkipv3acjtww5hx3x2fz46af7zde



53687091200 RWO oci-bv 14m w7qsiq tt-persistent-samplescaleout-data-2-2 Bound ocid1.volume.oc1.phx.abyhqljtiekqulpbadtwigehsuml75sngcuqjqqmx77lfpi3wdeecb 53687091200 oci-bv ecavfa RWO 14m tt-persistent-samplescaleout-mgmt-0 Bound ocid1.volume.oc1.phx.abyhqljtacxwop7r2wqx6hsvun4haaydnco3y6q3rkqbwp5hq35ala v7uwaq 53687091200 RWO oci-bv 14m tt-persistent-samplescaleout-zk-0 Bound ocid1.volume.oc1.phx.abyhqljtyoa5hdchax4sus652jtp665ckaef2cq3lakac2lq52vfbl 53687091200 RWO oci-bv s6kkcq 14m tt-persistent-samplescaleout-zk-1 Bound ocid1.volume.oc1.phx.abyhqljtongpcoggzpg2is25vmumijmah5gustwc3avgnijrjigtqp 53687091200 RWO oci-bv 13m htrana tt-persistent-samplescaleout-zk-2 Bound ocid1.volume.oc1.phx.abyhqljttmgoljskb2ruawzv365uit7lsln2sbfno5e4vhh6plbgh4 tiblfq 53687091200 RWO oci-bv 10m

The Operator automatically creates one or two Persistent Volume Claims (PVCs) per Pod. These PVCs cause Persistent Volumes (PVs) to be allocated by Kubernetes and attached to the TimesTen Pods. TimesTen uses these PVs to hold the TimesTen instance and database. If you specify two PVCs, one PV holds the TimesTen instance and the checkpoint files and the second PV holds the TimesTen transaction logs. In this example, the Operator creates one PVC for each Pod for a total of six PVCs. Each of the six PVs hold the TimesTen instance and an element of the database.

The Operator creates one PVC for the Pod that contains the management instance.

The Operator creates one PVC for each Pod that runs a ZooKeeper instance. This PVC causes a PV to be allocated by Kubernetes and attached to the ZooKeeper Pod. Each PV holds ZooKeeper's persistent data. In this example, since there are three ZooKeeper instances, the Operator created three PVCs.

Connect to the Database

You can establish a shell in a TimesTen Pod and connect to the TimesTen database in the grid. You can then run operations in this TimesTen database.

1. Establish a shell in the TimesTen samplescaleout-data-1-0 Pod.

kubectl exec -it samplescaleout-data-1-0 -c tt -- /bin/bash

2. Connect to the samplescaleout database. Verify the information from the metadata files is in the database. This example connects to the database as the sampleuser user. (This user was created in the schema.sql file). The example then calls the ttConfiguration built-in procedure to check that the PermSize connection attribute has a value of 200 for this element of the database. The example then verifies the sampleuser.emp table exists.

ttisql -connstr "DSN=samplescaleout;uid=sampleuser;pwd=sampleuserpwd";

The output is similar to the following:

Copyright (c) 1996, 2025, Oracle and/or its affiliates. All rights reserved. Type ? or "help" for help, type "exit" to quit ttIsql.



```
connect "DSN=samplescaleout;uid=sampleuser;pwd=*******";
Connection successful:
DSN=samplescaleout;Durability=0;UID=sampleuser;DataStore=/tt/home/timesten/
datastore/samplescaleout;
DatabaseCharacterSet=AL32UTF8;ConnectionCharacterSet=AL32UTF8;PermSize=200;
Connections=100;
(Default setting AutoCommit=1)
Command> call ttConfiguration ('PermSize');
< PermSize, 200 >
1 row found.
Command> tables;
SAMPLEUSER.EMP
1 table found.
```

Manage a Grid and Its Database

You can establish a shell in the TimesTen Pod of the management instance. You can then use the ttGridAdmin utility to manage and monitor the grid, including the health of the TimesTen database and its elements.

1. Establish a shell in the TimesTen samplescaleout-management-0 Pod.

```
kubectl exec -it samplescaleout-mgmt-0 -c tt -- /bin/bash
```

2. Run the ttGridAdmin dbStatus -all to check the status of the TimesTen database.

ttGridAdmin dbStatus -all

The output is the following:

Database samplescaleout summary status as of Sun Jan 15 17:00:11 UTC 2025

```
created,loaded-complete,open
Completely created elements: 6 (of 6)
Completely loaded elements: 6 (of 6)
Completely created replica sets: 3 (of 3)
Completely loaded replica sets: 3 (of 3)
```

Open elements: 6 (of 6)

Database samplescaleout element level status as of Sun Jan 15 17:00:11 UTC 2025

16:28:56 samplescaleout-data-2-0 instance1 4 opened stopped 2025-01-15 16:28:57 samplescaleout-data-2-1 instance1 5 opened stopped 2025-01-15 16:28:56 samplescaleout-data-2-2 instance1 6 opened stopped 2025-01-15 16:28:56 Database samplescaleout Replica Set status as of Wed Jan 15 17:00:11 UTC 2025 RS DS Elem Host Instance Status Cache Agent Date/Time of Event Message _____ _ 1 1 1 samplescaleout-data-1-0 instance1 opened stopped 2025-01-15 16:28:57 1 2 4 samplescaleout-data-2-0 instance1 opened stopped 2025-01-15 16:28:57 2 1 2 samplescaleout-data-1-1 instance1 opened stopped 2025-01-15 16:28:57 2 2 5 samplescaleout-data-2-1 instance1 opened stopped 2025-01-15 16:28:56 31 3 samplescaleout-data-1-2 instance1 opened stopped 2025-01-15 16:28:56 6 samplescaleout-data-2-2 instance1 opened stopped 32 2025-01-15 16:28:56 Database samplescaleout Data Space Group status as of Wed Jan 15 17:00:11 UTC 2025 DS RS Elem Host Instance Status Cache Agent Date/Time of Event Message __ __ ____ ____ _____ _ 1 1 1 samplescaleout-data-1-0 instance1 opened stopped 2025-01-15 16:28:57 1 2 2 samplescaleout-data-1-1 instance1 opened stopped 2025-01-15 16:28:57 1 3 3 samplescaleout-data-1-2 instance1 opened stopped 2025-01-15 16:28:56 2 1 4 samplescaleout-data-2-0 instance1 opened stopped 2025-01-15 16:28:57 2 2 5 samplescaleout-data-2-1 instance1 opened stopped 2025-01-15 16:28:56 2 3 6 samplescaleout-data-2-2 instance1 opened stopped 2025-01-15 16:28:56

Use Helm in Your TimesTen Kubernetes Operator Environment

You can use Helm and TimesTen Helm charts to install and upgrade TimesTen CRDs and the TimesTen Operator. You can also use Helm to create and upgrade TimesTen databases.

Topics:

- Overview of Helm and TimesTen Helm Charts
- Install the TimesTen CRDs and the TimesTen Operator
- Create TimesTen Databases and Test TimesTen
- Upgrade
- Roll Back an Upgrade
- Clean Up

Overview of Helm and TimesTen Helm Charts

This section provides an overview of Helm, discusses Helm charts, and details the Helm charts for TimesTen. It covers the following topics:

- About Helm
- About a Helm Chart
- About Helm Charts for TimesTen
- About Installing and Testing a Release
- About Versions in a Chart.yaml File
- About the Helm Substitution Engine and Language

About Helm

Helm is the package manager for Kubernetes. It is an open source project by the Cloud Native Computing Foundation (CNCF).

Kubernetes provides a rich interface for deploying applications through an extensible object model. Typically, you write manifest YAML files describing objects and then use the kubect1 command to feed the YAML files to Kubernetes. Kubernetes then configures the Kubernetes cluster to match. Built-in object types include Pods, Services, and Persistent Volumes. The TimesTen Operator extends the Kubernetes object type system by adding objects of type TimesTenClassic and TimesTenScaleout.

Helm is an optional layer on top of this architecture. Rather than manually creating YAML files and using kubectl to apply the files, Helm gives application developers the ability to package a set of YAML files into a single entity called a chart. Such a chart can be installed into a Kubernetes cluster using a single command.



The YAML files provided in a chart are templates. When installing a chart, you can specify a set of values to replace into the chart. Helm processes the YAML in the chart, substituting values provided by you, and submits the final version to Kubernetes through the kubectl command.

Helm also provides advanced facilities to manage applications:

- Charts can depend on other charts. Helm can assist in maintaining the dependency relationships. Helm can install an application and its dependencies as a unit.
- You can list charts that have been installed in your Kubernetes cluster and refer to them by name.
- You can use the helm test command to verify the functioning of the application.
- You can upgrade an installed application by applying a new version of the chart that installed the application.
- You can delete charts by name.

For more information about Helm, see https://helm.sh/docs/.

About a Helm Chart

A Helm chart is a collection of files in a directory. A chart consists of at minimum the following:

- Chart.yaml: This file contains metadata about the chart, including its name and version.
- templates: This is a directory containing one or more YAML file templates. A sophisticated templating language allows variables to be substituted into these files. Ultimately, when the chart is installed, Helm substitutes values into these templates and presents the result to Kubernetes with the kubectl command.
- values.yaml: This file contains default values for the variables, which are used in the chart's templates.

About Helm Charts for TimesTen

TimesTen provides TimesTen Helm charts in its container images. You have several options for obtaining container images and downloading the Helm charts from these container images. The examples assume that you have downloaded the charts into the kube_files/helm directory. See Prepare to Use the TimesTen Kubernetes Operator.

The following charts are included in the TimesTen container images's helm directory:

- ttcrd: Install the TimesTen Custom Resource Definitions (CRDs) into the Kubernetes cluster.
- ttoperator: Installs the TimesTen Operator.
- ttclassic: Creates TimesTen databases.

Each chart requires a different level of authority. As a result, the charts are packaged as separate charts so they can be installed with appropriate privileges. In addition, a single TimesTen Operator can simultaneously create and manage multiple TimesTen databases. Because the relationship between TimesTen Operators and TimesTen databases is not one to one, there is a ttoperator chart for installing the TimesTen Operator and a separate ttclassic chart for creating TimesTen databases.



About Installing and Testing a Release

You use Helm to install the ttcrd, ttoperator, and ttclassic Helm charts.

After installation, you can test the installation of the TimesTen Operator and the creation of TimesTen databases by running the helm test command. This command creates a test Pod that runs a command specific to the chart being tested. If the command finishes with exit code 0, the test succeeds and the test Pod is deleted. Any other error code indicates failure. In the case of failure, the test Pod is not deleted.

The examples in later sections show you how to install the charts and how to use the helm test command.

About Versions in a Chart.yaml File

Each chart contains metadata in its Chart.yaml file.

As an example, let's look at the Chart.yaml file for the ttoperator chart. In particular, let's look at the appVersion and version metadata.

```
apiVersion: v2
name: ttoperator
description: A Helm chart for Kubernetes
# A chart can be either an 'application' or a 'library' chart.
#
# Application charts are a collection of templates that can be packaged into
versioned archives
# to be deployed.
#
# Library charts provide useful utilities or functions for the chart
developer. They're included as
# a dependency of application charts to inject those utilities and functions
into the rendering
# pipeline. Library charts do not define any templates and therefore cannot
be deployed.
type: application
# This is the chart version. This version number should be incremented each
time you make changes
# to the chart and its templates, including the app version.
# Versions are expected to follow Semantic Versioning (https://semver.org/)
version: 2211340.1.0
# This is the version number of the application being deployed. This version
number should be
# incremented each time you make changes to the application. Versions are not
expected to
# follow Semantic Versioning. They should reflect the version the application
is using.
# It is recommended to use it with quotes.
appVersion: "22.1.1.34.0"
```

The value of version reflects the TimesTen release number without dots (.) followed by .1.0. For example, for release 22.1.1.34.0, the value of version is 2211340.1.0.

Similarly, the value of appVersion reflects the TimesTen release number, but is in a different format. For example, for release 22.1.1.34.0, the value of appversion is 22.1.1.34.0.

About the Helm Substitution Engine and Language

Helm has a robust substitution engine and language that lets you specify your own values for variables in a chart's template YAML manifest files. For more information about chart templates, see https://helm.sh/docs/chart_template_guide/getting_started/.

For each chart, the TimesTen Operator provides variables specific to that chart. For example, the ttoperator chart has variables and default values specific to the TimesTen Operator deployment. Similarly, the ttclassic chart has variables and default values specific to the deployment of TimesTen Classic databases. You can change the default values for the provided variables.

You can define variables and values for those variables either on the helm command line or by providing a YAML file with the values. The values may be strings, lists, arbitrary YAML, or entire files.

For information about the variables for the TimesTen-specific Helm charts, see Helm Charts for the TimesTen Kubernetes Operator.

Install the TimesTen CRDs and the TimesTen Operator

This section shows you how to install the TimesTen CRDs and the TimesTen Operator using TimesTen Helm charts. It covers the following topics:

- Install the TimesTen CRDs
- Install the TimesTen Operator
- Test the TimesTen Operator

Install the TimesTen CRDs

TimesTen provides the ttcrd chart in TimesTen container images. You use this chart to install the TimesTen CRDs in your cluster. For information about obtaining TimesTen Helm charts from TimesTen container images, see Obtain TimesTen YAML Manifest Files and Helm Charts.

Note:

Since CRDs are cluster-wide, the CRDs are available to all namespaces in the cluster after they are installed.

In this example, the location of the ttcrd chart is in the kube_files/helm directory on your development host. For more information about this directory and obtaining Helm charts, see Obtain TimesTen YAML Manifest Files and Helm Charts.



Note:

Do not modify this chart.

1. On your development host, change to the helm directory.

cd kube files/helm

2. Install the ttcrd chart.

helm install ttcrd ./ttcrd

The output is the following.

NAME: ttcrd LAST DEPLOYED: Thu Jan 16 20:50:47 2025 NAMESPACE: default STATUS: deployed REVISION: 1 TEST SUITE: None NOTES: Version 22.1.1.34.0.1.0 the ttcrd chart has been installed. This release is named "ttcrd". To learn more about the release, try: \$ helm status ttcrd \$ helm get all ttcrd \$ helm history ttcrd To rollback to a previous version of the chart, run: \$ helm rollback ttcrd <REVISION> - run 'helm history ttcrd' for a list of revisions.

3. Confirm the CRDs are defined in your Kubernetes cluster.

kubectl get crds | grep timesten

The output is the following:

timestenclassics.timesten.oracle.com 2025-01-01T20:50:48Z

Congratulations! You successfully installed the TimesTen CRDs.

Install the TimesTen Operator

TimesTen provides the ttoperator chart in TimesTen container images. You use this chart to install the TimesTen Operator in your namespace. The chart contains a default configuration for installing the TimesTen Operator Deployment. The Deployment causes Kubernetes to create one or more Pods, each of which runs the TimesTen Operator.



For information about obtaining TimesTen Helm charts from TimesTen container images, see Obtain TimesTen YAML Manifest Files and Helm Charts.

The TimesTen Operator requires a TimesTen container image. You need to specify the TimesTen container image, container registry, and image pull secret that you are using for the TimesTen Operator. The ttoperator chart provides default settings as well as variables that enable you to change the defaults. For information about these variables, see The ttoperator Chart.

Note:

The service account, roles, and role bindings that the TimesTen Operator requires to run properly are automatically installed when you install the ttoperator chart.

After you decide the variables you want to customize, you have two options:

- Create a YAML file that defines the variables that you want to use to configure your environment. Next, to install the chart, pass this YAML file to the helm install command by specifying the -f option.
- Specify the variables on the command line by running the helm install command with the --set option.

For more information about these options, see https://helm.sh/docs/intro/using_helm/.

The example creates a YAML file. Let's review how to make customizations for the container image, container registry, and image pull secret using a YAML file:

```
image:
   repository: container-registry.oracle.com/timesten/timesten
   tag: "22.1.1.34.0"
imagePullSecrets:
   - name: sekret
```

For the image variable, specify the following:

- repository: The name of your container image. This example uses containerregistry.oracle.com/timesten/timesten.
- tag: The name of the image tag. This example uses "22.1.1.34.0", corresponding to TimesTen release 22.1.1.34.0.

Let's install the TimesTen Operator. Let's assume you have previously created a kube_files/ helm/customyaml directory for your customized YAML file.

Note:

The installation process creates and deploys the service account, role, and rolebinding objects required to run the TimesTen Operator in your namespace.

1. On your development host, change to the helm directory.

```
cd kube files/helm
```


2. Create a YAML file that defines the variables for your configuration.

```
vi customyaml/tt_operator.yaml
image:
   repository: container-registry.oracle.com/timesten/timesten
   tag: "22.1.1.34.0"
imagePullSecrets:
   - name: sekret
```

3. Install the TimesTen Operator by installing the ttoperator chart.

```
helm install -f customyaml/tt operator.yaml ttoper ./ttoperator
```

Let's look at this helm install command:

- The -f option is used to pass the customized YAML file to the helm install command.
- The name of the file that contains the customizations is tt_operator.yaml, which is located in the customyaml directory.
- The name of the release is ttoper.
- The name of the chart that installs the TimesTen Operator is ttoperator.

Let's look at the output from the helm install command.

```
NAME: ttoper
LAST DEPLOYED: Thu Jan 16 02:54:39 2025
NAMESPACE: default
STATUS: deployed
REVISION: 1
NOTES:
Version 2211340.1.0 of the ttoperator chart has been installed.
This release is named "ttoper".
To learn more about the release, try:
  $ helm status ttoper
  $ helm get all ttoper
  $ helm history ttoper
To rollback to a previous version of the chart, run:
  $ helm rollback ttoper <REVISION>
    - run 'helm history ttoper' for a list of revisions.
To test the operator, run:
  $ helm test ttoper
```

Note the following:

• The ttoperator chart version is 2211340.1.0 corresponding to TimesTen release 22.1.1.34.0.



- The release name is ttoper.
- The status of the release is deployed.
- 4. (Optional) Verify the release.

helm list

Output.

```
NAME NAMESPACE REVISION

UPDATED STATUS CHART

APP VERSION

ttoper default 1 2025-01-16 02:54:39.916210368

+0000 UTC deployed ttoperator-2211340.1.0 22.1.1.34.0
```

The helm list command shows the ttoper release exists and is installed in your namespace.

5. Confirm the TimesTen Operator is running in your namespace.

kubectl get pods

The output is the following:

NAME	READY	STATUS	RESTARTS	AGE
timesten-operator-55c6f99-zqlct	1/1	Running	0	10m

Congratulations! You successfully installed the ttoperator chart. The TimesTen Operator is running in your namespace.

Test the TimesTen Operator

To test the TimesTen Operator, the test Pod uses the curl command to access the TimesTen Operator's readiness probe endpoint. If the TimesTen Operator self-reports that it is ready, the test succeeds.

If you previously disabled exposing probes from the TimesTen Operator by setting probes.expose=0 in your customized YAML file, helm test does not work. We recommend you do not disable exposing probes.

To test the TimesTen Operator, you test the ttoperator chart release running in your namespace.

1. Examine the ttoperator chart release.

helm list

The output is similar to the following:

NAME NAMESPACE REVISION UPDATED APP VERSION

STATUS CHART



```
ttoperdefault12025-01-1602:54:39.916210368+0000UTC deployed ttoperator-2211340.1.022.1.1.34.0
```

The ttoper release of the ttoperator chart is installed in your namespace.

2. Test the release.

helm test ttoper

The output is the following:

```
NAME: ttoper
LAST DEPLOYED: Thu Jan 16 02:54:39 2025
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE:
              ttoper-ttoperator-test
Last Started: Thu Jan 16 02:55:17 2025
Last Completed: Thu Jan 16 02:55:20 2025
Phase:
               Succeeded
NOTES:
Version 2211340.1.0 of the ttoperator chart has been installed.
This release is named "ttoper".
To learn more about the release, try:
  $ helm status ttoper
  $ helm get all ttoper
  $ helm history ttoper
To rollback to a previous version of the chart, run:
  $ helm rollback ttoper <REVISION>
    - run 'helm history ttoper' for a list of revisions.
To test the operator, run:
  $ helm test ttoper
```

The test for the ttoper release succeeded.

Congratulations! You successfully tested the TimesTen Operator. The test succeeded, indicating the TimesTen Operator is running in your namespace and operating properly.

Create TimesTen Databases and Test TimesTen

This section shows you how to create replicated and non-replicated TimesTen databases. It covers the following topics:

- About Creating TimesTen Databases
- About Using the ttclassic Helm Chart
- Create Replicated TimesTen Databases



- Test TimesTen for a Replicated Configuration
- Create Non-Replicated TimesTen Databases
- Test TimesTen for a Non-Replicated Configuration

About Creating TimesTen Databases

The TimesTen Operator creates, manages, and monitors TimesTen Classic databases. It supports the following topologies:

- Replicated active standby pair configurations: The TimesTen Operator configures one TimesTen database as the active database, copies the active database to the standby, and configures an active standby pair replication scheme between them.
- Non-replicated configurations: The TimesTen Operator configures one or more TimesTen Classic databases. Each database is independent and has no relationship to each other.

You use the ttclassic Helm chart to create TimesTen Classic databases for both replicated and non-replicated configurations. To distinguish the two configurations, the ttclassic chart provides the replicationTopology variable. Let's look at an example and then discuss the specifics.

Here are two YAML files that define the variables for both configurations. How you use these files to create TimesTen databases is discussed in a later section.

Replicated:

The replicationTopology variable has a value of activeStandbyPair, indicating a replicated configuration in which the TimesTen Operator creates an active standby pair of TimesTen databases. Since the default value for the replicationTopology variable is activeStandbyPair, you have the option of not specifying the replicationTopology variable for replicated configurations.

Non-replicated:

```
storageClassName: oci-bv
storageSize: 10Gi
image:
   repository: container-registry.oracle.com/timesten/timesten
   tag: "22.1.1.34.0"
imagePullSecret: sekret
replicationTopology: none
replicas: 3
rollingUpdatePartition: 2
dbConfigMap:
```



```
- name: norepsamplehelm directory: cm
```

The replicationTopology variable has a value of none, indicating a non-replicated configuration in which the TimesTen Operator creates one or more non-replicated TimesTen databases, each of which operates independently without replication configured. In addition, the ttclassic chart provides additional variables to assist in your customizations:

- replicas: Determines the number of Pods to be configured. Each Pod contains a TimesTen database. For example, if replicas is 3, there are three Pods each containing a TimesTen database. The default is 1.
- rollingUpdatePartition: Determines how many databases are upgraded. The upgrade procedure is discussed in a later section.

The following customizations apply to both configurations:

- storageClassName and storageSize: Since TimesTen is a database and is persistent, you have to specify a storage class, which is used to request persistent volumes for the database. The storageSize datum determines the amount of storage to be provisioned for TimesTen and its database.
- image: For repository, you must specify the container registry that contains the TimesTen container image. For tag, you specify an image tag. Taken together, image in this example represents the container registry that contains the TimesTen container image along with the image tag representing the release of TimesTen.
- imagePullSecret: You have to specify a Kubernetes Secret that authenticates
 Kubernetes to pull a TimesTen container image from the specified container registry.
- dbConfigMap: Optionally, you can specify one or more ConfigMaps that contain the metadata files for your TimesTen databases. You do not create these ConfigMaps. Rather, the ttclassic chart installation process creates the ConfigMaps for you. How this is done is discussed later in this chapter.

There are many more customizations available to you. See The ttclassic Chart. In addition, the TimesTen Operator provides metadata files that enable you to customize and define the configuration for your TimesTen databases. These files include testUser, db.ini, adminUser, cacheUser, schema, sql, cachegroups.sql, and others. The TimesTen Operator provides several methods for you to make those files available to the TimesTen Operator and TimesTen. For details, see Use Configuration Metadata.

About Using the ttclassic Helm Chart

The ttclassic Helm chart contains the information needed to deploy replicated and nonreplicated TimesTen database in your Kubernetes namespace. There are many customizations available to meet your preferred configuration. There are variables specific to the ttclassic chart for such customizations. The About Creating TimesTen Databases section discusses some of these customizations. For additional customizations, see The ttclassic Chart.

The TimesTen Operator provides metadata files to further customize your database. One file of particular interest is testUser. After TimesTen is deployed, you can use the helm test command to test TimesTen. The test operation requires that a TimesTen test user exist in the TimesTen database. This user connects to the database as part of the testing process. This

user must be defined before installing the ttclassic chart. The TimesTen Operator provides the testUser metadata file for this purpose. The file contains one line of the form:

```
testuser/testuserpassword
```

where testuser is the name of the TimesTen user to use for testing TimesTen and testuserpassword is the password for this test user. For details, see Use Configuration Metadata.

If you are using Kubernetes ConfigMaps or Secrets as the facilities to place metadata files into the TimesTen containers, you do not need to create them. Instead, when you install the ttclassic chart, the ConfigMaps and Secrets are automatically created as part of the installation process. To facilitate the process, you must create a directory within the ttclassic chart directory tree and create the metadata files in this directory. There are examples later in this section that show you how to do this.

After you decide your configurations needs and review the variables you are using for these customizations, you have the following options for supplying these variables as part of the ttclassic chart's installation process:

- Create a YAML file for the variables. When installing the ttclassic chart, pass this YAML file to the helm install command by specifying the -f option.
- Specify the variables on the command line by running the helm install command with the --set option.

The examples use a YAML file. For specifics about these options including the syntax, see https://helm.sh/docs/intro/using_helm/ in the Helm documentation.

The examples show you how to use the ttclassic Helm chart to define and deploy replicated and non-replicated TimesTen databases. Both examples use the same set of metadata files and both use Kubernetes ConfigMaps to get the metadata files into the tt containers.

For simplicity, let's create the metadata files and ConfigMaps used in the examples. For complete details about metadata files, see Use Configuration Metadata and Populate the / ttconfig Directory.

 On your development host, change to the ttclassic directory. In this example, the directory location is kube files/helm/ttclassic.

```
cd kube files/helm/ttclassic
```

2. Create a directory for the metadata files.

mkdir cm

- 3. Create metadata files in the directory used for the metadata files (cm in this example). These metadata files are used in the examples throughout the chapter.
 - a. Create the testUser file.

vi cm/testUser

sampletestuser/sampletestuserpwd1



b. Create the db.ini file.

```
vi cm/db.ini
```

PermSize=200 DatabaseCharacterSet=AL32UTF8

c. Create the adminUser file.

```
vi cm/adminUser
```

adminuser/adminuserpwd

d. Create the schema.sql file.

```
vi cm/schema.sql
```

```
create table adminuser.emp (id number not null primary key, name char
(32));
```

You created the metadata files and the directory for them. Save the location of the directory. You need it later.

Create Replicated TimesTen Databases

This example shows you how to create replicated TimesTen Classic databases using the ttclassic Helm chart. The example uses a YAML manifest file and assumes you have created the kube files/helm/customyaml directory for the file.

1. On your development host, change to the helm directory.

cd kube files/helm

2. Create a YAML file that defines the variables for your replicated configuration.

Note the following:

- The storageClassName is oci-bv. Replace oci-bv with the name of your storage class.
- The storageSize is 10Gi. Replace 10Gi with the amount of storage that needs to be requested for each Pod to hold TimesTen.



- For the image variable:
 - repository: The repository is container-registry.oracle.com/timesten/ timesten. Replace container-registry.oracle.com/timesten/timesten with the name and location of your TimesTen container image.
 - tag: The tag is 22.1.1.34.0. Replace tag with the tag for the TimesTen release.
- The imagePullSecret is sekret. Replace sekret with the image pull secret that Kubernetes needs to use to fetch the TimesTen container image.
- The replicationTopology is activeStandbyPair, indicating a replicated configuration consisting of an active standby pair of TimesTen databases.
- The name of the ConfigMap is repsamplehelm. The location of the metadata files is in the cm directory, which is located within the kube file/helm/ttclassic directory tree..
- 3. Install the ttclassic chart.

```
helm install -f customyaml/repsamplehelm.yaml repsamplehelm ./ttclassic
```

Let's look at this helm install command:

- The -f option indicates that a YAML file is passed to the helm install command.
- The name of the YAML file that contains the customizations is repsamplehelm.yaml, which is located in the customyaml directory.
- The name of the release is repsamplehelm.
- The name of the chart is ttclassic, which is located in the kube_files/helm/ ttclassic directory.

Let's look at the output from the helm install command.

```
NAME: repsamplehelm
LAST DEPLOYED: Thu Jan 16 14:58:40 2025
NAMESPACE: default
STATUS: deployed
REVISION: 1
NOTES:
Version 2211340.1.0 of the ttclassic chart has been installed.
This release is named "repsamplehelm".
To learn more about the release, try:
    $ helm status repsamplehelm
    $ helm get all repsamplehelm
    $ helm history repsamplehelm
    $ helm history repsamplehelm
    $ helm history repsamplehelm
    $ helm nollback to a previous version of the chart, run:
    $ helm rollback repsamplehelm <REVISION>
    - run 'helm history repsamplehelm' for a list of revisions.
```

Note the following:

The ttclassic chart version is 2211340.1.0, corresponding to TimesTen release 22.1.1.34.0.

- The release name is repsamplehelm.
- The status of the release is deployed.
- 4. (Optional) Verify the release.

helm list

The output is similar to the following:

```
NAME NAMESPACE REVISION

UPDATED STATUS

CHART APP VERSION

repsamplehelm default 1 2025-01-16

14:58:40.109402333 +0000 UTC deployed ttclassic-2211340.1.0

22.1.1.34.0

...
```

The helm list command shows the repsamplehelm release exists and is installed in your namespace.

5. Monitor the progress.

kubectl get ttc repsamplehelm

The output is similar to the following:

NAME	STATE	ACTIVE	AGE
repsamplehelm	Initializing	None	44s

The provisioning starts, but is not yet complete as indicated by the Initializing state.

Wait a few minutes. Then, check again.

```
kubectl get ttc repsamplehelm
```

The output is similar to the following:

NAME	STATE	ACTIVE	AGE
repsamplehelm	Normal	repsamplehelm-0	4m23s

The provisioning process completes. The active standby pair of TimesTen databases are up and running and operational as indicated by the Normal state.

6. Confirm the ConfigMap and the metadata files exist.

kubectl get configmap repsamplehelm

NAME	DATA	AGE
repsamplehelm	4	8m50s



Check the metadata files.

kubectl describe configmap repsamplehelm

The output is similar to the following:

```
Name:
            repsamplehelm
Namespace: default
Labels:
            app.kubernetes.io/managed-by=Helm
Annotations: meta.helm.sh/release-name: repsamplehelm
            meta.helm.sh/release-namespace: default
Data
====
adminUser:
____
adminuser/adminuserpwd
db.ini:
____
PermSize=200
DatabaseCharacterSet=AL32UTF8
schema.sql:
____
create table adminuser.emp (id number not null primary key, name char
(32));
testUser:
____
sampletestuser/sampletestuserpwd1
BinaryData
====
Events: <none>
```

The repsamplehelm ConfigMap exists and contains the metadata files. Since the testUser file exists, you can use Helm to test TimesTen. See Test TimesTen for a Replicated Configuration.

7. (Optional) Confirm the Pods.

kubectl get pods

NAME	READY	STATUS	RESTARTS	AGE
repsamplehelm-0	3/3	Running	0	11m
repsamplehelm-1	2/3	Running	0	11m



The repsamplehelm-0 and repsamplehelm-1 Pods are running in your namespace. Each Pod is running TimesTen.

Congratulations! You successfully installed the ttclassic chart for a replicated configuration. An active standby pair of TimesTen Classic databases are up and running and fully operational.

Test TimesTen for a Replicated Configuration

To test TimesTen, a test Pod is created. It runs a script that attempts to connect to a TimesTen database by using TimesTen's client/server access. If the script can connect to the database, the script then looks at internal metadata to verify that the database is functioning properly. If the test succeeds, the test Pod is deleted.

The test connects to the database as the test user. You define this test user in the testUser metadata file. This user has CONNECT privileges, but does not have any other privileges. This limits the test and the test user's access to the database. If you do not define a test user, the test fails.

The test verifies that the active standby pair of TimesTen Classic databases are up and running and that replication between them is configured.

Let's test TimesTen.

1. Confirm the ttclassic release.

helm list

The output is similar to the following:

```
NAMENAMESPACEREVISIONUPDATEDSTATUSCHARTAPP VERSIONrepsamplehelmdefault114:58:40.109402333 +0000UTC deployed22.1.1.34.0
```

The repsamplehelm release exists in your namespace.

2. Test TimesTen.

helm test repsamplehelm

```
NAME: repsamplehelm
LAST DEPLOYED: Thu Jan 16 14:58:40 2025
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: repsamplehelm-ttclassic-test
Last Started: Thu Jan 16 15:21:45 2025
Last Completed: Thu Jan 16 15:21:49 2025
Phase: Succeeded
NOTES:
```



Version 2211340.1.0 of the ttclassic chart has been installed. This release is named "repsamplehelm". To learn more about the release, try: \$ helm status repsamplehelm \$ helm get all repsamplehelm \$ helm history repsamplehelm To rollback to a previous version of the chart, run: \$ helm rollback repsamplehelm <REVISION> - run 'helm history repsamplehelm' for a list of revisions.

The test for the repsamplehelm release succeeded.

Congratulations! You successfully tested TimesTen. The active and standby databases are up and running and replication between them is configured.

Create Non-Replicated TimesTen Databases

This example shows you how to create non-replicated TimesTen Classic databases using the ttclassic Helm chart. The example uses a YAML manifest file and assumes you have created the kube files/helm/customyaml directory for the file.

1. On your development host, change to the helm directory.

vi customyaml/norepsamplehelm.yaml

cd kube files/helm

2. Create a YAML file that defines the variables for your non-replicated configuration.

Note the following:

- The storageClassName is oci-bv. Replace oci-bv with the name of your storage class.
- The storageSize is 10Gi. Replace 10Gi with the amount of storage that needs to be requested for each Pod to hold TimesTen.
- For the image variable:



- repository: The repository is container-registry.oracle.com/timesten/ timesten. Replace container-registry.oracle.com/timesten/timesten with the name and location of your TimesTen container image.
- tag: The tag is 22.1.1.34.0. Replace tag with the tag for the TimesTen release.
- The imagePullSecret is sekret. Replace sekret with the image pull secret that Kubernetes needs to use to fetch the TimesTen container image.
- For a non-replicated configuration:
 - The replicationTopology is none, indicating a non-replicated configuration consisting of replicas number of Pods. Each Pod contains an independent TimesTen database.
 - The number of replicas is 3, indicating the number of Pods, each of which contains a TimesTen database. Replace 3 with the number of Pods you want provisioned. Valid values are between 1 and 3, with 1 being the default.
 - The rollingUpdatePartition is 2. This variable is specific to upgrades and determines the number of TimesTen databases to upgrade. Kubernetes upgrades Pods with an ordinal value that is greater than or equal to the rollingUpdatePartition value. For example, if you have three non-replicated Pods (replicas = 3 and Pods are norepsamplehelm-0, norepsamplehelm-1, and norepsamplehelm-2) and you set rollingUpdatePartition to 2, the norepsamplehelm-0 Pods is upgraded, but the norepsamplehelm-1 and norepsamplehelm-0 Pods are not. There are examples in the upgrade section that show you how rollingUpdatePartition works. You have the option of changing the value during the upgrade process.
- The name of the ConfigMap is norepsamplehelm. The location of the metadata files is in the cm directory, which is located within the kube_file/helm/ttclassic directory
 tree..
- 3. Install the ttclassic chart.

helm install -f customyaml/norepsamplehelm.yaml norepsamplehelm ./ttclassic

Let's look at this helm install command:

- The -f option indicates that a YAML file is passed to the helm install command.
- The name of the YAML file that contains the customizations is norepsamplehelm.yaml, which is located in the customyaml directory.
- The name of the release is norepsamplehelm.
- The name of the chart is ttclassic, which is located in the kube_files/helm/ ttclassic directory.

Let's look at the output from the helm install command.

```
NAME: norepsamplehelm
LAST DEPLOYED: Thu Jan 16 17:42:47 2025
NAMESPACE: default
STATUS: deployed
REVISION: 1
NOTES:
Version 2211340.1.0 of the ttclassic chart has been installed.
```



```
This release is named "norepsamplehelm".

To learn more about the release, try:

$ helm status norepsamplehelm

$ helm get all norepsamplehelm

$ helm history norepsamplehelm

To rollback to a previous version of the chart, run:

$ helm rollback norepsamplehelm <REVISION>

- run 'helm history norepsamplehelm' for a list of revisions.
```

Note the following:

- The ttclassic chart version is 2211340.1.0, corresponding to TimesTen release 22.1.1.34.0.
- The release name is norepsamplehelm.
- The status of the release is deployed.
- 4. (Optional) Verify the release.

helm list

The output is similar to the following:

NAMENAMESPACEREVISIONUPDATEDSTATUSCHARTAPP VERSIONnorepsamplehelm default12025-01-1617:42:47.180635098 +0000 UTC deployedttclassic-2211340.1.022.1.1.26.0

The helm list command shows the norepsamplehelm release exists and is installed in your namespace.

5. Monitor the progress.

NAME STATE ACTIVE AGE norepsamplehelm NoReplicasReady N/A 94s

The provisioning starts, but is not yet complete as indicated by the NoReplicasReady state.

Wait a few minutes. Then, check again.

kubectl get ttc norepsamplehelm

NAME	STATE	ACTIVE	AGE
norepsamplehelm	AllReplicasReady	N/A	12m



The provisioning process completes. Databases are up and running and operational as indicated by the AllReplicasReady state.

6. Confirm the ConfigMap and the metadata files exist.

kubectl get configmap norepsamplehelm

The output is similar to the following:

NAME	DATA	AGE
norepsamplehelm	4	15m

Check the metadata files.

kubectl describe configmap norepsamplehelm

The output is similar to the following:

```
Name: norepsamplehelm
Namespace: default
Labels: app.kubernetes.io/managed-by=Helm
Annotations: meta.helm.sh/release-name: norepsamplehelm
            meta.helm.sh/release-namespace: default
Data
====
testUser:
____
sampletestuser/sampletestuserpwd1
adminUser:
____
adminuser/adminuserpwd
db.ini:
____
PermSize=200
DatabaseCharacterSet=AL32UTF8
schema.sql:
____
create table adminuser.emp (id number not null primary key, name char
(32));
BinaryData
====
Events: <none>
```

The norepsamplehelm ConfigMap exists and contains the metadata files. Since the testUser file exists, you can use Helm to test TimesTen. See Test TimesTen for a Non-Replicated Configuration.



7. (Optional) Confirm the Pods.

kubectl get pods

The output is similar to the following:

NAME	READY	STATUS	RESTARTS	AGE
norepsamplehelm-0	3/3	Running	0	17m
norepsamplehelm-1	3/3	Running	0	17m
norepsamplehelm-2	3/3	Running	0	17m
•••				

There are three Pod running in your namespace, each of which contains an independent TimesTen database.

Congratulations! You successfully installed the ttclassic chart for a non-replicated configuration. TimesTen databases are up and running and fully operational.

Test TimesTen for a Non-Replicated Configuration

To test TimesTen, a test Pod is created. It runs a script that attempts to connect to all of the TimesTen databases (the replicas) by using TimesTen's client/server access. If the script can connect to the database, the script then looks at internal metadata to verify that the database is functioning properly. If the test succeeds, the test Pod is deleted.

The test connects to the database as the test user. You define this test user in the testUser metadata file. This user has CONNECT privileges, but does not have any other privileges. This limits the test and the test user's access to the database. If you do not define a test user, the test fails.

The test verifies TimesTen databases are up and running and fully operational.

Let's test TimesTen.

1. Confirm the ttclassic release.

helm list

The output is similar to the following:

NAME NAMESPACE REVISION UPDATED STATUS CHART APP VERSION norepsamplehelm default 1 2025-01-16 17:42:47.180635098 +0000 UTC deployed ttclassic-2211340.1.0 22.1.1.26.0

The norepsamplehelm release exists in your namespace.

2. Test TimesTen.

helm test norepsamplehelm



The output is similar to the following:

```
NAME: norepsamplehelm
LAST DEPLOYED: Thu Jan 16 17:42:47 2025
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE:
              norepsamplehelm-ttclassic-test
Last Started: Thu Jan 16 18:17:22 2025
Last Completed: Thu Jan 16 18:17:26 2025
Phase:
               Succeeded
NOTES:
Version 2211340.1.0 of the ttclassic chart has been installed.
This release is named "norepsamplehelm".
To learn more about the release, try:
  $ helm status norepsamplehelm
  $ helm get all norepsamplehelm
  $ helm history norepsamplehelm
To rollback to a previous version of the chart, run:
  $ helm rollback norepsamplehelm <REVISION>
    - run 'helm history norepsamplehelm' for a list of revisions.
```

The test for the norepsamplehelm release succeeded.

Congratulations! You successfully tested TimesTen. TimesTen databases are up and running and fully operational.

Upgrade

This section shows you how to upgrade the TimesTen CRDs, the TimesTen Operator, and both replicated and non-replicated TimesTenClassic objects and associated TimesTen databases. It covers the following topics:

- About Upgrading
- Upgrade the TimesTen CRDs
- Upgrade the TimesTen Operator
- Upgrade Replicated TimesTen Databases
- Upgrade Non-Replicated TimesTen Databases

About Upgrading

You can use Helm to upgrade TimesTen CRDs and the TimesTen Operator. You can also use Helm to perform an automated upgrade of both replicated and non-replicated TimesTen Classic databases.

TimesTen provides several options for obtaining container images that contain the release of TimesTen that you want to use for the upgrade. These container images contain the Helm



charts for the upgrade. For information about obtaining container images and downloading Helm charts from a container image, see Prepare to Use the TimesTen Kubernetes Operator.

The upgrade examples make the following assumptions:

- Container image: The container image that contains the new release of TimesTen is container-registry.oracle.com/timesten/timesten:22.1.1.35.0.
- Download directory: The directory on your development host that contain the Helm charts for the upgrade is new kube files/helm.

Upgrade the TimesTen CRDs

To upgrade the TimesTen CRDs in your cluster, use the ttcrd chart from the new release.

Note the following:

- You cannot downgrade TimesTen CRDs.
- Since CRDs are cluster-scoped, if you delete TimesTen CRDs from your Kubernetes cluster, all TimesTenClassic objects that are deployed in this cluster are also deleted.
- 1. Confirm the current ttcrd release.

helm list

The output is similar to the following:

NAME NAMESPACE	REVISION	
UPDATED		STATUS CHART
APP VERSION		
ttcrd default	1	2025-01-16 20:50:47.406557985
+0000 UTC deployed	ttcrd-2211340.1.0	22.1.1.34.0

The ttcrd chart version is 2211340.1.0 and appversion is 22.1.1.34.0 corresponding to TimesTen release 22.1.1.34.0.

2. On your development host, change to the new_kube_files/helm directory.

cd new_kube_files/helm

Upgrade to the new release.

helm upgrade ttcrd ./ttcrd

```
Release "ttcrd" has been upgraded. Happy Helming!
NAME: ttcrd
LAST DEPLOYED: Thu Jan 16 19:07:52 2025
NAMESPACE: default
STATUS: deployed
REVISION: 2
TEST SUITE: None
NOTES:
Version 2211350.1.0 the ttcrd chart has been installed.
```



```
This release is named "ttcrd".

To learn more about the release, try:

$ helm status ttcrd

$ helm get all ttcrd

$ helm history ttcrd

To rollback to a previous version of the chart, run:

$ helm rollback ttcrd <REVISION>

- run 'helm history ttcrd' for a list of revisions.
```

The TimesTen CRDs are upgraded to the new release.

Congratulations! You successfully upgraded the TimesTen CRDs.

Upgrade the TimesTen Operator

You can upgrade the TimesTen Operator while there are TimesTenClassic objects running in your namespace. To upgrade the TimesTen Operator in your namespace, use the new release of the ttoperator chart.

The upgrade process requires you to complete the following steps:

- Create a new YAML file. In this file, you specify the container image that contains the new release of TimesTen.
- Use the helm get values command for the upgrade. The example shows you how to do this.

The example assumes you have created a new kube files/helm/customyaml directory.

1. On your development host, change to the helm directory that contains the new Helm charts.

cd new kube files/helm

2. Create a YAML file and in it specify the name of the TimesTen image you want to use for the upgrade.

vi customyaml/upgradeoperator.yaml

```
image:
    repository: container-registry.oracle.com/timesten/timesten
    tag: "22.1.1.35.0"
```

- 3. Before the upgrade, do the following:
 - a. Verify that the TimesTen Operator is running.

kubectl get pods



The output is the following:

NAME	READY	STATUS	RESTARTS	AGE
norepsamplehelm-0	3/3	Running	0	6h19m
norepsamplehelm-1	3/3	Running	0	6h19m
norepsamplehelm-2	3/3	Running	0	6h19m
repsamplehelm-0	3/3	Running	0	9h
repsamplehelm-1	2/3	Running	0	9h
timesten-operator-55c6f99-zqlct	1/1	Running	0	2d21h

b. Confirm the image that the TimesTen Operator is running.

kubectl describe deployment timesten-operator | grep Image

The output is the following:

Image: container-registry.oracle.com/timesten/timesten:22.1.1.34.0

c. Confirm the state of the TimesTenClassic objects.

kubectl get ttc

The output is similar to the following:

NAME	STATE	ACTIVE	AGE
norepsamplehelm	AllReplicasReady	N/A	6h28m
repsamplehelm	Normal	repsamplehelm-0	9h

The TimesTenClassic objects and associated databases are functioning properly.

4. Confirm the Helm chart release.

helm list

The output is similar to the following:

NAME NAMESPACE REVISION UPDATED STATUS CHART APP VERSION ... ttoper default 1 2025-01-16 02:54:39.916210368 +0000 UTC deployed ttoperator-2211340.1.0 22.1.1.34.0

5. Upgrade the TimesTen Operator to the new release. To upgrade the TimesTen Operator, use the ttoperator chart from the new release and use the customized YAML file that references the new image.



Note:

We recommend this syntax. This ensures existing customizations are preserved.

helm get values ttoper --all > prev-values-ttop.yaml

```
helm upgrade -f prev-values-ttop.yaml -f customyaml/upgradeoperator.yaml
ttoper ./ttoperator
```

Let's look at this helm upgrade command:

- The get values ttoper Helm command retrieves the values for the current release, including existing customizations. The result of this command is piped into the prevvalues-ttop.yaml file. You can choose any name for this file.
- The helm upgrade command uses the prev-values-ttop.yaml file with the customized upgradeoperator.yaml file to do the upgrade. Specifically, the values in the prev-values-ttop.yaml file are used except when overridden by the values in the customized upgradeoperator.yaml file. The upgradeoperator.yaml file takes precedence over the prev-values-ttop.yaml file since it was listed last on the command line.

Let's look at the output from the helm upgrade command.

```
Release "ttoper" has been upgraded. Happy Helming!
NAME: ttoper
LAST DEPLOYED: Thu Jan 16 00:20:34 2025
NAMESPACE: default
STATUS: deployed
REVISION: 2
NOTES:
Version 2211350.1.0 of the ttoperator chart has been installed.
This release is named "ttoper".
To learn more about the release, try:
  $ helm status ttoper
  $ helm get all ttoper
  $ helm history ttoper
To rollback to a previous version of the chart, run:
  $ helm rollback ttoper <REVISION>
    - run 'helm history ttoper' for a list of revisions.
To test the operator, run:
```

Note the following:

- The ttoper release is upgraded. The release revision is 2.
- The status of the release is deployed.



- The ttoperator chart version is 2211350.1.0 corresponding to TimesTen release 22.1.1.35.0.
- 6. Test the TimesTen Operator is running.

helm test ttoper

The output is similar to the following:

```
NAME: ttoper
LAST DEPLOYED: Thu Jan 16 00:20:34 2025
NAMESPACE: default
STATUS: deployed
REVISION: 2
TEST SUITE:
              ttoper-ttoperator-test
Last Started: Thu Jan 16 00:20:54 2025
Last Completed: Thu Jan 16 00:20:57 2025
              Succeeded
Phase:
NOTES:
Version 2211350.1.0 of the ttoperator chart has been installed.
This release is named "ttoper".
To learn more about the release, try:
  $ helm status ttoper
  $ helm get all ttoper
  $ helm history ttoper
To rollback to a previous version of the chart, run:
  $ helm rollback ttoper <REVISION>
    - run 'helm history ttoper' for a list of revisions.
To test the operator, run:
  $ helm test ttoper
```

- 7. After the upgrade, do the following:
 - a. Verify that the TimesTen Operator is running.

kubectl get pods

The output is the following:

READY	STATUS	RESTARTS	AGE
3/3	Running	0	6h38m
3/3	Running	0	6h38m
3/3	Running	0	6h38m
3/3	Running	0	9h
2/3	Running	0	9h
1/1	Running	0	31s
	3/3 3/3 3/3 3/3 2/3	3/3Running3/3Running3/3Running3/3Running2/3Running	3/3 Running 0 2/3 Running 0

There is a new TimesTen Operator running.

b. Confirm the TimesTen Operator is running the new image.

kubectl describe deployment timesten-operator | grep Image

The output is the following:

Image: container-registry.oracle.com/timesten/timesten:22.1.1.35.0

The TimesTen Operator is running the new release.

c. Confirm the state of the TimesTenClassic objects.

kubectl get ttc

The output is similar to the following:

NAME	STATE	ACTIVE	AGE
norepsamplehelm	AllReplicasReady	N/A	6h39m
repsamplehelm	Normal	repsamplehelm-0	9h

The TimesTen Operator resumes the management and monitoring of the TimesTenClassic objects. The objects and the associated TimesTen databases are functioning properly.

Congratulations! You successfully upgraded the ttoperator chart. There is a new TimesTen Operator that is running in your namespace and it is using the new image.

Upgrade Replicated TimesTen Databases

You can upgrade a replicated TimesTen configuration, consisting of an active standby pair of TimesTen databases.

The upgrade process requires you to perform the following steps:

- Create a YAML file. In this file, you specify the container image that contains the new release of TimesTen.
- Uses the helm values command for the upgrade. The example shows you how to do this.

The example assumes you have created a new kube files/helm/customyaml directory.



Note:

The following actions occur during an upgrade:

- The standby is terminated first. It takes some time for the standby to come back up. When the standby comes back up, it is upgraded to the new release. During the upgrade of the standby, depending on your replication configuration, there may be disruption on the active database.
- Next, the failover from the active to the standby occurs:
 - The active is terminated. It takes some time for the former active to come back up. When the active comes back up, it is upgraded to the new release.
 - The standby database is promoted to the active and the former active becomes the standby.

For more information about how TimesTen performs an upgrade of an active standby pair of TimesTen databases, see Performing an Upgrade with Active Standby Pair Replication in the Oracle TimesTen In-Memory Database Installation, Migration, and Upgrade Guide.

Ensure you perform an upgrade at the appropriate time. We recommend that you do not perform upgrades at the busiest time of a production day. Applications may experience short outages and perhaps reduced performance as a result of the upgrade procedure.

1. On your development host, change to the helm directory that contains the new Helm charts.

cd new kube files/helm

Create a YAML file and in it specify the name of the TimesTen image you want to use for the upgrade.

vi customyaml/upgradereplicated.yaml

```
image:
    repository: container-registry.oracle.com/timesten/timesten
    tag: "22.1.1.35.0"
```

- 3. Before the upgrade, do the following:
 - a. Confirm the replicated TimesTenClassic object exists in your namespace.

kubectl get ttc

The output is similar to the following:

NAME	STATE	ACTIVE	AGE
norepsamplehelm	AllReplicasReady	N/A	8h
repsamplehelm	Normal	repsamplehelm-0	11h

The repsamplehelm TimesTenClassic object exists and is in the Normal state.



b. Verify the image.

kubectl describe ttc repsamplehelm | grep Image

The output is similar to the following:

Image: container-registry.oracle.com/timesten/timesten:22.1.1.34.0
...

4. Confirm the ttclassic chart release.

helm list

The output is similar to the following:

NAME NAMESPACE REVISION UPDATED STATUS CHART APP VERSION repsamplehelm default 1 2025-01-16 14:58:40.109402333 +0000 UTC deployed ttclassic-2211340.1.0 22.1.1.34.0

5. Upgrade to the new release. Use the ttclassic chart from the new release and use the customized YAML file that references the new image.

Note:

We recommend the following syntax. This ensures existing customizations are preserved.

helm get values repsamplehelm --all > prev-values-repttc.yaml

```
helm upgrade -f prev-values-repttc.yaml -f customyaml/
upgradereplicated.yaml repsamplehelm ./ttclassic
```

Let's look at this helm upgrade command:

- The get values repsamplehelm Helm command retrieves the values for the current release, including existing customizations. The result of this command is piped into the prev-values-repttc.yaml file. You can choose any name for this file.
- The helm upgrade command uses the prev-values-repttc.yaml file with the customized upgradereplicated.yaml file to do the upgrade.

Let's look at the output from the helm upgrade command.

```
Release "repsamplehelm" has been upgraded. Happy Helming!
NAME: repsamplehelm
LAST DEPLOYED: Thu Jan 16 02:25:13 2025
NAMESPACE: default
STATUS: deployed
```



```
REVISION: 2
NOTES:
Version 2211350.1.0 of the ttclassic chart has been installed.
This release is named "repsamplehelm".
To learn more about the release, try:
   $ helm status repsamplehelm
   $ helm get all repsamplehelm
   $ helm history repsamplehelm
To rollback to a previous version of the chart, run:
   $ helm rollback repsamplehelm <REVISION>
   - run 'helm history repsamplehelm' for a list of revisions.
```

Note the following:

- The repsamplehelm chart release is upgraded. The release revision is 2.
- The status of the release is deployed.
- The ttclassic chart version is 2211350.1.0 corresponding to TimesTen release 22.1.1.35.0.
- 6. Observe the automatic upgrade and how the TimesTenClassic object transitions from and to various states.

kubectl get events -w

11m	Normal	Upgrade	timestenclassic/
repsamplehe	lm Imag	ge updated, auto	matic upgrade started
11m	Normal	Upgrade	timestenclassic/
repsamplehe	lm Dele	eted standby pod	repsamplehelm-1 during upgrade
11m	Warning	Failed	timestenclassic/
repsamplehe	lm Pod	repsamplehelm-1	was replaced
11m	Normal	StateChange	timestenclassic/
repsamplehe	lm Pod	repsamplehelm-1	is Not Ready
11m	Warning	StateChange	timestenclassic/
repsamplehe	lm Time	esTenClassic was	Normal, now ActiveTakeover
11m	Normal	StateChange	timestenclassic/
repsamplehe	lm Time	esTenClassic was	ActiveTakeover, now StandbyDown
9m26s	Normal	Info	timestenclassic/
repsamplehe	lm Pod	repsamplehelm-1	Agent Up
9m26s	Normal	Info	timestenclassic/
repsamplehe	lm Pod	repsamplehelm-1	Instance Exists
9m26s	Normal	Info	timestenclassic/
repsamplehe	lm Pod	repsamplehelm-1	Daemon Down
9m26s	Normal	Info	timestenclassic/
repsamplehe	lm Pod	repsamplehelm-1	
9m26s	Normal	Info	timestenclassic/
repsamplehe	lm Pod	÷ ÷	Database Unloaded
9m24s	Normal	Info	timestenclassic/



repsamplehelm Pod repsamplehelm-1 Database None 9m21s Normal Info timestenclassic/ repsamplehelm pollAsyncStatus: Async polling for RepDuplicate, timeout in 597 secs 9m18s Normal Info timestenclassic/ repsamplehelm pollAsyncStatus: Async polling for RepDuplicate, timeout in 594 secs 9m15s Normal Info timestenclassic/ repsamplehelm pollAsyncStatus: Async polling for RepDuplicate, timeout in 591 secs 9m12s Normal Info timestenclassic/ repsamplehelm pollAsyncStatus: Async polling for RepDuplicate, timeout in 588 secs 9m9s Normal Info timestenclassic/ repsamplehelm pollAsyncStatus: Async polling for RepDuplicate, timeout in 585 secs 9m6s Normal Info timestenclassic/ repsamplehelm RepDuplicate completed in 16 secs 9m5s Normal Info timestenclassic/ repsamplehelm Pod repsamplehelm-1 Database Loaded 9m5s Normal Info timestenclassic/ repsamplehelm Pod repsamplehelm-1 RepAgent Not Running 9m5s Normal Info timestenclassic/ repsamplehelm Pod repsamplehelm-1 RepScheme Exists Normal StateChange 9m5s timestenclassic/ repsamplehelm Pod repsamplehelm-1 RepState IDLE 9m Normal Info timestenclassic/ repsamplehelm Pod repsamplehelm-1 Database Loaded 9m Normal Info timestenclassic/ repsamplehelm Pod repsamplehelm-1 RepAgent Running Normal Info timestenclassic/ 9m repsamplehelm Pod repsamplehelm-1 RepScheme Exists Normal StateChange timestenclassic/ 9m repsamplehelm Pod repsamplehelm-1 RepState STANDBY StateChange Normal timestenclassic/ 9m repsamplehelm Pod repsamplehelm-1 is Ready timestenclassic/ 8m59s Normal Upgrade repsamplehelm Upgrade of standby complete 8m59s Normal StateChange timestenclassic/ repsamplehelm TimesTenClassic was StandbyDown, now Normal timestenclassic/ 8m28s Upgrade Normal repsamplehelm Deleted active pod repsamplehelm-0 during upgrade 7m27s Warning Error timestenclassic/ repsamplehelm Pod repsamplehelm-0 Unreachable for 119 seconds 7m27s Normal StateChange timestenclassic/ repsamplehelm Pod repsamplehelm-0 is Not Ready 7m27s Normal StateChange timestenclassic/ repsamplehelm Pod repsamplehelm-0 is Not Active Ready 7m27s Warning StateChange timestenclassic/ repsamplehelm TimesTenClassic was Normal, now ActiveDown 7m27s timestenclassic/ Warning Failed repsamplehelm Pod repsamplehelm-0 was replaced 7m26s Normal Info timestenclassic/ repsamplehelm Pod repsamplehelm-1 Database Updatable 7m26s Normal StateChange timestenclassic/ repsamplehelm Pod repsamplehelm-1 RepState ACTIVE

7m26s Normal StateChange timestenclassic/ repsamplehelm Pod repsamplehelm-1 is Not Ready 7m26s StateChange Normal timestenclassic/ repsamplehelm TimesTenClassic was ActiveDown, now ActiveTakeover 7m21s Normal StateChange timestenclassic/ repsamplehelm Pod repsamplehelm-1 is Ready 7m21s Normal StateChange timestenclassic/ repsamplehelm Pod repsamplehelm-1 is Active Ready 7m21s Normal StateChange timestenclassic/ TimesTenClassic was ActiveTakeover, now StandbyDown repsamplehelm 6m17s Normal Info timestenclassic/ repsamplehelm Pod repsamplehelm-0 Agent Up 6m17s Normal Info timestenclassic/ repsamplehelm Pod repsamplehelm-0 Instance Exists 6m17s Normal Info timestenclassic/ repsamplehelm Pod repsamplehelm-0 Daemon Down timestenclassic/ 6m16s Normal Info repsamplehelm Pod repsamplehelm-0 Daemon Up Normal Info timestenclassic/ 6m16s repsamplehelm Pod repsamplehelm-0 Database Unloaded 6m14s Normal Info timestenclassic/ repsamplehelm Pod repsamplehelm-0 Database None Info timestenclassic/ 6m11s Normal repsamplehelm pollAsyncStatus: Async polling for RepDuplicate, timeout in 597 secs 6m8s Normal Info timestenclassic/ repsamplehelm pollAsyncStatus: Async polling for RepDuplicate, timeout in 594 secs 6m5s Info Normal timestenclassic/ repsamplehelm pollAsyncStatus: Async polling for RepDuplicate, timeout in 591 secs 6m2s Normal Info timestenclassic/ repsamplehelm pollAsyncStatus: Async polling for RepDuplicate, timeout in 588 secs 5m59s Normal Info timestenclassic/ RepDuplicate completed in 13 secs repsamplehelm 5m58s Normal Info timestenclassic/ repsamplehelm Pod repsamplehelm-0 Database Loaded 5m58s Normal Info timestenclassic/ repsamplehelm Pod repsamplehelm-0 RepAgent Not Running 5m58s Info timestenclassic/ Normal repsamplehelm Pod repsamplehelm-0 RepScheme Exists 5m58s Normal StateChange timestenclassic/ repsamplehelm Pod repsamplehelm-0 RepState IDLE 5m53s Info timestenclassic/ Normal repsamplehelm Pod repsamplehelm-0 Database Loaded 5m53s Normal Info timestenclassic/ repsamplehelm Pod repsamplehelm-0 RepAgent Running 5m53s Info timestenclassic/ Normal repsamplehelm Pod repsamplehelm-0 RepScheme Exists 5m53s Normal StateChange timestenclassic/ repsamplehelm Pod repsamplehelm-0 RepState STANDBY 5m53s Normal StateChange timestenclassic/ repsamplehelm Pod repsamplehelm-0 is Ready 5m52s Normal Upgrade timestenclassic/ repsamplehelm Upgrade of active complete

5m52sNormalUpgradetimestenclassic/repsamplehelmUpgradecompletedin 350secs5m52sNormalStateChangetimestenclassic/repsamplehelmTimesTenClassicwasStandbyDown, now Normal

The automated upgrade process completes. The TimesTenClassic object is in the Normal state. TimesTen databases are up and running and functioning properly. Active standby pair replication is configured between them.

- 7. After the upgrade, do the following:
 - a. Confirm the replicated TimesTenClassic object is in the Normal state and the active is repsamplehelm-1.

kubectl get ttc

The output is similar to the following:

repsamplehelm	Normal	repsamplehelm-1	11h
NAME	STATE	ACTIVE	AGE

b. Verify the image.

kubectl describe ttc repsamplehelm | grep Image

The output is similar to the following:

Image: container-registry.oracle.com/timesten/timesten:22.1.1.35.0
...

8. (Optional): Use helm test to test TimesTen.

helm test repsamplehelm

```
NAME: repsamplehelm
LAST DEPLOYED: Thu Jan 16 02:25:13 2025
NAMESPACE: default
STATUS: deployed
REVISION: 2
TEST SUITE:
              repsamplehelm-ttclassic-test
Last Started: Thu Jan 16 02:49:22 2025
Last Completed: Thu Jan 16 02:49:26 2025
Phase:
               Succeeded
NOTES:
Version 2211350.1.0 of the ttclassic chart has been installed.
This release is named "repsamplehelm".
To learn more about the release, try:
  $ helm status repsamplehelm
  $ helm get all repsamplehelm
```



```
$ helm history repsamplehelm
To rollback to a previous version of the chart, run:
  $ helm rollback repsamplehelm <REVISION>
  - run 'helm history repsamplehelm' for a list of revisions.
```

The test succeeeds.

Congratulations! You successfully completed an automated upgrade for a replicated TimesTenClassic object. The TimesTenClassic object is upgraded and is in the Normal state. The replicated active standby pair of TimesTen databases are running the new release and are fully operational.

Upgrade Non-Replicated TimesTen Databases

Let's perform an upgrade of a non-replicated TimesTenClassic object consisting of three independent TimesTen databases.

The upgrade process requires you to perform the following steps:

- Create a YAML file. In this file, you specify the container image that contains the new release of TimesTen.
- Use the helm values command for the upgrade. The example shows you how to do this.

In addition, the example shows you how to change the value for the <code>rollingUpdatePartition</code> variable.

1. On your development host, change to the helm directory that contains the new Helm charts.

cd new kube files/helm

Create a YAML file and in it specify the name of the TimesTen image you want to use for the upgrade.

vi customyaml/upgradenonreplicated.yaml

```
image:
    repository: container-registry.oracle.com/timesten/timesten
    tag: "22.1.1.35.0"
```

- 3. Before the upgrade, do the following:
 - a. Confirm the non-replicated TimesTenClassic object exists in your namespace.

kubectl get ttc

NAME	STATE	ACTIVE	AGE
norepsamplehelm	AllReplicasReady	N/A	10h
repsamplehelm	Normal	repsamplehelm-1	13h



The norepsamplehelm TimesTenClassic object exists and is in the AllReplicasReady state.

b. Verify the image and the value of rollingUpdatePartition.

```
kubectl get ttc norepsamplehelm -o yaml | grep 'image\|
rollingUpdatePartition'
```

The output is similar to the following:

```
image: container-registry.oracle.com/timesten/timesten:22.1.1.34.0
...
rollingUpdatePartition: 2
...
```

The value for rollingUpdatePartition is 2, indicating that Kubernetes upgrades Pods with an ordinal value greater than or equal to 2. For the norepsamplehelm object, since the value of replicas is 3, there are three Pods named norepsamplehelm-0, norepsamplehelm-1, and norepsamplehelm-2. Therefore, Kubernetes only upgrades the norepsamplehelm-2 Pod.

4. Confirm the ttclassic chart release.

helm list

The output is similar to the following:

```
NAMENAMESPACEREVISIONUPDATEDSTATUSCHARTAPP VERSIONnorepsamplehelm default117:42:47.180635098 +0000UTC deployed22.1.1.34.0
```

5. Upgrade to the new release. Use the ttclassic chart from the new release and use the customized YAML file that references the new image.

Note:

We recommend the following syntax. This ensures existing customizations are preserved.

helm get values norepsamplehelm --all > prev-values-norepttc.yaml

```
helm upgrade -f prev-values-norepttc.yaml -f customyaml/
upgradenonreplicated.yaml norepsamplehelm ./ttclassic
```

Let's look at this helm upgrade command:



- The get values norepsamplehelm Helm command retrieves the values for the current release, including existing customizations. The result of this command is piped into the prev-values-norepttc.yaml file. You can choose any name for this file.
- The helm upgrade command uses the prev-values-norepttc.yaml file with the customized upgradenonreplicated.yaml file to do the upgrade.

Let's look at the output from the helm upgrade command.

```
Release "norepsamplehelm" has been upgraded. Happy Helming!
NAME: norepsamplehelm
LAST DEPLOYED: Thu Jan 16 04:22:15 2025
NAMESPACE: default
STATUS: deployed
REVISION: 2
NOTES:
Version 2211350.1.0 of the ttclassic chart has been installed.
This release is named "norepsamplehelm".
To learn more about the release, try:
  $ helm status norepsamplehelm
  $ helm get all norepsamplehelm
  $ helm history norepsamplehelm
To rollback to a previous version of the chart, run:
  $ helm rollback norepsamplehelm <REVISION>
    - run 'helm history norepsamplehelm' for a list of revisions.
```

Note the following:

- The norepsamplehelm chart release is upgraded. The release revision is 2.
- The status of the release is deployed.
- The ttclassic chart version is 2211350.1.0 corresponding to TimesTen release 22.1.1.35.0.
- 6. Monitor the progress.

kubectl get ttc norepsamplehelm

The output is similar to the following:

NAME	STATE	ACTIVE	AGE
norepsamplehelm	SomeReplicasReady	N/A	10h

The object is in the SomeReplicasReady state.

Wait a few minutes. Then, monitor again.

```
kubectl get ttc norepsamplehelm
```

The output is similar to the following:

NAME	STATE	ACTIVE	AGE
norepsamplehelm	AllReplicasReady	N/A	10h

The object is in the AllReplicasReady state. All replicas are ready and available. TimesTen databases are up and running and functioning properly.

7. Check the image that the norepsamplehelm-2 Pod is running.

kubectl describe pod norepsamplehelm-2 | grep Image

The output is similar to the following:

```
Image:container-registry.oracle.com/timesten/timesten:22.1.1.35.0Image:container-registry.oracle.com/timesten/timesten:22.1.1.35.0Image:container-registry.oracle.com/timesten/timesten:22.1.1.35.0Image:container-registry.oracle.com/timesten/timesten:22.1.1.35.0
```

The containers in the Pod are running the new container image.

- 8. Check the image for the norepsamplehelm-1 and norepsamplehelm-0 Pods.
 - a. Check the norepsamplehelm-1 Pod.

kubectl describe pod norepsamplehelm-1 | grep Image

The output is similar to the following:

```
Image: container-registry.oracle.com/timesten/
timesten:22.1.1.34.0
Image: container-registry.oracle.com/timesten/
timesten:22.1.1.34.0
Image: container-registry.oracle.com/timesten/
timesten:22.1.1.34.0
Image: container-registry.oracle.com/timesten/
timesten:22.1.1.34.0
```

The containers in the Pod are not running the new image. Due to the rollingUpdatePartition value of 2, Kubernetes does not upgrade this Pod with the new image. This is correct behavior.

b. Check the norepsamplehelm-0 Pod.

kubectl describe pod norepsamplehelm-0 | grep Image

```
Image: container-registry.oracle.com/timesten/
timesten:22.1.1.34.0
Image: container-registry.oracle.com/timesten/
timesten:22.1.1.34.0
Image: container-registry.oracle.com/timesten/
timesten:22.1.1.34.0
```



```
Image: container-registry.oracle.com/timesten/
timesten:22.1.1.34.0
```

The containers in the Pod are not running the new image. Due to the rollingUpdatePartition value of 2, Kubernetes does not upgrade this Pod with the new image. This is correct behavior.

After you confirm the upgrade is working, you can upgrade the remaining Pods.

9. Create a YAML file and in it specify the rollingUpdatePartition variable and set the value to 0.

```
vi customyaml/changerollingupdate.yaml
```

```
rollingUpdatePartition: 0
```

10. Change the rollingUpdatePartition value.

Note:

We recommend the following syntax. This ensures existing customizations are preserved.

```
helm get values norepsamplehelm --all > prev-values-noreprolling.yaml &&
helm upgrade -f prev-values-noreprolling.yaml -f customyaml/
changerollingupdate.yaml norepsamplehelm ./ttclassic
```

```
Release "norepsamplehelm" has been upgraded. Happy Helming!
NAME: norepsamplehelm
LAST DEPLOYED: Thu Jan 16 04:56:32 2025
NAMESPACE: default
STATUS: deployed
REVISION: 3
NOTES:
Version 2211350.1.0 of the ttclassic chart has been installed.
This release is named "norepsamplehelm".
To learn more about the release, try:
  $ helm status norepsamplehelm
  $ helm get all norepsamplehelm
  $ helm history norepsamplehelm
To rollback to a previous version of the chart, run:
  $ helm rollback norepsamplehelm <REVISION>
    - run 'helm history norepsamplehelm' for a list of revisions.
```



Kubernetes automatically begins to terminate Pods and replace them with new ones. These new Pods run the new image. Since <code>rollingUpdatePartition</code> is now 0, you should expect to see the <code>norepsamplehelm-1</code> and <code>norepsamplehelm-0</code> Pods upgraded with the new image.

- **11**. Monitor the progress.
 - a. TimesTenClassic object:

kubectl get ttc norepsamplehelm

The output is similar to the following

NAME	STATE	ACTIVE	AGE
norepsamplehelm	SomeReplicasReady	N/A	11h

b. Pods:

kubectl get pods

The output is similar to the following:

NAME	READY	STATUS	RESTARTS	AGE
norepsamplehelm-0	3/3	Running	0	11h
norepsamplehelm-1	0/3	Init:0/1	0	31s
norepsamplehelm-2	3/3	Running	0	34m

Kubernetes replaces the norepsamplehelm-1 Pod first. Wait a few minutes. Then monitor again.

kubectl get pods

The output is similar to the following:

NAME	READY	STATUS	RESTARTS	AGE
norepsamplehelm-0	3/3	Running	0	3m14s
norepsamplehelm-1	3/3	Running	0	6m8s
norepsamplehelm-2	3/3	Running	0	40m

Kubernetes replaced the norepsamplehelm-1 and the norepsamplehelm-0 Pods. All Pods are running.

12. Confirm the Pods are running the new container image.

Pod norepsamplehelm-1:

kubectl describe pod norepsamplehelm-1 | grep Image

Image:	container-registry.oracle.com/timesten/timesten:22.1.1.35.0
Image:	<pre>container-registry.oracle.com/timesten/timesten:22.1.1.35.0</pre>



Image: container-registry.oracle.com/timesten/timesten:22.1.1.35.0 Image: container-registry.oracle.com/timesten/timesten:22.1.1.35.0

Pod norepsamplehelm-0:

kubectl describe pod norepsamplehelm-0 | grep Image

The output is similar to the following:

Image:	container-registry.oracle.com/timesten/timesten:22.1.1.35.0
Image:	<pre>container-registry.oracle.com/timesten/timesten:22.1.1.35.0</pre>
Image:	<pre>container-registry.oracle.com/timesten/timesten:22.1.1.35.0</pre>
Image:	<pre>container-registry.oracle.com/timesten/timesten:22.1.1.35.0</pre>

The Pods are running the new image.

13. Confirm the state of the TimesTenClassic object.

kubectl get ttc norepsamplehelm

The output is similar to the following:

NAME	STATE	ACTIVE	AGE
norepsamplehelm	AllReplicasReady	N/A	11h

Congratulations! You successfully performed an automated upgrade for a non-replicated TimesTenClassic object. All replicas are ready and available. Pods are running the new TimesTen container image. TimesTen databases are upgraded and fully operational.

Roll Back an Upgrade

This section shows you how to roll back a TimesTen upgrade as well as a TimesTen Operator upgrade. You cannot roll back an upgrade of TimesTen CRDs.

The section covers the following topics:

- Roll Back a Replicated TimesTen Upgrade
- Roll back a Non-Replicated TimesTen Upgrade
- Roll Back a TimesTen Operator Upgrade
Roll Back a Replicated TimesTen Upgrade

You can use Helm to roll back an upgrade of a replicated TimesTenClassic object. This reverts the replicated TimesTenClassic object and its associated TimesTen databases to the downgraded release.

Note:

Ensure you perform a downgrade at the appropriate time. We recommend that you do not perform a downgrade at the busiest time of a production day. Applications may experience outages and perhaps reduced performance as a result of the downgrade procedure.

- **1**. Before the rollback, do the following:
 - a. Confirm the replicated TimesTenClassic object exists in your namespace.

kubectl get ttc

The output is similar to the following:

NAME	STATE	ACTIVE	AGE
norepsamplehelm	AllReplicasReady	N/A	19h
repsamplehelm	Normal	repsamplehelm-1	22h

The repsamplehelm TimesTenClassic object exists and is in the Normal state.

b. Verify the image.

kubectl describe ttc repsamplehelm | grep Image

The output is similar to the following:

Image: container-registry.oracle.com/timesten/timesten:22.1.1.35.0
...

2. Review the revision history.

helm history repsamplehelm

The output is similar to the following:

REVISION	UPDATED				STATUS
CHART		APP	VERSION	DESCRIP:	FION
1	Thu Jan	16	14:58:40	2025	superseded
ttclassic-2211	340.1.0	22.2	L.1.34.0	Install	complete
2	Fri Jan	17	02:25:13	2025	deployed
ttclassic-2211	350.1.0	22.2	L.1.35.0	Upgrade	complete



Revision 2 of the repsamplehelm chart is running release 22.1.1.35.0. Revision 1 is running release 22.1.1.34.0.

3. Roll back to revision 1.

helm rollback repsamplehelm 1

The output is the following:

Rollback was a success! Happy Helming!

 Observe the downgrade and how the TimesTenClassic object transitions from and to various states.

kubectl get events -w

The output is similar to the following:

Upgrade Normal timestenclassic/ 17m repsamplehelm Image updated, automatic upgrade started 17m Normal Upgrade timestenclassic/ repsamplehelm Deleted standby pod repsamplehelm-0 during upgrade Warning Failed 17m timestenclassic/ repsamplehelm Pod repsamplehelm-0 was replaced Normal StateChange 17m timestenclassic/ repsamplehelm Pod repsamplehelm-0 is Not Ready Warning StateChange 17m timestenclassic/ repsamplehelm TimesTenClassic was Normal, now ActiveTakeover Normal StateChange 17m timestenclassic/ repsamplehelm TimesTenClassic was ActiveTakeover, now StandbyDown 14m Normal Info timestenclassic/ repsamplehelm Pod repsamplehelm-0 Agent Up Normal Info timestenclassic/ 14m repsamplehelm Pod repsamplehelm-0 Instance Exists Normal Info timestenclassic/ 14m repsamplehelm Pod repsamplehelm-0 Daemon Down 14m Normal Info timestenclassic/ repsamplehelm Pod repsamplehelm-0 Daemon Up Normal Info timestenclassic/ 14m repsamplehelm Pod repsamplehelm-0 Database Unloaded 14m Normal Info timestenclassic/ repsamplehelm Pod repsamplehelm-0 Database None 14m Normal Info timestenclassic/ repsamplehelm pollAsyncStatus: Async polling for RepDuplicate, timeout in 597 secs 14m Normal Info timestenclassic/ repsamplehelm pollAsyncStatus: Async polling for RepDuplicate, timeout in 594 secs 14m Normal Info timestenclassic/ repsamplehelm pollAsyncStatus: Async polling for RepDuplicate, timeout in 591 secs 14m Normal Info timestenclassic/ repsamplehelm pollAsyncStatus: Async polling for RepDuplicate, timeout in 588 secs 14m Normal Info timestenclassic/

```
repsamplehelm RepDuplicate completed in 14 secs
14m
    Normal
                  Info
                                         timestenclassic/
repsamplehelm Pod repsamplehelm-0 Database Loaded
14m
         Normal
                  Info
                                         timestenclassic/
repsamplehelm Pod repsamplehelm-0 RepAgent Not Running
14m
         Normal
                  Info
                                         timestenclassic/
repsamplehelm Pod repsamplehelm-0 RepScheme Exists
       Normal
                  StateChange
14m
                                         timestenclassic/
repsamplehelm Pod repsamplehelm-0 RepState IDLE
14m
    Normal
                  Info
                                         timestenclassic/
repsamplehelm Pod repsamplehelm-0 Database Loaded
14m
         Normal Info
                                         timestenclassic/
repsamplehelm Pod repsamplehelm-0 RepAgent Running
14m
         Normal Info
                                         timestenclassic/
repsamplehelm Pod repsamplehelm-0 RepScheme Exists
      Normal StateChange
14m
                                         timestenclassic/
repsamplehelm Pod repsamplehelm-0 RepState STANDBY
    Normal StateChange
14m
                                         timestenclassic/
repsamplehelm Pod repsamplehelm-0 is Ready
14m
         Normal Upgrade
                                         timestenclassic/
repsamplehelm Upgrade of standby complete
14m
    Normal StateChange
                                         timestenclassic/
repsamplehelm TimesTenClassic was StandbyDown, now Normal
      Normal Upgrade
                                         timestenclassic/
14m
repsamplehelm Deleted active pod repsamplehelm-1 during upgrade
14m
        Warning Failed
                                         timestenclassic/
repsamplehelm Pod repsamplehelm-1 has failed
14m
         Normal
                  StateChange
                                         timestenclassic/
repsamplehelm Pod repsamplehelm-1 is Not Ready
        Normal StateChange
                                        timestenclassic/
14m
repsamplehelm Pod repsamplehelm-1 is Not Active Ready
        Warning StateChange
14m
                                        timestenclassic/
repsamplehelm TimesTenClassic was Normal, now ActiveDown
13m
         Warning Failed
                                         timestenclassic/
repsamplehelm Pod repsamplehelm-1 was replaced
13m
         Normal Info
                                         timestenclassic/
repsamplehelm Pod repsamplehelm-0 Database Updatable
     Normal StateChange
13m
                                         timestenclassic/
repsamplehelm Pod repsamplehelm-0 RepState ACTIVE
   Normal StateChange
13m
                                         timestenclassic/
repsamplehelm Pod repsamplehelm-0 is Not Ready
13m
      Normal StateChange
                                         timestenclassic/
repsamplehelm TimesTenClassic was ActiveDown, now ActiveTakeover
13m
         Normal
                  StateChange
                                        timestenclassic/
repsamplehelm Pod repsamplehelm-0 is Ready
         Normal
                  StateChange
                                         timestenclassic/
13m
repsamplehelm Pod repsamplehelm-0 is Active Ready
13m
         Normal
                  StateChange
                                         timestenclassic/
repsamplehelm TimesTenClassic was ActiveTakeover, now StandbyDown
12m
          Normal
                  Info
                                         timestenclassic/
repsamplehelm Pod repsamplehelm-1 Agent Up
    Normal
12m
                  Info
                                         timestenclassic/
repsamplehelm Pod repsamplehelm-1 Instance Exists
                                         timestenclassic/
12m
    Normal
                  Info
repsamplehelm Pod repsamplehelm-1 Daemon Down
12m
                  Info
                                         timestenclassic/
        Normal
```

repsamplehelm Pod repsamplehelm-1 Daemon Up 12m Normal Info timestenclassic/ repsamplehelm Pod repsamplehelm-1 Database Unloaded 12m Normal Info timestenclassic/ repsamplehelm Pod repsamplehelm-1 Database None 12m Normal Info timestenclassic/ repsamplehelm pollAsyncStatus: Async polling for RepDuplicate, timeout in 597 secs 12m Normal Info timestenclassic/ repsamplehelm pollAsyncStatus: Async polling for RepDuplicate, timeout in 594 secs 11m Normal Info timestenclassic/ repsamplehelm pollAsyncStatus: Async polling for RepDuplicate, timeout in 591 secs 11m Normal Info timestenclassic/ repsamplehelm pollAsyncStatus: Async polling for RepDuplicate, timeout in 588 secs 11m Normal Info timestenclassic/ repsamplehelm RepDuplicate completed in 15 secs 11m Normal Info timestenclassic/ repsamplehelm Pod repsamplehelm-1 Database Loaded 11m Normal Info timestenclassic/ repsamplehelm Pod repsamplehelm-1 RepAgent Not Running Normal Info timestenclassic/ 11m repsamplehelm Pod repsamplehelm-1 RepScheme Exists 11m Normal StateChange timestenclassic/ repsamplehelm Pod repsamplehelm-1 RepState IDLE 11m Normal Info timestenclassic/ repsamplehelm Pod repsamplehelm-1 Database Loaded Normal Info timestenclassic/ 11m repsamplehelm Pod repsamplehelm-1 RepAgent Running Info 11m Normal timestenclassic/ repsamplehelm Pod repsamplehelm-1 RepScheme Exists 11m Normal StateChange timestenclassic/ repsamplehelm Pod repsamplehelm-1 RepState STANDBY timestenclassic/ 11m Normal StateChange repsamplehelm Pod repsamplehelm-1 is Ready Normal Upgrade timestenclassic/ 11m repsamplehelm Upgrade completed in 326 secs StateChange 11m Normal timestenclassic/ repsamplehelm TimesTenClassic was StandbyDown, now Normal

The downgrade process completes. The TimesTenClassic object is in the Normal state. TimesTen databases are up and running and functioning properly. Active standby pair replication is configured between them.

- 5. After the downgrade, do the following:
 - a. Confirm the replicated TimesTenClassic object is in the Normal state and the active is repsamplehelm-0.

kubectl get ttc



NAME	STATE	ACTIVE	AGE
repsamplehelm	Normal	repsamplehelm-0	23h

The object is in the Normal state. The active is repsamplehelm-0.

b. Verify the image.

kubectl describe ttc repsamplehelm | grep Image

The output is similar to the following:

Image: container-registry.oracle.com/timesten/timesten:22.1.1.34.0
...

6. Check the history.

helm history repsamplehelm

The output is similar to the following:

REVISION UPDATED STATUS APP VERSION DESCRIPTION CHART 1 Thu Jan 16 14:58:40 2025 superseded ttclassic-2211340.1.0 22.1.1.34.0 Install complete 2 Fri Jan 17 02:25:13 2025 superseded ttclassic-2211350.1.0 22.1.1.35.0 Upgrade complete 3 Fri Jan 17 13:54:33 2025 deployed ttclassic-2211340.1.0 22.1.1.34.0 Rollback to 1

Revision 3 is downgraded.

7. Confirm the repsamplehelm release is downgraded.

helm list

The output is similar to the following:

NAME NAMESPACE REVISION UPDATED STATUS CHART APP VERSION repsamplehelm default 3 2025-01-16 13:54:33.909336613 +0000 UTC deployed ttclassic-2211340.1.0 22.1.1.34.0 ...

The repsamplehelm chart release is downgraded.

Congratulations! You successfully completed the rollback. The TimesTenClassic object is downgraded to the desired release and is in the Normal state. The replicated active standby pair of TimesTen databases are running the downgraded release and are fully operational.



Roll back a Non-Replicated TimesTen Upgrade

You can use Helm to roll back an upgrade of a non-replicated TimesTenClassic object. This reverts the non-replicated TimesTenClassic object and its associated TimesTen databases to the downgraded release.

- 1. Before the rollback, do the following:
 - a. Confirm the non-replicated TimesTenClassic object exists in your namespace.

kubectl get ttc

The output is similar to the following:

NAME	STATE	ACTIVE	AGE
norepsamplehelm	AllReplicasReady	N/A	20h
repsamplehelm	Normal	repsamplehelm-0	23h

The norepsamplehelm TimesTenClassic object exists and is in the AllReplicasReady state.

b. Verify the image and the value of rollingUpdatePartition.

```
kubectl get ttc norepsamplehelm -o yaml | grep 'image\|
rollingUpdatePartition'
```

The output is similar to the following:

```
image: container-registry.oracle.com/timesten/timesten:22.1.1.35.0
...
rollingUpdatePartition: 0
...
```

The current value for rollingUpdatePartition is 0. However, the value used for the downgrade is the original value of rollingUpdatePartition. In the example, the original value is 2. If you roll back to the original chart release in which the rollingUpdatePartition value is 2, Kubernetes only downgrades the -2 Pod. An upcoming step illustrates this.

2. Review the revision history.

helm history norepsamplehelm

The output is similar to the following:

REVISIONUPDATEDSTATUSCHARTAPP VERSIONDESCRIPTION1Thu Jan16 17:42:472025supersededttclassic-2211340.1.022.1.1.34.0Install complete2Fri Jan17 04:22:152025supersededttclassic-2211350.1.022.1.1.35.0Upgrade complete



3 Fri Jan 17 04:56:32 2025 deployed ttclassic-2211350.1.0 22.1.1.35.0 Upgrade complete

Revision 3 of the norepsamplehelm chart is running release 22.1.1.35.0. Revision 1 is running release 22.1.1.34.0.

3. Roll back to revision 1.

helm rollback norepsamplehelm 1

The output is the following:

Rollback was a success! Happy Helming!

4. Monitor the progress.

kubectl get ttc norepsamplehelm

The output is similar to the following:

NAME	STATE	ACTIVE	AGE
norepsamplehelm	SomeReplicasReady	N/A	20h

The object is in the SomeReplicasReady state.

Wait a few minutes. Then, monitor again.

kubectl get ttc norepsamplehelm

The output is similar to the following:

NAME	STATE	ACTIVE	AGE
norepsamplehelm	AllReplicasReady	N/A	20h

The object is in the AllReplicasReady state. All replicas are ready and available. TimesTen databases are up and running and functioning properly.

- 5. Check the rollingUpdatePartition value and the image for the Pods.
 - a. Check the value of rollingUpdatePartition.

kubectl get ttc norepsamplehelm -o yaml | grep rollingUpdatePartition

The output is similar to the following:

rollingUpdatePartition: 2

As part of the downgrade process, the value of rollingUpdatePartition changes to its original value of 2.



b. Check the norepsamplehelm-2 Pod.

kubectl describe pod norepsamplehelm-2 | grep Image

The output is similar to the following:

```
Image: container-registry.oracle.com/timesten/
timesten:22.1.1.34.0
Image: container-registry.oracle.com/timesten/
timesten:22.1.1.34.0
Image: container-registry.oracle.com/timesten/
timesten:22.1.1.34.0
Image: container-registry.oracle.com/timesten/
timesten:22.1.1.34.0
```

The containers in the Pod are running the downgraded image. Due to the rollingUpdatePartition value of 2, Kubernetes updated this Pod with the downgraded image. This is correct behavior.

c. Check the norepsamplehelm-1 Pod.

kubectl describe pod norepsamplehelm-1 | grep Image

The output is similar to the following:

```
Image: container-registry.oracle.com/timesten/
timesten:22.1.1.35.0
Image: container-registry.oracle.com/timesten/
timesten:22.1.1.35.0
Image: container-registry.oracle.com/timesten/
timesten:22.1.1.35.0
Image: container-registry.oracle.com/timesten/
timesten:22.1.1.35.0
```

The containers in the Pod are not running the downgraded image. Due to the rollingUpdatePartition value of 2, Kubernetes does not update this Pod with the downgraded image. This is correct behavior.

d. Check the norepsamplehelm-0 Pod.

kubectl describe pod norepsamplehelm-0 | grep Image

The output is similar to the following:

```
Image: container-registry.oracle.com/timesten/
timesten:22.1.1.35.0
Image: container-registry.oracle.com/timesten/
timesten:22.1.1.35.0
Image: container-registry.oracle.com/timesten/
timesten:22.1.1.35.0
Image: container-registry.oracle.com/timesten/
timesten:22.1.1.35.0
```



The containers in the Pod are not running the downgraded image. Due to the rollingUpdatePartition value of 2, Kubernetes does not updated this Pod with the downgraded image. This is correct behavior.

After you confirm the downgrade is working, you can downgrade the remaining Pods.

6. Create a new YAML file and in it specify the rollingUpdatePartition variable and set the value to 0.

vi customyaml/changerollingupdate0.yaml

rollingUpdatePartition: 0

7. Change the rollingUpdatePartition value.

Note:

We recommend the following syntax. This ensures existing customizations are preserved.

```
helm get values norepsamplehelm --all > prev-values-noreprolling0.yaml &&
helm upgrade -f prev-values-noreprolling0.yaml -f customyaml/
changerollingupdate0.yaml norepsamplehelm ./ttclassic
```

The output is similar to the following:

```
Release "norepsamplehelm" has been upgraded. Happy Helming!
NAME: norepsamplehelm
LAST DEPLOYED: Fri Jan 17 15:09:42 2025
NAMESPACE: default
STATUS: deployed
REVISION: 5
NOTES:
Version 2211340.1.0 of the ttclassic chart has been installed.
This release is named "norepsamplehelm".
To learn more about the release, try:
  $ helm status norepsamplehelm
  $ helm get all norepsamplehelm
  $ helm history norepsamplehelm
To rollback to a previous version of the chart, run:
  $ helm rollback norepsamplehelm <REVISION>
    - run 'helm history norepsamplehelm' for a list of revisions.
```

Kubernetes automatically begins to terminate Pods and replace them with new ones. These new Pods run the downgraded image. Since <code>rollingUpdatePartition</code> is now 0, you should expect to see the <code>norepsamplehelm-1</code> and <code>norepsamplehelm-0</code> Pods updated with the downgraded image.



8. Monitor the progress.

a. TimesTenClassic object:

kubectl get ttc norepsamplehelm

The output is similar to the following

NAME	STATE	ACTIVE	AGE
norepsamplehelm	SomeReplicasReady	N/A	21h

b. Pods:

kubectl get pods

The output is similar to the following:

NAME	READY	STATUS	RESTARTS
AGE norepsamplehelm-0	3/3	Running	0
10h norepsamplehelm-1	0/3	Init:0/1	0
73s norepsamplehelm-2 34m	3/3	Running	0

Kubernetes replaces the norepsamplehelm-1 Pod first. Wait a few minutes. Then monitor again.

kubectl get pods

The output is similar to the following:

NAME	READY	STATUS	RESTARTS
AGE			
norepsamplehelm-0	3/3	Running	0
7m24s norepsamplehelm-1	3/3	Running	0
9m49s	5/5	Ruillillig	0
norepsamplehelm-2	3/3	Running	0
42m		-	

Kubernetes replaced the norepsamplehelm-1 and the norepsamplehelm-0 Pods. All Pods are running.

9. Confirm the Pods are running the downgraded container image.

```
Pod norepsamplehelm-1:
```

```
kubectl describe pod norepsamplehelm-1 | grep Image
```



Image:	container-registry.oracle.com/timesten/timesten:22.1.1.34.0
Image:	<pre>container-registry.oracle.com/timesten/timesten:22.1.1.34.0</pre>
Image:	<pre>container-registry.oracle.com/timesten/timesten:22.1.1.34.0</pre>
Image:	<pre>container-registry.oracle.com/timesten/timesten:22.1.1.34.0</pre>

Pod norepsamplehelm-0:

kubectl describe pod norepsamplehelm-0 | grep Image

The output is similar to the following:

Image:	<pre>container-registry.oracle.com/timesten/timesten:22.1.1.34.0</pre>
Image:	<pre>container-registry.oracle.com/timesten/timesten:22.1.1.34.0</pre>
Image:	<pre>container-registry.oracle.com/timesten/timesten:22.1.1.34.0</pre>
Image:	<pre>container-registry.oracle.com/timesten/timesten:22.1.1.34.0</pre>

The Pods are running the new image.

10. Confirm the value of rollingUpdatePartition.

kubectl get ttc norepsamplehelm -o yaml | grep rollingUpdatePartition

The output is similar to the following:

rollingUpdatePartition: 0

11. Confirm the state of the TimesTenClassic object.

kubectl get ttc norepsamplehelm

The output is similar to the following:

NAME	STATE	ACTIVE	AGE
norepsamplehelm	AllReplicasReady	N/A	21h

12. Check the history.

helm history norepsamplehelm

The output is similar to the following:

REVISION	UPDATED				STATUS
CHART		APP	VERSION	DESCRIP	FION
1	Thu Jan	16	17:42:47	2025	superseded
ttclassic-221	1340.1.0	22.2	1.1.34.0	Install	complete
2	Fri Jan	17	04:22:15	2025	superseded
ttclassic-221	1350.1.0	22.2	1.1.35.0	Upgrade	complete
3	Fri Jan	17	04:56:32	2025	superseded
ttclassic-221	1350.1.0	22.2	1.1.35.0	Upgrade	complete
4	Fri Jan	17	14:36:43	2025	superseded



ttclassic-2211340.1.0 22.1.1.34.0 Rollback to 1 5 Fri Jan 17 15:09:42 2025 deployed ttclassic-2211340.1.0 22.1.1.34.0 Upgrade complete

Revision 5 is running the 2211340.1.0 chart.

13. Confirm the norepsamplehelm release is downgraded.

helm list

The output is similar to the following:

NAMENAMESPACEREVISIONUPDATEDSTATUSCHARTAPP VERSIONnorepsamplehelm default52025-01-0715:09:42.959017268 +0000 UTC deployedttclassic-2211340.1.022.1.1.34.0

The norepsamplehelm chart release is downgraded.

Congratulations! You successfully completed the rollback. The TimesTenClassic object is downgraded to the desired release and is in the AllReplicasReady state. All replicas are ready and available. Pods are running the downgraded TimesTen container image. TimesTen databases are upgraded and fully operational.

Roll Back a TimesTen Operator Upgrade

You can use Helm to roll back an upgrade of the TimesTen Operator.

- 1. Before the rollback, do the following:
 - a. Confirm the TimesTen Operator is running in your namespace.

kubectl get pods

The output is similar to the following:

NAME	READY	STATUS	RESTARTS
AGE			
 timesten-operator-57b7949f97-xdlwl 15h	1/1	Running	0

The TimesTen Operator is running in your namespace.

b. Verify the image.

kubectl describe deployment timesten-operator | grep Image

Image: container-registry.oracle.com/timesten/timesten:22.1.1.35.0

2. Review the current release of the TimesTen Operator chart.

helm list

The output is similar to the following:

NAME NAMESPACE REVISION UPDATED STATUS CHART APP VERSION ... ttoper default 2 2025-01-16 00:20:34.757004974 +0000 UTC deployed ttoperator-2211350.1.0 22.1.1.35.0

The chart's release is 22.1.1.35.0.1.0.

3. Review the revision history.

helm history ttoper

The output is similar to the following:

REVISIONUPDATEDSTATUSCHARTAPP VERSIONDESCRIPTION1Thu Jan16 02:54:39 2025supersededttoperator-2211340.1.022.1.1.34.0Install complete2Fri Jan17 00:20:34 2025deployedttoperator-2211350.1.022.1.1.35.0Upgrade complete

Revision 2 of the ttoper chart is running release 22.1.1.35.0. Revision 1 is running release 22.1.1.34.0.

4. Roll back to revision 1.

helm rollback ttoper 1

Output.

Rollback was a success! Happy Helming!

- 5. After the downgrade, do the following:
 - a. Verify that the TimesTen Operator is running.

kubectl get pods



NAME	READY	STATUS	RESTARTS	AGE
•••				
timesten-operator-55c6f99-2djfr	1/1	Running	0	4m13s

There is a new TimesTen Operator running.

b. Confirm the TimesTen Operator is running the downgraded image.

kubectl describe deployment timesten-operator | grep Image

The output is the following:

```
Image: container-registry.oracle.com/timesten/timesten:22.1.1.34.0
```

The TimesTen Operator is running the downgraded release.

c. Confirm the state of the TimesTenClassic objects.

kubectl get ttc

The output is similar to the following:

NAME	STATE	ACTIVE	AGE
norepsamplehelm	AllReplicasReady	N/A	22h
repsamplehelm	Normal	repsamplehelm-0	25h

The TimesTen Operator resumes the management and monitoring of the TimesTenClassic objects. The objects and the associated TimesTen databases are functioning properly.

6. Confirm the revision history.

helm history ttoper

The output is similar to the following:

 REVISION
 UPDATED
 STATUS

 CHART
 APP VERSION
 DESCRIPTION

 1
 Thu Jan
 16 02:54:39
 2025
 superseded

 ttoperator-2211340.1.0
 22.1.1.34.0
 Install complete

 2
 Fri Jan
 16 00:20:34
 2025
 superseded

 ttoperator-2211350.1.0
 22.1.1.35.0
 Upgrade complete

 3
 Fri Jan
 17 16:11:37
 2025
 deployed

 ttoperator-2211340.1.0
 22.1.1.34.0
 Rollback to 1

Revision 3 of the chart is running release 2211340.

7. Confirm the chart's release is downgraded.

helm list



NAME NAMESPACE REVISION UPDATED STATUS CHART APP VERSION ... ttoper default 3 2025-01-17 16:11:37.408437395 +0000 UTC deployed ttoperator-2211340.1.0 22.1.1.34.0

The chart contains the correct release.

Congratulations! You successfully completed the rollback. The TimesTen Operator is using the downgraded image and is functioning properly. It has resumed management and monitoring of TimesTenClassic objects.

Clean Up

This section shows you how to delete replicated and non-replicated TimesTenClassic objects and associated databases from your namespace. It also shows you how to delete the TimesTen Operator Deployment and TimesTen CRDs. It covers the following topics:

- About Uninstalling a Release
- Delete TimesTen Databases
- Delete the TimesTen Operator
- Delete the TimesTen CRDs

About Uninstalling a Release

You can uninstall a release by using the helm uninstall command. This command deletes the Kubernetes objects created by the helm install command. Uninstalling a release results in TimesTen databases being deleted or being unmanaged or both. You can also uninstall the TimesTen Operator and the TimesTen CRDs. If you uninstall TimesTen CRDs, Kubernetes deletes the CRDs along with all TimesTenClassic objects associated with the CRDs.

The examples are for demonstration purposes.

Delete TimesTen Databases

You can uninstall replicated and non-replicated TimesTenClassic objects. This results in the deletion of TimesTen databases associated with the object. Since the Persistent Volume Claims (PVCs) are not automatically deleted, you must manually delete them. The example shows you how to do this.

1. Confirm the TimesTenClassic objects that are running in your namespace.

kubectl get ttc



NAME	STATE	ACTIVE	AGE
norepsamplehelm	AllReplicasReady	N/A	24h
repsamplehelm	Normal	repsamplehelm-0	27h

2. List the releases.

helm list

The output is similar to the following:

NAME NAMESPACE REVISION UPDATED STATUS APP VERSION CHART norepsamplehelm default 5 2025-01-16 15:09:42.959017268 +0000 UTC deployed ttclassic-2211340.1.0 22.1.1.34.0 13:54:33.909336613 +0000 UTC deployed 22.1.1.34 0 repsamplehelm default 3 2025-01-16 ttclassic-2211340.1.0 22.1.1.34.0 default 3 2025-01-16 ttoper 16:11:37.408437395 +0000 UTC deployed ttoperator-2211340.1.0 22.1.1.34.0

The norepsamplehelm and repsamplehelm releases exist in your namespace.

- 3. Uninstall the release for the replicated TimesTenClassic object.
 - a. Uninstall.

helm uninstall repsamplehelm

The output is the following:

release "repsamplehelm" uninstalled

b. Confirm the TimesTenClassic object no longer exists.

kubectl get ttc repsamplehelm

The output is the following:

Error from server (NotFound): timestenclassics.timesten.oracle.com
"repsamplehelm" not found

c. Confirm the ConfigMap is deleted.

kubectl get configmap repsamplehelm



The output is the following:

Error from server (NotFound): configmaps "repsamplehelm" not found

d. Delete the PVCs.

kubectl get pvc

The output is the following:

```
NAME
                               STATUS
VOLUME
                                       CAPACITY ACCESS MODES
STORAGECLASS
             AGE
tt-persistent-repsamplehelm-0
                               Bound csi-4ef3c13e-ae49-4604-8586-
b78343142481 50Gi
                  RWO
                                     oci-bv
                                                   28h
tt-persistent-repsamplehelm-1
                               Bound
csi-6786e82d-0bde-458f-8acd-047308e391a0
                                       50Gi
                                                 RWO
oci-bv
            28h
. . .
```

Delete:

kubectl delete pvc tt-persistent-repsamplehelm-0

kubectl delete pvc tt-persistent-repsamplehelm-1

- 4. Uninstall the release for the non-replicated TimesTenClassic objects.
 - a. Uninstall.

helm uninstall norepsamplehelm

The output is the following:

release "norepsamplehelm" uninstalled

b. Confirm the TimesTenClassic object no longer exists.

kubectl get ttc norepsamplehelm

The output is the following:

Error from server (NotFound): timestenclassics.timesten.oracle.com
"norepsamplehelm" not found

c. Confirm the ConfigMap is deleted.

kubectl get configmap repsamplehelm



The output is the following:

Error from server (NotFound): configmaps "norepsamplehelm" not found

d. Delete the PVCs.

kubectl get pvc

The output is the following:

NAME STATUS VOLUME CAPACITY ACCESS MODES STORAGECLASS AGE tt-persistent-norepsamplehelm-0 Bound csi-b5302a0c-f533-418e-a152a9399ed2be7b 50Gi RWO oci-bv 25h tt-persistent-norepsamplehelm-1 Bound csi-cb4b0e04-16f1-4e87a952-781ac213ff77 50Gi RWO oci-bv 25h tt-persistent-norepsamplehelm-2 Bound csi-353d4f18-50fe-4bd6-8d57fe17b973e0be 50Gi RWO oci-bv 25h

Delete:

kubectl delete pvc tt-persistent-norepsamplehelm-0

kubectl delete pvc tt-persistent-norepsamplehelm-1

kubectl delete pvc tt-persistent-norepsamplehelm-2

You successfully deleted the TimesTenClassic objects and associated databases from your namespace.

Delete the TimesTen Operator

You can uninstall the TimesTen Operator from your namespace.

1. Confirm the TimesTen Operator is running.

kubectl get pods

The output is similar to the following:

NAME	READY	STATUS	RESTARTS	AGE
timesten-operator-55c6f99-2djfr	1/1	Running	0	3h1m

2. List the release.

helm list



NAME NAMESPACE REVISION UPDATED STATUS CHART APP VERSION ttoper default 3 2025-01-16 16:11:37.408437395 +0000 UTC deployed ttoperator-2211340.1.0 22.1.1.34.0

The ttoper release exists in your namespace.

3. Uninstall.

helm uninstall ttoper

The output is similar to the following:

release "ttoper" uninstalled

4. Confirm the TimesTen Operator Pod is deleted.

kubectl get pods

The output is similar to the following:

No resources found in default namespace.

5. Confirm the TimesTen Operator Deployment is deleted.

kubectl get deployment timesten-operator

The output is similar to the following:

Error from server (NotFound): deployments.apps "timesten-operator" not found

You successfully deleted The TimesTen Operator. It is no longer running in your namespace.

Delete the TimesTen CRDs

You can delete TimesTen CRDs from your cluster.

Note:

Use caution when deleting the TimesTen CRDs. This action causes the deletion of all TimesTenClassic objects and associated databases that are running in all namespaces.



1. List the release.

helm list

The output is similar to the following:

```
NAME NAMESPACE REVISION

UPDATED STATUS CHART

APP VERSION

ttcrd default 2 2025-01-16 19:07:52.249429601

+0000 UTC deployed ttcrd-2211350.1.0 22.1.1.35.0
```

2. Uninstall.

helm uninstall ttcrd

The output is similar to the following:

release "ttcrd" uninstalled

3. Confirm the TimesTen CRDs no longer exist in the cluster.

kubectl get crd | grep timesten

You successfully deleted the TimesTen CRDs.



10 Use TimesTen Databases

This chapter explains how to use direct mode applications and Client/Server drivers to access and use TimesTen Classic databases.

Topics:

- About Using Direct Mode Applications
- About Using Client/Server Drivers

About Using Direct Mode Applications

You can run direct mode applications inside of the Pods in your TimesTenClassic and TimesTenScaleout deployments. For information on direct mode applications, see Managing TimesTen Databases in the Oracle TimesTen In-Memory Database Operations Guide.

TimesTen Pods are created with the Kubernetes *shareProcessNamespace* option. This option allows direct mode applications running in other containers within the same Pod to function properly.

Note:

The standard security issues that surround direct mode apply in this environment as in a non-Kubernetes environment. Segregating your applications into separate containers from TimesTen is intended for ease of management and ease of upgrade. It is not intended as a security barrier and provides no additional security.

Use the .spec.template.spec.containers attribute of your TimesTenClassic object or the .spec.dataTemplate.spec.containers of your TimesTenScaleout object to cause one or more containers to be created within each of the TimesTen Pods that runs the tt container. Such containers are created in addition to the tt container that runs TimesTen.

This example illustrates how to include the .spec.template.spec.containers attribute in your TimesTenClassic object definition. For a TimesTenScaleout object, replace template with one or more dataTemplates.

```
apiVersion: timesten.oracle.com/v4
kind: TimesTenClassic
metadata:
    name: directmode
spec:
    ttspec:
    storageClassName: oci-bv
    storageSize: 250Gi
    image: container-registry.oracle.com/timesten/timesten:22.1.1.34.0
    imagePullSecret: sekret
    dbConfigMap:
    - directmode
template:
```



spec: containers: name: yourapp image: phx.ocir.io/youraccount/yourapplication:2 command: ["/bin/yourapp"] name: anotherapp image: phx.ocir.io/youraccount/anotherapplication:2 command: ["/bin/anotherapp"]

You can specify any other Kubernetes configuration for these containers.

The Operator automatically adds appropriate mounts to the containers. This gives your containers the ability to access TimesTen.

To use TimesTen in direct mode, your application containers must know how TimesTen is configured in the tt container. You must configure your application containers similarly.

In particular:

- You must know the name of the TimesTen users group. If you are using a container image located at container-registry.oracle.com/timesten, the name of the TimesTen users group is timesten. If you built the TimesTen container image, and changed the timesten default, ensure you use the name you used when you built the container image.
- You must know the name of the Linux operating system user that runs TimesTen. The default is timesten. As was mentioned in the previous bullet, if you changed this default, ensure you use the name you used when you built the container image.
- You must configure your application containers to run your applications as a member of the TimesTen users group. Only members of this group can run TimesTen in direct mode.
- You can run your direct mode applications as a user with the same UID as that of the TimesTen user that runs TimesTen (3429 is the default) . However, this grants the application instance administrator permissions on the TimesTen instance. Alternatively, you can create a group with the same GID as that of the TimesTen users group of and then create a user whose primary or secondary group is that group, but with a UID that is not the UID of the TimesTen user that is running TimesTen. In this case, you can run your application as this user and also use TimesTen in direct mode. You can then grant this user privileges up to and including the ADMIN privilege. For more information on primary and secondary groups, see Creating an Installation on Linux/UNIX in the Oracle TimesTen In-Memory Database Installation, Migration, and Upgrade Guide. For information on TimesTen privileges, see System Privileges and Object Privileges in the Oracle TimesTen In-Memory Database SQL Reference.
- The direct mode application must use the TimesTen instance that is configured at /tt/ home/timesten/instances/instance1. The scripts to configure the TimesTen environment variables are located at /tt/home/timesten/instances/instance1/bin/ttenv.*.
- Do not modify any file that is located in the TimesTen instance. In addition, ensure you do not create any new files in the \$TIMESTEN_HOME directory tree of the instance.
- Do not add entries to the /tt/home/timesten/instances/instance1/conf/sys.odbc.ini file. These files can be overwritten by the Operator. However, you can store your own DSN entries in the \$HOME/.odbc.ini file located in your application container.
- Do not create additional TimesTen databases.

Kubernetes, not the Operator, is responsible for monitoring and managing the life cycle of the direct mode containers. In particular:



- Applications are started by Kubernetes regardless of the state of TimesTen (located in its own container). Kubernetes manages the life cycle of containers individually. It does not sequence. Your application must know how to wait for TimesTen to become available.
- For replicated TimesTenClassic objects, a direct mode application runs in the Pod containing the active TimesTen database and in the Pod containing the standby TimesTen database. The application may need to use the ttRepStateGet built-in procedure to determine whether it is running on the active or on the standby and perhaps quiesce itself on the standby. For more information on the ttRepStateGet built-in procedure, see ttRepStateGet in the Oracle TimesTen In-Memory Database Reference.
- Kubernetes may start the application before the TimesTen database exists or before it is loaded into memory and ready for use. It is the responsibility of the direct mode application to verify the state of the TimesTen database in its Pod and to use it appropriately.
- If your application exits, the container terminates, and Kubernetes spawns another container. This does not impact TimesTen that is running in the tt container.

About Using Client/Server Drivers

Applications that are running in other Pods in your Kubernetes cluster can use your TimesTen database by using the standard TimesTen Client/Server drivers. You must configure your application containers with a TimesTen client instance. That instance must contain a configured *\$TIMESTEN_HOME/conf/sys.odbc.ini* file, or your application must use an appropriate Client/Server connection string.

In TimesTen Classic, if you chose to configure a sys.odbc.ini file, the contents of sys.odbc.ini contains a client DSN definition that references the Pods that are running your TimesTen databases. In TimesTen Scaleout, you can export a sys.odbc.ini file for use by client/server clients outside of the grid. Use the ttGridAdmin utility with the gridClientExport or gridClientExportAll for this purpose. See ttGridAdmin in the Oracle TimesTen In-Memory Database Reference.

This example creates the sample DSN and references the sample TimesTenClassic object in the mynamespace namespace.

```
% vi $TIMESTEN_HOME/conf/sys.odbc.ini
```

```
[sample]
TTC_SERVER_DSN=sample
TTC_SERVER1=sample-0.sample.mynamespace.svc.cluster.local
TTC_SERVER2=sample-1.sample.mynamespace.svc.cluster.local
```

Applications connect to the TimesTen database using this DSN. In a TimesTen Classic active standby pair configuration, TimesTen automatically routes application connections to the active database.

Client/Server applications must connect to the database using a defined username and password. The Operator can create such a user with ADMIN privileges. You can then connect to the database as that user to create other users and grant those users the CREATE SESSION privilege. See Overview of Configuration Metadata and Kubernetes Facilities.

In this example, use a connection string to connect to the sample database as the sampleuser user. (If you use a connection string that requires all the required connection attributes, you do not need to define them in the sys.odbc.ini file.) The sampleuser user was created by the Operator and already exists in the sample database. After connecting, you can verify that the sampleuser.emp table exists. (The Operator also previously created this table. See schema.sql for information on how the Operator created this table.)



```
% ttIsqlCS -connstr "TTC SERVER1=sample-0.sample.mynamespace.svc.cluster.local;
TTC SERVER2=sample-1.sample.mynamespace.svc.cluster.local;
TTC SERVER DSN=sample;UID=sampleuser;PWD=samplepw";
Copyright (c) 1996, 2025, Oracle and/or its affiliates. All rights
reserved.
Type ? or "help" for help, type "exit" to quit ttIsql.
connect "TTC SERVER1=sample-0.sample.mynamespace.svc.cluster.local;
TTC SERVER DSN=sample;uid=sampleuser;pwd=*******";
Connection successful:
DSN=;TTC SERVER=sample-0.sample.mynamespace.svc.cluster.local;
TTC SERVER DSN=sample;UID=sampleuser;DATASTORE=/tt/home/timesten/datastore/sample;
DATABASECHARACTERSET=AL32UTF8; CONNECTIONCHARACTERSET=AL32UTF8; PERMSIZE=200;
DDLREPLICATIONLEVEL=3;
(Default setting AutoCommit=1)
Command> tables;
 SAMPLEUSER.EMP
1 table found.
```

If you are using TimesTen client/server from applications within your Kubernetes cluster, you do not have to list all the TimesTen Pods in your connection string. Instead, you can create a Kubernetes Service that routes incoming client connections to ready instances. For this Service to work correctly, you need to use a readiness probe.

If you are using TimesTen client/server from applications outside your Kubernetes cluster, you must use a Kubernetes NodePort Service and a readiness probe.

For more information:

- About readiness probes: See About Readiness Probes for TimesTen Containers in this book.
- About Kubernetes Services: See Kubernetes Services in the Kubernetes documentation.
- About NodePort Service: See NodePort Service in the Kubernetes documentation.

11

Manage and Monitor TimesTen Classic Databases

The TimesTen Operator maintains high level states for TimesTenClassic objects and TimesTen Pods. These states describe the health of TimesTenClassic objects and the health of TimesTen databases in Pods.

Let's learn about these states and let's learn about some manual operations you can perform.

Topics:

- About the High Level State of TimesTenClassic Objects
- About the High Level State of TimesTen Pods
- About the BothDown State
- About the ManualInterventionRequired State for Replicated Objects
- About Bringing Up One Database
- About Suspending Management of a TimesTenClassic Object
- About Manual Operations

About the High Level State of TimesTenClassic Objects

The TimesTen Operator keeps track of the high level state of TimesTenClassic objects in replicated and non-replicated configurations. For replicated TimesTenClassic objects, the high level state describes the TimesTen databases in an active standby pair. Since each TimesTen database in a non-replicated configuration is independent, the high level state for non-replicated TimesTenClassic object describes the health of the replicas of TimesTen databases. For example, if all replicas for the TimesTenClassic object are up and running and functioning properly, the high level state is AllReplicasReady. Use kubectl get events or kubectl get ttc to monitor the state of TimesTenClassic objects.

The following table shows the high level states for TimesTenClassic objects and indicates if the state is supported for replicated or non-replicated objects or both:

State	Replicated Objects	Non-Replicated Objects
ActiveDown	Y	Ν
ActiveTakeover	Y	Ν
AllReplicasReady	Ν	Y
BothDown	Y	Ν
ConfiguringActive	Y	Ν
Failed	Y	Y
Initializing	Y	Y
ManualInterventionRequired	Y	Y
NoReplicasReady	Ν	Y

State	Replicated Objects	Non-Replicated Objects
Normal	Y	Ν
Reexamine	Y	Y
SomeReplicasReady	Ν	Y
StandbyCatchup	Y	Ν
StandbyDown	Y	Ν
StandbyStarting	Y	Ν
WaitingForActive	Y	Ν

ActiveDown

If the TimesTen Operator detects that TimesTen in the Pod containing the active database has failed, then the TimesTenClassic object immediately enters the ActiveDown state.

The unreachableTimeout timeout value controls how long the state of the Pod containing the active database can be Unknown before the TimesTenClassic object's state becomes ActiveDown.

When the TimesTenClassic object's state becomes ActiveDown, the standby database immediately becomes the active, and the state of the TimesTenClassic object becomes StandbyDown.

ActiveTakeover

When the TimesTenClassic object is in the Normal state, and the standby database goes down, the state briefly changes to ActiveTakeover.

When AWT cache groups are used, the standby is normally responsible for pushing updates from TimesTen to Oracle Database. However, if the standby fails, the active database takes over this responsibility. This occurs during the ActiveTakeover state.

AllReplicasReady

For a non-replicated TimesTenClassic object, all replicas are ready and available. The TimesTen databases are up and running and functioning properly.

BothDown

Neither the active nor the standby database is functioning properly. The TimesTen Operator attempts to bring up the pair of databases.

If both Pods in the active standby pair fail, the TimesTen Operator uses the information in TimesTenClassicStatus to minimize data loss. See About the BothDown State.

ConfiguringActive

When the TimesTenClassic object is in the WaitingForActive state, and when the database that should be the active database comes up, the TimesTenClassic object enters the ConfiguringActive state. The TimesTen Operator then configures this database to be the



active. Once the database is configured as the active, the TimesTenClassic object enters the StandbyDown state. See About the BothDown State.

Failed

If a problem occurs while Initializing a TimesTenClassic object, the object transitions to the Failed state. Once in this state, the TimesTen Operator does not attempt to repair the object. You must delete it. Use the kubectl get events command to determine the cause of the problem and then recreate the object.

Initializing

After you create a TimesTenClassic object, the TimesTen Operator creates the Kubernetes StatefulSets and Secrets that are required for the TimesTenClassic object. As the TimesTen Pods are starting up, the TimesTen Operator assigns the initializing state to the TimesTenClassic object.

ManualInterventionRequired

When a TimesTenClassic object enters the ManualInterventionRequired state, the TimesTen Operator takes no further action for the object. It does not query the TimesTen agents associated with the object to determine the state of TimesTen and it does not command TimesTen to do anything.

For replicated TimesTenClassic objects, see About the ManualInterventionRequired State for Replicated Objects and About Bringing Up One Database.

NoReplicasReady

For a non-replicated TimesTenClassic object, there are no replicas ready or available. TimesTen databases are not running.

Normal

For replicated objects, the Normal state indicates that TimesTen databases are up and running and functioning properly.

Reexamine

When a TimesTenClassic object is in the ManualInterventionRequired state, you can specify the .spec.ttspec.reexamine datum to cause the TimesTen Operator to take over management of the object. The TimesTen Operator moves the object to the Reexamine state. The Operator then examines the state of TimesTen.

For a replicated TimesTen Classic object, the TimesTen Operator does the following:

- If you correctly repaired TimesTen, the Operator moves the TimesTenClassic object to either the Normal or StandbyDown state, depending on the nature of your repair.
- If you did not correctly repair TimesTen, the Operator moves the TimesTenClassic object to the ManualInterventionRequired state.

For a non-replicated TimesTenClassic object, the TimesTen Operator does the following:



- If you correctly repaired TimesTen, the Operator moves the TimesTenClassic object to the AllReplicasReady state.
- If you did not correctly repair TimesTen, the Operator moves the TimesTenClassic object to the ManualInterventionRequired state.

SomeReplicasReady

For a non-replicated TimesTenClassic object, some but not all replicas are ready and available.

StandbyCatchup

This state is entered after the <code>StandbyStarting</code> state. During the <code>StandbyStarting</code> state, the standby copies the active database to the standby Pod. When the duplicate process is complete, the state changes from <code>StandbyStarting</code> to <code>StandbyCatchup</code>. See <code>StandbyStarting</code>. In the <code>StandbyCatchup</code> state, the duplicate process has completed. Transactions that ran during this duplicate process must now be copied over to the standby. Thus the <code>StandbyCatchup</code> state is the state when the newly created standby catches up to any transactions that ran on the active while the duplicate operation was running. Applications can continue to use the active without restriction.

StandbyDown

The active database is functioning properly, but the standby database is not. The TimesTen Operator automatically attempts to restart and reconfigure the standby database. Applications can continue to use the active database without restriction.

StandbyStarting

The standby is duplicating the database from the active. The StandbyStarting state is complete when the duplicate operation completes. The StandbyCatchup state is then entered. See StandbyCatchup. Applications can continue to use the active without restriction.

WaitingForActive

When the TimesTenClassic object is in the BothDown state, if the TimesTen Operator can determine which database contains the most up-to-date data, the TimesTenClassic object enters the WaitingForActive state. The object remains in this state until the Pod that contains the database is running, and the TimesTen agent within the tt container (within that Pod) is responding to the TimesTen Operator. See About the BothDown State.

About the High Level State of TimesTen Pods

The TimesTen Operator keeps track of the individual health and state of each TimesTen Pod. How often the Operator checks the health is defined by the value of the pollingInterval. For information on pollingInterval, see TimesTenClassicSpecSpec

The TimesTen Operator maintains similar states for replicated and non-replicated objects. However, the values for the states may differ. The following table indicates if the state is supported for replicated or non-replicated objects or both:



State	Replicated Objects	Non-Replicated Objects
CatchingUp	Y	N
Down	Y	Y
Healthy	Y	N
HealthyActive	Y	N
HealthyStandby	Y	N
Initializing	Y	Υ
ManualInterventionRequired	Ν	Y
Normal	Y	Y
OtherDown	Y	N
Reexamine	Ν	Y
Terminal	Y	Y
Unknown	Y	Y
UpgradeFailed	Y	N

CatchingUp

The standby has completed the process of duplicating the database from the active. The newly created standby is catching up to any transactions that ran on the active while the duplicate operation was running.

Down

Either the Pod or the TimesTen components within the Pod (or both) are not functioning properly.

Healthy

The Pod and the TimesTen components within the Pod are in a healthy state, given this Pod's role in the active standby pair.

HealthyActive

When a TimesTenClassic object is in the Reexamine state, the TimesTen Operator examines the state of both TimesTen instances. The TimesTen Operator does not know which instance (if any) contains a properly configured active database (or a properly configured standby database). The Operator must examine both instances to see. If a healthy instance is found and that instance contains a properly configured active database, the state of the Pod is reported as HealthyActive.

HealthyStandby

When a TimesTenClassic object is in the Reexamine state, the TimesTen Operator examines the state of both TimesTen instances. The TimesTen Operator does not know which instance (if any) contains a properly configured standby database (or a properly configured active database). The Operator must examine both instances to see. If a healthy instance is found



and that instance contains a properly configured standby database, the state of the Pod is reported as HealthyStandby.

Initializing

When a new Pod and PVC are started by Kubernetes as part of a TimesTenClassic object's StatefulSet, the TimesTen Operator instructs the TimesTen Agent to perform the following actions in the tt container of the Pod:

- Create a TimesTen installation.
- Create a TimesTen instance. This instance uses the installation that was created.
- Create TimesTen configuration files in the instance.
- Start the TimesTen instance.
- Create a TimesTen database.
- If applicable, create database users.
- If applicable, create database schemas, tables, views, procedures and other database objects.

While the previous operations are running, the state of the TimesTen Pod is Initializing. If these operations complete successfully, the TimesTen Operator changes the TimesTen Pod state to Normal. If any of the operations fail, the TimesTen Operator changes the state to Terminal.

ManualInterventionRequired

For non-replicated objects, when a TimesTen Pod enters the ManualInterventionRequired state, the TimesTen Operator does not know the state of TimesTen or the TimesTen databases. The TimesTen Operator takes no further action. Examine the TimesTen Pods and the TimesTenClassic object. When you have corrected the problem, you can instruct the TimesTen Operator to move the object and TimesTen Pods to the Reexamine state.

Normal

When a new Pod and PVC are started by Kubernetes as part of a TimesTenClassic object's StatefulSet, the TimesTen Operator instructs the TimesTen Agent to perform various actions in the tt container of the Pod. While these actions are being performed, the state is Initializing. If the operations complete successfully, the TimesTen Operator changes the TimesTen Pod state to Normal.

OtherDown

The Pod and the TimesTen components within the Pod are in a healthy state, but TimesTen in this Pod believes that TimesTen in the other Pod has failed. In particular, the OtherDown state indicates that this Pod contains an active database, and the database's peer has reached the failThreshold. The database in this Pod is no longer keeping transaction logs for its peer, as the peer is too far behind. Recovering the peer requires re-duplicating the active database, which the TimesTen Operator performs automatically.



Reexamine

For non-replicated objects, when a TimesTen object is in the ManualInterventionRequired state, you can specify the .spec.ttspec.reexamine datum to cause the TimesTen Operator to take over management of the object. The TimesTen Operator moves the object to the Reexamine state. The Operator then examines the state of TimesTen and the TimesTen databases in the Pods.

Based on the examination, the TimesTen Operator does the following:

- If healthy, the Operator moves the TimesTenClassic object to the AllReplicasReady state and the TimesTen Pods to the Normal state.
- If not healthy, the Operator moves the TimesTenClassic object and the unhealthy Pods to the ManualInterventionRequired state.

Terminal

The TimesTen Operator is unable to repair TimesTen in a Pod.

Unknown

The state of this Pod is unknown. Either the Pod is unreachable or the TimesTen agent contained within the Pod has failed.

UpgradeFailed

For replicated objects, an automated upgrade was attempted on TimesTen in this Pod and the upgrade failed. See About Upgrading TimesTen Classic Databases.

About the BothDown State

In replicated configurations, the TimesTen Operator provisions, monitors, and manages active standby pairs of TimesTen databases. It detects and reacts to the failure of the active or the standby database. For example, when one database in the active standby pair is down, the TimesTen Operator does the following:

- If the active database fails, the Operator promotes the standby to be the active.
- If the standby database fails, the Operator keeps the active running and repairs the standby.

However, if both databases fail at the same time, it is essential that the databases are brought back up appropriately. TimesTen replication does not atomically commit transactions in both database simultaneously. Transactions are committed in one database and then later are committed in the other database. (The database on which transactions are committed first is considered the database that is *ahead*.) Depending on how replication is configured, transactions on the active database may be ahead of the standby or the standby may be ahead of the active. To avoid data loss, the database that is ahead must become the active database after the failure is corrected.

In most cases, the TimesTen Operator can determine which database was ahead at the time of the failure. However, there are cases where the Operator cannot determine which database was ahead. In particular, the Operator cannot determine which database is ahead if all of the following conditions occur:



- Both databases failed during the polling interval. Specifically, the Operator examined both databases and the TimesTen Pods were in the Healthy state. The Operator waited pollingInterval seconds, and when the Operator examined the databases again (after this pollingInterval), both databases were down and
- RETURN TWOSAFE replication was configured and
- DISABLE RETURN **or** LOCAL COMMIT ACTION COMMIT **(or both) were configured.**

See TimesTenClassicSpecSpec for more information on the .spec.ttspec.pollingInterval datum and on the RETURN TWOSAFE and DISABLE RETURN replication configurations options. For information about defining an active standby pair replication scheme, see CREATE ACTIVE STANDBY PAIR in the Oracle TimesTen In-Memory Database SQL Reference and Defining an Active Standby Pair Replication Scheme in the Oracle TimesTen In-Memory Database Replication Guide

This combination of events indicates that some transactions may have committed on the standby and not on the active and/or some transactions may have committed on the active and not on the standby. The TimesTen Operator takes no action in this case.

When both databases fail, the TimesTenClassic object enters the BothDown state. The Operator must then determine the appropriate action to take. The Operator first examines the value of the .spec.ttspec.bothDownBehavior datum to determine what to do.

If .spec.ttspec.bothDownBehavior is set to Manual, the TimesTenClassic object immediately enters the ManualInterventionRequired state. The Operator takes no further action even if either TimesTen container subsequently becomes available. See About the ManualInterventionRequired State for Replicated Objects.

If .spec.ttspec.bothDownBehavior is set to Best (the default setting), the Operator attempts to determine which database was ahead at the time of failure.

- If the Operator cannot determine which database is ahead, the TimesTenClassic object immediately enters the ManualInterventionRequired state. See About the ManualInterventionRequired State for Replicated Objects.
- If the Operator can determine which database is ahead:
 - The TimesTenClassic object enters the WaitingForActive state. The object remains in this state until the Pod containing that database is running and the TimesTen agent located in the tt container within that Pod is responding to the Operator. At this point, the TimesTenClassic object enters the ConfiguringActive state.
 - While the TimesTenClassic object is in the ConfiguringActive state, TimesTen in this Pod is started, the database is loaded and is configured for use as the new active database. If there are any problems with these steps, the TimesTenClassic object enters the ManualInterventionRequired state. If the database is successfully loaded and successfully configured as the new active, the TimesTenClassic object enters the StandbyDown state. See About the High Level State of TimesTenClassic Objects for information on the states of your TimesTenClassic object.
 - You can specify the maximum amount of time (expressed in seconds) that the TimesTenClassic object remains in the WaitingForActive state by specifying a value for the spec.ttspec.waitingForActiveTimeout datum. After this period of time, if the object is still in the WaitingForActive state, the object automatically transitions to the ManualInterventionRequired state. The default is 0, which indicates that there is no timeout, and the object will remain in this state indefinitely. See TimesTenClassicSpecSpec for more information on the spec.ttspec.waitingForActiveTimeout datum.

- The time to recover the database varies by the size of the database. You should consider the size of your database when deciding the value for spec.ttspec.waitingForActiveTimeout.
- If the database that is ahead cannot be loaded, the TimesTenClassic object enters the ManualInterventionRequired state. See About the ManualInterventionRequired State for Replicated Objects.

About the ManualInterventionRequired State for Replicated Objects

When a replicated TimesTenClassic object enters the ManualInterventionRequired state, the TimesTen Operator takes no further action for this object. It does not query the TimesTen agents associated with the object to determine the state of TimesTen and does not command TimesTen to do anything. It is important for you to address why the TimesTenClassic object is in this state.

If your TimesTenClassic object is in the ManualInterventionRequired state and it is not the result of it first being in the BothDown state, perform the operations necessary to manually repair one of the databases. Then, perform the steps to bring up this database. These steps are covered in About Bringing Up One Database later in this chapter.

If, however, the TimesTenClassic object is in the ManualInterventionRequired state as a result of it first being in the BothDown state:

- It may be unclear which database, if either, is suitable to be the new active. There may be transactions that have committed on the active database and not on the standby database, and simultaneously there may be transactions that have committed on the standby database and not on the active database.
- You need to manually examine both databases and may need to reconcile the data before you can choose which database should be the new active.
- If you can reconcile the data, and can manually fix one of the databases, then you can perform the steps to bring up one database. These steps are covered in About Bringing Up One Database later in this chapter. If you cannot reconcile the data, contact Oracle Support for further assistance.

In order for you to direct the Operator to move the TimesTenClassic object out of the ManualInterventionRequired state, you must either:

- Bring up exactly one database: The Operator treats this database as the active database. All of these conditions must be met:
 - The TimesTen agent in the container is running.
 - The TimesTen the instance in the container is running.
 - The TimesTen database is loaded.
 - There is no replication scheme in the database.
 - The replication agent is not running.
 - The replication state is IDLE.

If these conditions are met, the Operator moves the TimesTenClassic object to the StandbyDown state. If any of these conditions are not met, the TimesTenClassic object remains in the ManualInterventionRequired state. Note that when no replication scheme exists in the database, the Operator will still create the appropriate replication scheme

based on how it is defined in the TimesTenClassic object definition. See About Bringing Up One Database for an example of how you can direct the Operator to take action once one database is up and running.

- Bring up both databases: In this case, you must configure the active standby pair. Specifically, each database must meet all of the following conditions:
 - The TimesTen agent in the container is running.
 - The TimesTen instance in the container is running.
 - The database is loaded.
 - The replication scheme is defined in both databases.
 - The replication agents are started and are running.
 - One database must be in the ACTIVE state and the other database must be in the STANDBY state.

If these conditions are met, the Operator moves the TimesTenClassic object to the Normal state. If any of these conditions are not met, the TimesTenClassic object remains in the ManualInterventionRequired state.

If you cannot bring up either database, the TimesTenClassic object remains in the ManualInterventionRequired state.

You direct the Operator to examine the databases by specifying the .spec.ttspec.reexamine datum. Every .spec.ttspec.pollingInterval, the Operator examines the value of .spec.ttspec.reexamine. If the value has changed since the last iteration for this TimesTenClassic object, the Operator examines the state of the TimesTen containers for this object. See TimesTenClassicSpecSpec for more information on the .spec.ttspec.pollingInterval and the .spec.ttspec.reexamine datum.

The examination of the databases is performed exactly one time after you change the .spec.ttspec.reexamine value. If the required conditions were not met, you may again attempt to meet them. You must then modify the .spec.ttspec.reexamine value again to cause the Operator to reexamine the databases.

Note that whenever a TimesTenClassic object changes state, a Kubernetes Event is created. You can monitor these events with the kubectl describe command to be informed of such state transitions.

About Bringing Up One Database

This section assumes you have manually repaired or have manually performed maintenance on one of the databases associated with a replicated TimesTenClassic object. The TimesTenClassic object is currently in the ManualInterventionRequired state. You now want to direct the TimesTen Operator to treat the repaired database as the active, to perform the necessary steps to duplicate this database to the standby, and to bring up both databases, such that both are running and operating successfully.

Recall that all of these conditions must be met for the database:

- TimesTen agent in the container is running.
- TimesTen daemon (the instance) in the container is running.
- TimesTen database is loaded.
- There is no replication scheme in the database.
- The replication agent is not running.



• The replication state is IDLE.

These sections show you how to verify the conditions are met for the database and how to set the reexamine value:

- Verify Conditions Are Met for the Database
- Set the reexamine Value

Verify Conditions Are Met for the Database

Perform these steps to ensure the conditions are met for the database (the database to be the active). In this example, sample-1 will be the new active.

Note: These steps require you to use TimesTen utilities and TimesTen built-in procedures. See Utilities and Built-In Procedures in the *Oracle TimesTen In-Memory Database Reference* for details.

1. Confirm the TimesTenClassic object (sample, in this example) is in the ManualInterventionRequired state (represented in **bold**).

sample	ManualInterventionRequired	sample-0	12h
NAME	STATE	ACTIVE	AGE
% kubect	l get ttc sample		

2. Use the kubectl exec -it command to invoke the shell within the sample-1 Pod that contains the TimesTen database. (This database will be the new active.)

The remaining procedures take place within this shell.

% kubectl exec -it sample-1 -c tt -- /bin/bash

 Use the ttDaemonAdmin utility to start TimesTen daemon (if not already started). Then use the ttAdmin utility to load the TimesTen database into memory (if not already loaded).

```
% ttDaemonAdmin -start
TimesTen Daemon (PID: 5948, port: 6624) startup OK.
% ttAdmin -ramLoad sample
RAM Residence Policy
                             : manual
Manually Loaded In RAM
                             : True
Replication Agent Policy
                             : manual
Replication Manually Started : False
Cache Agent Policy
                             : manual
                             : False
Cache Agent Manually Started
Database State
                             : Open
```

4. Use the ttIsql utility to connect to the sample database. Then, call the ttRepStop built-in procedure to stop the replication agent.

```
% ttIsql sample
```

```
Copyright (c) 1996, 2025, Oracle and/or its affiliates. All rights reserved. Type ? or "help" for help, type "exit" to quit ttIsql.
```

```
connect "DSN=sample";
Connection successful: DSN=sample;UID=timesten;DataStore=/tt/home/timesten/datastore/
sample;
DatabaseCharacterSet=AL32UTF8;ConnectionCharacterSet=AL32UTF8;AutoCreate=0;
PermSize=200;DDLReplicationLevel=3;ForceDisconnectEnabled=1;
(Default setting AutoCommit=1)
Command> call ttRepStop;
```



5. From within ttlsql, use the SQL DROP ACTIVE STANDBY PAIR statement to drop the active standby pair replication scheme. Then use the ttlsql repschemes command to verify there are no replication schemes in the database. Exit from ttlsql.

```
Command> DROP ACTIVE STANDBY PAIR;
Command> repschemes;
```

0 replication schemes found.

6. Use the ttStatus utility to verify the TimesTen daemon is running and the replication agent is not running. Note: The ttStatus output may change from release to release.

```
% ttStatus
 TimesTen status report as of Thu Jan 16 02:14:15 2025
 Daemon pid 5948 port 6624 instance instance1
 TimesTen server pid 5955 started on port 6625
  _____
  _____
 Data store /tt/home/timesten/datastore/sample
 Daemon pid 5948 port 6624 instance instance1
 TimesTen server pid 5955 started on port 6625
 There are 15 connections to the data store
 Shared Memory KEY 0x0a100c60 ID 196609
PL/SQL Memory Key 0x0b100c60 ID 229378 Address 0x500000000TypePIDContextConnection NameProcess104180x00000000218a6e0sampleProcess83380x000000001cbb6e0sampleSubdaemon59530x0000000015075f0ManagerSubdaemon59530x000000001607210CheckpointSubdaemon59530x00007f132c008c0FlusherSubdaemon59530x00007f132c080370Log MarkerSubdaemon59530x00007f1334008c0MonitorSubdaemon59530x00007f1334008c0MonitorSubdaemon59530x00007f1338008c0AgingSubdaemon59530x00007f1338008c0AgingSubdaemon59530x00007f1330008c0Deadlock DetectorSubdaemon59530x00007f133c008c0Collock DetectorSubdaemon59530x00007f133c008c0Deadlock DetectorSubdaemon59530x00007f13c008c0Collock DetectorSubdaemon59530x00007f13c008c0Deadlock DetectorSubdaemon59530x00007f13c008c0Garbage CollectorSubdaemon59530x00007f13c008c0Garbage CollectorSubdaemon59530x00007f13c008c0Garbage CollectorSubdaemon59530x00007f13c008c0Garbage CollectorSubdaemon59530x00007f13c008c0Garbage CollectorSubdaemon59530x00007f13c008c0Garbage CollectorSubdaemon59530x00007f13c008c0Garb
 PL/SQL Memory Key 0x0b100c60 ID 229378 Address 0x500000000
                                                                                                                                    ConnID
                                                                                                                                              2
                                                                                                                                              1
                                                                                                                                         2047
                                                                                                                                         2046
                                                                                                                                         2041
                                                                                                                                        2045
                                                                                                                                        2040
                                                                                                                                        2044
                                                                                                                                        2037
                                                                                                                                        2042
                                                                                                                                        2039
                                                                                                                                       2043
                                                                                                                                        2038
                                                                                                                                        2035
                                                                                                                                         2036
 Open for user connections
 RAM residence policy: Manual
 Data store is manually loaded into RAM
 Replication policy : Manual
 Cache Agent policy : Manual
 PL/SQL enabled.
 _____
                                                  _____
 Accessible by group timesten
 End of report
```

You successfully verified the conditions for the database. The database is up and running. The Operator will treat this database as the active. You are now ready to set the value for the .spec.ttspec.reexamine datum.

Set the reexamine Value

This example shows you how to set the reexamine value for a replicated TimesTenClassic object. The example also illustrates the action the TimesTen Operator takes after the reexamine value has been changed.
Set the reexamine value. The value must be different than the current value for the TimesTenClassic object. When the TimesTen Operator examines this value and notices it has changed since the last iteration, it takes appropriate action.

Use the kubectl edit command to edit the TimesTenClassic object.

- If there is a line for reexamine in the file, then modify its value. It must be different than the current value.
- If there is no line for reexamine in the file, then add a line and specify a value.

In this example, there is no reexamine line. This example adds the reexamine line and sets the value for reexamine to April22reexamine1 (represented in **bold**).

Note: Not all output is shown.

```
% kubectl edit timestenclassic sample
# Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving this
# file will be reopened with the relevant failures.
#
apiVersion: timesten.oracle.com/v4
kind: TimesTenClassic
metadata:
. . .
 name: sample
  namespace: mynamespace
. . .
repCreateStatement: |
  create active standby pair
    "{{tt-name}}" on "{{tt-node-0}}",
    "{{tt-name}}" on "{{tt-node-1}}"
  RETURN TWOSAFE
  store "{{tt-name}}" on "{{tt-node-0}}"
    PORT {{tt-rep-port}} FAILTHRESHOLD 0 TIMEOUT 999
  store "{{tt-name}}" on "{{tt-node-1}}"
    PORT {{tt-rep-port}} FAILTHRESHOLD 0 TIMEOUT 999
spec:
  ttspec:
   bothDownBehavior: Best
    dbConfigMap:
    - sample
    image: container-registry.oracle.com/timesten/timesten:22.1.1.34.0
    imagePullSecret: sekret
    storageClassName: oci-bv
    storageSize: 250Gi
    reexamine: April22reexamine1
```

timestenclassic.timesten.oracle.com/sample edited

2. Use the kubectl get command to assess the state of the sample TimesTenClassic object. Observe how the state changes as you issue multiple kubectl get commands. Also note that the Operator has successfully configured sample-1 to be the active.

```
% kubectl get ttc sample
NAME
    STATE ACTIVE AGE
sample Reexamine None
                        68m
% kubectl get ttc sample
NAME
      STATE
                        ACTIVE AGE
sample ConfiguringActive None
                                 68m
% kubectl get ttc sample
                  ACTIVE
NAME
      STATE
                             AGE
sample StandbyDown sample-1 68m
% kubectl get ttc sample
```



NAME STATE ACTIVE AGE sample Normal sample-1 71m

3. Use the kubectl describe command to further review the actions of the Operator (represented in **bold**).

Not all output is shown:

```
% kubectl describe ttc sample
Name:
       sample
Namespace: mynamespace
. . .
Kind:
             TimesTenClassic
. . .
Rep Create Statement: create active standby pair
  "{{tt-name}}" on "{{tt-node-0}}",
  "{{tt-name}}" on "{{tt-node-1}}"
RETURN TWOSAFE
store "{{tt-name}}" on "{{tt-node-0}}"
 PORT {{tt-rep-port}} FAILTHRESHOLD 0 TIMEOUT 999
store "{{tt-name}}" on "{{tt-node-1}}"
  PORT {{tt-rep-port}} FAILTHRESHOLD 0 TIMEOUT 999
Spec:
  Ttspec:
   Both Down Behavior: Best
   Db Config Map:
     sample
   Image:
                        container-registry.oracle.com/timesten/timesten:22.1.1.34.0
    Image Pull Policy: Always
   Image Pull Secret: sekret
   Reexamine: April22reexamine1
   Stop Managing:
                      April21Stop1
    Storage Class Name: oci-bv
   Storage Size:
                       250Gi
Status:
  Classic Upgrade Status:
   Active Start Time:
                                0
   Active Status:
   Image Update Pending:
                                false
   Last Upgrade State Switch:
                                0
   Prev Reset Upgrade State:
   Prev Upgrade State:
   Standby Start Time:
                                0
   Standby Status:
   Upgrade Start Time:
                                0
   Upgrade State:
  Active Pods:
                                sample-1
  High Level State:
                                Normal
  Last Event:
                                54
  Last High Level State Switch: 1619230912
  Pod Status:
   Cache Status:
     Cache Agent:
                         Not Running
     Cache UID Pwd Set: true
     N Cache Groups:
                         0
   Db Status:
     Db:
                                  Loaded
     Db Id:
                                  475
     Db Updatable:
                                  No
    Initialized:
                                  true
    Last High Level State Switch: ?
    Pod Status:
```



```
Agent:
                      Up
 Last Time Reachable: 1619231126
 Pod IP:
                     10.244.7.89
 Pod Phase:
                    Running
Prev High Level State: Healthy
Prev Image:
Replication Status:
 Last Time Rep State Changed: 0
  Rep Agent:
                             Running
 Rep Peer P State:
                             start
 Rep Scheme:
                             Exists
 Rep State:
                              STANDBY
Times Ten Status:
 Daemon: Up
 Instance: Exists
Release: 22.1.1
                22.1.1.34.0
Admin User File: false
Cache User File: false
Cg File:
                false
Disable Return: false
High Level State: Healthy
Intended State: Standby
Local Commit:
                false
Name:
                sample-0
               false
Schema File:
Using Twosafe: false
Cache Status:
 Cache Agent: Not Running
 Cache UID Pwd Set: true
 N Cache Groups: 0
Db Status:
 Db:
                             Loaded
 Db Id:
                             476
 Db Updatable:
                             Yes
Initialized:
                             true
Last High Level State Switch: ?
Pod Status:
 Agent:
                      Up
 Last Time Reachable: 1619231126
 Pod IP:
                     10.244.6.149
 Pod Phase:
                     Running
Prev High Level State: Healthy
Prev Image:
Replication Status:
 Last Time Rep State Changed: 1619228670
                             Running
 Rep Agent:
 Rep Peer P State:
                             start
 Rep Scheme:
                             Exists
 Rep State:
                             ACTIVE
Times Ten Status:
 Daemon:
                    Up
                 Exists
 Instance:
 Release:
                    22.1.1.34.0
Admin User File:
                    false
Cache User File:
                    false
Cg File:
                    false
Disable Return:
                    false
High Level State: Healthy
Intended State: Active
Local Commit:
                    false
Name:
                    sample-1
Schema File:
                    false
```



```
Using Twosafe:
                        false
 Prev High Level State: StandbyDown
                        April22reexamine1
 Prev Reexamine:
 Prev Stop Managing:
                        April21Stop1
 Rep Create Statement: create active standby pair "sample" on
"sample-0.sample.mynamespace.svc.cluster.local", "sample" on
"sample-1.sample.mynamespace.svc.cluster.local" NO RETURN store "sample" on
"sample-0.sample.mynamespace.svc.cluster.local" PORT 4444 FAILTHRESHOLD 0 store
"sample" on "sample-1.sample.mynamespace.svc.cluster.local" PORT 4444 FAILTHRESHOLD 0
 Rep Port:
                       4444
 Status Version:
                        1.0
Events:
 Type Reason
                 Age From
                                  Message
 _____
                   ____
                         ----
                                    _____
       StateChange 58m ttclassic TimesTenClassic was Normal, now
ManualInterventionRequired
      StateChange 46m ttclassic Pod sample-0 Daemon Down
 -
 _
       StateChange 41m ttclassic Pod sample-1 Daemon Down
      StateChange 41m ttclassic Pod sample-1 Daemon Up
  _
     StateChange 41m ttclassic Pod sample-1 Database Unloaded
  _
     StateChange 40m ttclassic Pod sample-1 Database Loaded
  _
     StateChange 40m ttclassic Pod sample-1 RepState IDLE
      StateChange 40m ttclassic Pod sample-1 RepAgent Not Running
      StateChange 17m ttclassic Pod sample-1 Database Updatable
       StateChange 17m ttclassic Pod sample-1 RepScheme None
       StateChange 4m21s ttclassic TimesTenClassic was
ManualInterventionRequired, now Reexamine
               4m16s ttclassic Active error: Daemon Down
       Error
       StateChange 4m16s ttclassic TimesTenClassic was Reexamine, now
ConfiguringActive
       StateChange 4m10s ttclassic Pod sample-1 RepState ACTIVE
       StateChange 4m10s ttclassic Pod sample-1 RepScheme Exists
       StateChange 4m10s ttclassic Pod sample-1 RepAgent Running
       StateChange 4m8s ttclassic TimesTenClassic was ConfiguringActive, now
StandbyDown
       StateChange 4m3s ttclassic Pod sample-0 Daemon Up
 -
       StateChange 4m3s ttclassic Pod sample-0 Database Unloaded
       StateChange 3m56s ttclassic Pod sample-0 Database None
 _
 _
       StateChange 3m42s ttclassic Pod sample-0 Database Loaded
 _
       StateChange 3m42s ttclassic Pod sample-0 Database Not Updatable
       StateChange 3m42s ttclassic Pod sample-0 RepAgent Not Running
  _
  _
       StateChange 3m42s ttclassic Pod sample-0 RepState IDLE
       StateChange 3m36s ttclassic Pod sample-0 RepAgent Running
  _
       StateChange 3m36s ttclassic Pod sample-0 RepState STANDBY
       StateChange 3m36s ttclassic TimesTenClassic was StandbyDown, now Normal
```

4. Use the kubectl exec -it command to invoke the shell within the sample-1 Pod that contains the TimesTen database. Then, verify you can connect to the active database.

```
% kubectl exec -it sample-1 -c tt -- /bin/bash
$ ttIsql sample
Copyright (c) 1996, 2025, Oracle and/or its affiliates. All rights reserved.
Type ? or "help" for help, type "exit" to quit ttIsql.
```

```
connect "DSN=sample";
Connection successful: DSN=sample;UID=timesten;DataStore=/tt/home/timesten/datastore/
sample;
DatabaseCharacterSet=AL32UTF8;ConnectionCharacterSet=AL32UTF8;AutoCreate=0;PermSize=2
00;DDLReplicationLevel=3;ForceDisconnectEnabled=1;
(Default setting AutoCommit=1)
```



```
Command> call ttRepStateGet;
< ACTIVE >
1 row found.
```

5. Use the kubectl exec -it command to invoke the shell within the sample-0 Pod that contains the TimesTen database. Then, verify you can connect to the standby database.

```
% kubectl exec -it sample-0 -c tt -- /bin/bash
% ttIsql sample
Copyright (c) 1996, 2025, Oracle and/or its affiliates. All rights reserved.
Type ? or "help" for help, type "exit" to quit ttIsql.
connect "DSN=sample";
Connection successful: DSN=sample;UID=timesten;DataStore=/tt/home/timesten/datastore/
sample;
DatabaseCharacterSet=AL32UTF8;ConnectionCharacterSet=AL32UTF8;AutoCreate=0;
PermSize=200;DDLReplicationLevel=3;ForceDisconnectEnabled=1;
(Default setting AutoCommit=1)
Command> call ttRepStateGet;
< STANDBY >
1 row found.
```

The TimesTen Operator is now managing and monitoring your TimesTenClassic object. The TimesTenClassic object is in the Normal state. Both databases are up and running and ready for use.

About Suspending Management of a TimesTenClassic Object

The TimesTen Operator examines the state of the TimesTen instances and the databases associated with each TimesTenClassic object. It takes actions to repair anything that is broken. You may have a situation in which you want to manually perform maintenance operations. In such a situation, you do not want the TimesTen Operator to interfere and attempt to perform repair operations.

You could stop the TimesTen Operator by deleting the timesten-operator Deployment. This action prevents the Operator from interfering. See Revert to Manual Control. However, if you have more than one TimesTenClassic object and you delete the TimesTen Operator, this interferes with the management of all the TimesTenClassic objects, when perhaps only one of them needs manual intervention.

You can direct the TimesTen Operator to take no action for one or more TimesTenClassic objects by specifying the .spec.ttspec.stopManaging datum in the TimesTenClassic object's definition. The TimesTen Operator examines the value of .spec.ttspec.stopManaging and if it has changed since the last time the Operator examined it, the Operator changes the state of the TimesTenClassic object to ManualInterventionRequired. This causes the Operator to no longer examine the status of the TimesTenClassic object. The Operator takes no action on the object or its Pods.

When you want the TimesTen Operator to manage the TimesTenClassic object again, you use the .spec.ttspec.reexamine datum. These actions enable you to perform manual operations on TimesTen without deleting the TimesTen Operator.

For an example, see Suspend Management.

Suspend Management

You can instruct the TimesTen Operator to stop managing one or more TimesTenClassic objects that are running in your namespace. To do so, use the TimesTenClassic object's .spec.ttspec.stopManaging datum.

Let's look at an example.

1. Review the TimesTenClassic objects that are running in your namespace.

kubectl get ttc

The output is similar to the following:

NAME	STATE	ACTIVE	AGE
samplenorep	AllReplicasReady	N/A	47h
samplerep	Normal	samplerep-0	2d

There are two TimesTenClassic objects. The samplenorep is a non-replicated object and all replicas are ready. The samplerep is a replicated object that is in the Normal state.

2. Review the Pods.

kubectl get pods

The output is similar to the following:

RESTARTS	AGE
0	47h
0	47h
0	47h
0	2d
0	2d
0	2d1h
	0 0 0 0 0

For the samplenorep non-replicated object, there are three Pods running representing three databases (replicas=3). For the samplerep replicated object, there are two Pods running. One Pod represents the active database (samplerep-0) and one Pod represents the standby database (samplerep-1).

- 3. Instruct the TimesTen Operator to stop managing the samplenorep object.
 - a. Use the kubectl edit command to edit the object.

If there is a line for .spec.ttspec.stopManaging in the file, then modify its value. It must be different than the current value. If there is no line, add a line and specify a value.

This example adds the .spec.ttspec.stopManaging datum and sets the value to stopManagingNow.

kubectl edit ttc samplenorep

Please edit the object below. Lines beginning with a '#' will be



```
ignored,
# and an empty file will abort the edit. If an error occurs while
saving this file will be
# reopened with the relevant failures.
#
apiVersion: timesten.oracle.com/v4
kind: TimesTenClassic
metadata:
. . .
spec:
 ttspec:
   additionalMemoryRequest: 2Gi
   automaticMemoryRequests: true
   cacheCleanup: true
   createASReadinessProbe: true
    daemonLogCPURequest: 200m
    daemonLogMemoryRequest: 200Mi
    stopManaging: stopManagingNow
. . .
```

- b. Save and close the file.
- 4. Confirm the samplenorep object is in the ManualInterventionRequired state.

kubectl get ttc

The output is similar to the following:

NAME	STATE	ACTIVE	AGE
samplenorep	ManualInterventionRequired	N/A	2d
samplerep	Normal	samplerep-0	2d1h

The TimesTen Operator moved the samplenorep object to the ManualInterventionRequired state.

5. Confirm the TimesTen Operator moved each of the Pods for the samplenorep object to the ManualInterventionRequired state.

kubectl get events

The output is similar to the following. Not all output is displayed.

LAST SEEN	TYPE	REASON	OBJECT	
MESSAGE				
11m	Normal	StateChange	timestenclassic/samplenorep	Pod
samplenorep	-0 state	was Normal, now	ManualInterventionRequired	
11m	Normal	StateChange	timestenclassic/samplenorep	Pod
samplenorep	-1 state	was Normal, now	ManualInterventionRequired	
samplenorep 11m	-1 state Normal		ManualInterventionRequired timestenclassic/samplenorep	Pod
11m	Normal	StateChange	-	Pod
11m	Normal	StateChange was Normal, now	timestenclassic/samplenorep	Pod

The TimesTen Operator successfully moved the Pods to the ManualInterventionRequired state.

- 6. Instruct the TimesTen Operator to stop managing the samplenorep object.
 - a. Use the kubectl edit command to edit the object.

If there is a line for .spec.ttspec.stopManaging in the file, then modify its value. It must be different than the current value. If there is no line, add a line and specify a value.

This example adds the .spec.ttspec.stopManaging datum and sets the value to stopManagingNow.

kubectl edit ttc samplerep

Please edit the object below. Lines beginning with a '#' will be ignored, # and an empty file will abort the edit. If an error occurs while saving this file will be # reopened with the relevant failures. # apiVersion: timesten.oracle.com/v4 kind: TimesTenClassic metadata: . . . spec: ttspec: additionalMemoryRequest: 2Gi automaticMemoryRequests: true cacheCleanup: true createASReadinessProbe: true daemonLogCPURequest: 200m daemonLogMemoryRequest: 200Mi stopManaging: stopManagingNow . . .

- b. Save and close the file.
- 7. Confirm the samplerep object is in the ManualInterventionRequired state.

kubectl get ttc

The output is similar to the following:

NAME	STATE	ACTIVE	AGE
samplenorep	ManualInterventionRequired	N/A	2d
samplerep	ManualInterventionRequired	samplerep-0	2d1h

The TimesTen Operator moved the samplerep object to the ManualInterventionRequired state.

You successfully instructed the TimesTen Operator to move TimesTenClassic objects to the ManualInterventionRequired state. The TimesTen Operator is no longer monitoring or managing the objects. It takes no further action on the objects or its Pods. You can now perform manual operations on your TimesTen databases. When you are ready for the TimesTen Operator to resume management of one or both objects, do the following:



- For non-replicated objects, such as samplenorep, proceed to Resume Management.
- For replicated objects, such as samplerep, proceed to About Bringing Up One Database.

Resume Management

If a TimesTenClassic object is in the ManualInterventionRequired state, you can instruct the TimesTen Operator to examine the state of TimesTen and its databases to see if they are healthy. For the procedures for replicated objects, proceed to About Bringing Up One Database.

For non-replicated objects, set the object's .spec.ttspec.reexamine datum. Doing so instructs the TimesTen Operator to move the object to the Reexamine state. In the Reexamine state, the TimesTen Operator examines the state of TimesTen and its databases. If healthy, the Operator returns the object to the AllReplicasReady state. If not healthy, the object re-enters the ManualInterventionRequired state.

1. Review the TimesTenClassic objects that are running in your namespace.

```
kubectl get ttc
```

The output is similar to the following:

NAME	STATE	ACTIVE	AGE
samplenorep	ManualInterventionRequired	N/A	2d1h

The replicated samplenorep object is in the ManualInterventionRequired state.

- 2. Instruct the TimesTen Operator to resume management of the samplenorep object.
 - a. Use the kubectl edit command to edit the object.

If there is a line for .spec.ttspec.reexamine in the file, then modify its value. It must be different than the current value. If there is no line, add a line and specify a value.

This example adds the .spec.ttspec.reexamine datum and sets the value to reexamineNow.

```
kubectl edit ttc samplenorep
# Please edit the object below. Lines beginning with a '#' will be
ignored,
# and an empty file will abort the edit. If an error occurs while
saving this file will be
# reopened with the relevant failures.
#
apiVersion: timesten.oracle.com/v4
kind: TimesTenClassic
metadata:
. . .
spec:
  ttspec:
   additionalMemoryRequest: 2Gi
   automaticMemoryRequests: true
    cacheCleanup: true
```



```
createASReadinessProbe: true
daemonLogCPURequest: 200m
daemonLogMemoryRequest: 200Mi
reexamine: rexamineNow
```

- • •
- b. Save and close the file.
- 3. Observe the behavior.

kubectl get events

The output is similar to the following. Not all output is displayed.

```
Normal StateChange timestenclassic/samplenorep
27s
                                                               Pod
samplenorep-0 state was ManualInterventionRequired, now Reexamine
    Normal StateChange timestenclassic/samplenorep
27s
                                                               Pod
samplenorep-1 state was ManualInterventionRequired, now Reexamine
27s
    Normal StateChange timestenclassic/samplenorep
                                                               Pod
samplenorep-2 state was ManualInterventionRequired, now Reexamine
    Normal StateChange timestenclassic/samplenorep
27s
TimesTenClassic was ManualInterventionRequired, now Reexamine
22s Normal StateChange timestenclassic/samplenorep
                                                               Pod
samplenorep-0 state was Reexamine, now Normal
         Normal StateChange timestenclassic/samplenorep
22s
                                                               Pod
samplenorep-1 state was Reexamine, now Normal
22s Normal StateChange timestenclassic/samplenorep
                                                               Pod
samplenorep-2 state was Reexamine, now Normal
     Normal StateChange timestenclassic/samplenorep
22s
TimesTenClassic was Reexamine, now AllReplicasReady
```

The TimesTen Operator changed the state of each TimesTen Pod from ManualInterventionRequired to Reexamine and changed the TimesTenClassic object's state from ManualInterventionRequired to Reexamine. The TimesTen Operator examined TimesTen and its databases. Since TimesTen and its databases are healthy and functioning properly, the TimesTen Operator changed the state of the TimesTen Pods to Normal and the state of the TimesTenClassic object to AllReplicasReady.

About Manual Operations

The TimesTen Operator strives to keep TimesTen databases running once they are deployed. Kubernetes manages the lifecycle of the Pods. It recreates the Pods if they fail. If the nodes on which the Pods are running fail, Kubernetes recreates the Pods on available Kubernetes cluster nodes. The TimesTen Operator monitors TimesTen running in the Pods and initiates the appropriate operations to keep the databases operational. The TimesTen Operator performs these operations automatically and there is minimal human intervention required.

If necessary, there are the manual operations you can perform:

- Manually Invoke TimesTen Utilities
- Revert to Manual Control
- Delete TimesTen Databases
- Locate the TimesTen Operator



Modify TimesTen Connection Attributes

Manually Invoke TimesTen Utilities

You can manually invoke TimesTen utilities on TimesTen instances. Use the kubectl exec -it command to invoke a shell in a container in a TimesTen Pod. Once in the container, you can invoke TimesTen utilities and perform additional operations.

Note:

The TimesTen Operator continues to query the status of the Pod and the status of TimesTen within the Pod. If you invoke a command that disrupts the functioning of either the Pod or TimesTen, the Operator may act to try to fix what you did.

This example shows how to use the kubectl exec -it command to invoke a shell within the sample-0 Pod that contains the active TimesTen database in an active standby pair configuration. The example then runs the ttIsql utility.

```
% kubectl exec -it sample-0 -c tt -- /bin/bash
% ttIsql sample
Copyright (c) 1996, 2025, Oracle and/or its affiliates. All rights reserved.
Type ? or "help" for help, type "exit" to quit ttIsql.
connect "DSN=sample";
Connection successful:
DSN=sample;UID=timesten;DataStore=/tt/home/timesten/datastore/sample;
DatabaseCharacterSet=AL32UTF8;ConnectionCharacterSet=AL32UTF8;PermSize=200;
```

Revert to Manual Control

DDLReplicationLevel=3;

(Default setting AutoCommit=1)

If you want to manually operate and control TimesTen databases, you can delete the timesten-operator Deployment. This causes the TimesTen Operator to stop. It does not restart. This affects all of the TimesTenClassic objects that are running in your Kubernetes namespace. If you do not want the TimesTen Operator to stop managing all of the TimesTenClassic objects, you can suspend the management of individual TimesTenClassic objects. See About Suspending Management of a TimesTenClassic Object.

A TimesTenClassic object remains in Kubernetes as do other Kubernetes objects associated with the TimesTenClassic object. You can use the kubectl exec -it command to invoke a shell in one or more Pods and you can control Timesten running in those Pods.

If a Pod fails, Kubernetes creates new ones to replace them. This is due to the StatefulSet object that the TimesTen Operator had previously created in Kubernetes. However, since the TimesTen Operator is not running the new Pods, it cannot automatically start TimesTen. In this case, your TimesTen databases cannot be configured or started. You are responsible for the operation of TimesTen in the Pods.

If you want the TimesTen Operator to take control, you must redeploy the Operator. Once the Operator is redeployed, it automatically identifies the TimesTenClassic objects in your Kubernetes cluster and attempts to manage them again.



This example shows you how to manually control TimesTen.

1. Verify the Operator and the TimesTen databases are running.

% kubectl get pods					
NAME	READY	STATUS	RESTARTS	AGE	
sample-0	3/3	Running	0	18h	
sample-1	2/3	Running	0	18h	
timesten-operator-5d7dcc7948-pzj58	1/1	Running	0	18h	

Navigate to the /deploy directory where the operator.yaml resides. (kube_files/deploy, in this example.)

```
% cd kube files/deploy
```

3. Use the kubectl delete command to delete the Operator. The Operator is stopped and no longer deployed.

```
% kubectl delete -f operator.yaml
deployment.apps "timesten-operator" deleted
```

4. Verify the Operator is no longer running, but the TimesTen databases are.

% kubectl	get pods			
NAME	READY	STATUS	RESTARTS	AGE
sample-0	3/3	Running	0	19h
sample-1	2/3	Running	0	19h

5. Use the kubectl exec -it command to invoke a shell in a Pod that runs TimesTen.

```
% kubectl exec -it sample-0 -c tt -- /bin/bash
Last login: Thu Jan 16 14:30:45 UTC 2025 on pts/0
```

6. Run the ttStatus utility.

```
% ttStatus
TimesTen status report as of Thu Jan 16 14:36:31 2025
```

```
Daemon pid 183 port 6624 instance instance1
TimesTen server pid 190 started on port 6625
```

```
      Data store /tt/home/timesten/datastore/sample

      Daemon pid 183 port 6624 instance instance1

      TimesTen server pid 190 started on port 6625

      There are 20 connections to the data store

      Shared Memory KEY 0x02200bbc ID 32769

      PL/SQL Memory Key 0x03200bbc ID 65538 Address 0x500000000

      Type
      PID

      Context
      Connection Name
      ConnID

      Replication
      263
      0x00007f9960008c0
      LOGFORCE:140299698493184
      2029

      Replication
      263
      0x00007f9a040008c0
      XLA_PAREN:140300350273280
      2031

      Replication
      263
      0x00007f9a08008c0
      REDISTENER:14030037123456
      2030

      Replication
      263
      0x00007f9a08008c0
      FAILOVER:140299429472000
      2028

      Replication
      263
      0x00007f9a08008c0
      FAILOVER:14030035423104
      2032

      Replication
      263
      0x00007f9a20009b0
      TRANSMITTER(M):140299695343360
      2034

      Replication
      263
      0x00000023365b0
      Manager
      2047

      Subdaemon
      187
      0x000000023365b0
      Manager
      2041

      Subdaemon
      187
      0x00000002432cf0
      Log Marker
      2041

      Subdaemon<
```



```
187
                     0x00007f90d80008c0 Monitor
                                                                     2043
Subdaemon
            187 0x00007f90f808b360 IndexGC
Subdaemon
                                                                     2037
Subdaemon1870x00007f90fc0008c0 FlusherSubdaemon1870x00007f910004efd0 XactId Rollback
                                                                     2045
                                                                     2040
Open for user connections
RAM residence policy: Always
Replication policy : Manual
Replication agent is running.
Cache Agent policy : Manual
PL/SQL enabled.
                     _____
Accessible by group timesten
End of report
```

7. Run the ttIsql utility to connect to the sample database and perform various operations.

```
% ttIsql sample
Copyright (c) 1996, 2025, Oracle and/or its affiliates. All rights reserved.
Type ? or "help" for help, type "exit" to quit ttIsql.
connect "DSN=sample";
Connection successful:
DSN=sample;UID=timesten;DataStore=/tt/home/timesten/datastore/sample;
DatabaseCharacterSet=AL32UTF8; ConnectionCharacterSet=AL32UTF8; PermSize=200;
DDLReplicationLevel=3;
(Default setting AutoCommit=1)
Command> describe sampleuser.emp;
Table SAMPLEUSER.EMP:
 Columns:
   *TD
                                    NUMBER NOT NULL
    NAME
                                    CHAR (32)
1 table found.
(primary key columns are indicated with *)
Command> INSERT INTO sampleuser.emp VALUES (1, 'This is a test.');
1 row inserted.
Command> SELECT * FROM sampleuser.emp;
< 1, This is a test.
                                      >
1 row found.
```

Delete TimesTen Databases

If you delete a TimesTenClassic object, Kubernetes automatically deletes all of the Kubernetes objects and the resources it is using, including the StatefulSet, Service, and Pods. However, Kubernetes does not delete the PersistentVolumeClaims that contain the TimesTen databases. After you delete a TimesTenClassic object, you must manually delete the PersistentVolumeClaims (PVCs) associated with the object. After you manually delete the PVCs, Kubernetes recycles PersistentVolumes that are holding the databases. You may be able to control the recycling operation by using a Kubernetes volume retention policy. However, the TimesTen Operator has no control over this process.

The following example deletes PVCs associated with a replicated TimesTenClassic object called sample.

```
% kubectl delete pvc tt-persistent-sample-0
persistentvolumeclaim "tt-persistent-sample-0" deleted
```



```
% kubectl delete pvc tt-persistent-sample-1
persistentvolumeclaim "tt-persistent-sample-1" deleted
```

Locate the TimesTen Operator

The TimesTen Operator is configured in your Kubernetes cluster using a Deployment. Kubernetes automatically monitors the TimesTen Operator and restarts it if it fails. The TimesTen Operator runs in a Pod and the name of the Operator begins with timestenoperator, followed by arbitrary characters to make the name unique. If you specify multiple replicas when you deploy the Operator, there are multiple Pods. Only one Pod is active at a time. The remainder of the Pods wait for the active to fail, and if it does, then one of the Pods becomes active. TimesTenClassic objects continue to function if the Operator fails. When a new Operator is started by Kubernetes, the Operator automatically monitors and manages existing TimesTenClassic objects.

To display the Pods that are running the TimesTen Operator, use the kubectl get pods command. In the following example, there is one Pod that is running the TimesTen Operator. For more information about the TimesTen Operator, see About the TimesTen Operator.

% kubectl get pods				
NAME	READY	STATUS	RESTARTS	AGE
timesten-operator-5d7dcc7948-8mnz4	1/1	Running	0	3m21s

Modify TimesTen Connection Attributes

TimesTen uses connection attributes to define the attributes of a database. There are three types of connection attributes:

- Data store attributes: Define the characteristics of a database that can only be changed by destroying and recreating the database.
- First connection attributes: Define the characteristics of a database that can be changed by unloading and reloading the database into memory.
- General connection attributes: Control how applications access the database. These attributes persist for the duration of the connection.

For more information on TimesTen connection attributes, see List of Connection Attributes in the *Oracle TimesTen In-Memory Database Reference* and Connection Attributes for Data Manager DSNs or Server DSNs in the *Oracle TimesTen In-Memory Database Operations Guide*.

In a Kubernetes environment:

- You can only modify data store attributes by deleting the TimesTenClassic object and the PersistentVolumeClaims associated with the TimesTenClassic object. Doing so results in the deletion of the TimesTen databases.
- You can modify first connection and general connection attributes without deleting a TimesTenClassic object (which deletes the databases) and the PersistentVolumeClaims associated with the TimesTenClassic object. Note that there are TimesTen restrictions when modifying some of the first connection attributes.

To modify first or general connection attributes:

• You must first edit the db.ini file. Complete the procedure in the Manually Edit the db.ini File section. This section must be completed first.

Then, take these steps:



- If you are modifying first connection attributes, follow the procedure in the Modify First Connection Attributes section.
- If you are modifying general connection attributes, follow the procedure in the Modify General Connection Attributes section.

Manually Edit the db.ini File

Complete this section if you are modifying first or general connection attributes or both. This section must be completed before proceeding to the Modify First Connection Attributes or the Modify General Connection Attributes sections.

To modify first or general connection attribute requires a change in the sys.odbc.ini file.

If you have already created a TimesTenClassic object and you now want to change one or more first or general connection attributes in your sys.odbc.ini file, you must change the db.ini file.

The details as to how you should modify your db.ini file depends on the facility originally used to contain the db.ini file. (Possible facilities include ConfigMaps, Secrets, or init containers. See Populate the /ttconfig Directory.)

In this example, the ConfigMap facility was originally used to contain the db.ini file and to populate the /ttconfig directory of the TimesTen containers. The example modifies the sample ConfigMap.

The steps are as follows:

1. Use the kubectl describe command to review the contents of the db.ini file (represented in **bold**) located in the original sample ConfigMap.

```
% kubectl describe configmap sample
Name:
        sample
Namespace: mynamespace
Labels:
          <none>
Annotations: <none>
Data
____
adminUser:
sampleuser/samplepw
db.ini:
PermSize=200
DatabaseCharacterSet=AL32UTF8
ConnectionCharacterSet=AL32UTF8
schema.sql:
____
create sequence sampleuser.s;
create table sampleuser.emp (id number not null primary key, name char (32));
Events: <none>
```

 Use the kubectl edit command to modify the db.ini file in the original sample ConfigMap. Change the PermSize first connection attribute to 600 (represented in **bold**). Add the TempSize first connection attribute and set its value to 300 (represented in **bold**). Add the ConnectionCharacterSet connection attribute.



```
% kubectl edit configmap sample
# Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving this
# file will be reopened with the relevant failures.
apiVersion: v1
data:
 adminUser: |
   sampleuser/samplepw
 db.ini: |
   PermSize=600
   TempSize=300
   DatabaseCharacterSet=AL32UTF8
   ConnectionCharacterSet=AL32UTF8
 schema.sql: |
   create sequence sampleuser.s;
   create table sampleuser.emp (id number not null primary key, name char (32));
kind: ConfigMap
metadata:
 creationTimestamp: "2025-01-16T19:23:59Z"
 name: sample
 namespace: mynamespace
 resourceVersion: "71907255"
 selfLink: /api/v1/namespaces/mynamespace/configmaps/sample
uid: 0171ff7f-f789-11ea-82ad-0a580aed0453
. . .
configmap/sample edited
```

3. Use the kubectl describe command to verify the changes to the sample ConfigMap. (The changes are represented in **bold**.)

```
% kubectl describe configmap sample
Name: sample
Namespace: mynamespace
Labels:
             <none>
Annotations: <none>
Data
====
schema.sql:
____
create sequence sampleuser.s;
create table sampleuser.emp (id number not null primary key, name char (32));
adminUser:
____
sampleuser/samplepw
db.ini:
____
PermSize=600
TempSize=300
DatabaseCharacterSet=AL32UTF8
ConnectionCharacterSet=AL32UTF8
Events: <none>
```

You have successfully changed the sample ConfigMap. If you are modifying first connection attributes, proceed to the Modify First Connection Attributes section. If you are modifying only general connection attributes, proceed to the Modify General Connection Attributes section.



Modify First Connection Attributes

If you have not modified the db.ini file, proceed to the Manually Edit the db.ini File section. You must now delete the standby Pod and then delete the active Pod. When you delete a Pod that contains a container running TimesTen, the Operator creates a new Pod to replace the deleted Pod. This new Pod contains a new sys.odbc.ini file which is created from the contents of the db.ini file located in the /ttconfig directory.

Perform these steps to delete the standby Pod.

 Use the kubectl get command to determine which Pod is the standby Pod for the sample TimesTenClassic object. The active Pod is the Pod represented in the ACTIVE column. The standby Pod is the other Pod (not represented in the ACTIVE column). Therefore, for the sample TimesTenClassic object, the active Pod is sample-0, (represented in **bold**) and the standby Pod is sample-1.

% kubectl get ttc sample NAME STATE ACTIVE AGE sample Normal sample-0 47h

2. Delete the standby Pod (sample-1, in this example). This results in the Operator creating a new standby Pod to replace the deleted Pod. When the new standby Pod is created, it will use the newly modified sample ConfigMap. (You modified this ConfigMap in the Manually Edit the db.ini File section.)

```
% kubectl delete pod sample-1
pod "sample-1" deleted
```

 Use the kubectl get command to verify the standby Pod is up and running and the state is Normal.

Note that the state is StandbyDown (represented in **bold**).

% kubectl get ttc sample NAME STATE ACTIVE AGE sample StandbyDown sample-0 47h

Wait a few minutes, then run the command again. Note that the state has changed to Normal (represented in **bold**).

% kubectl get ttc sample NAME STATE ACTIVE AGE sample Normal sample-0 47h

4. Use the kubectl exec -it command to invoke the shell in the standby Pod (sample-1, in this example). Then, run the ttIsql utility to connect to the sample database. Note the new PermSize value of 600 and the new TempSize value of 300 in the connection output (represented in **bold**).

```
% kubectl exec -it sample-1 -c tt -- /bin/bash
% ttIsql sample
Copyright (c) 1996, 2025, Oracle and/or its affiliates. All rights reserved.
Type ? or "help" for help, type "exit" to quit ttIsql.
connect "DSN=sample";
Connection successful:
DSN=sample;UID=timesten;DataStore=/tt/home/timesten/datastore/sample;
DatabaseCharacterSet=AL32UTF8;ConnectionCharacterSet=AL32UTF8;
AutoCreate=0;PermSize=600;TempSize=300;DDLReplicationLevel=3;
ForceDisconnectEnabled=1;
(Default setting AutoCommit=1)
```



5. Fail over from the active Pod to the standby Pod. See "Fail Over" for details of the fail over process. Before you begin this step, ensure you quiesce your applications and you use the ttRepAdmin -wait command to wait until replication is caught up, such that all transactions that were executed on the active database have been replicated to the standby database. Once the standby is caught up, fail over from the active database to the standby by deleting the active Pod. When you delete the active Pod, the Operator automatically detects the failure and promotes the standby database to be the active.

Delete the active Pod (sample-0, in this example).

% kubectl delete pod sample-0
pod "sample-0" deleted

 Wait a few minutes, then use the kubectl get command to verify the active Pod is now sample-1 for the sample TimesTenClassic object and the state is Normal (represented in bold).

% kubectl get ttc sample NAME STATE ACTIVE AGE sample Normal sample-1 47h

7. Use the kubectl exec -it command to invoke the shell in the active Pod (sample-1, in this example). Then, run the ttIsql utility to connect to the sample database. Note the new PermSize value of 600 and the new TempSize value of 300 in the connection output (represented in **bold**).

```
% kubectl exec -it sample-1 -c tt -- /bin/bash
Last login: Thu Jan 16 15:50:29 UTC 2025 on pts/0
[timesten@sample-1 ~]$ ttIsql sample
```

Copyright (c) 1996, 2025, Oracle and/or its affiliates. All rights reserved. Type ? or "help" for help, type "exit" to quit ttIsql.

```
connect "DSN=sample";
Connection successful:
DSN=sample;UID=timesten;DataStore=/tt/home/timesten/datastore/sample;
DatabaseCharacterSet=AL32UTF8;ConnectionCharacterSet=AL32UTF8;
AutoCreate=0;PermSize=600;TempSize=300;DDLReplicationLevel=3;
ForceDisconnectEnabled=1;
(Default setting AutoCommit=1)
```

8. Use the kubectl exec -it command to invoke the shell in the standby Pod (sample-0, in this example). Then, run the ttIsql utility to connect to the sample database. Note the new PermSize value of 600 and the new TempSize value of 300 in the connection output (represented in **bold**).

```
% kubectl exec -it sample-0 -c tt -- /bin/bash
% ttIsql sample
```

Copyright (c) 1996, 2025, Oracle and/or its affiliates. All rights reserved. Type ? or "help" for help, type "exit" to quit ttIsql.

```
connect "DSN=sample";
Connection successful:
DSN=sample;UID=timesten;DataStore=/tt/home/timesten/datastore/sample;
DatabaseCharacterSet=AL32UTF8;ConnectionCharacterSet=AL32UTF8;
AutoCreate=0;PermSize=600;TempSize=300;DDLReplicationLevel=3;
ForceDisconnectEnabled=1;
(Default setting AutoCommit=1)
```



You have successfully modified the PermSize and the TempSize first connection attributes.

Modify General Connection Attributes

If you have not modified the db.ini file, proceed to the Manually Edit the db.ini File section. You can either directly modify the sys.odbc.ini file for the active TimesTen database and the sys.odbc.ini file for the standby TimesTen database or you can follow the steps in the Modify First Connection Attributes section. The first approach (modifying the sys.odbc.ini file directly) is less disruptive.

This section discusses the procedure for directly modifying the sys.odbc.ini files.

The sys.odbc.ini file is located in the TimesTen container of the Pod containing the active TimesTen database and in the TimesTen container of the Pod containing the standby TimesTen database. After you complete the modifications to the sys.odbc.ini files, subsequent applications can connect to the database using these general connection attributes.

This example illustrates how to edit the sys.odbc.ini files.

1. Use the kubectl exec -it command to invoke a shell in the active Pod. (In this example, sample-0 is the active Pod.)

```
% kubectl exec -it sample-0 -c tt -- /bin/bash
Last login: Thu Jan 16 22:43:26 UTC 2025 on pts/8
```

Verify the current directory (/tt/home/timesten).

% pwd
/tt/home/timesten

 Navigate to the directory where the sys.odbc.ini file is located. The sys.odbc.ini file is located in the /tt/home/timesten/instances/instance1/conf directory. Therefore, navigate to the instances/instance1/conf directory.

% cd instances/instance1/conf

4. Edit the sys.odbc.ini file, adding, modifying, or deleting the general connection attributes for your DSN. (sample, in this example.)

Note:

Ensure that you only make modifications to the TimesTen general connection attributes. Data store attributes and first connection attributes cannot be modified by directly editing the sys.odbc.ini file.

This example modifies the sample DSN, adding the ConnectionCharacterSet general connection attribute and setting its value equal to AL32UTF8 (represented in **bold**).

```
vi sys.odbc.ini
```

```
[ODBC Data Sources]
sample=TimesTen 22.1 Driver
tt=TimesTen 22.1 Driver
```

```
[sample]
Datastore=/tt/home/timesten/datastore/sample
PermSize=200
DatabaseCharacterSet=AL32UTF8
```



```
ConnectionCharacterSet=AL32UTF8
DDLReplicationLevel=3
AutoCreate=0
ForceDisconnectEnabled=1
```

5. Use the ttIsql utility to connect to the sample database and verify the value of the ConnectionCharacterSet attribute is AL32UTF8 (represented in **bold**).

```
% ttIsql sample
Copyright (c) 1996, 2025, Oracle and/or its affiliates. All rights reserved.
Type ? or "help" for help, type "exit" to quit ttIsql.
```

```
connect "DSN=sample";
Connection successful:
DSN=sample;UID=timesten;DataStore=/tt/home/timesten/datastore/sample;
DatabaseCharacterSet=AL32UTF8;ConnectionCharacterSet=AL32UTF8;
AutoCreate=0;PermSize=200;DDLReplicationLevel=3;ForceDisconnectEnabled=1;
(Default setting AutoCommit=1)
```

You have successfully modified the sys.odbc.ini file located in the TimesTen container of the active Pod (in this example, sample-0). Use the same procedure to modify the sys.odbc.ini file located in the TimesTen container of the standby Pod (in this example, sample-1).

For example:

Use the kubectl exec -it command to invoke a shell in the standby Pod (sample-1, in this example).

% kubectl exec -it sample-1 -c tt -- /bin/bash Last login: Thu Jan 16 23:08:08 UTC 2025 on pts/0

2. Verify the current directory (/tt/home/timesten).

```
% pwd
/tt/home/timesten
```

 Navigate to the directory where the sys.odbc.ini file is located. The sys.odbc.ini file is located in the /tt/home/timesten/instances/instance1/conf directory. Therefore, navigate to the instances/instance1/conf directory.

```
% cd instances/instance1/conf
```

4. Edit the sys.odbc.ini file, adding, modifying, or deleting the same general connection attributes that you modified for the active database. Therefore, this example modifies the sample DSN, adding the ConnectionCharacterSet general connection attribute and setting its value equal to AL32UTF8 (represented in **bold**).

```
vi sys.odbc.ini
```

```
[ODBC Data Sources]
sample=TimesTen 22.1 Driver
tt=TimesTen 22.1 Driver
```

```
[sample]
Datastore=/tt/home/timesten/datastore/sample
PermSize=200
DatabaseCharacterSet=AL32UTF8
ConnectionCharacterSet=AL32UTF8
DDLReplicationLevel=3
AutoCreate=0
```



```
ForceDisconnectEnabled=1 ...
```

5. Use the ttIsql utility to connect to the sample database and verify the value of the ConnectionCharacterSet attribute is AL32UTF8 (represented in **bold**).

```
% ttIsql sample
```

```
Copyright (c) 1996, 2025, Oracle and/or its affiliates. All rights reserved. Type ? or "help" for help, type "exit" to quit ttIsql.
```

```
connect "DSN=sample";
Connection successful:
DSN=sample;UID=timesten;DataStore=/tt/home/timesten/datastore/sample;
DatabaseCharacterSet=AL32UTF8;ConnectionCharacterSet=AL32UTF8;
AutoCreate=0;PermSize=200;DDLReplicationLevel=3;ForceDisconnectEnabled=1;
(Default setting AutoCommit=1)
```

You have successfully modified the sys.odbc.ini file located in the TimesTen container of the active Pod (sample-0) and the sys.odbc.ini file located in the TimesTen container of the standby Pod (sample-1). The ConnectionCharacterSet general connection attribute has also been modified.

12 Manage TimesTen Scaleout

This chapter discusses how the TimesTen Operator manages and repairs TimesTen Scaleout. Topics:

- About Managing TimesTen Scaleout
- About Single Data Instance Failure
- About Management Instance Failure
- About the waiting for seed State
- About Failure of All Data Instances
- About High Level States
- About Management States
- About Database and Element States
- About the ManualInterventionRequired State
- About Suspending Management
- Simulate Single Data Instance Failure
- Simulate Management Instance Failure
- Simulate Replica Set Failure with Restart
- Simulate Replica Set Failure with Manual
- Suspend Management
- Set reexamine Datum

About Managing TimesTen Scaleout

TimesTen Scaleout delivers high performance, fault tolerance, and scalability within a highly available in-memory database that provides persistence and recovery. Since a database is distributed across multiple hosts, some components of the database may fail while others continue to operate.

TimesTen Scaleout supports error and failure detection with automatic recovery for many error and failure situations in order to maintain a continuous operation for all applications.

The TimesTen Operator implements best practices for how to handle failures for TimesTen Scaleout. For more information about how TimesTen Scaleout handles failures, see Recovering from Failure in the Oracle TimesTen In-Memory Database Scaleout User's Guide

In particular, the Operator detects and handles the following failure cases:

- If a TimesTen instance or element fails, the Operator restarts it.
- If an entire replica set fails and if all elements in the replica set reach the waiting for seed state, the Operator unloads and reloads the database to resolve it (by default). For details about how TimesTen Scaleout recovers from a down replica set, see Recovering from a Down Replica Set in the Oracle TimesTen In-Memory Database Scaleout User's Guide.



• If all data instances fail, the Operator detects and reports the failure.

The Operator communicates to the TimesTen agent running in the tt container in each Pod running TimesTen. The agent determines information about the state of TimesTen running in the container and sends that information back to the Operator. The Operator analyzes this information and determines the health and state of TimesTen. This information is summarized in well-defined states. The Operator uses state machines to determine the appropriate set of commands to be executed to detect failures and, if possible, repair TimesTen. These states are discussed later in the chapter.

Let's take a deeper look at how the Operator detects and repairs TimesTen Scaleout. Specifically, let's look at how the Operator handles single data instance failure, management instance failure, entire replica set failure, and total database failure.

About Single Data Instance Failure

TimesTen Scaleout is intended to be resilient to single data instance failures. In your Kubernetes environment, if a data instance fails, the TimesTen Operator starts it back up. Once it is restarted, TimesTen Scaleout reloads the element of the database. See Recovering When a Data Instance Is Down in the Oracle TimesTen In-Memory Database Scaleout User's Guide.

About Management Instance Failure

The TimesTen Operator supports one management instance per TimesTenScaleout object. If the management instance is down, the Operator ignores the state of the data instances until the management instance is back up. Without a management instance, the grid and TimesTen database continue to function. However, should a data instance fail without a working management instance, it is not possible to repair the data instance.

About the waiting for seed State

In TimesTen Scaleout, there could be a situation where an entire replica set fails simulataneously. If all elements in a replica set fail at the same time, it may not be possible to reload any of the elements without unloading the entire database and reloading it. In addition, a reload may cause TimesTen Scaleout to discard transactions against other replica sets that were committed prior to the unload operation.

For more information about replica set failure in TimesTen Scaleout, see Recovering from a Down Replica Set in the Oracle TimesTen In-Memory Database Scaleout User's Guide.

If this case occurs in your Kubernetes environment, the default behavior is for the TimesTen Operator to detect this situation and forcibly unload and reload the database when the situation occurs. This operation is triggered when the Operator notices that all elements in a replica set are in the waiting for seed state.

You have the option of controlling the behavior of the Operator when all elements in a replica set are in the waiting for seed state. You do this by using a TimesTenScaleout object's .spec.ttspec.replicaSetRecovery datum, and setting a particular value for the datum. Accepted values are the following:

- Restart: The Operator forcibly unloads and reloads the database when a total replica set failure occurs. This is the default.
- Manual: The Operator changes the state of the TimesTenScaleout object to ManualInterventionRequired when a total replica set failure occurs. The Operator takes



no further action to repair the grid. You must repair it. See About the ManualInterventionRequired State and Set reexamine Datum for details.

For more information on the .spec.ttspec.replicaSetRecovery datum for a TimesTenScaleout object, see TimesTenScaleoutSpecSpec.

About Failure of All Data Instances

If all elements in a database fail simultaneously, there must be caution taken in reloading the database. TimesTen Scaleout does not automatically reload a database if all data instances fail. You must perform this reload. It is essential that all possible data instances are up before a reload is attempted.

For more information about failure of all data instances in TimesTen Scaleout, see Database Recovery in the Oracle TimesTen In-Memory Database Scaleout User's Guide.

If this case occurs in a Kubernetes environment, the TimesTen Operator does not attempt to reload the database after all data instances fail. However, the Operator does detect this case and changes the state of a TimesTenScaleout object to ManualInterventionRequired. See About the ManualInterventionRequired State and Set reexamine Datum.

About High Level States

The TimesTen Operator maintains a High Level state for TimesTenScaleout objects.

These High Level states are as follows:

- DatabaseDown
- DatabaseImpeded
- DatabasePartial
- DatabaseRestarting
- DatabaseRestartRequired
- Failed
- Initializing
- ManualInterventionRequired
- Normal
- Reexamine
- Unmanaged

DatabaseDown

The database is unusable. The Operator attempts to fix it. If it cannot be fixed, the Operator moves the object to the ManualInterventionRequired state.

DatabaseImpeded

The database within the grid is fully operational, but one or more elements are not functional. All data in the database is available and all SQL is accepted.



DatabasePartial

The database is up, but some data is not available. One or more replica sets are not available.

DatabaseRestarting

The database is in the process of being forcibly unloaded and reloaded after a DatabaseRestartRequired condition.

DatabaseRestartRequired

While the database is up (at least partially), it must be stopped and restarted (unloaded and reloaded) in order to restore functionality. This can occur when all elements in a replica set fail simultaneously and all elements are unloadable due to a waiting for seed condition. When this occurs, the database must be unloaded and reloaded. During this time, committed transactions may be lost. See Database Recovery and Recovering from a Down Replica Set in the *Oracle TimesTen In-Memory Database Scaleout User's Guide*.

Failed

If a problem occurs while Initializing a TimesTenScaleout object, the object transitions to the Failed state. Once in this state, the Operator does not attempt to repair the object. You must delete it. Use the kubectl describe command to examine the Operator logs to determine the cause of the problem and then recreate the object.

Initializing

After you create a TimesTenScaleout object in your Kubernetes cluster, the Operator creates the StatefulSets and Services that are required to deploy a TimesTenScaleout grid and database. The Operator assigns a High Level state of Initializing to the TimesTenScaleout object.

ManualInterventionRequired

If the Operator encounters a problem that it does not know how to fix, the Operator places a TimesTenScaleout object into this state. The Operator takes no further action to fix the object. You can set the object's .spec.ttspec.reexamine datum to cause the Operator to re-engage with the object. For information about .spec.ttspec.reexamine, see Set reexamine Datum.

Normal

The grid and database are up and operating as they should.

Reexamine

If a TimesTenScaleout object is in the ManualInterventionRequired state, and you modify the object's .spec.ttspec.reexamine datum, the TimesTen Operator moves the object into the Reexamine state. The Operator examines the state of grid and database. If healthy, the Operator returns the object to the Normal state. Otherwise, the object re-enters the ManualInterventionRequired state.



Unmanaged

The grid does not have a functional management instance. Until the management instance is fixed, the grid cannot be further monitored, managed or controlled. The Operator will attempt to fix the management instance.

About Management States

The Operator uses the TimesTen ttGridAdmin mgmtExamine utility to determine the health of the management instance. The Operator also synthesizes a management state that describes the status of the management instance in a single value.

For information about the TimesTen ttGridAdmin mgmtExamine utility, see Examine Management Instances (mgmtExamine) in the Oracle TimesTen In-Memory Database Reference.

For example, you can use the kubectl get tts command to observe the state of a TimesTenScaleout object.

kubectl get tts samplescaleoutOVERALLMGMTCREATELOADOPENAGEsamplescaleoutNormalNormalcreatedloaded-completeopen10m

Note that the management state is **Normal**. There are additional examples in the chapter where you have the opportunity to observe the management state.

Here are the management states:

- ActiveAgentUp
- ActiveDaemonUp
- ActiveDown
- Error
- Normal
- Unknown

ActiveAgentUp

The TimesTen agent in the container that should be running the management instance is up, but the management instance has not yet been started.

ActiveDaemonUp

The management instance has been started, but is not yet functional.

ActiveDown

The management instance is down.



Error

There is an unexpected error with the management instance.

Normal

The management instance is functioning normally.

Unknown

The state of the management instance cannot be determined.

About Database and Element States

TimesTen Scaleout defines a set of overall database and element status values to detemine the status of a database or element. The TimesTen Operator uses these status values to assess the state of the database represented by a TimesTenScaleout object.

In TimesTen Scaleout, the overall database status is encoded in three strings:

- How created is the database?
- How loaded is the database?
- How open is the database?

This triplet of strings is returned as part of the output of the TimesTen ttGridAdmin dbStatus utility.

For information about the database states, see Display the Status of the Database and All Elements in the *Oracle TimesTen In-Memory Database Scaleout User's Guide*. For information about the ttGridAdmin dbstatus utility, see Monitor the Status of a Database (dbStatus) in the *Oracle TimesTen In-Memory Database Reference*.

The TimesTen Operator uses TimesTen Scaleout database states to report the state of the database for a TimesTenScaleout object.

For example, this code snippet uses the kubectl get command to return the status for a deployed TimesTenScaleout object.

complete	open	2d				
samplescaleou	t Nor	nal		Normal	created	loaded-
LOAD		OPEN	AGE			
NAME	OVE	RALL		MGMT	CREATE	
kubectl get t	ts samp	lescale	eout			

Note the following:

- The High Level state of the TimesTenScaleout object is Normal (as indicated by the OVERALL field).
- The management instance state is Normal (as indicated by the MGMT field).
- The database creation state is created (as indicated by the CREATE field).
- The database loaded state is loaded-complete (as indicated by the LOAD field).



• The database open state is open (as indicated by the OPEN field).

TimesTen Scaleout also keeps a state for each element in a database. See Troubleshooting Based on Element Status in the *Oracle TimesTen In-Memory Database Scaleout User's Guide* for details on element states.

The Operator does not monitor these element states. However, if all the elements in a replica set are in the waiting for seed state, the Operator checks the value of a TimesTenScaleout object's .spec.ttspec.replicaSetRecovery datum:

- If the value is Restart, the DatabaseRestartRequired High Level state is triggered.
- If the value is Manual, the Operator moves the TimesTenScaleout object to the ManualInterventionRequired state.

About the ManualInterventionRequired State

If the TimesTen Operator determines it cannot repair a TimesTenScaleout object, the Operator changes the High Level state of the object to ManualInterventionRequired. The Operator does not further manage an object in this state. In addition, the Operator makes no attempt to determine its state nor to repair it.

In TimesTen Scaleout, there are several troubleshooting scenarios that you can review to identify and possibly fix the problem. For more information about the troubleshooting scenarios, see Recovering from Failure in the *Oracle TimesTen In-Memory Database Scaleout User's Guide*.

If you identify and fix the problem, you can cause the Operator to resume management of a TimesTenScaleout object. You do this by specifying a TimesTenScaleout object's spec.ttspec.reexamine datum. When this datum is specified, the Operator moves the object to the Reexamine state. For information about the Reexamine state, see Set reexamine Datum.

About Suspending Management

There may be a situation in which you want to manually perform maintenance operations. In such a situation, you do not want the Operator to interfere and attempt to perform repair or recovery operations on your grid and database.

One alternative is to stop the Operator (by deleting the timesten-operator Deployment). This action prevents the Operator from interfering or performing repair/recovery operations. However, if you have more than one TimesTenScaleout object deployed in your Kubernetes environment and you delete the Operator, this interferes with the management of all the TimesTenScaleout objects, when perhaps only one of them needs manual intervention.

Another approach is to ask the Operator to take no action for one TimesTenScaleout object. To do this, specify a TimesTenScaleout object's .spec.ttspec.stopManaging datum. The Operator examines the value of .spec.ttspec.stopManaging and if it has changed since the last time the Operator examined it, the Operator changes the state of the TimesTenScaleout object to ManualInterventionRequired. This causes the Operator to no longer examine the status of the grid and database. Nor does the Operator examine the Pods, the containers, and the instances associated with this particular TimesTenScaleout object. For an example showing how to set the .spec.ttspec.stopManaging datum for a TimesTenScaleout object, see Suspend Management.

To cause the Operator to resume management of the TimesTenScaleout object, change the value of the object's .spec.ttspec.reexamine datum. See Set reexamine Datum for details.



See TimesTenScaleoutSpecSpec for information about the TimesTenScaleout object definition.

Simulate Single Data Instance Failure

Let's simulate a single data instance failure and observe how a TimesTenScaleout object transitions through various state changes.

Note:

This example is for demonstration purposes only. Do not attempt this example in a production environment.

In the example, there is a deployed TimesTenScaleout object that is functioning properly.

NAME	OVERALL	MGMT	CREATE	LOAD	OPEN	AGE
samplescaleout	Normal	Normal	created	loaded-complete	open	10m

Note the High Level state is Normal, the management state is Normal, and the database state is created, loaded-complete, open.

To simulate a single data instance failure, let's delete a Pod that contains a data instance. Here are the Pods:

kubectl get pods				
NAME	READY	STATUS	RESTARTS	AGE
samplescaleout-data-1-0	2/2	Running	0	11m
samplescaleout-data-1-1	2/2	Running	0	11m
samplescaleout-data-1-2	2/2	Running	0	11m
samplescaleout-data-2-0	2/2	Running	0	11m
samplescaleout-data-2-1	2/2	Running	0	11m
samplescaleout-data-2-2	2/2	Running	0	11m
samplescaleout-mgmt-0	2/2	Running	0	11m
samplescaleout-zk-0	1/1	Running	0	11m
samplescaleout-zk-1	1/1	Running	0	10m
samplescaleout-zk-2	1/1	Running	0	9m35s
timesten-operator-7677964df9-sp2zp	1/1	Running	0	7d3h

Let's delete the samplescaleout-data-1-0 Pod and observe the behavior.

1. Delete the Pod.

```
kubectl delete pod samplescaleout-data-1-0
pod "samplescaleout-data-1-0" deleted
```

2. Use the kubectl get command to observe state transitions.

kubect	l get tts	samplescaleout			
NAME		OVERALL	MGMT	CREATE	LOAD
OPEN	AGE				
sample	scaleout	DatabaseImpeded	Normal	created	loaded-functional
open	16m				



The High Level state is DatabaseImpeded, indicating that the database within the grid is fully operational, but one or more elements is not functional. The database loaded state is loaded-functional, indicating loading is in progress and at least one element from each replica set is loaded.

kubectl get tts samplescaleoutOVERALLMGMTCREATELOADOPENAGEsamplescaleoutNormalNormalcreatedloaded-completeopen18m

The object transitioned to the Normal High Level state, indicating the grid and database are functioning normally. The database state is loaded-complete, indicating the element loaded successfully.

Even though there was a single data instance failure, TimesTen Scaleout fully recovered. There was no manual intervention required.

Simulate Management Instance Failure

Let's simulate a management instance failure. Let's observe how a TimesTenScaleout object transitions through various state changes.

Note:

This example is for demonstration purposes only. Do not attempt this example in a production environment.

In the example, there is a deployed TimesTenScaleout object that is functioning properly.

kubectl get tts	samplescal	eout				
NAME	OVERALL	MGMT	CREATE	LOAD	OPEN	AGE
samplescaleout	Normal	Normal	created	loaded-complete	open	68m

Note the High Level state is Normal, the management state is Normal, and the database state is created, loaded-complete, open.

To simulate a management instance failure, let's delete the Pod that contains the management instance. Here are the Pods:

kubectl get pods				
NAME	READY	STATUS	RESTARTS	AGE
samplescaleout-data-1-0	2/2	Running	0	57m
samplescaleout-data-1-1	2/2	Running	0	73m
samplescaleout-data-1-2	2/2	Running	0	73m
samplescaleout-data-2-0	2/2	Running	0	73m
samplescaleout-data-2-1	2/2	Running	0	73m
samplescaleout-data-2-2	2/2	Running	0	73m
samplescaleout-mgmt-0	2/2	Running	0	73m
samplescaleout-zk-0	1/1	Running	0	73m
samplescaleout-zk-1	1/1	Running	0	72m
samplescaleout-zk-2	1/1	Running	0	71m
timesten-operator-7677964df9-sp2zp	1/1	Running	0	7d4h



Let's delete the samplescaleout-mgmt-0 Pod and observe the behavior.

1. Delete the Pod.

```
kubectl delete pod samplescaleout-mgmt-0
pod "samplescaleout-mgmt-0" deleted
```

2. Use the kubectl get command to observe state transitions.

kubectl get tts samplescaleout NAME OVERALL MGMT CREATE LOAD OPEN AGE samplescaleout **Unmanaged ActiveDown** created loaded-complete open 79m

The High Level state is Unmanaged, indicating that the grid has no functional management instance. As a result, the grid cannot be further managed, monitored, or controlled. The management instance state is ActiveDown, indicating the management instance is down. Note that since the management instance is down, the Operator ignores the state of the data instances until the management instance is back up.

kubectl get tts samplescaleoutNAMEOVERALLMGMTCREATELOADOPENAGEsamplescaleoutUnmanagedActiveDaemonUpcreatedloaded-completeopen81m

The object remains in the Unmanaged High Level state. The management state transitions to ActiveDaemonUp, indicating the management instance has been started, but is not yet functional.

kubectl get tts	samplescal	eout				
NAME	OVERALL	MGMT	CREATE	LOAD	OPEN	AGE
samplescaleout	Normal	Normal	created	loaded-complete	open	81m

The object transitioned to the Normal High Level state, indicating the grid and database are functioning normally. The management state is Normal, indicating that management instance is functioning normally.

Even though there was a management instance failure, TimesTen Scaleout fully recovered. There was no manual intervention required.

Simulate Replica Set Failure with Restart

Let's simulate a replica set failure. In this example, a TimesTenScaleout object's .spec.ttspec.replicaSetRecovery datum has not been specified. The default of Restart is assumed, indicating that the TimesTen Operator forcibly unloads and reloads the database when a total replica set failure occurs.

Let's observe how a TimesTenScaleout object transitions through various state changes.



Note:

This example is for demonstration purposes only. Do not attempt this example in a production environment.

In the example, there is a deployed TimesTenScaleout object that is functioning properly.

samplescaleout	Normal	Normal	created	loaded-complete	open	99m
NAME	OVERALL	MGMT	CREATE	LOAD	OPEN	AGE
kubectl get tts	samplescal	eout				

Note the High Level state is Normal, the management state is Normal, and the database state is created, loaded-complete, open.

In this example, there are three replica sets.

READY	STATUS	RESTARTS	AGE
2/2	Running	0	11m
2/2	Running	0	11m
2/2	Running	0	11m
2/2	Running	0	11m
2/2	Running	0	11m
2/2	Running	0	11m
2/2	Running	0	11m
1/1	Running	0	11m
1/1	Running	0	10m
1/1	Running	0	9m35s
1/1	Running	0	7d3h
	2/2 2/2 2/2 2/2 2/2 2/2 2/2 2/2 1/1 1/1	2/2Running2/2Running2/2Running2/2Running2/2Running2/2Running2/2Running1/1Running1/1Running1/1Running1/1Running	2/2Running02/2Running02/2Running02/2Running02/2Running02/2Running02/2Running01/1Running01/1Running01/1Running01/1Running0

Let's delete the samplescaleout-data-1-0 and samplescaleout-data-2-0 Pods that belong to one of the replica sets.

1. Delete the Pods.

```
kubectl delete pod samplescaleout-data-1-0;kubectl delete pod
samplescaleout-data-2-0
pod "samplescaleout-data-1-0" deleted
pod "samplescaleout-data-2-0" deleted
```

2. Use the kubectl get command to observe state transitions.

kubect	l get tts	samplescaleout			
NAME		OVERALL	MGMT	CREATE	LOAD
OPEN	AGE				
sample	scaleout	DatabasePartial	Normal	created	loaded-incomplete
open	111m				

The High Level state is DatabasePartial indicating that the database is up, but some data is not available. One or more replica sets have failed completely. The database loaded



state is loaded-incomplete, indicating that at least one replica set has no elements that finished loading successfully.

 kubectl get tts samplescaleout
 MGMT
 CREATE

 NAME
 OVERALL
 MGMT
 CREATE

 LOAD
 OPEN
 AGE

 samplescaleout
 DatabaseRestartRequired
 Normal
 created
 loading

 incomplete
 open
 112m
 Image: Create description of the second descrip

The object transitions to the DatabaseRestartRequired High Level state. This state indicates that while the database is up (at least partially), the database must be stopped and restarted (unloaded and reloaded) in order to restore functionality. This can occur when all elements in a replica set fail simultaneously and such elements are unloadable due to a waiting for seed condition. When this happens the database must be unloaded and reloaded. At that time, committed transactions may be lost.

kubectl get tts samplescaleoutNAMEOVERALLMGMTCREATELOADOPENAGEsamplescaleoutDatabaseRestartingNormalcreatedincompleteopen114m

The object transitions to the DatabaseRestarting High Level state. This state indicates that the database is being forcibly unloaded and reloaded after a DatabaseRestartRequired condition.

kubectl get tts samplescaleout NAME OVERALL MGMT CREATE LOAD OPEN AGE samplescaleout **Normal** Normal created **loaded-complete closed** 114m

The object transitions to the Normal High Level state, indicating that the grid and database are functioning normally. The database loaded state changed to loaded-complete, indicating every element was loaded successfully. The database open state is closed, indicating the database is closed for connections.

kubectl get tts samplescaleoutOVERALLMGMTCREATELOADOPENAGEsamplescaleoutNormalNormalcreatedloaded-completeopen114m

The object remains in the Normal High Level state. The database open state changed to open, indicating the database is now open for connections.

Even though there was a total replica set failure, TimesTen Scaleout forcibly unloaded and reloaded the database. The Operator returned the grid and database to a Normal state, indicating both are functioning normally. There was no manual intervention required.

Simulate Replica Set Failure with Manual

Let's simulate a total replica set failure where a TimesTenScaleout object's .spec.ttspec.replicaSetRecovery datum has a value of Manual.



Note:

This example is for demonstration purposes only. Do not attempt this example in a production environment.

Let's take a look at the TimesTenScaleout object definition.

```
cat samplescaleout2.yaml
apiVersion: timesten.oracle.com/v4
kind: TimesTenScaleout
metadata:
  name: samplescaleout2
spec:
  ttspec:
    storageClassName: oci-bv
    storageSize: 250G
    image: container-registry.oracle.com/timesten/timesten:22.1.1.34.0
    imagePullSecret: sekret
    replicaSetRecovery: Manual
    dbConfigMap:
    - samplescaleout2
    k: 2
    nReplicaSets: 3
    nMgmt: 1
    nZookeeper: 3
```

Note the .spec.ttspec.replicaSetRecovery datum for the object has a value of Manual, indicating that the Operator will set this TimesTenScaleout object's High Level state to ManualInterventionRequired when a total replica set failure occurs.

Let's simulate a total replica set failure with this object. Before we begin, let's do a quick check of the state of the object.

kubectl get tts	samplescale	out2				
NAME	OVERALL	MGMT	CREATE	LOAD	OPEN	AGE
samplescaleout2	Normal	Normal	created	loaded-complete	open	11m

Note the High Level state is Normal, the management state is Normal, and the database state is created, loaded-complete, open.

In this example, there are three replica sets.

kubectl get pods				
NAME	READY	STATUS	RESTARTS	AGE
samplescaleout2-data-1-0	2/2	Running	0	12m
samplescaleout2-data-1-1	2/2	Running	0	12m
samplescaleout2-data-1-2	2/2	Running	0	12m
samplescaleout2-data-2-0	2/2	Running	0	12m
samplescaleout2-data-2-1	2/2	Running	0	12m
samplescaleout2-data-2-2	2/2	Running	0	12m
samplescaleout2-mgmt-0	2/2	Running	0	13m
samplescaleout2-zk-0	1/1	Running	0	13m
samplescaleout2-zk-1	1/1	Running	0	12m



samplescaleout2-zk-2	1/1	Running	0	11m
timesten-operator-7677964df9-sp2zp	1/1	Running	0	7d6h

Let's delete the samplescaleout2-data-1-0 and samplescaleout2-data-2-0 Pods that belong to one of the replica sets.

1. Delete the Pods.

```
kubectl delete pod samplescaleout2-data-1-0;kubectl delete pod
samplescaleout2-data-2-0
pod "samplescaleout2-data-1-0" deleted
pod "samplescaleout2-data-2-0" deleted
```

2. Use the kubectl get command to observe state transitions.

kubectl get tts s	amplescaleout2			
NAME	OVERALL	MGMT	CREATE	LOAD
OPEN AGE				
samplescaleout2	DatabasePartial	Normal	created	loaded-incomplete
open 18m				

The High Level state is DatabasePartial indicating that the database is up, but some data is not available. One or more replica sets have failed completely. The database loaded state is loaded-incomplete, indicating that at least one replica set has no elements that finished loading successfully.

 kubectl get tts samplescaleout2
 MGMT
 CREATE

 NAME
 OVERALL
 MGMT
 CREATE

 LOAD
 OPEN
 AGE

 samplescaleout2
 ManualInterventionRequired
 Normal
 created
 loading

 incomplete
 open
 20m
 Omega
 OPEN
 Normal
 CREATE

The object transitions to the ManualInterventionRequired High Level state. The Operator takes no further action to fix the object. The database loaded state remains loading-incomplete. Recall that you can set the .spec.ttspec.reexamine datum to cause the Operator to re-engage with the object. See Set reexamine Datum for details.

There was a total replica set failure. Because the TimesTen Scaleout object's .spec.ttspec.replicaSetRecovery datum had a value of Manual, the Operator set the object to the ManualInterventionRequired state. Review the information in Recovering from a Down Replica Set in the Oracle TimesTen In-Memory Database Scaleout User's Guide for details about how to fix this failure. Next, review Set reexamine Datum in this book to give

Suspend Management

control back to the Operator.

Let's walk through an example that shows you how to suspend management of a TimesTenScaleout object.

In the example, there is a deployed TimesTenScaleout object that is functioning properly.

kubectl get tts samplescaleoutOVERALLMGMTCREATELOADOPENAGEsamplescaleoutNormalNormalcreatedloaded-completeopen3h33m

Note the High Level state is Normal, the management state is Normal, and the database state is created, loaded-complete, open.

- 1. Use the kubectl edit command to edit the TimesTenScaleout object, making the following changes:
 - If there is a line for .spec.ttspec.stopManaging in the file, then modify its value. It must be different than the current value.
 - If there is no line for .spec.ttspec.stopManaging in the file, then add a line and specify a value.

In this example, there is no .spec.ttspec.stopManaging line. This example adds the .spec.ttspec.stopManaging line and adds a value of Suspend.

```
kubectl edit timestenscaleout samplescaleout
# Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving
this file will be
# reopened with the relevant failures.
#
apiVersion: timesten.oracle.com/v4
kind: TimesTenScaleout
metadata:
  creationTimestamp: "2024-01-18T23:57:56Z"
  generation: 1
. . .
spec
. . .
  ttspec:
    . . .
    dbConfigMap:
    - samplescaleout
    . . .
    k: 2
    . . .
    nReplicaSets: 3
    nZookeeper: 3
    replicaSetRecovery: Restart
    stopManaging: Suspend
. . .
timestenscaleout.timesten.oracle.com/samplescaleout edited
```

2. Use the kubectl get command to observe the behavior..

kubectl get tts	samplescaleout			
NAME	OVERALL	MGMT	CREATE	
LOAD	OPEN AGE			
samplescaleout	ManualInterventionRequired	Normal	created	loaded-
complete open	3h33m			


The Operator sets the TimesTenScaleout object to the ManualInterventionRequired High Level state. The Operator takes no further action on the object. You can now perform manual operations on your grid and database. When you have completed such operations and are ready for the Operator to resume management, you can set the .spec.ttspec.rexamine datum for the object. For an example that shows you how to set a TimesTenScaleout object's .spec.ttspec.rexamine datum, see Set reexamine Datum.

Set reexamine Datum

If a TimesTenScaleout object is in the ManualInterventionRequired state, you can set/modify the object's .spec.ttspec.reexamine datum to instruct the TimesTen Operator to move the object into the Reexamine state. In the Reexamine state, the Operator examines the state of TimesTen and the database. If both are healthy, the Operator returns the object to the High Level Normal state. If not healthy, the object re-enters the ManualInterventionRequired state.

Let's walk through an example that shows you how to reexamine a TimesTenScaleout object.

1. Verify a TimesTenScaleout object is in the ManualInterventionRequired state.

kubectl ge	t tts	samplesca	aleout			
NAME		OVERALL		MGMT	CREATE	
LOAD		OPEN	AGE			
samplescal	eout	ManualI	nterventionRequired	Normal	created	loaded-
complete	open	3h48m				

- 2. Use the kubectl edit command to edit the TimesTenScaleout object, making the following changes:
 - If there is a line for .spec.ttspec.reexamine in the file, then modify its value. It must be different than the current value.
 - If there is no line for .spec.ttspec.reexamine in the file, then add a line and specify a value.

In this example, there is no .spec.ttspec.reexamine line. This example adds the .spec.ttspec.reexamine line and assigns a value of Reexamine1.

```
kubectl edit timestenscaleout samplescaleout
# Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving
this file will be
# reopened with the relevant failures.
apiVersion: timesten.oracle.com/v4
kind: TimesTenScaleout
metadata:
 creationTimestamp: "2024-01-18T23:57:56Z"
  generation: 1
. . .
spec
. . .
  ttspec:
    . . .
    dbConfiqMap:
    - samplescaleout
    . . .
```



```
k: 2
...
nReplicaSets: 3
nZookeeper: 3
replicaSetRecovery: Restart
reexamine: Reexamine1
...
timestenscaleout.timesten.oracle.com/samplescaleout edited
```

3. Use the kubectl get command to observe the behavior.

kubectl get tts samplescaleout NAME OVERALL MGMT CREATE LOAD OPEN AGE samplescaleout **Reexamine** Normal created loaded-complete open 3h52m

The object is in the Reexamine High Level state. The Operator examines the state of TimesTen and the database. If healthy, the Operator moves the object to the High Level Normal state. If not healthy, the object re-enters the ManualInterventionRequired state.

kubectl get tts samplescaleout NAME OVERALL MGMT CREATE LOAD OPEN AGE samplescaleout **Normal** Normal created loaded-complete open 3h53m

The object transitioned to the Normal High Level state, indicating the grid and database are functioning normally.



13 Optimize Client/Server Performance

This chapter discusses how to optimize client/server performance. It discusses NodePort Services and the TimesTen Connection Manager.

Topics:

- About Client/Server Challenges
- About NodePort Services
- About the Connection Manager
- How to Use the Connection Manager
- About the NodePort Service for the Connection Manager
- About the TimesTen Operator Configuration

About Client/Server Challenges

You may want to access TimesTen databases that are inside a Kubernetes cluster by using client/server from outside the cluster. While this is possible, it presents the following challenges:

- Clients outside the Kubernetes cluster access TimesTen databases inside the cluster by using a NodePort Service. Client applications can connect to any node in the Kubernetes cluster. The node that the application connects to may not be a node on which TimesTen is running. Kubernetes routes the connection to a TimesTen instance in the cluster. This works seamlessly, but results in additional network round trips and reduced performance.
- If changes from DML operations are coming from client/server applications, it is important to route those connections to active TimesTen instances as they are the only writable instances. For read-only applications, it is desirable to route connections to both active and standby instances, so that standby instances can participate in the workload.

The TimesTen Operator uses NodePort Services and provides a Connection Manager that provides solutions to these client/server challenges and allows applications to avoid these issues.

About NodePort Services

To implement this feature, the TimesTen Operator creates additional NodePort Services. The TimesTen Operator creates one Service for each Pod that the Operator creates.

For example, if you configure a TimesTenClassic object called sample to have three Pods, the TimesTen Operator creates three NodePort Services called sample-np-0, sample-np-1, and sample-np-2. Each Service maps to the corresponding Pod. For example, NodePort Service sample-np-1 maps to Pod sample-1. Kubernetes deletes these Services when you delete a TimesTenClassic object.

Since these are NodePort Services, Kubernetes automatically assigns a port number on the cluster's external network that can be used by applications outside the cluster to access it. The port numbers that Kubernetes assigns are unpredictable. However, after creating NodePort



Services, the TimesTen Operator can query the Service to see what external port number Kubernetes has assigned.

About the Connection Manager

The TimesTen Operator functions as an https server. It provides http and https servers running on a number of ports, including:

- Metrics (port 8080): Used by Prometheus to scrape metrics about the TimesTen Operator. The /metrics URL endpoint is available for this purpose. It exposes metrics from the TimesTen Operator to Prometheus.
- Probe (port 8081): Used by Kubernetes to determine the readiness of the TimesTen Operator. The /probe URL endpoint is available for this purpose.

This feature adds several new endpoints to the /metrics server. These endpoints allow applications to query information about the location and health of TimesTen databases, which are managed by the TimesTen Operator.

These endpoints allow applications to use an https GET to retrieve a TimesTen connection string that can then be used to access TimesTen databases optimally. Applications can request connection strings that are usable either from inside the cluster or from outside the cluster.

Here is the syntax:

/connstr/v1/namespace/ttc name/access mode/in or out

where:

- connstr/v1: Is a hardcoded, constant string that is part of the API. Do not change this
 parameter.
- namespace: Is the namespace in which the TimesTenClassic object being queried resides.
- ttc name: Is the name of the TimesTenClassic object being queried.
- access_mode: Describes the type of access the application wants in order to get to the TimesTen database. Valid values are as follows:
 - readable: Indicates that the application wants to issue reads against the database.
 - writable: Indicates that the application wants to make changes to the database.
 - readonly: Indicates that the application wants to access a read-only, non-writable instance of the database.

The difference between readable and readonly is that databases that are writable are also readable, but they are not readonly.

- in or out: Is either:
 - internal: Indicates that the application wants to receive a connection string that is viable from inside the cluster.
 - external: Indicates that the application wants to receive a connection string that is viable from outside the cluster.

The response is a JSON object that includes a TimesTen client/server connection string containing TTC_SERVER and TTC_SERVER_DSN connection attributes that can be used to access the desired database in the desired mode. For more information about these connection



attributes, see Client and Server Connection Attributes in the Oracle TimesTen In-Memory Database Reference.

The TimesTen Operator updates this information and stores it in the TimesTenClassic object's status field. When an application queries this data, the TimesTen Operator GETs the TimesTenClassic object from Kubernetes and retrieves the information from the object's status. The TimesTen Operator then returns the information to the querying application.

How to Use the Connection Manager

Let's look at an example to illustrate how the Connection Manager works.

Consider the following scenario:

- The cluster consists of the following nodes:
 - node1: External IP address 1.2.3.4
 - node2: External IP address 1.2.3.5
 - node3: External IP address 1.2.3.6
- An active/standby TimesTenClassic object called sample in the payroll namespace exists. The following Pods also exist for this object:
 - sample-0:
 - * Running on node1
 - * Contains the active database
 - * Internal IP address: 9.0.0.1
 - * Associated NodePort Service sample-np-0: Is assigned external port number 32444
 - sample-1:
 - * Running on node3
 - * Contains the standby database
 - * Internal IP address: 9.0.0.77
 - * Associated NodePort Service sample-np-1: Is assigned external port number 32445

If you issue a GET to /connstr/v1/payroll/sample/readable/external, you receive the following connection string:

TTC_SERVER1=1.2.3.4/32444;TTC_SERVER2=1.2.3.6/32445;TTC_SERVER_DSN=sample

If you issue a GET to /connstr/1/payroll/sample/writable/external, you receive the following connection string:

TTC_SERVER1=1.2.3.4/32444;TTC_SERVER_DSN=sample

If you issue a GET to /connstr/v1/payroll/sample/readable/internal, you receive the following connection string:

TTC_SERVER1=9.0.0.1/6625;TTC_SERVER2=9.0.0.77/6625;TTC_SERVER_DSN=sample



If you issues a GET to /connstr/v1/payroll/sample/writable/internal, you receive the following connection string:

```
TTC SERVER1=9.0.0.77/6625;TTC SERVER DSN=sample
```

About Accessing the Endpoint from Inside the Cluster

Applications running inside the cluster can access the TimesTen Operator at timestenoperator.namespace.svc.cluster.local:8080 where namespace is the name of your namespace.

The metrics endpoint by default uses https, secured with a self-signed certificate created by the TimesTen Operator. Applications running inside the cluster can get the appropriate certificate to allow them to access the endpoint by mounting the Kubernetes Secret called timesten-operator-metrics-client. The TimesTen Operator automatically creates this Secret.

The Secret contains the following entries:

- ca.crt : Adds the TimesTen Operator's certificate to the list of certificates that the application accepts (or trusts).
- client.crt: Contains a client certificate that the TimesTen Operator trusts.
- client.key: Contains the private key for the client certificate.

To access the Connection Manager API, applications must authenticate to the TimesTen Operator using this information.

About Accessing the Endpoint from Outside the Cluster

When the TimesTen Operator starts up, it creates a NodePort Service that maps to the metrics endpoint. This allows applications outside the cluster to access this information. The TimesTen Operator attempts to create this NodePort Service on an external port number that you provide. If successful, applications outside the cluster can then issue GETs to this port number on any node in the cluster in order to retrieve the TimesTen connection strings. It is up to you to make the IP addresses of cluster nodes and the external port number available to applications.

The metrics endpoint by default uses https, secured with a self-signed certificate created by the TimesTen Operator. Applications running inside the cluster can get the appropriate certificate to allow them to access the endpoint by mounting the Kubernetes Secret called timesten-operator-metrics-client. The TimesTen Operator automatically creates this Secret.

However, applications outside the cluster cannot directly access this Secret or other Secrets within the cluster. You must manually extract the contents of the timesten-operator-metricsclient Secret and make it available to applications outside the cluster that want to use the Connection Manager API.

The Secret contains the following entries:

- ca.crt : Adds the TimesTen Operator's certificate to the list of certificates that the application accepts (or trusts).
- client.crt: Contains a client certificate that the TimesTen Operator trusts.
- client.key: Contains the private key for the client certificate.



To access the Connection Manager API, applications must authenticate to the TimesTen Operator using this information.

About Accessing TimesTen

Once applications receive a connection string from the Connection Manager, they may then add additional information to the connection string before presenting it to TimesTen. For example, the UID and PWD connection attributes can be added to authenticate the database as well as additional attributes can be added, such as character set, lock timeouts, and so on.

About Handling Failures

Once connected to a database, the application can use TimesTen normally. At some point, the application's connection to TimesTen may fail. In such circumstances, applications should not attempt to reconnect to the database using the connection string that applications were previously using. Rather, applications should request a new connection string from the Connection Manager and should use it instead.

About the NodePort Service for the Connection Manager

The TimesTen Operator creates a NodePort Service that enables access to the Connection Manager API from outside the cluster. The Service routes incoming requests to the current leader Operator instance. The service is called timesten-operator-np.

By default the NodePort Service is configured to use port number 32625. You can override this port number by using the TT_CONNECTION_MANAGER_NODEPORT environment variable in the operator.yaml or cluster_operator.yaml YAML files or by using the operatorNodePort value in the ttoperator or ttclusteroperator Helm charts.

Note:

Kubernetes allows NodePort Services to use only port numbers between 30000 and 32767.

About the TimesTen Operator Configuration

Here is the TimesTen Operator configuration for using the Connection Manager to optimize client/server performance:

• TT_CONNECTION_MANAGER: If defined and set to 1, this environment variable enables the Connection Manager. By default, the value is set to 1 in the operator.yaml and cluster operator.yaml YAML manifest files.

connectionManager: If defined and set to true in the ttoperator or ttclusteroperator Helm charts, the Connection Manager is enabled. The default is true and is set in the ttoperator and ttclusteroperator Helm charts.

If you disable metrics for the TimesTen Operator, or you are not using https for metrics, the Connection Manager is not enabled. For example, if you set METRICS_SCHEME to http or EXPOSE_METRICS to 0 (or the equivalent in the Helm charts), then TT_CONNECTION_MANAGER (and connectionManager) is ignored and treated as 0 (or false).

- TT_CONNECTION_MANAGER_NODEPORT environment variable (in operator.yaml and cluster_operator.yaml) and the operatorNodePort value (in the ttoperator and ttclusteroperator Helm charts): Use to change the port number on which the Connection Manager API is available for applications outside the cluster. The default is 32625.
- TT_OPERATOR_SAN environment variable (in operator.yaml and cluster_operator.yaml) and the operatorSAN value (in the ttoperator and ttclusteroperator Helm charts): Use to add subject alternate names (SANs) to the TLS certificate that the TimesTen Operator creates to control access to the TimesTen Operator metrics and to the Connection Manager API.
 - If you use metrics and the Connection Manager API from inside the cluster only, you
 do not need to define this environment variable in the YAML manifest files (or define
 the value in the Helm charts).
 - If you use either metrics or the Connection Manager API or both from outside the cluster, you must define a SAN for the nodes in the cluster. By default, no SAN is specified.

For more information, see TimesTen Kubernetes Operator Environment Variables and Helm Charts for the TimesTen Kubernetes Operator.

Normal http error codes indicate if the request is invalid.

If a valid request is received, a JSON object is returned in the following format:

{ "status": 0 [, "error": "the error"] [, "connstr": "TTC SERVER1=..."]}

where:

- status: Is a numeric indication as to whether the request was successful or not. A value of 0 indicates success.
- error: If status is not equal to 0, this is a string that explains the error.
- connstr: If status is equal to 0, this is the connection string that is returned.



Expose TimesTen Metrics with the TimesTen Kubernetes Operator

The TimesTen Kubernetes Operator (TimesTen Operator) can expose TimesTen metrics to Prometheus or any other scraping mechanism. This chapter shows you how.

Topics:

- Overview of TimesTen Metrics
- Overview of the TimesTen Kubernetes Operator and the TimesTen Exporter
- About the Prometheus Operator
- About Exposing TimesTen Metrics
- About Using http or https for TimesTen Metrics
- About Transport Layer Security (mutual TLS) Certificates for TimesTen Metrics
- About Creating PodMonitor Objects
- About the TimesTen Metrics Service
- About Choosing to Expose TimesTen Metrics
- Expose TimesTen Metrics Automatically

Overview of TimesTen Metrics

There are several TimesTen metrics that can be exposed to Prometheus or another scraping mechanism. These metrics are collected from a variety of sources, including TimesTen system tables and views and TimesTen built-in procedures and utilities. For more information about TimesTen metrics, see The Metrics Supported by the TimesTen Exporter in the Oracle TimesTen In-Memory Database Monitoring and Troubleshooting Guide.

Prometheus is an open source monitoring and alerting toolkit. It collects and stores metrics from monitored targets by scraping the http (or https) metrics endpoint on these targets.

TimesTen exports TimesTen metrics to Prometheus by using the TimesTen ttExporter (TimesTen exporter) utility. The TimesTen exporter presents itself as an http or https server. When the TimesTen exporter receives an http or https request to the /metrics endpoint, it retrieves the TimesTen metrics from each database that it monitors and prepares a plain text http or https response with the metrics.

The TimesTen exporter converts TimesTen metrics into a form supported by Prometheus. This simple integration lets you monitor the health and operation of your TimesTen databases.

The TimesTen exporter is supported in TimesTen Classic and in TimesTen Scaleout. In TimesTen Scaleout, the TimesTen exporter is deployed on each data instance and on the management instance.

For information about Prometheus and the TimesTen exporter, see About Prometheus and About the TimesTen Exporter in the Oracle TimesTen In-Memory Database Monitoring and Troubleshooting Guide.



Overview of the TimesTen Kubernetes Operator and the TimesTen Exporter

The TimesTen Operator supports the TimesTen exporter and lets you configure TimesTenClassic and TimesTenScaleout objects to use this TimesTen exporter. Once configured, the TimesTen Operator starts, stops, and manages the TimesTen exporter. As the TimesTen exporter collects metrics from the TimesTen databases running in your Kubernetes environment, the TimesTen Operator works to expose these metrics to Prometheus.

The TimesTen exporter runs in its own exporter container. This container exists in the same TimesTen Pods as the tt and the daemonlog containers. These TimesTen Pods run in your Kubernetes cluster.

The TimesTen Operator creates the Pods that are running TimesTen with the Kubernetes shareProcessNamespace Pod attribute. This attribute allows the TimesTen exporter that is running in the exporter container to access TimesTen that is running in the tt container, both of which are in the TimesTen Pod. For more information about the shareProcessNamespace attribute, see Share Process Namespace between Containers in a Pod in the Kubernetes documentation.

The TimesTen Operator configures the exporter container with the same TimesTen container image as the tt and the daemonlog containers. If the TimesTen exporter fails or exits, Kubernetes destroys the exporter container and creates another one to take its place. Kubernetes monitors and manages the exporter container. Because the lifecycle of individual containers in a Pod are independent, the TimesTen Operator ensures that the ttExporter command starts after TimesTen. The TimesTen Operator waits until the TimesTen agent creates the TimesTen instance and waits until the TimesTen main daemon is running in the tt container of the TimesTen Pod. The Operator then starts the TimesTen exporter in the exporter container of the same TimesTen Pod.

Once configured, the TimesTen exporter functions as an http or https server. It listens for incoming GET requests and responds to them by gathering a set of metrics from TimesTen. It then returns these metrics as the response to the GET request.

To facilitate the listening process, the TimesTen Operator creates a Kubernetes headless Service. This Service exposes the port on which the TimesTen exporter is listening to the remainder of the Kubernetes cluster. This lets a Prometheus server running in the cluster to fetch the TimesTen metrics from TimesTen and process them.

About the Prometheus Operator

In Overview of the TimesTen Kubernetes Operator and the TimesTen Exporter, we discussed how the TimesTen Operator and the TimesTen exporter work together to collect and export TimesTen metrics. Now let's look at how we can expose these metrics to Prometheus.

Note:

You can expose TimesTen metrics to any scraping target. Our documentation focuses on Prometheus.



There are several ways to configure Prometheus in Kubernetes. One of the most popular is the Prometheus Operator. The Prometheus Operator simplifies the deployment, monitoring, and management of Prometheus in Kubernetes.

Similar to the TimesTen Kubernetes Operator, the Prometheus Operator add several custom resource definitions (CRDs) to Kubernetes. Just as you deploy TimesTen by creating an object of type TimesTenClassic or type TimesTenScaleout, deploying Prometheus in Kubernetes involves creating an object of type *Prometheus*. The Prometheus Operator automatically detects the creation of such an object and responds by starting a server using the configuration included in the object.

Note:

It is beyond the scope of this book to detail how to create an object of type Prometheus.

The Prometheus Operator also simplifies the configuration of Prometheus servers that it deploys. The Prometheus Operator can automatically edit the Prometheus server's configuration files to include data sources that should be scraped from the /metrics endpoint. This is done through the creation of Kubernetes objects of type *ServiceMonitor* and of type *PodMonitor*. (The ServiceMonitor object type is discussed in About Creating ServiceMonitor Objects.)

Let's look at a basic example of a PodMonitor object:

```
apiVersion: monitoring.coreos.com/v1
kind: PodMonitor
metadata:
   name: sample
spec:
   selector:
    matchLabels:
        app: sample
podMetricsEndpoints:
        port: metrics
```

If you create such a PodMonitor object, the Prometheus Operator automatically responds by editing the Prometheus server's configuration files to cause the Prometheus server to scrape metrics from any Pod created with a label of app: sample. All Pods with the specified label automatically appear as scrape targets in Kubernetes. As Pods matching the selector are created or deleted, the Prometheus configuration is automatically kept in synchronization.

Using a Prometheus Operator offers a simplified approach to not only deploying, managing, and monitoring Prometheus, but also configuring Prometheus servers.

About Exposing TimesTen Metrics

The TimesTen Operator supports the TimesTen exporter, providing the ability for TimesTen databases deployed in your Kubernetes cluster to have TimesTen metrics exported. In addition, the Prometheus Operator configures and stands up Prometheus servers and creates and manages configuration files associated with these Prometheus servers. However, there are additional tasks that Prometheus requires to scrape TimesTen metrics.

For example:



- If you are using https to serve TimesTen metrics, an Oracle Wallet and certificates must be created.
- The certificates must be made available to Prometheus.
- Prometheus configuration files must be edited to cause Prometheus to scrape metrics from TimesTen. This is true whether TimesTen metrics are served using https or http.

Let's look at these tasks, examine our options for completing these tasks, and discuss how the TimesTen Operator can automatically perform these tasks for you.

Startung with TimesTen release 22.1, the TimesTen Operator automatically exports and exposes TimesTen metrics. In addition, if the Prometheus Operator is installed in your Kubernetes cluster, the TimesTen Operator provides Prometheus with all the information it needs to scrape TimesTen metrics from TimesTen databases.

Let's look at how the TimesTen Operator does this.

By default, the TimesTen Operator does the following:

- Exports and exposes TimesTen metrics from TimesTen databases deployed in your Kubernetes cluster.
- Uses https/Transport Layer Security (mutual TLS) to make these metrics available.
- If the Prometheus Operator is installed, creates the necessary objects to cause Prometheus to be automatically modified, which then causes Prometheus to scrape TimesTen metrics from the TimesTen databases.
- Exposes TimesTen metrics outside of the TimesTen Pods.

Although this default behavior is recommended, you have the option of changing the default behavior. Let's examine our options:

- https or http?: TimesTen metrics are available by https (default) or http.
- Create a Kubernetes PodMonitor object?: If the Prometheus Operator is installed in your Kubernetes cluster, the TimesTen Operator can create a PodMonitor object. This object contains the information needed by the Prometheus Operator to configure TimesTen databases as a scrape target. If the Prometheus Operator has been installed in your Kubernetes cluster, then by default the TimesTen Operator attempts to create a PodMonitor object.
- Expose TimesTen metrics outside of the TimesTen Pods?: The TimesTen exporter listens on and accepts GETs on the *sampleobject.mynamespace.svc.cluster.local:8080/* metrics endpoint (where *sampleobject* is the name of a TimesTenClassic or TimesTenScaleout object and *mynamespace* is the name of your namespace). By default, the TimesTen Operator deploys a Kubernetes Service that makes this endpoint available to other Pods in the Kubernetes cluster.

Note:

Having the TimesTen Operator automatically expose TimesTen metrics is not available in TimesTen release 18.1. This functionality is available in TimesTen release 22.1 and above.

Now let's look at how we can change this default behavior.

A TimesTenClassic or TimesTenScaleout object that is deployed in your namespace has specific datum associated with the object. These datum define the characteristics of a

TimesTenClassic or TimesTenScaleout object, including the TimesTen databases associated with the object. You have the option of customizing a TimesTenClassic or TimesTenScaleout object by specifying values for particular datum in the object's YAML manifest file.

For exposing TimesTen metrics, the TimesTen Operator provides the .spec.ttspec.prometheus datum/clause and provides specific datum for use within this clause. This clause and its associated datum determine if and how the TimesTen Operator exports and exposes TimesTen metrics.

Let's look at the specific datum in the .spec.ttspec.prometheus clause in greater detail:

- .spec.ttspec.prometheus.publish: Determines if the TimesTen Operator provisions an exporter container for the TimesTen exporter. The default is true. We discuss this in About Choosing to Expose TimesTen Metrics.
- .spec.ttspec.prometheus.insecure: Determines if TimesTen metrics are served using https or http. If the value is false or not specified, TimesTen metrics are served using https. If the value is true, TimesTen metrics are served using http. For more information, see About Using http or https for TimesTen Metrics.
- .spec.ttspec.prometheus.certSecret: If specified, contains a Kubernetes Secret that you have previously created. This Secret contains an Oracle Wallet and the necessary certificates for https. If not specified, the TimesTen Operator automatically creates an Oracle Wallet, the necessary certificates, and the Kubernetes Secrets.

We recommend that you do not specify .spec.ttspec.prometheus.certSecret and instead let the TimesTen Operator automate this process for you. For more information, see About Transport Layer Security (mutual TLS) Certificates for TimesTen Metrics. If you want to specify .spec.ttspec.prometheus.certSecret, see Create Your Own Oracle Wallet, Certificates, and Secrets for Exposing TimesTen Metrics.

• .spec.ttspec.prometheus.createPodMonitors: Determines if the TimesTen Operator should create a PodMonitor object. The default is true. We discuss this in About Creating PodMonitor Objects.

For additional information about .spec.ttspec.prometheus and its associated datum, see TimesTenClassicSpecSpecPrometheus for a TimesTenClassic object and TimesTenScaleoutSpecSpecPrometheus for a TimesTen Scaleout object.

About Using http or https for TimesTen Metrics

TimesTen metrics are available by http or https.

The default behavior is https and is discussed in About Transport Layer Security (mutual TLS) Certificates for TimesTen Metrics.

To cause the TimesTen Operator to use http for TimesTen metrics, specify the .spec.ttspec.prometheus.insecure datum in a TimesTenClassic or TimesTenScaleout object YAML manifest file.

Here is a code snippet of a TimesTenClassic object YAML manifest file showing you how to do this:

```
apiVersion: timesten.oracle.com/v4
kind: TimesTenClassic
metadata:
   name: usehttp
spec:
   ttspec:
```



```
prometheus:
insecure: true
port: 7777
```

Note the following:

• The .spec.ttspec.insecure datum is specified in the .spec.ttspec.prometheus clause of the TimesTenClassic object. This causes the TimesTen Operator to configure the TimesTen exporter to serve TimesTen metrics using http.

The .spec.ttspec.port datum is specified. This is the port on which the TimesTen exporter listens. The causes the TimesTen Operator to set up the http server on TCP port 7777 in each TimesTen Pod.

About Transport Layer Security (mutual TLS) Certificates for TimesTen Metrics

When https is used, the TimesTen Operator automatically creates self-signed certificates. The TimesTen Operator also creates two Kubernetes Secrets to hold these certificates.

As an example, for a TimesTen object called sample, these Secrets are automatically created:

- sample-metrics: This Secret is automatically mounted in the TimesTen exporter containers of the TimesTen Pods. It contains an Oracle Wallet, which contains all certificates needed by the TimesTen exporter for https.
- sample-metrics-client: This Secret contains files that a Prometheus server (or other scraper) requires to scrape TimesTen metrics. This Secret contains the following three files:
 - ca.crt: The Certificate Authority certificate needed by the client to authenticate the self-signed certificate used by the TimesTen exporter.
 - client.crt: The client certificate that the TimesTen exporter uses to authenticate any clients that try to scrape metrics from it.
 - client.key: The private key that is associated with the client.crt client certificate.

Both Secrets are created with appropriate Kubernetes owner references. If you delete the associated TimesTenClassic or TimesTenScaleout object, these Secrets are automatically deleted.

Although not recommended, you have the option of creating your own certificates to serve TimesTen metrics using https. See Create Your Own Oracle Wallet, Certificates, and Secrets for Exposing TimesTen Metrics.

Note:

If TimesTen metrics are served by using either http or https whereby you create your own self-signed certificates, then the TimesTen Operator does not automatically create certificates, Oracle Wallets, or Kubernetes Secrets.



About Creating PodMonitor Objects

By default, if the Prometheus Operator has been installed in your Kubernetes cluster, the TimesTen Operator automatically creates PodMonitor objects when it provisions TimesTen in your namespace. Specifically, the TimesTen Operator automatically creates a PodMonitor object for each deployed TimesTenClassic or TimesTenScaleout object.

This automates all of the steps required to get TimesTen metrics into Prometheus. As TimesTen objects are created and deleted, the appropriate scrape targets are automatically provisioned and removed from the Prometheus configuration without any intervention by you.

Although recommended, this is an optional capability. If you choose not to use the Prometheus Operator, you can export TimesTen metrics into Prometheus yourself. The Prometheus Operator is not a prerequisite for using the TimesTen Operator.

If you have installed the Prometheus Operator and you do not want the TimesTen Operator to automatically create PodMonitor objects, set

the .spec.ttspec.prometheus.createPodMonitors datum of your TimesTenClassic or TimesTenScaleout object to false.

This capability is available whether a TimesTenClassic or TimesTenScaleout object is using http or https. It is also available with TimesTen Operator-created certificates and Secrets.

However, this capability is not available if you specify

the .spec.ttspec.prometheus.certSecret datum in a TimesTenClassic or TimesTenScaleout object definition. If you provide your own Kubernetes Secret that contains your own Oracle Wallet (and this Oracle Wallet contains the necessary certificates for using https), the TimesTen Operator cannot automatically produce a PodMonitor object with sufficient information to allow Prometheus to access TimesTen. In this case, you must create your own PodMonitor objects, or otherwise edit Prometheus configuration files to access TimesTen and have TimesTen metrics scraped.

If the TimesTen Operator attempts to create a PodMonitor object and is unable to do so, the TimesTen Operator generates an appropriate Event and continues on with provisioning and other normal operations. This can happen if the Prometheus Operator is not installed in your Kubernetes cluster or if the TimesTen Operator has not been granted appropriate role-based access control (RBAC) privileges in your Kubernetes cluster. For more information about RBAC privileges, see https://kubernetes.io/docs/reference/access-authn-authz/rbac/.

A PodMonitor object created by the TimesTen Operator has an owner reference to the appropriate TimesTenClassic or TimesTenScaleout object. This ensures that the PodMonitor object is automatically deleted if the TimesTenClassic or TimesTenScaleout object is deleted.

Note:

The .spec.ttspec.prometheus.createPodMonitors datum can be added to new TimesTenClassic and TimesTenScaleout objects, but cannot be added to existing TimesTenClassic and TimesTenScaleout objects (by using the kubectl upgrade or the kubectl patch commands or by other means).

For more information about PodMonitor objects, see https://prometheus-operator.dev/docs/getting-started/design/#podmonitor.



About the TimesTen Metrics Service

The TimesTen Operator automatically creates a Kubernetes Service for the TimesTen exporter. This Service exposes the port on which the TimesTen exporter listens to the Pods in the Kubernetes cluster. This Service exposes the port at

sample.mynamespace.svc.cluster.local:port/metrics where:

- sample is the name of a TimesTenClassic or TimesTenScaleout object.
- mynamespace is the name of your namespace.
- port is the port number on which the TimesTen exporter listens.

The default port is 8888.

You can change the port on which the TimesTen exporter listens by specifying the .spec.ttspec.prometheus.port datum in a TimesTenClassic or TimesTenScaleout object YAML manifest file.

Here is a code snippet of a TimesTenClassic object YAML manifest file showing the .spec.ttspec.prometheus.port datum.

```
apiVersion: timesten.oracle.com/v4
kind: TimesTenClassic
metadata:
   name: sample
spec:
   ttspec:
...
prometheus:
   port: 7777
```

In this snippet, the TimesTen exporter listens on port 7777.

About Choosing to Expose TimesTen Metrics

By default, the TimesTen Operator exposes TimesTen metrics for TimesTenClassic and TimesTenScaleout objects.

You have the option of not exposing TimesTen metrics by setting

the .spec.ttspec.prometheus.publish datum to false in a TimesTenClassic or TimesTenScaleout object YAML manifest file. Doing so causes the TimesTen Operator to not provision an exporter container for the TimesTen exporter. If an exporter container is not provisioned, the TimesTen exporter is not configured, started, or managed. In this case, TimesTen metrics for the databases associated with this TimesTenClassic or TimesTenScaleout object are not exported or exposed.

Here is a code snippet showing how to set the value of the .spec.ttspec.prometheus.publish datum to false in a TimesTenClassic object YAML manifest file:

```
apiVersion: timesten.oracle.com/v4
kind: TimesTenClassic
metadata:
   name: sample
```



```
spec:
   ttspec:
...
prometheus:
   publish: false
```

In this example, the TimesTen Operator does not provision an exporter container for the sample TimesTenClassic object.

The default value for the .spec.ttspec.prometheus.publish datum is true. If you want the TimesTen Operator to provision an exporter container, you can choose the default or you can specify a value of true for the .spec.ttspec.prometheus.publish datum.

If you do not specify the publish datum, and any other datum is specified in the .spec.ttspec.prometheus clause, the default value for the publish datum is true.

If you do not specify the .spec.ttspec.prometheus clause and the TimesTen release is 22.1 or greater, the default value for the publish datum is dependent on the value of the EXPOSE METRICS TimesTen Operator environment variable:

- If EXPOSE_METRICS is "1" (or not specified), the TimesTen Operator treats the publish datum as true.
- If EXPOSE_METRICS is "0", the TimesTen Operator treats the publish datum as false.

For more information about the .spec.ttspec.prometheus clause of a TimesTenClassic or TimesTenScaleout object, see TimesTenClassicSpecSpecPrometheus or TimesTenScaleoutSpecSpecPrometheus respectively.

For information about TimesTen Operator environment variables, see TimesTen Kubernetes Operator Environment Variables.

We recommend that you let the TimesTen Operator automatically export, expose, and publish TimesTen metrics. For a complete example, see Expose TimesTen Metrics Automatically.

Expose TimesTen Metrics Automatically

Let's walk through an example showing how the TimesTen Operator automatically exports and exposes TimesTen metrics. In this example, let's create a TimesTenClassic object and observe how the TimesTen Operator automatically creates the objects needed to automatically export and expose TimesTen metrics to Prometheus.

Let's assume you have installed the Prometheus Operator in your Kubernetes cluster and there is a Prometheus server running in your namespace. Let's also assume you have started the TimesTen Operator following the steps in About the TimesTen Operator and there are no TimesTenClassic or TimesTenScaleout objects deployed in your namespace.

1. Create a TimesTenClassic object.

```
vi samplepublish.yaml
apiVersion: timesten.oracle.com/v4
kind: TimesTenClassic
metadata:
   name: samplepublish
spec:
   ttspec:
```



```
storageClassName: oci
storageSize: 250G
image: container-registry.oracle.com/timesten/timesten:22.1.1.34.0
imagePullSecret: sekret
prometheus:
    port: 6666
```

In this example, the port on which the TimesTen exporter listens is 6666.

2. Deploy the TimesTenClassic object.

kubectl create -f samplepublish.yaml

The output is the following:

timestenclassic.timesten.oracle.com/samplepublish created

3. Wait a few minutes, then confirm the TimesTenClassic object is in the Normal state.

kubectl get ttc samplepublish

The output is similar to the following:

NAME	STATE	ACTIVE	AGE
samplepublish	Normal	samplepublish-0	3m56s

4. Confirm the TimesTen Operator provisioned an exporter container in each TimesTen Pod

kubectl get pods

The output is similar to the following:

NAME	READY	STATUS	RESTARTS	AGE
 samplepublish-0 samplepublish-1 	3/3 3/3	Running Running	0 0	7m33s 7m33s

The TimesTen Operator automatically provisioned 3 containers in each TimesTen Pod. One of these containers is the exporter container, which is running the TimesTen exporter.

5. Confirm the TimesTen Operator automatically created the appropriate Kubernetes Secrets.

kubectl get secrets

The output is similar to the following:

NAME		
TYPE	DATA	AGE
•••		
samplepublish-metrics		
Opaque	1	15m



samplepublish-metrics-client
Opaque 3 15m

By default, TimesTen metrics are served using https. As a result, the TimesTen Operator automatically created an Oracle Wallet, the certificates, and the Kubernetes Secrets needed for https. For more information, see About Transport Layer Security (mutual TLS) Certificates for TimesTen Metrics.

6. Confirm the appropriate files are in the samplepublish-metrics-client Kubernetes Secret.

kubectl describe secret samplepublish-metrics-client

The output is similar to the following:

Name: samplepublish-metrics-client Namespace: mynamespace Labels: <none> Annotations: <none> Type: Opaque Data ==== ca.crt: 1461 bytes client.crt: 1273 bytes client.key: 1675 bytes

The TimesTen Operator automatically created the ca.crt, client.crt, and client.key files. The samplepublish-metrics-client Kubernetes Secret holds these files. See About Transport Layer Security (mutual TLS) Certificates for TimesTen Metrics.

 Confirm the TimesTen Operator automatically created the samplepublish PodMonitor object.

kubectl describe podmonitor samplepublish

The output is similar to the following:

Name:	samplepublis	h
Namespace:	mynamespace	
Labels:	app=samplepu	blish
	database.tim	esten.oracle.com=samplepublish
Annotations:	<none></none>	
API Version:	monitoring.c	oreos.com/v1
Kind:	PodMonitor	
Metadata:		
Creation Ti	mestamp: 202	5-01-25T15:36:08Z
Generation:	1	
Owner Refer	ences:	
API Versi	.on:	timesten.oracle.com/v4
Block Own	er Deletion:	true
Controlle	er:	true
Kind:		TimesTenClassic



```
samplepublish
    Name:
    UID:
                           d0f46bf7-b1d8-4499-876c-51410a469772
                           284346942
  Resource Version:
                           5a61ec9e-df7d-4a98-be47-dce1e7c3d217
 UID:
Spec:
  Namespace Selector:
  Pod Metrics Endpoints:
    Bearer Token Secret:
      Key:
    Interval: 15s
    Path:
              /metrics
    Port:
              exporter
              https
    Scheme:
    Tls Config:
      Ca:
        Secret:
          Key: ca.crt
          Name: samplepublish-metrics-client
      Cert:
        Secret:
          Key: client.crt
         Name: samplepublish-metrics-client
      Key Secret:
                    client.key
        Key:
                    samplepublish-metrics-client
        Name:
      Server Name:
samplepublish.samplepublish.mynamespace.svc.cluster.local
  Selector:
    Match Labels:
      database.timesten.oracle.com: samplepublish
Events:
                                     <none>
```

Let's look at the important information in this PodMonitor object:

- There is an app=samplepublish label. If there are Pods with a label that matches app=samplepublish, Prometheus scrapes metrics from them. The TimesTen Pods contain the app=samplepublish label. Prometheus will therefore scrape metrics from these Pods. We will see this later.
- Prometheus scrapes metrics from the /metrics endpoint.
- Metrics are exposed using https.
- The TimesTen Operator placed the samplepublish-metrics and samplepublishmetrics-client Kubernetes Secrets in the PodMonitor object. These Secrets and their contents are used by the Prometheus Operator.

The Prometheus Operator edits the Prometheus server configuration files based on the information in this PodMonitor object.

8. Confirm the TimesTen Operator automatically created the appropriate Kubernetes Service.

```
kubectl describe service samplepublish
```



The output is similar to the following:

```
Name:
                       samplepublish
                    mynamespace
Namespace:
Labels:
                      app=samplepublish
Annotations:
                    <none>
Selector:
                      app=samplepublish
                       ClusterIP
Type:
IP Family Policy: SingleStack
IP Families:
                      IPv4
IP:
                      None
IPs:
                      None
                     cs 6625/TCP
Port:

      TargetPort:
      6625/TCP

      Endpoints:
      10.244.0.120:6625,10.244.1.144:6625

      Port:
      exporter 6666/TCP

TargetPort: 6666/TCP
Endpoints: 10.244.0.120:6666,10.244.1.144:6666
Session Affinity: None
Events:
                       <none>
```

The TimesTen exporter listens on port 6666.

9. Confirm there is a Prometheus server running in your namespace.

kubectl get pods

The output is similar to the following:

NAME	READY	STATUS	RESTARTS	AGE
prometheus-sampleprometheusserver-0	2/2	Running	2	18d

The prometheus-sampleprometheusserver-0 Prometheus server is running in your namespace.

10. Confirm the Prometheus Operator edited the Prometheus server configuration file based on the information in the samplepublish PodMonitor object.

```
kubectl exec prometheus-sampleprometheusserver-0 -c prometheus --
cat /etc/prometheus/config out/prometheus.env.yaml
```

The output is similar to the following:

```
global:
  evaluation_interval: 30s
  scrape_interval: 30s
  external_labels:
    prometheus: mynamespace/sampleprometheusserver
    prometheus_replica: prometheus-sampleprometheusserver-0
scrape_configs:
    job_name: podMonitor/mynamespace/samplepublish/0
    honor_labels: false
    kubernetes_sd_configs:
```

```
- role: pod
   namespaces:
     names:
      - mynamespace
  scrape interval: 15s
  metrics path: /metrics
  scheme: https
  tls config:
    insecure skip verify: false
    ca file: /etc/prometheus/certs/secret mynamespace samplepublish-
metrics-client ca.crt
    cert file: /etc/prometheus/certs/secret mynamespace samplepublish-
metrics-client_client.crt
    key file: /etc/prometheus/certs/secret mynamespace samplepublish-
metrics-client client.key
    server name: samplepublish.samplepublish.mynamespace.svc.cluster.local
  relabel configs:
  - source labels:
    - job
    target_label: __tmp_prometheus_job_name
  - action: drop
    source labels:
    - meta kubernetes pod phase
    regex: (Failed|Succeeded)
  - action: keep
    source labels:
    - __meta_kubernetes_pod_label_database_timesten oracle com
    - meta kubernetes pod labelpresent database timesten oracle com
   regex: (samplepublish);true
  - action: keep
    source labels:
    - meta_kubernetes_pod_container_port_name
    regex: exporter
  - source labels:
    - __meta_kubernetes_namespace
    target label: namespace
  - source labels:
    - meta kubernetes pod container name
    target label: container
  - source labels:
    - meta kubernetes pod name
    target_label: pod
  - target label: job
    replacement: mynamespace/samplepublish
  - target label: endpoint
    replacement: exporter
  - source labels:
    - address
    target label: tmp hash
    modulus: 1
    action: hashmod
  - source labels:
    - tmp hash
    regex: 0
    action: keep
  metric relabel configs: []
```

Prometheus has the information it needs to scrape TimesTen metrics.

- **11.** Review some of the TimesTen metrics: In your browser, go to your Prometheus server.
 - a. In the Prometheus server search bar, type a TimesTen metric. For example, timesten_databases. Next, click Execute.

The output is similar to the following:

```
timesten_databases{container="exporter", endpoint="exporter",
instance="10.244.0.120:6666", instancename="instance1",
job="mynamespace/samplepublish",
namespace="mynamespace", pod="samplepublish-0"} 1
```

```
timesten_databases{container="exporter", endpoint="exporter",
instance="10.244.1.144:6666", instancename="instance1",
job="mynamespace/samplepublish",
namespace="mynamespace", pod="samplepublish-1"} 1
```

There is one TimesTen database in the samplepublish-0 Pod and one TimesTen database in the samplepublish-1 Pod as evidenced by the value of **1** for the metric.

b. In the Prometheus server search bar, type a second TimesTen metric. For example, timesten_database_loaded. Next, click Execute.

The output is similar to the following:

```
timesten_database_loaded{container="exporter", dsn="samplepublish",
endpoint="exporter", instance="10.244.0.120:6666",
instancename="instance1",
job="mynamespace/samplepublish", namespace="mynamespace",
pod="samplepublish-0"} 1
timesten_database_loaded{container="exporter", dsn="samplepublish",
endpoint="exporter", instance="10.244.1.144:6666",
instancename="instance1",
job="mynamespace/samplepublish", namespace="mynamespace",
pod="samplepublish-1"} 1
```

The samplepublish database in the samplepublish-0 Pod and the samplepublish database in the samplepublish-1 Pod are both loaded into memory as evidenced by the value of **1** for the metric.

Congratulations! You successfully created TimesTen databases whose metrics are collected by Prometheus.



15

Expose Metrics from the TimesTen Kubernetes Operator

The TimesTen Kubernetes Operator (TimesTen Operator) can expose metrics about its own functionality as well as the status of TimesTenClassic and TimesTenScaleout objects to Prometheus or any other scraping mechanism.

This chapter shows you how.

Topics:

- About Exposing Metrics from the TimesTen Kubernetes Operator
- About Using http or https
- About Transport Layer Security (mutual TLS) Certificates
- About Creating ServiceMonitor Objects
- About the TimesTen Kubernetes Operator's Metrics Service
- About TimesTen Operator Metrics
- Demonstrate How to Expose TimesTen Kubernetes Operator Metrics

About Exposing Metrics from the TimesTen Kubernetes Operator

By default, the TimesTen Kubernetes Operator exposes metrics about its own functionality as well as the status of TimesTenClassic and TimesTenScaleout objects it manages.

Let's look at how the TimesTen Operator does this. Let's examine the default behavior and let's determine how you can change the default behavior if you choose to do so.

By default, the TimesTen Operator does the following:

- Exports and exposes metrics about its own functionality and the overall status of the TimesTenClassic and TimesTenScaleout objects that it manages.
- Uses https/Transport Layer Security (mutual TLS) to make these metrics available.
- Automatically creates certificates, Oracle Wallets, and Kubernetes Secrets required for TLS.
- If the Prometheus Operator is installed, creates the appropriate objects to cause Prometheus to be automatically modified, which then causes Prometheus to scrape metrics from the TimesTen Operator.
- Exposes metrics outside of the TimesTen Operator Pods.

Although this default behavior is recommended, you have the option of changing the default behavior. Let's examine our options:

- https or http?: TimesTen Operator metrics are available by https (default) or http.
- Create a Kubernetes ServiceMonitor object?: If the Prometheus Operator is installed in your Kubernetes cluster, the TimesTen Operator can create a Kubernetes ServiceMonitor object. This object contains the information needed by the Prometheus Operator to



configure the TimesTen Operator as a scrape target. If the Prometheus Operator has been installed in your Kubernetes cluster, then by default the TimesTen Operator creates a ServiceMonitor object.

• Expose TimesTen Operator metrics outside of the TimesTen Operator Pods?: The TimesTen Operator listens on and accepts GETs on the timestenoperator.mynamespace.svc.cluster.local:8080/metrics endpoint (where mynamespace is the name of your namespace). By default, the TimesTen Operator deploys a Kubernetes Service that makes this endpoint available to other Pods in the Kubernetes cluster. You can choose not to create this Service.

Now, let's look at how we can change the default behavior.

The TimesTen Operator is deployed as a Kubernetes Deployment. The provided operator.yaml and cluster_operator.yaml YAML manifest file that lets you customize this Deployment. These files are located in the deploy directory of the TimesTen Operator distribution. For more information, see About the TimesTen Operator.

The operator.yaml file contains various environment variables. Three of these variables govern if and how the TimesTen Operator exports and exposes metrics about its own operation and the status of the TimesTenClassic and TimesTenScaleout objects that it manages.

- METRICS_SCHEME: Determines if metrics should be made available by https or http. A setting of "https" (default) indicates https should be used. A setting of "http" indicates http should be used.
- CREATE_SERVICEMONITOR: Determines if the TimesTen Operator should create a Kubernetes ServiceMonitor object (discussed in a later section). A setting of "1" (default) indicates the TimesTen Operator should create a ServiceMonitor object. A setting of "0" indicates the TimesTen Operator should not create a ServiceMonitor object.
- EXPOSE_METRICS: Determines if metrics are exposed outside of the TimesTen Operator Pods. A setting of "1" (default) indicates metrics should be exposed outside of the TimesTen Operator Pods. A setting of "0" indicates metrics should not be exposed outside of the TimesTen Operator Pods.

Let's look at a snippet of an <code>operator.yaml</code> file that shows the existence of these environment variables and the default settings.

```
# Copyright (c) 2019 - 2025, Oracle and/or its affiliates.
apiVersion: apps/v1
kind: Deployment
metadata:
  name: timesten-operator
spec:
. . .
          env:
            . . .
            - name: EXPOSE METRICS
              value: "1"
             - name: METRICS SCHEME
               value: "https"
             - name: EXPOSE PROBES
              value: "1"
             - name: CREATE SERVICEMONITOR
               value: "1"
. . .
```



For reference information about these variables, see TimesTen Kubernetes Operator Environment Variables.

Note:

TimesTen Operator metrics are accurate when you have one TimesTen Operator defined in your TimesTen Operator Deployment (replicas:1 in your operator.yaml file).

About Using http or https

TimesTen Operator metrics are available by http or https.

The default behavior is https. To cause the TimesTen Operator to use http instead of https, set the METRICS SCHEME environment variable to "http" in your operator.yaml YAML manifest file.

Note:

In TimesTen release 18.1, if you specify https for the METRICS_SCHEME environment variable, the TimesTen Operator acts as though you specified http.

About Transport Layer Security (mutual TLS) Certificates

When https is used, the TimesTen Operator automatically creates self-signed certificates. The TimesTen Operator also creates two Kubernetes Secrets to hold these certificates:

- timesten-operator-metrics: This Secret is used internally by the TimesTen Operator. It contains all the certificates needed by the TimesTen Kubernetes Operator for https/TLS. You do not need to use or examine this Secret.
- timesten-operator-metrics-client: This Secret contains files that a Prometheus server (or other scraper) requires to scrape metrics from the TimesTen Operator. This Secret contains the following three files:
 - ca.crt: The Certificate Authority certificate needed by the client to authenticate the self-signed certificate used by the TimesTen Operator.
 - client.crt: The client certificate that the TimesTen Operator uses to authenticate any clients that try to scrape metrics from it.
 - client.key: The private key that is associated with the client.crt client certificate.

Both Secrets are created with appropriate Kubernetes owner references. If you delete the TimesTen Operator deployment, these Secrets are automatically deleted.

Note:

When http is used, these certificates, Wallets, and Secrets are not created.

About Creating ServiceMonitor Objects

By default, if the Prometheus Operator is installed in your Kubernetes cluster, the TimesTen Operator attempts to create a Kubernetes ServiceMonitor object called timesten-operator. This object includes the information needed by Prometheus to configure the TimesTen Operator as a scrape target. If installed in your cluster, the Prometheus Operator responds to the creation of this object by editing the Prometheus configuration files to scrape metrics from the corresponding Service. For more information about ServiceMonitor objects, see https://prometheus-operator.dev/docs/getting-started/design/#servicemonitor.

A ServiceMonitor object created by the TimesTen Operator has an owner reference to the appropriate TimesTen Operator deployment. This ensures that the ServiceMonitor object is automatically deleted if the TimesTen Operator deployment is deleted.

The CREATE_SERVICEMONITOR environment variable determines if the TimesTen Operator creates a ServiceMonitor object. If you do not want the TimesTen Operator to create a ServiceMonitor object, then set CREATE_SERVICEMONITOR to "0" in your operator.yaml YAML manifest file..

About the TimesTen Kubernetes Operator's Metrics Service

The TimesTen Operator always makes metrics available. However, you can limit access to the TimesTen Operator's /metrics endpoint. By default, the TimesTen Operator's /metrics endpoint is available to other Pods in the Kubernetes cluster. Specifically, the TimesTen Operator automatically creates a Kubernetes Service called timesten-operator. This Service exposes the metrics port in the active TimesTen Operator Pod to the rest of the cluster at timesten-operator.mynamespace.svc.cluster.local:8080/metrics (where mynamespace is the name of your namespace).

If you choose not to expose TimesTen Operator metrics outside of the TimesTen Operator's Pods, set the EXPOSE_METRICS environment variable to "0" in your operator.yaml YAML manifest file. In this case, the metrics port is not exposed by a Kubernetes Service. (You could still fetch metrics by using the kubectl exec command to run curl or a similar tool within the TimesTen Operator Pod itself.)

If you set the EXPOSE_METRICS environment variable to "0" and the CREATE_SERVICEMONITOR environment variable is set to "1" (default), the CREATE_SERVICEMONITOR environment variable is treated as though it was set to "0".

In addition, if you set the EXPOSE_METRICS environment variable to "0", the value of the METRICS SCHEME environment variable is ignored and http is always used.

About TimesTen Operator Metrics

The TimesTen Operator publishes metrics about its own functionality as well as the status of TimesTenClassic and TimesTenScaleout objects that it manages.

For details about the metrics, see TimesTen Kubernetes Operator Metrics.

Demonstrate How to Expose TimesTen Kubernetes Operator Metrics

Let's walk through an example illustrating how the TimesTen Kubernetes Operator exposes metrics about its own functionality as well as the status of TimesTenClassic or TimesTenScaleout objects. Let's assume Prometheus is installed in your Kubernetes cluster and you have a Prometheus server running in your namespace.

Let's make the following decisions:

- https or http?: Let's choose https. The METRICS_SCHEME environment variable determines if metrics should be made available by https or http. A setting of "1" (default) indicates https.
- Create a ServiceMonitor object?: Let's have the TimesTen Operator create a ServiceMonitor object. This object contains the information needed by Prometheus to configure the TimesTen Operator as a scrape target. The CREATE_SERVICEMONITOR environment variable determines if the TimesTen Operator should create a ServiceMonitor object. A setting of "1" (default) indicates the TimesTen Operator should create the object.
- Expose TimesTen Operator metrics outside of the TimesTen Operator Pods?: Let's have
 the metrics exposed outside of the TimesTen Operator Pods so that Prometheus can
 scrape metrics from it. The EXPOSE_METRICS environment variable determines if metrics are
 exposed outside of the TimesTen Operator Pods. A setting of "1" (default) indicates
 metrics should be exposed. This means that the TimesTen Operator will create a
 Kubernetes Service (called timesten-operator) that allows the /metrics endpoint to be
 available to other Pods in the Kubernetes cluster.

Since we have chosen the defaults for the TimesTen Operator environment variables, you do not have to modify these variables in the <code>operator.yaml YAML</code> manifest file.

- 1. Start the TimesTen Operator following the steps in About the TimesTen Operator.
- 2. (Optional) Review the TimesTen Operator deployment running in your namespace.

kubectl describe deployment timesten-operator

Output.

Name:	timesten-operator		
Namespace:	mynamespace		
CreationTimestamp:	Thu, 16 Jan 2025 20:48:48 +0000		
Labels:	<none></none>		
Annotations:	deployment.kubernetes.io/revision: 1		
Selector:	name=timesten-operator		
Replicas:	1 desired 1 updated 1 total 1 available 0		
unavailable			
StrategyType:	RollingUpdate		
MinReadySeconds:	0		
RollingUpdateStrategy:	25% max unavailable, 25% max surge		
Pod Template:			
Labels: nam	e=timesten-operator		
Service Account: time	esten-operator		
Containers:			
timesten-operator:			
Image: contai	ner-registry.oracle.com/timesten/		



```
timesten:22.1.1.34.0
   Ports: 8081/TCP, 8080/TCP
   Host Ports: 0/TCP, 0/TCP
   Command:
      /timesten/operator/operator/timesten-operator
   Liveness: http-get http://:probes/healthz delay=10s timeout=10s
period=30s #success=1 #failure=3
   Readiness: http-get http://:probes/healthz delay=10s timeout=10s
period=10s #success=1 #failure=1
   Environment:
     WATCH_NAMESPACE: (v1:metadata.namespace)

POD_NAME: (v1:metadata.name)

OPERATOR_NAME: timesten-operator
     EXPOSE METRICS:
                            1
     METRICS SCHEME:
                           https
     EXPOSE PROBES:
                           1
     CREATE SERVICEMONITOR: 1
   Mounts:
                             <none>
                             <none>
 Volumes:
Conditions:
 Туре
              Status Reason
  ____
               _____ ____
 Available
              True MinimumReplicasAvailable
 Progressing True
                        NewReplicaSetAvailable
OldReplicaSets: <none>
NewReplicaSet: timesten-operator-7f77c749fd (1/1 replicas created)
Events:
 Type Reason
                            Age From
                                                        Message
  -----
                            ----
                                                        _____
 Normal ScalingReplicaSet 73s deployment-controller Scaled up
replica set timesten-operator-7f77c749fd to 1
```

The EXPOSE_METRICS and CREATE_SERVICEMONITOR TimesTen Operator environment variables are set to 1 (defaults) and the METRICS SCHEME is set to https (default).

3. Confirm the TimesTen Operator created the appropriate Kubernetes Secrets.

kubectl get secrets

Output:

NAME		
TYPE	DATA	AGE
timesten-operator-metrics		
Opaque	1	11m
timesten-operator-metrics-client		
Opaque	3	11m

The TimesTen Operator created the timesten-operator-metrics Kubernetes Secret. This Secret contains the certificates needed by the TimesTen Operator for https/TLS. This Secret is used internally by the TimesTen Operator.



The TimesTen Operator also created the timesten-operator-metrics-client Kubernetes Secret. This Secret contains the files that a Prometheus server needs to scrape metrics from the TimesTen Operator Pods.

4. Confirm the appropriate files are in the timesten-operator-metrics-client Kubernetes Secret.

kubectl describe secret timesten-operator-metrics-client

Output:

```
Name: timesten-operator-metrics-client
Namespace: mynamespace
Labels: <none>
Annotations: <none>
Type: Opaque
Data
====
ca.crt: 1465 bytes
client.crt: 1277 bytes
client.key: 1675 bytes
```

The Secret contains the following files:

- ca.crt: The Certificate Authority file needed by the client to authenticate the selfsigned certificate used by the TimesTen Operator.
- client.crt: The client certificate that the TimesTen Operator uses to authenticate clients that are attempting to scrape metrics from it.
- client.key: The private key that is associated with the client.crt file.
- 5. Confirm the TimesTen Operator created the timesten-operator ServiceMonitor object.

kubectl describe servicemonitor timesten-operator

Output:

```
Name:
        timesten-operator
Namespace: mynamespace
Labels:
         app=timesten-operator
            database.timesten.oracle.com=timesten-operator
Annotations: deployment.kubernetes.io/revision: 1
API Version: monitoring.coreos.com/v1
Kind:
            ServiceMonitor
Metadata:
 Creation Timestamp: 2025-01-24T20:49:33Z
 Generation:
               1
 Owner References:
   API Version:
                       apps/v1
   Block Owner Deletion: true
   Controller:
                        true
   Kind:
                        Deployment
   Name:
                         timesten-operator
```



```
3084af2d-90ca-49d1-874a-cde1174f50b5
   UID:
 Resource Version:
                          282625424
                          6337b813-9d62-43b0-8594-60e387e7a30d
 UID:
Spec:
  Endpoints:
   Bearer Token Secret:
     Key:
   Interval: 15s
   Path:
              /metrics
   Port:
              metrics
    Scheme:
             https
   Tls Config:
      Ca:
       Secret:
         Key:
                ca.crt
         Name: timesten-operator-metrics-client
      Cert:
       Secret:
         Key: client.crt
         Name: timesten-operator-metrics-client
      Key Secret:
       Key:
                   client.key
                   timesten-operator-metrics-client
       Name:
      Server Name: timesten-operator.mynamespace.svc.cluster.local
 Namespace Selector:
  Selector:
   Match Labels:
     App: timesten-operator
Events:
          <none>
```

Let's look at the important information in this ServiceMonitor object:

- There is an app=timesten-operator label. If there are Pods with a label that matches app=timesten-operator, Prometheus scrapes metrics from them. The TimesTen Operator Pod contains the app=timesten-operator label. Prometheus will therefore scrape metrics from it. We will see this later.
- Prometheus scrapes metrics from the /metrics endpoint.
- Metrics are exposed using https.
- The TimesTen Operator placed the timesten-operator-metrics and timestenoperator-metrics-client Kubernetes Secrets in the ServiceMonitor object. These Secrets are used by the Prometheus Operator.

The Prometheus Operator edits the Prometheus server configuration files based on the information in this ServiceMonitor object.

6. Confirm the TimesTen Operator created the appropriate Kubernetes Service.

kubectl describe service timesten-operator

Output:

Labels:	app=timesten-operator
Namespace:	mynamespace
Name:	timesten-operator



database.timesten.oracle.com=timesten-operator Annotations: deployment.kubernetes.io/revision: 1 name=timesten-operator Selector: ClusterIP Type: IP Family Policy: SingleStack IP Families: IPv4 IP: 10.96.169.59 IPs: 10.96.169.59 metrics 8080/TCP Port: TargetPort: 8080/TCP Endpoints: 10.244.8.180:8080 probe 8081/TCP Port: TargetPort: 8081/TCP Endpoints: 10.244 8 10.244.8.180:8081 Endpoints: Session Affinity: None Events: <none>

7. Confirm there is a Prometheus server running in your namespace.

kubectl get pods

Output.

```
NAME READY STATUS RESTARTS AGE prometheus-sampleprometheusserver-0 2/2 Running 0 10d ...
```

The prometheus-sampleprometheusserver-0 Prometheus server is running in your namespace.

 Confirm the Prometheus Operator edited the Prometheus server configuration file based on the information in the timesten-operator ServiceMonitor object.

```
kubectl exec prometheus-sampleprometheusserver-0 -c prometheus --
cat /etc/prometheus/config out/prometheus.env.yaml
```

Output.

```
global:
  evaluation interval: 30s
 scrape interval: 30s
 external labels:
    prometheus: mynamespace/sampleprometheusserver
    prometheus replica: prometheus-sampleprometheusserver-0
scrape configs:
- job name: serviceMonitor/mynamespace/timesten-operator/0
  honor labels: false
  kubernetes sd configs:
  - role: endpoints
   namespaces:
     names:
     - mynamespace
  scrape interval: 15s
  metrics path: /metrics
```



```
scheme: https
  tls config:
    insecure skip verify: false
    ca file: /etc/prometheus/certs/secret mynamespace timesten-operator-
metrics-client ca.crt
    cert file: /etc/prometheus/certs/secret mynamespace timesten-operator-
metrics-client client.crt
    key file: /etc/prometheus/certs/secret mynamespace timesten-operator-
metrics-client client.key
    server name: timesten-operator.mynamespace.svc.cluster.local
  relabel configs:
  - source labels:
    - job
    target label: tmp prometheus job name
  - action: keep
    source labels:
    - meta kubernetes service label app
    - meta kubernetes service labelpresent app
    regex: (timesten-operator);true
  - action: keep
    source labels:
    - meta kubernetes endpoint port name
    regex: metrics
  - source labels:
    - meta kubernetes endpoint address target kind
    - meta kubernetes endpoint address target name
    separator: ;
    regex: Node;(.*)
    replacement: ${1}
    target label: node
  - source labels:
    - meta kubernetes endpoint address target kind
    - meta kubernetes endpoint address target name
    separator: ;
    regex: Pod;(.*)
    replacement: ${1}
    target label: pod
  - source labels:
    - meta kubernetes namespace
    target label: namespace
  - source labels:
    - __meta_kubernetes service name
    target label: service
  - source labels:
    - meta kubernetes pod name
    target label: pod
  - source labels:
    - __meta_kubernetes_pod_container_name
    target label: container
  - action: drop
    source labels:
    - meta kubernetes pod phase
    regex: (Failed|Succeeded)
  - source labels:
    - meta kubernetes service name
    target label: job
```

```
replacement: ${1}
- target_label: endpoint
replacement: metrics
- source_labels:
    __address__
    target_label: __tmp_hash
    modulus: 1
    action: hashmod
- source_labels:
    - __tmp_hash
    regex: 0
    action: keep
metric_relabel_configs: []
```

Prometheus has the information it needs to scrape TimesTen Operator metrics.

9. Review then deploy a TimesTenClassic object.

Review.

cat sample.yaml

Output.

```
apiVersion: timesten.oracle.com/v4
kind: TimesTenClassic
metadata:
   name: sample
spec:
   ttspec:
    storageClassName: oci
    storageSize: 250G
    image: container-registry.oracle.com/timesten/timesten:22.1.1.34.0
    imagePullSecret: sekret
```

Deploy:

kubectl create -f sample.yaml

Output:

timestenclassic.timesten.oracle.com/sample created

10. Wait a few minutes, then confirm the sample TimesTenClassic object is in the Normal state.

kubectl get ttc sample

Output:

NAME	STATE	ACTIVE	AGE
sample	Normal	sample-0	2m37s



- 11. Review some of the TimesTen Operator metrics. In your browser, go to your Prometheus server.
 - a. In the Prometheus server search bar, type a TimesTen Operator metric. For example, timesten_classic_state_normal. Next, click Execute.

Output.

```
timesten_classic_state_normal{container="timesten-operator",
endpoint="metrics", instance="10.244.8.180:8080", job="timesten-
operator",
name="sample", namespace="mynamespace", pod="timesten-
operator-7f77c749fd-2lt5x",
service="timesten-operator"} 1
```

There is one TimesTenClassic object (sample) and it is in the Normal state.

b. In the Prometheus server search bar, type a second TimesTen Operator metric. For example, timesten_classic_state_not_normal. Next, click Execute.

Output.

```
timesten_classic_state_not_normal{container="timesten-operator",
endpoint="metrics", instance="10.244.8.180:8080", job="timesten-
operator",
name="sample", namespace="mynamespace", pod="timesten-
operator-7f77c749fd-2lt5x",
service="timesten-operator"} 0
```

There is one TimesTenClassic object (sample) and it is in either Normal or Initializing state. A value of 0 indicates it is not in any other state.

c. In the Prometheus server search bar, type a third TimesTen Operator metric. For example, timesten_classic_state. Next, click Execute.

Output.

```
timesten_classic_state{container="timesten-operator",
endpoint="metrics",
instance="10.244.8.180:8080", job="timesten-operator", name="sample",
namespace="mynamespace", pod="timesten-operator-7f77c749fd-2lt5x",
service="timesten-operator", state="Initializing"} 0
```

```
timesten_classic_state{container="timesten-operator",
endpoint="metrics",
instance="10.244.8.180:8080", job="timesten-operator", name="sample",
namespace="mynamespace", pod="timesten-operator-7f77c749fd-2lt5x",
service="timesten-operator", state="Normal"} 1
```

The sample TimesTenClassic object is no longer in the Initializing state. It is now in the Normal state.

Congratulations! You successfully walked through an example demonstrating how TimesTen Operator metrics are exposed, scraped, and published.



16 Work with TimesTen Cache

The TimesTen Operator supports the use of TimesTen Cache in your Kubernetes environment.

About Using TimesTen Cache

The TimesTen Operator provides interfaces that you can use to configure cache groups.

These are the cache-related metadata files that you can provide:

 cacheUser: This file contains the TimesTen cache manager user. Its format is user/ttpwd/ orapwd, containing the username, TimesTen password, and Oracle password for the user. The Oracle user (called the cache administration user in Oracle Database) must exist prior to creating and deploying TimesTen Classic or TimesTen Scaleout. The Operator creates the TimesTen user in the the TimesTen database with the given name and password. The Operator also grants this user the appropriate privileges.

Here is an example:

cachemanageruser/ttmgrpwd/oramgrpwd

See cacheUser for more information.

 cachegroups.sql: This file may contain CREATE CACHE GROUP statements as well as LOAD CACHE GROUP statements for the Operator to automatically run when the TimesTen database is created. The file also contains TimesTen built-in procedures to update statistics on the cache group tables (such as, ttOptEstimateStats and ttOptUpdateStats).

Here is an example:

```
CREATE READONLY CACHE GROUP readcache
AUTOREFRESH
INTERVAL 5 SECONDS
FROM oratt.readtab (
keyval NUMBER NOT NULL PRIMARY KEY,
str VARCHAR2(32)
);
LOAD CACHE GROUP readcache COMMIT EVERY 256 ROWS;
```

See cachegroups.sql for more information.

If you provide the cacheUser and cachegroups.sql files, the Operator uses them to provision TimesTen Cache when a new database is created.

The following metadata files are also relevant for TimesTen Cache:

• tnsnames.ora: This file is required. It defines Oracle Net Services that applications connect to. For TimesTen Cache, this file configures the connectivity between the TimesTen and the Oracle Database (from which data is being cached). In this context, TimesTen is the application that is the connection to the Oracle Database. See tnsnames.ora for details.


sqlnet.ora: This file is optional. However, it may be necessary depending on your Oracle
Database configuration. The file defines options for how client applications communicate
with the Oracle Database. In this context, TimesTen is the application. The tnsnames.ora
and sqlnet.ora files together define how an application communicates with the Oracle
Database.

See sqlnet.ora for details.

• db.ini: This file is required. Its contents contain TimesTen connection attributes for your TimesTen database. The files are included in TimesTen's sys.odbc.ini file in TimesTen Classic or in the database definition file (dbDef) in TimesTen Scaelout. You must specify the OracleNetServiceName and the DatabaseCharacterSet connection attributes in this file. The DatabaseCharacterSet value must match the value of the Oracle database character set value.

See db.ini for details.

• schema.sql: This file may be required. In TimesTen Cache, one or more cache table users own the cache tables. If this cache table user is not the cache manager user, then you must specify the schema.sql file and in it include the schema user. You must also assign the appropriate privileges to this schema user. For example, if the oratt schema user was created in the Oracle Database, and this user is not the TimesTen cache manager user, you must create the TimesTen oratt user in this file.

The instance administrator uses the ttlsql utility to run this file immediately after the database is created. This file is run before the Operator configures TimesTen Cache, so ensure there are no cache definitions in this file.

See Create the Oracle Database Users for more information on the schema users in the Oracle Database. See schema.sql for details about the schema.sql file.

In TimesTen Classic, the contents of the cachegroups.sql file runs on the active database before it is duplicated to the standby. If there are autorefresh cache groups specified in the cachegroups.sql file, they are paused by the agent prior to duplicating the active database to the standby. After the duplication process completes, these autorefresh cache groups are re-enabled.

In TimesTen Scaleout, the contents of the cachegroups.sql file runs during database creation.

Once created and rolled out, the Operator does not monitor or manage TimesTen Cache. Specifically, the Operator does not monitor the health of the cache agents, nor does it take further action to start or stop them. In addition, the Operator does not verify that data is propagating correctly between the TimesTen database and the Oracle Database.

If you delete your TimesTenClassic or TimesTenScaleout object, the Operator automatically cleans up the Oracle Database metadata. If, however, you want to retain the Oracle Database metadata, specify the cacheCleanUp datum in your TimesTenClassic or TimesTen Scaleout object definition and set its value to false. See cacheCleanup datum in TimesTenClassicSpecSpec and TimesTenScaleoutSpecSpec.

For a complete example, see TimesTen Cache in TimesTen Classic Example.



17 Use Encryption for Data Transmission

TimesTen replication and TimesTen Client/Server support the use of Transport Layer Security (TLS) for communication between TimesTen instances.

This chapter details the process for configuring and using TLS in your Kubernetes environment. This enables encrypted data transmission between your replicated TimesTen databases and, if in a Client/Server environment, between your TimesTen client applications and your TimesTen Server (your TimesTen database).

Topics include:

- Create TLS Certificates for Replication and Client/Server
- Configure TLS for Replication
- Configure TLS for Client/Server

Create TLS Certificates for Replication and Client/Server

By default, TimesTen replication transmits data between your TimesTen databases unencrypted. In addition, in a TimesTen Client/Server environment, by default data is transmitted unencrypted between your application and your TimesTen database.

You can choose to enable encryption for replication and for Client/Server through the use of Transport Layer Security (TLS). TimesTen provides the ttCreateCerts utility to generate self-signed certificates for TLS. For more information on TLS certificates and wallets, see About Using Certificates with Client/Server in the Oracle TimesTen In-Memory Database Security Guide.

Note:

Java must be installed on your development host in order for you to use the ttCertsCreate utility. The utility searches for Java according to the JRE_HOME, JAVA HOME, and PATH settings.

The ttCreateCerts utility is located in the /bin directory of a TimesTen instance. The utility creates three wallets: rootWallet, clientWallet, and serverWallet.

From your Linux development host, perform these steps to create the certificates.

1. Navigate to the bin directory of the installation and run the ttInstanceCreate utility interactively to create an instance. Recall that the *installation_dir* directory was created when you unpacked the TimesTen distribution.

You have to create a TimesTen instance as the ttCreateCerts utility is run from a TimesTen instance. For more information on the ttInstanceCreate utility, see ttInstanceCreate in the Oracle TimesTen In-Memory Database Reference.



Create the instance directory (/scratch/ttuser/instance_dir, in this example), then run the ttInstanceCreate utility, supplying the -name and the -location parameters. This example uses instance1 as the name of the instance and uses /scratch/ttuser/ instance dir as the location of the instance.

The 22.1 Release Notes are located here : 'installation dir/tt22.1.1.34.0/README.html'

2. Set the <code>TIMESTEN_HOME</code> environment variable. This variable must be set before you run the <code>ttCertsCreate</code> utility. From the <code>bin</code> directory of the instance, source the <code>ttenv.csh</code> or the <code>ttenv.sh</code> script.

This example uses the bash Bourne-type shell. (Not all output is shown.)

```
% . /scratch/ttuser/instance_dir/instance1/bin/ttenv.sh
LD_LIBRARY_PATH set to
...
PATH set to
...
CLASSPATH set to
TIMESTEN HOME set to /scratch/ttuser/instance dir/instance1
```

3. Run the ttCreateCerts utility from the bin directory of the instance. This example uses the -verbose qualifier to show detailed output. See Using TLS for Client/Server in TimesTen Classic in the Oracle TimesTen In-Memory Database Security Guide for more information on the ttCreateCerts utility.

The default wallet directory is *timesten_home/conf*, where *timesten_home* is the TimesTen instance home directory. This example uses this default wallet directory.

```
% /scratch/ttuser/instance_dir/instancel/bin/ttCreateCerts -verbose
Requested Certificates:
User Certificates:
Subject: CN=server1,C=US
Trusted Certificates:
Subject: CN=ecRoot,C=US
Requested Certificates:
User Certificates:
Subject: CN=client1,C=US
Trusted Certificates:
Subject: CN=ecRoot,C=US
ttCreateCerts : certificates created in /scratch/ttuser/instance_dir/
instancel/conf
```

4. Review the wallet locations and the certificates (represented in **bold**). The cwallet.sso in the serverWallet directory is the file you will supply as the replicationWallet metadata



file for replication and for the server in a Client/Server environment. The cwallet.sso in the clientWallet directory is the file you will use for the client in a Client/Server environment. See "About Configuration Metadata Details" for information on the replicationWallet and the clientWallet metadata files. Also see "Configure TLS for Replication" and "Configure TLS for Client/Server" for information on using these metadata files.

(These cwallet.sso files are also represented in **bold**).

% ls \$TIMESTEN_HOME/conf clientl.cert root.cert serverl.cert snmp.ini sys.ttconnect.ini clientWallet rootWallet serverWallet sys.odbc.ini timesten.conf % ls \$TIMESTEN_HOME/conf/*Wallet* /scratch/ttuser/instance_dir/instancel/conf/clientWallet: cwallet.sso cwallet.sso.lck /scratch/ttuser/instance_dir/instancel/conf/rootWallet: cwallet.sso cwallet.sso.lck /scratch/ttuser/instance_dir/instancel/conf/serverWallet: cwallet.sso cwallet.sso.lck

You have successfully created the certificates that can be used for TLS for both replication and TimesTen Client/Server. You are now ready to configure and use TLS for replication, for Client/Server, or for both replication and Client/Server.

Configure TLS for Replication

You can configure TLS for replication to ensure secure network communication between your replicated TimesTen databases. See Transport Layer Security for TimesTen Replication in the *Oracle TimesTen In-Memory Database Security Guide* for detailed information.

These sections describe how to configure and use TLS for replication:

- Create Metadata Files and Kubernetes Facilities
- Create a TimesTenClassic Object
- Monitor Deployment of a TimesTenClassic Object
- Verify TLS Is Being Used for Replication

Create Metadata Files and Kubernetes Facilities

The /ttconfig/replicationWallet metadata file is required for TLS support for replication. (The /ttconfig directory is located in the containers of your TimesTen databases.) This file must contain the cwallet.sso file (the Oracle wallet) that was generated when you created the TLS certificates. Recall that this file was located in the /scratch/ttuser/instance_dir/ instance1/conf/serverWallet directory. See Create TLS Certificates for Replication and Client/Server for information on creating these certificates. This wallet contains the credentials that are used by TimesTen replication for configuring TLS encryption between your active standby pair of TimesTen databases.

In addition to the /ttconfig/replicationWallet metadata file, you may use the other supported metadata files. See About Configuration Metadata Details for information on these supported metadata files.

You can include these metadata files in one or more Kubernetes facilities (for example, in a Kubernetes Secret, in a ConfigMap, or in an init container). This ensures the metadata files are



populated in the /ttconfig directory of the TimesTen containers. Note that there is no requirement as to how to get the metadata files into this /ttconfig directory. See Populate the /ttconfig Directory for more information.

The example in the following sections illustrates how to include the replicationWallet metadata file in a Kubernetes Secret. It also creates the db.ini, the adminUser, and the schema.sql metadata files and includes these metadata files in a ConfigMap:

- Create a Kubernetes Secret
- Create a ConfigMap

Create a Kubernetes Secret

This section creates the repl-tls Kubernetes Secret. The repl-tls Secret will contain the replicationWallet metadata file.

On your Linux development host:

1. From the directory of your choice, create an empty subdirectory. This example creates the serverWallet subdirectory. (The serverWallet directory is used in the remainder of this example to denote this directory.)

```
% mkdir -p serverWallet
```

 Copy the /scratch/ttuser/instance_dir/instance1/conf/serverWallet/cwallet.sso file into the serverWallet directory that you just created. Recall that this file was generated when you used the ttCreateCerts utility to create the TLS certificates. See "Create TLS Certificates for Replication and Client/Server" for information.

% cp /scratch/ttuser/instance_dir/instance1/conf/serverWallet/cwallet.sso serverWallet/cwallet.sso

3. Create the Kubernetes Secret.

In this example:

- The name of the Secret is repl-tls. Replace repl-tls with a name of your choosing. (repl-tls is represented in **bold**.)
- The name of the metadata file required for TLS replication is replicationWallet (represented in **bold**).
- The location of the wallet directory is serverWallet (in this example, represented in **bold**). If you use a different directory, replace serverWallet with the name of your directory.
- The name of the Oracle wallet is cwallet.sso (represented in bold).

Use the kubectl create command to create the Secret:

```
% kubectl create secret generic repl-tls
--from-file=replicationWallet=serverWallet/cwallet.sso
secret/repl-tls created
```

You have successfully created and deployed the <code>repl-tls</code> Kubernetes Secret. The <code>replicationWallet/cwallet.sso</code> file will later be available in the <code>/ttconfig</code> directory of the TimesTen containers. In addition, the file will be available in the <code>/tt/home/timesten/</code> replicationWallet directory of the TimesTen containers.



Create a ConfigMap

This section creates the repl-tls ConfigMap. This ConfigMap contains the db.ini, the adminUser, and the schema.sql metadata files.

These metadata files are not required for TLS, but are included as additional attributes for your TimesTen databases. See "Overview of Configuration Metadata and Kubernetes Facilities" for information on the metadata files and the ConfigMap facility.

On your Linux development host:

1. From the directory of your choice, create an empty subdirectory for the metadata files. This example creates the cm_replTLS subdirectory. (The cm_replTLS directory is used in the remainder of this example to denote this directory.)

```
% mkdir -p cm_replTLS
```

2. Navigate to the ConfigMap directory.

```
% cd cm_replTLS
```

3. Create the db.ini file in this ConfigMap directory (cm_replTLS, in this example). In this db.ini file, define the PermSize and DatabaseCharacterSet connection attributes.

```
vi db.ini
```

```
PermSize=200
DatabaseCharacterSet=AL32UTF8
```

4. Create the adminUser file in this ConfigMap directory (cm_replTLS, in this example). In this adminUser file, create the sampleuser user with the samplepw password.

vi adminUser

sampleuser/samplepw

5. Create the schema.sql file in this ConfigMap directory (cm_replTLS, in this example). In this schema.sql file, define the s sequence and the emp table for the sampleuser user. The Operator will automatically initialize your database with these object definitions.

```
vi schema.sql
create sequence sampleuser.s;
create table sampleuser.emp (
   id number not null primary key,
   name char(32)
);
```

6. Create the ConfigMap. The files in the cm_replTLS directory are included in the ConfigMap and, later, will be available in the TimesTen containers.

In this example:

• The name of the ConfigMap is repl-tls. Replace repl-tls with a name of your choosing. (repl-tls is represented in **bold** in this example.)



• This example uses cm_replTLS as the directory where the files that will be copied into the ConfigMap reside. If you use a different directory, replace cm_replTLS with the name of your directory. (cm replTLS is represented in **bold** in this example.)

Use the kubectl create command to create the ConfigMap:

```
% kubectl create configmap repl-tls --from-file=cm_replTLS
configmap/repl-tls created
```

 Use the kubectl describe command to verify the contents of the ConfigMap. (repl-tls, in this example.)

```
% kubectl describe configmap repl-tls
Name: repl-tls
Namespace: mynamespace
Labels: <none>
Annotations: <none>
Data
====
adminUser:
____
sampleuser/samplepw
db.ini:
____
PermSize=200
DatabaseCharacterSet=AL32UTF8
schema.sql:
____
create sequence sampleuser.s;
create table sampleuser.emp (id number not null primary key, name char (32));
Events: <none>
```

You have successfully created and deployed the repl-tls ConfigMap.

Create a TimesTenClassic Object

This section creates the TimesTenClassic object. For detailed information about the TimesTenClassic object type, see About the TimesTenClassic Object Type.

Perform these steps:

 Create an empty YAML file. You can choose any name, but you may want to use the same name you used for the name of the TimesTenClassic object. (In this example, repltls.) The YAML file contains the definitions for the TimesTenClassic object. See "TimesTenClassicSpecSpec" for information on the fields that you must specify in this YAML file as well as the fields that are optional.

In this example, the fields of particular interest for TLS replication are:

- dbSecret: This example uses one Kubernetes Secret (called repl-tls) for the replicationWallet metadata file.
- replicationCipherSuite: This field is required for TLS for replication. In this example, the value is SSL_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256. See Task 3: Configure TLS for Replication in the Oracle TimesTen In-Memory Database Security Guide and see the replicationCipherSuite entry in "Table 20-3" in this book for more information.



• replicationSSLMandatory: This field is optional. In this example, set replicationSSLMandatory equal to 1. See Task 3: Configure TLS for Replication in the *Oracle TimesTen In-Memory Database Security Guide* and see the replicationSSLMandatory entry in Table 20-3 in this book for more information.

In addition, this example includes:

- name: Replace repltls with the name of your TimesTenClassic object.
- storageClassName: Replace oci-by with the name of the storage class used to allocate PersistentVolumes to hold TimesTen.
- storageSize: Replace 250Gi with the amount of storage that should be requested for each Pod to hold TimesTen. Note: This example assumes a production environment and uses a value of 250Gi for storageSize. For demonstration purposes, a value of 50Gi is adequate. See the storageSize and the logStorageSize entries in "Table 20-3" for information.
- image: Replace container-registry.oracle.com/timesten/timesten:22.1.1.34.0 with the location and the name of image.
- imagePullSecret: Replace sekret with the image pull secret that Kubernetes should use to fetch the TimesTen image.
- dbConfigMap: This example uses one ConfigMap (called repl-tls) for the db.ini, the adminUser, and the schema.sql metadata files.

```
% vi repltls.yaml
```

```
apiVersion: timesten.oracle.com/v4
kind: TimesTenClassic
metadata:
 name: repltls
spec:
  ttspec:
   storageClassName: oci-bv
    storageSize: 250Gi
    image: container-registry.oracle.com/timesten/timesten:22.1.1.34.0
    imagePullSecret: sekret
    dbConfigMap:
    - repl-tls
    dbSecret:
    - repl-tls
    replicationCipherSuite: SSL ECDHE ECDSA WITH AES 128 GCM SHA256
    replicationSSLMandatory: 1
```

2. Use the kubectl create command to create the TimesTenClassic object from the contents of the YAML file (in this example, repltls.yaml). Doing so begins the process of deploying your active standby pair of TimesTen databases in the Kubernetes cluster.

```
% kubectl create -f repltls.yaml
timestenclassic.timesten.oracle.com/repltls created
```

You have successfully created the TimesTenClassic object in the Kubernetes cluster. The process of deploying your TimesTen databases begins, but is not yet complete.

Monitor Deployment of a TimesTenClassic Object

Use the kubectl get and the kubectl describe commands to monitor the progress of the active standby pair as it is provisioned.



1. Use the kubectl get command and review the STATE field. Observe the value is Initializing. The active standby pair provisioning has begun, but is not yet complete.

```
% kubectl get ttc repltls
NAME STATE ACTIVE AGE
repltls Initializing None 50s
```

2. Use the kubectl get command again to see if value of the STATE field has changed. In this example, the value is Normal, indicating the active standby pair of databases are now provisioned and the process is complete.

```
% kubectl get ttc repltls
NAME STATE ACTIVE AGE
repltls Normal repltls-0 3m45s
```

3. Use the kubectl describe command to view the active standby pair provisioning in detail.

Note the following have been correctly set in the repltls TimesTenClassic object definition:

- The repl-tls Secret has been correctly referenced in the dbSecret field (represented in bold).
- The repl-tls Configmap has been correctly referenced in the dbConfigMap field (represented in **bold**).
- The replicationCipherSuite field has been correctly set to SSL ECDHE ECDSA WITH AES 128 GCM SHA256 (represented in bold).
- The replicationSSLMandatory field has been correctly set to 1 (represented in bold).

Note: Not all of the output is shown in this example.

```
% kubectl describe ttc repltls
Name: repltls
Namespace: mynamespace
Labels: <none>
Annotations: <none>
API Version: timesten.oracle.com/v4
         TimesTenClassic
Kind:
Metadata:
 Creation Timestamp: 2025-01-16T18:51:43Z
 Generation: 1
 Resource Version: 75029797
 Self Link:
/apis/timesten.oracle.com/v4/namespaces/mynamespace/timestenclassics/repltls
 UID:
                     a2915ef3-0fe0-11eb-8b9a-aaa0151611fe
Spec:
  Ttspec:
   Db Config Map:
     repl-tls
   Db Secret:
     repl-tls
   Image:
                             container-registry.oracle.com/timesten/
timesten:22.1.1.34.0
   Image Pull Policy:
                           Always
   Image Pull Secret:
                            sekret
   Replication Cipher Suite: SSL_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
   Replication SSL Mandatory: 1
   Storage Class Name: oci-bv
   Storage Size:
                             250Gi
. . .
Events:
  Type Reason Age From
                                    Message
```

```
-----
 -
      Create
                 4m17s ttclassic Secret tta2915ef3-0fe0-11eb-8b9a-aaa0151611fe
created
                  4m17s ttclassic Service repltls created
      Create
      Create 4m17s ttclassic StatefulSet repltls created
 _
       StateChange 3m10s ttclassic Pod repltls-1 Agent Up
       StateChange 3ml0s ttclassic Pod repltls-1 Release 22.1.1.34.0
 _
      StateChange 3ml0s ttclassic Pod repltls-1 Daemon Up
 _
 _
       StateChange 2m3s ttclassic Pod repltls-0 Agent Up
      StateChange 2m3s ttclassic Pod repltls-0 Release 22.1.1.34.0
 _
      StateChange 2mls ttclassic Pod repltls-0 Daemon Up
      StateChange 68s ttclassic Pod repltls-0 Database Loaded
      StateChange 68s ttclassic Pod repltls-0 Database Updatable
 _
      StateChange 68s ttclassic Pod repltls-0 CacheAgent Not Running
 _
      StateChange 68s ttclassic Pod repltls-0 RepAgent Not Running
 _
      StateChange 67s ttclassic Pod repltls-0 RepState IDLE
      StateChange 67s ttclassic Pod repltls-0 RepScheme None
 _
 _
      StateChange 66s ttclassic Pod repltls-0 RepAgent Running
 _
      StateChange 66s ttclassic Pod repltls-0 RepScheme Exists
      StateChange 66s ttclassic Pod repltls-0 RepState ACTIVE
 _
      StateChange 47s ttclassic Pod repltls-1 Database Loaded
 _
 _
      StateChange 47s ttclassic Pod repltls-1 Database Not Updatable
     StateChange 47s ttclassic Pod repltls-1 CacheAgent Not Running
      StateChange 47s ttclassic Pod repltls-1 RepAgent Not Running
      StateChange 47s ttclassic Pod repltls-1 RepScheme Exists
      StateChange 47s ttclassic Pod repltls-1 RepState IDLE
      StateChange 41s ttclassic Pod repltls-1 RepAgent Running
      StateChange 36s ttclassic Pod repltls-1 RepState STANDBY
       StateChange 36s ttclassic TimesTenClassic was Initializing, now Normal
```

Your active standby pair of TimesTen databases are successfully deployed (as indicated by Normal.) You are now ready to verify that TLS is being used for replication.

Verify TLS Is Being Used for Replication

To verify TLS is being used for replication, perform the following steps:

1. Review the active (repltls-0, in this example) pod and the standby pod (repltls-1, in this example).

% kubectl get pods					
NAME	READY	STATUS	RESTARTS	AGE	
repltls-0	2/2	Running	0	6m35s	
repltls-1	2/2	Running	0	6m34s	
timesten-operator-f84766548-tch7s	1/1	Running	0	28d	

2. Optional: Use the kubectl exec -it command to invoke the shell in the active Pod (repltls-0, in this example).

```
% kubectl exec -it repltls-0 -c tt -- /bin/bash
```

 Optional: From the shell in the active pod, verify the cwallet.sso file is located in the /tt/ home/timesten/replicationWallet directory.

```
% ls /tt/home/timesten/replicationWallet
cwallet.sso
```

4. Optional: From the shell in the active pod, verify that the TLS replication-specific values are correct in the timesten.conf configuration file. (This file is located in the /tt/home/timesten/instances/instance1/conf directory.)

In particular, note that:



- replication_wallet is correctly set to /tt/home/timesten/replicationWallet (represented in bold).
- replication_cipher_suite is correctly set to SSL ECDHE ECDSA WITH AES 128 GCM SHA256 (represented in bold).
- replication_ssl_mandatory is correctly set to 1 (represented in bold).

See Task 3: Configure TLS for Replication in the Oracle TimesTen In-Memory Database Security Guide for more information on these timesten.conf attributes.

```
% cat /tt/home/timesten/instances/instance1/conf/timesten.conf
admin uid=3429
admin user=timesten
daemon port=6624
group name=timesten
hostname=repltls-0
instance guid=48AC5964-56A1-4C66-AB89-5646A2431EA3
instance name=instance1
replication_cipher_suite=SSL_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
replication ssl mandatory=1
replication wallet=/tt/home/timesten/replicationWallet
server port=6625
show date=1
timesten release=22.1.1
tns admin=/ttconfig
verbose=1
```

5. From the shell in the active pod, run the ttRepAdmin utility with the -showstatus -detail options to verify the replication agent transmitters and receivers are using TLS (as indicated by SSL, represented in **bold**). See ttRepAdmin in the *Oracle TimesTen In-Memory Database Reference* for information on this utility.

Note: Not all output is shown in this example.

```
% ttRepAdmin -showstatus -detail repltls
Replication Agent Status as of: 2025-01-16 19:01:55
DSN : repltls
...
TRANSMITTER thread(s) (TRANSMITTER(M):139870727366400):
For : REPLTLS (track 0) (SSL)
Start/Restart count : 1
Current state : STATE_META_PEER_INFO
RECEIVER thread(s) (RECEIVER:139870719887104):
For : REPLTLS (track 0) (SSL)
Start/Restart count : 1
Current state : STATE_RCVR_READ_NETWORK_LOOP
```

You have successfully verified that TLS for replication is being used.

Automatically Configure Client/Server TLS

The TimesTen Operator can automatically create self-signed certificates and configure TimesTen to use those certificates for client/server TLS encryption. This automation eliminates the requirement for you to manually run the TimesTen ttCreateCerts utility to create selfsigned certificates as well to perform additional manual steps to make the resulting certificates available to TimesTen on TimesTen servers and TimesTen clients. Topics in this section include the following:

- About Configuring a TimesTenClassic Object for Automatic Client/Server TLS Encryption
- About the Automation Process
- How-to Example

About Configuring a TimesTenClassic Object for Automatic Client/Server TLS Encryption

You can customize your TimesTenClassic object to enable automatic client/server TLS encryption by including a .spec.ttspec.clientTLS section in your TimesTenClassic object definition. When you include the .spec.ttspec.clientTLS section in your TimesTenClassic object definition and set .spec.ttspec.clientTLS.auto to true, the TimesTen Operator notices and takes action to create self-signed certificates and configure TimesTen to use those certificates for client/server encryption.

Here is a snippet of a TimesTenClassic object that uses .spec.ttspec.clientTLS.

```
apiVersion: timesten.oracle.com/v4
kind: TimesTenClassic
metadata:
  name: sampletls
spec:
  ttspec:
    storageClassName: oci-bv
    clientTLS:
      auto: true
      ciphersuites: SSL ECDHE ECDSA WITH AES 256 GCM SHA384
      eccurve: p256
      encryption: required
      signAlg: ecdsasha384
      validity: 365
    storageSize: 10Gi
    image: container-registry.oracle.com/timesten/timesten:22.1.1.34.0
    imagePullSecret: sekret
    dbConfigMap:
    - sampletls
```

Let's look at .spec.ttspec.clientTLS in more detail:

- auto: Determines if automatic client/server encryption is enabled. Set auto to true to enable automatic client/server encryption. The default is false. If set to false, there is no automatic client/server encryption for this TimesTenClassic object.
- ciphersuites: Defines the cipher suite(s) used for client/server communication. You can specify one or more cipher suites. If there is more than one cipher suite, use a comma-separated list, and list the cipher suites in order of preference. This example specifies SSL ECDHE ECDSA WITH AES 256 GCM SHA384.

The TimesTen Operator first checks to see if there is a ciphersuites entry in the db.ini file.

- If there is an entry, the TimesTen Operator uses it.
- If there is no entry, the TimesTen Operator uses the value specified in .spec.ttspec.clientTLS.ciphersuites.



 If .spec.ttspec.clientTLS.auto is true, and there is no value specified in either the db.ini file or in .spec.ttspec.clientTLS.ciphersuites, the TimesTen Operator sets the value to SSL ECDHE ECDSA WITH AES 128 GCM SHA256.

The TimesTen Operator adds the ciphersuites value to the TimesTen Server's sys.odbc.ini file.

• eccurve: Defines the elliptical curve signing algorithm. This example specifies p256.

When the TimesTen Operator runs the TimesTen ttCreateCerts utility, it supplies the .spec.ttspec.clientTLS.eccurve value to the -eccurve option of the TimesTen ttCreateCerts utility. If you do not specify a value, the default is p384. For a list of supported values, see TimesTenClassicSpecSpecClientTLS.

 encryption: Defines the encryption setting for client/server access. This example specifies required.

The TimesTen Operator first checks to see if there is an encryption entry in the db.ini file for the object.

- If there is an entry, the TimesTen Operator uses it.
- If there is no entry, the TimesTen Operator uses the value specified in .spec.ttspec.clientTLS.encryption.
- If .spec.ttspec.clientTLS.auto is true, and there is no value specified in either the db.ini file or in .spec.ttspec.clientTLS.encryption, the default is accepted. For a list of supported values, see TimesTenClassicSpecSpecClientTLS.

The TimesTen Operator adds the encryption value to the TimesTen Server's sys.odbc.ini file.

• signAlg: Defines the elliptical curve signing algorithm. This example specifies ecdsasha384.

When the TimesTen Operator runs the TimesTen ttCreateCerts utility, it supplies the .spec.ttspec.clientTLS.signAlg value to the -sign_alg option of the TimesTen ttCreateCerts utility. If you do not specify a value, the default is ecdsasha384. For a list of supported values, see TimesTenClassicSpecSpecClientTLS.

• validity: Defines the number of days the created certificate is valid. This example specifies 365.

When the TimesTen Operator runs the TimesTen ttCreateCerts utility, it supplies the .spec.ttspec.clientTLS.validity value to the -validity option of the TimesTen ttCreateCerts utility. If you do not specify a value, the default is 3650. For a list of supported values, see TimesTenClassicSpecSpecClientTLS.

For more information, see the following:

- TimesTenClassicSpecSpecClientTLS in this book.
- Configuration for TLS for Client/Server in the Oracle TimesTen In-Memory Database Security Guide.

About the Automation Process

Let's review the actions the TimesTen Operator takes when you specify .spec.ttspec.clientTLS and set .spec.ttspec.clientTLS.auto: true in a TimesTenClassic object definition:



 The TimesTen Operator runs the TimesTen ttCreateCerts -dir <temporary_directory> utility. The TimesTen Operator may supply additional options to the ttCreateCerts utility depending on the settings you provided in .spec.ttspec.clientTLS.

For more information, see

- About Configuring a TimesTenClassic Object for Automatic Client/Server TLS Encryption
- ttCreateCerts utility in the Oracle TimesTen In-Memory Database Reference
- The TimesTen Operator creates two directories that contain Oracle wallets. These wallets contain the self-signed certificates that were created by the ttCreateCerts utility:
 - /serverWallet/cwallet.sso: Contains the self-signed certificate for TimesTen servers.
 - /clientWallet/cwallet.sso: Contains the self-signed certificate for TimesTen clients.

As an example, let's look at a directory structure for the tt container in a Pod named sampletls-0:

```
[timesten@sampletls-0 /]$ pwd
/
[timesten@sampletls-0 /]$ ls -a serverWallet
. .. cwallet.sso
[timesten@sampletls-0 /]$ ls -a clientWallet
. .. cwallet.sso
```

Note that the /serverWallet/cwallet.sso and the /clientWallet/cwallet.sso wallets exist. These wallets contain the self-signed certificates required for client/server TLS encryption.

- The TimesTen Operator creates two Kubernetes Secrets:
 - <name_of_TimesTenClassic_object>-server where <name_of_TimesTenClassic_object> is the name of the TimesTenClassic object: This is the Kubernetes Secret that contains the self-signed certificate that is stored in / serverWallet/cwallet.sso and used for TimesTen servers.

The containers that run TimesTen are configured to mount this Secret. As a result, this file is available in the TimesTen containers.

- <name_of_TimesTenClassic_object>-client where <name_of_TimesTenClassic_object> is the name of the TimesTenClassic object: This is the Kubernetes Secret that contains the self-signed certificate that is stored in / clientWallet/cwallet.sso and used for TimesTen clients.

As an example, let's look at the Kubernetes Secrets created by the TimesTen Operator for a TimesTenClassic object named sampletls.

kubect.	l get secrets	
NAME		TYPE
DATA	AGE	
sample	tls-client	Opaque
1	21m	
sample	tls-server	Opaque
1	21m	



 The TimesTen Operator checks to see if the /serverWallet directory in the tt container exists. If it does, appropriate Wallet, Ciphersuites, and Encryption entries are added to the sys.odbc.ini file being generated. These entries could be located in the db.ini metadata file or in .spec.ttspec.clientTLS or both. See About Configuring a TimesTenClassic Object for Automatic Client/Server TLS Encryption.

After completing these steps, the TimesTen Operator has automatically configured TimesTen servers for client/server TLS encryption.

For client applications, when you define Pods that run your applications, you can cause the client Secret to be mounted in these Pods. You can also include a wallet entry in your sys.odbc.ini file that directly references the wallet from this Secret. There are steps in the Configure and Deploy the TimesTenClassic Object that show you how to do this.

How-to Example

Let's walk-through an example that illustrates how to configure your TimesTenClassic object for automatic client/server TLS encryption and let's observe how the TimesTen Operator automates the process.

- Before You Begin
- Configure and Deploy the TimesTenClassic Object

Before You Begin

This example assumes that the TimesTenClassic CRD is deployed in your Kubernetes cluster and the TimesTen Operator is running in your namespace at namespace-scope. The example also assumes you have created metadata files and created a Kubernetes Configmap for these metadata files.

Let's confirm the TimesTenClassic CRD is deployed and the TimesTen Operator is running. Next, let's create some metadata files and create a Configmap for these metadata files. These metadata files and their content are not required for automatically configuring client/server TLS. They are used for explanatory purposes.

1. Confirm the TimesTenClassic CRD exists in your Kubernetes cluster.

kubectl get crds | grep timesten

The output is similar to the following:

timestenclassics.timesten.oracle.com

2025-03-28T16:34:33Z

2. Confirm the TimesTen Operator is running in your namespace at namespace-scope.

kubectl get pods

The output is similar to the following:

NAME	READY	STATUS	RESTARTS	AGE
timesten-operator-79bddcc446-nktcz	1/1	Running	0	3m54s

3. Create the metadata files.



a. From a directory of your choice, create an empty subdirectory for the metadata files. This example creates the cm subdirectory.

mkdir cm

b. Using an editor of your choice, create your desired metadata files. Exit from your editor after creating each file.

db.ini:

vi cm/db.ini

PermSize=200 DatabaseCharacterSet=AL32UTF8

adminUser:

vi cm/adminUser

adminuser/adminuserpwd

schema.sql

vi cm/schema.sql

create table adminuser.emp (id number not null primary key, name char
(32));

4. Create a Kubernetes Configmap for the metadata files.

a. Create the configmap.

kubectl create configmap sampletls --from-file cm

b. (Optional) Review the Configmap in your namespace.

kubectl describe configmap sampletls

The output is similar to the following:

Name: sampletls Namespace: mynamespace Labels: <none> Annotations: <none> Data ==== adminUser: ---adminuser/adminuserpwd db.ini: -----PermSize=200



```
DatabaseCharacterSet=AL32UTF8
schema.sql:
----
create table adminuser.emp (id number not null primary key, name char
(32));
BinaryData
====
Events: <none>
```

You are now ready to perform the steps to have the TimesTen Operator automatically configure client/server TLS encryption.

Configure and Deploy the TimesTenClassic Object

This example shows you how to configure and deploy a TimesTenClassic object for automatic client/server TLS encryption. This example uses a YAML manifest file.

1. From a directory of your choice, create the YAML manifest file.

```
vi sampletls.yaml
apiVersion: timesten.oracle.com/v4
kind: TimesTenClassic
metadata:
 name: sampletls
spec:
  ttspec:
    storageClassName: oci-bv
    clientTLS:
      auto: true
      ciphersuites: SSL ECDHE ECDSA WITH AES 256 GCM SHA384
      eccurve: p256
      encryption: required
      signAlg: ecdsasha384
      validity: 365
    storageSize: 10Gi
    image: container-registry.oracle.com/timesten/timesten:22.1.1.34.0
    imagePullSecret: sekret
    dbConfigMap:
    - sampletls
```

Note the following:

- .spec.ttspec.clientTLS is specified and .spec.ttspec.clientTLS is set to true.
 This directs the TimesTen Operator to automatically create self-signed certificates and configure TimesTen to use those certificates for client/server TLS encryption.
- .spec.ttspec.clientTLS.ciphersuites is specified. The TimesTen Operator adds a ciphersuites entry with a value of SSL_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 to the sys.odbc.ini file.



- .spec.ttspec.clientTLS.eccurve is specified. The TimesTen Operator adds the eccurve option to the ttCreateCerts utility and supplies of value of p256 for this option.
- .spec.ttspec.clientTLS.encryption is specified. The TimesTen Operator adds an encryption entry with a value of required to the sys.odbc.ini file.
- .spec.ttspec.clientTLS.signAlg is specified. The TimesTen Operator adds the sign_alg option to the ttCreateCerts utility and supplies of value of ecdsasha384 for this option.
- .spec.ttspec.clientTLS.validity is specified. The TimesTen Operator adds the -validity option to the ttCreateCerts utility and supplies of value of 365 for this option.
- 2. Deploy the TimesTenClassic object.

kubectl create -f sampletls.yaml

The output is the following:

timestenclassic.timesten.oracle.com/sampletls created

- 3. Monitor deployment.
 - a. Check status.

kubectl get ttc sampletls

The output is similar to the following:

NAME STATE ACTIVE AGE sampletls Initializing None 2m33s

The provisioning starts, but is not yet completed.

b. Wait a few minutes, then check status again.

kubectl get ttc sampletls

The output is similar to the following:

NAME	STATE	ACTIVE	AGE
sampletls	Normal	sampletls-0	4m37s

The provisioning process completes. The databases are up and running and operational, as indicated by the Normal state.

- 4. Verify the TimesTen Operator created the Kubernetes Secrets that contain the certificates for TimesTen servers and TimesTen clients.
 - a. Review the Secrets.

kubectl get secrets



The output is similar to the following:

NAME		
TYPE	DATA	AGE
sampletls-client		
Opaque	1	21m
sampletls-server		
Opaque	1	21m

b. Confirm the contents of the sampletls-server Secret.

kubectl describe secret sampletls-server

The output is similar to the following:

Name: sampletls-server Namespace: mynamespace Labels: <none> Annotations: <none> Type: Opaque Data ==== cwallet.sso: 1525 bytes

The Secret contains the cwallet.sso wallet. This wallet contains the self-signed certificate for TimesTen servers.

c. Confirm the contents of the sampletls-client Secret.

kubectl describe secret sampletls-client

The output is similar to the following:

Name: sampletls-client Namespace: mynamespace Labels: <none> Annotations: <none> Type: Opaque Data ==== cwallet.sso: 1525 bytes

The Secret contains the cwallet.sso wallet. This wallet contains the self-signed certificate for TimesTen clients.

The TimesTen Operator ran the TimesTen ttCreateCerts utility to create self-signed certificates for client/server TLS encryption. The TimesTen Operator stored these certificates in Kubernetes Secrets.



- Verify the TimesTen Operator automaticlly configured TimesTen to use the certificates for client/server TLS encryption.
 - a. Establish a shell in the tt container of the sampletls-0 Pod.

kubectl exec -it sampletls-0 -c tt -- /bin/bash

b. Confirm the existence of the server and client wallets.

```
[timesten@sampletls-0 /]$ pwd
/
[timesten@sampletls-0 /]$ ls -a serverWallet
. .. cwallet.sso
[timesten@sampletls-0 /]$ ls -a clientWallet
```

. .. cwallet.sso

The /serverWallet/cwallet.sso wallet contains the self-signed certificate for TimesTen Servers. The /clientWallet/cwallet.sso wallet contains the self-signed certificate for TimesTen clients.

c. (Optional) Confirm the contents of sys.odbc.ini file.

```
[timesten@sampletls-0 ~]$ cat /tt/home/timesten/instances/instance1/
conf/sys.odbc.ini
```

The output is similar to the following:

```
[ODBC Data Sources]
sampletls=TimesTen 22.1 Driver
tt=TimesTen 22.1 Driver
```

```
[sampletls]
Datastore=/tt/home/timesten/datastore/sampletls
```

```
PermSize=200
DatabaseCharacterSet=AL32UTF8
DDLReplicationLevel=3
AutoCreate=0
ForceDisconnectEnabled=1
Wallet=/serverWallet
```

```
Ciphersuites=SSL_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
Encryption=required
```

```
[tt]
```

```
Datastore=/tt/home/timesten/datastore/sampletls
PermSize=200
DatabaseCharacterSet=AL32UTF8
DDLReplicationLevel=3
AutoCreate=0
ForceDisconnectEnabled=1
WaitForConnect=0
```



```
Wallet=/serverWallet
Ciphersuites=SSL_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
Encryption=required
```

The TimesTen Operator correctly added appropriate Wallet, Ciphersuites, and Encryption entries to the sys.odbc.ini file. The values for these entries were supplied in .spec.ttspec.clientTLS and are correct.

- d. Exit from the shell.
- 6. Verify TLS is being used.

The following steps assume you have successfully copied the client wallet to the application container that is running your TimesTen client instance and that you have configured the appropriate client-side attributes. For more information about TimesTen client-side attributes, see Task 3: Set Client Configuration for TLS in TimesTen Classic in the *Oracle TimesTen In-Memory Database Security Guide*.

a. Establish a shell in the application container that contains your TimesTen client instance.

kubectl exec -it client-0 -c tt -- /bin/bash

b. Use ttIsqlCS to connect to the sampletls database.

```
[timesten@sampletls-0]$ ttisqlcs -connstr
"TTC_SERVER1=sampletls-0.sampletls.mynamespace.svc.cluster.local;TTC_SER
VER2=sampletls-1.sampletls.mynamespace.svc.cluster.local;TTC_SERVER_DSN=
sampletls;UID=adminuser;PWD=adminuserpwd;wallet=/
clientWallet;ciphersuites=SSL_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384;encryp
tion=required";
```

The output is similar to the following:

```
Copyright (c) 1996, 2025, Oracle and/or its affiliates. All rights reserved.
Type ? or "help" for help, type "exit" to quit ttIsql.
```

```
connect
"TTC_SERVER1=sampletls-0.sampletls.mynamespace.svc.cluster.local;TTC_SER
VER2=sampletls-1.sampletls.mynamespace.svc.cluster.local;TTC_SERVER_DSN=
sampletls;UID=adminuser;PWD=*******;wallet=/
clientWallet;ciphersuites=SSL_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384;encryp
tion=required";
Connection successful:
DSN=;TTC_SERVER=sampletls-0.sampletls.mynamespace.svc.cluster.local;TTC_
SERVER_DSN=sampletls;UID=adminuser;DATASTORE=/tt/home/timesten/
datastore/
sampletls;DATABASECHARACTERSET=AL32UTF8;CONNECTIONCHARACTERSET=US7ASCII;
AUTOCREATE=0;PERMSIZE=200;DDLREPLICATIONLEVEL=3;FORCEDISCONNECTENABLED=1
;Encryption=Required;Wallet=/
clientWallet;CipherSuites=SSL_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384;
(Default setting AutoCommit=1)
```



c. Use the TimesTen sqlgetconnectattr command in ttIsqlCS to verify TLS is being used. A return value of 1 indicates TLS is being used.

Command> sqlgetconnectattr tt_tls_session;

The output is the following:

TT TLS SESSION = 1 (SQL TRUE)

Congratulations! You successfully configured a TimesTenClassic object for automatic client/ server TLS encryption. After you deployed the TimesTenClassic object in your namespace, the TimesTen Operator automatically created self-signed certificates and configured TimesTen to use those certificates for client/server TLS encryption.

Configure TLS for Client/Server

You can configure TLS for Client/Server to ensure secure network communication between TimesTen clients and servers. See Transport Layer Security for TimesTen Client/Server in the Oracle TimesTen In-Memory Database Security Guide for detailed information.

There are both server-side and client-side configuration requirements for using TLS for Client/ Server. These requirements are detailed in these sections:

- Configuration Requirements for the Server
- Configuration Requirements for the Client

Configuration Requirements for the Server

These sections discuss the configuration requirements for the server. The sections also include an example of how to configure TLS for the server in your Kubernetes cluster.

- Overview of Metadata Files and Kubernetes Facilities
- Create a Kubernetes Secret for the csWallet Metadata File
- Create a ConfigMap for the Server-Side Attributes
- Create a TimesTenClassic Object
- Monitor Deployment of the TimesTenClassic Object

Overview of Metadata Files and Kubernetes Facilities

The /ttconfig/csWallet metadata file is required for TLS support for Client/Server. (The / ttconfig directory is located in the containers of your TimesTen databases.) This file must contain the cwallet.sso file (the Oracle wallet) that was generated when you created the TLS certificates. This file is the Oracle wallet required for the server. Recall that this file was located in the /scratch/ttuser/instance_dir/instance1/conf/serverWallet directory. See Create TLS Certificates for Replication and Client/Server for information on creating these certificates. This wallet contains the credentials that are used for configuring TLS encryption between your TimesTen database and your Client/Server applications.

There are also server-side connection attributes that must be set. You can define these attributes in the db.ini metadata file. After the db.ini file is placed in the /ttconfig directory of the TimesTen containers, the Operator copies the contents of the db.ini file to the

timesten_home/conf/sys.odbc.ini file located in the TimesTen containers. (Note that timesten_home is the TimesTen instance directory. This instance directory is /tt/home/ timesten/instances/instance1 in your TimesTen containers.)

These required server-side attributes are: Wallet, CipherSuites, and Encryption. See Create a ConfigMap for the Server-Side Attributes for information on these attributes. Also see Server Attributes for TLS in the Oracle TimesTen In-Memory Database Security Guide.

In addition to the csWallet and the db.ini metadata files, you may use other supported metadata files. See About Configuration Metadata Details for information on these supported metadata files.

You can include these metadata files in one or more Kubernetes facilities (for example, in a Kubernetes Secret, in a ConfigMap, or in an init container). This ensures the metadata files are populated in the /ttconfig directory of the TimesTen containers. Note that there is no requirement as to how to get the metadata files into this /ttconfig directory. See Populate the /ttconfig Directory.

The following example includes the csWallet metadata file in a Kubernetes Secret. It also creates the db.ini, the adminUser, and the schema.sql metadata files and includes these metadata files in a ConfigMap.

Create a Kubernetes Secret for the csWallet Metadata File

This section creates the cs-tls Kubernetes Secret. The cs-tls Secret will contain the csWallet metadata file.

On your Linux development host:

1. From the directory of your choice, create an empty subdirectory. This example creates the serverWallet subdirectory. (The serverWallet directory is used in the remainder of this example to denote this directory.)

% mkdir -p serverWallet

2. Copy the cwallet.sso file into the serverWallet directory that you just created. Recall that the cwallet.sso file was generated when you used the ttCreateCerts utility to create the TLS certificates. Also recall that this file was located in the /scratch/ttuser/ instance_dir/instance1/conf/serverWallet directory. See "Create TLS Certificates for Replication and Client/Server" for information.

```
% cp /scratch/ttuser/instance_dir/instance1/conf/serverWallet/cwallet.sso
serverWallet/cwallet.sso
```

3. Create the Kubernetes Secret.

In this example:

- The name of the Secret is cs-tls. Replace cs-tls with a name of your choosing. (cs-tls is represented in **bold**.)
- The name of the metadata file required for TLS for Client/Server is csWallet (represented in **bold**).
- The location of the wallet directory is serverWallet (in this example, represented in **bold**). If you use a different directory, replace serverWallet with the name of your directory.
- The name of the Oracle wallet: cwallet.sso (represented in **bold**).

Use the kubectl create command to create the Secret:



```
% kubectl create secret generic cs-tls
--from-file=csWallet=serverWallet/cwallet.sso
secret/cs-tls created
```

You have successfully created and deployed the cs-tls Kubernetes Secret. The csWallet/ cwallet.sso file will later be available in the /ttconfig directory of the TimesTen containers. In addition, the file will be available in the /tt/home/timesten/csWallet directory of the TimesTen containers.

Create a ConfigMap for the Server-Side Attributes

This section creates the cs-tls ConfigMap. This ConfigMap contains the db.ini, the adminUser, and the schema.sql metadata files.

On your Linux development host:

1. From the directory of your choice, create an empty subdirectory for the metadata files. This example creates the cm_csTLS subdirectory. (The cm_csTLS directory is used in the remainder of this example to denote this directory.)

% mkdir -p cm_csTLS

2. Navigate to the ConfigMap directory.

% cd cm_csTLS

3. Create the db.ini file in this ConfigMap directory (cm_csTLS, in this example). In this db.ini file, define the server-side attributes for TLS for Client/Server. These server-side attributes will later be included in the sys.odbc.ini file located in the timesten_home/ conf directory in your TimesTen containers. (Note that timesten_home is the TimesTen instance directory. This instance directory is tt/home/timesten/instances/instance1 in your TimesTen containers.)

These are the required server-side attributes for TLS for Client/Server:

- wallet: This is the directory in your TimesTen containers that contains the server wallet. Specify /tt/home/timesten/csWallet.
- ciphersuites: This is the cipher suite setting. You can specify more than one value. If you specify more than one value, separate each value by a comma. List the values in order of preference. There is no default value. Values are the following:
 - SSL_ECDHE_ECDSA_WITH_AES_128_GCM_256
 - SSL ECDHE ECDSA WITH AES 256 GCM 384
 - SSL RSA WITH AES 128 CBC SHA256

For TLS to be used, the server and the client settings must include at least one common suite. This example specifies SSL_ECDHE_ECDSA_WITH_AES_128_GCM_256. See Server Attributes for TLS in the Oracle TimesTen In-Memory Database Security Guide.

• encryption: This is the encryption setting for the server. This example specifies the required setting. See Server Attributes for TLS in the *Oracle TimesTen In-Memory Database Security Guide* for information on the valid encryption settings.

This example also specifies the PermSize and the DatabaseCharacterSet connection attributes.

vi db.ini

PermSize=200

ORACLE

```
DatabaseCharacterSet=AL32UTF8
wallet=/tt/home/timesten/csWallet
ciphersuites=SSL_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
encryption=required
```

4. Create the adminUser file in this ConfigMap directory (cm_csTLS, in this example). In this adminUser file, create the sampleuser user with the samplepw password.

vi adminUser

sampleuser/samplepw

5. Create the schema.sql file in this ConfigMap directory (cm_csTLS, in this example). In this schema.sql file, define the s sequence and the emp table for the sampleuser user. The Operator will automatically initialize your database with these object definitions.

```
vi schema.sql
create sequence sampleuser.s;
create table sampleuser.emp (
   id number not null primary key,
   name char(32)
);
```

6. Create the ConfigMap. The files in the cm_csTLS directory are included in the ConfigMap and, later, will be available in the TimesTen containers.

In this example:

- The name of the ConfigMap is cs-tls. Replace cs-tls with a name of your choosing. (cs-tls is represented in **bold** in this example.)
- This example uses cm_csTLS as the directory where the files that will be copied into the ConfigMap reside. If you use a different directory, replace cm_csTLS with the name of your directory. (cm_csTLS is represented in **bold** in this example.)

Use the kubectl create command to create the ConfigMap:

```
% kubectl create configmap cs-tls --from-file=cm_csTLS
configmap/cs-tls created
```

7. Use the kubectl describe command to verify the contents of the ConfigMap. (cs-tls, in this example.)

```
% kubectl describe configmap cs-tls
Name: cs-tls
Namespace: mynamespace
Labels: <none>
Annotations: <none>
Data
====
db.ini:
____
PermSize=200
DatabaseCharacterSet=AL32UTF8
wallet=/tt/home/timesten/csWallet
ciphersuites=SSL_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
encryption=required
schema.sql:
____
create sequence sampleuser.s;
```



create table sampleuser.emp (id number not null primary key, name char (32));

```
adminUser:
----
sampleuser/samplepw
```

Events: <none>

You have successfully created and deployed the cs-tls ConfigMap.

Create a TimesTenClassic Object

This section creates the TimesTenClassic object. For detailed information about the TimesTenClassic object type, see About the TimesTenClassic Object Type.

Perform these steps:

 Create an empty YAML file. You can choose any name, but you may want to use the same name you used for the name of the TimesTenClassic object. (In this example, cstls.) The YAML file contains the definitions for the TimesTenClassic object. See <u>TimesTenClassicSpecSpec</u> for information on the fields that you must specify in this YAML file as well as the fields that are optional.

In this example, the fields of particular interest for TLS Client/Server are:

- dbSecret: This example uses one Kubernetes Secret (called cs-tls) for the csWallet metadata file.
- dbConfigMap: This example uses one ConfigMap (called cs-tls). The db.ini file is contained in the cs-tls ConfigMap. Recall that the db.ini file contains the server-side attributes for TLS for Client/Server.

In addition, this example includes:

- name: Replace cstls with the name of your TimesTenClassic object.
- storageClassName: Replace oci-by with the name of the storage class used to allocate PersistentVolumes to hold TimesTen.
- storageSize: Replace 250Gi with the amount of storage that should be requested for each Pod to hold TimesTen. Note: This example assumes a production environment and uses a value of 250Gi for storageSize. For demonstration purposes, a value of 50Gi is adequate. See the storageSize and the logStorageSize entries in "Table 20-3" for information.
- image: Replace container-registry.oracle.com/timesten/timesten:22.1.1.34.0 with the location and name of your image.
- imagePullSecret: Replace sekret with the image pull secret that Kubernetes should use to fetch the TimesTen image.

```
% vi cstls.yaml
```

```
apiVersion: timesten.oracle.com/v4
kind: TimesTenClassic
metadata:
    name: cstls
spec:
    ttspec:
    storageClassName: oci-bv
    storageSize: 250Gi
    image: container-registry.oracle.com/timesten/timesten:22.1.1.34.0
    imagePullSecret: sekret
```



```
dbConfigMap:
- cs-tls
dbSecret:
- cs-tls
```

2. Use the kubectl create command to create the TimesTenClassic object from the contents of the YAML file (in this example, cstls.yaml). Doing so begins the process of deploying your active standby pair of TimesTen databases in the Kubernetes cluster.

```
% kubectl create -f cstls.yaml
timestenclassic.timesten.oracle.com/cstls created
```

You have successfully created the TimesTenClassic object in the Kubernetes cluster. The process of deploying your TimesTen databases begins, but is not yet complete.

Monitor Deployment of the TimesTenClassic Object

Use the kubectl get and the kubectl describe commands to monitor the progress of the active standby pair as it is provisioned.

1. Use the kubectl get command and review the STATE field. Observe the value is Initializing. The active standby pair provisioning has begun, but is not yet complete.

```
% kubectl get ttc cstls
NAME STATE ACTIVE AGE
cstls Initializing None 15s
```

2. Use the kubectl get command again to see if value of the STATE field has changed. In this example, the value is Normal, indicating the active standby pair of databases are now provisioned and the process is complete.

```
% kubectl get ttc cstls
NAME STATE ACTIVE AGE
cstls Normal cstls-0 3m30s
```

3. Use the kubectl describe command to view the active standby pair provisioning in detail.

Note the following have been correctly set in the cstls TimesTenClassic object definition:

- The cs-tls Secret has been correctly referenced in the dbSecret field (represented in **bold**).
- The cs-tls Configmap has been correctly referenced in the dbConfigMap field (represented in **bold**).

Note: Note all of the output is shown in this example.

```
% kubectl describe ttc cstls
Name:
      cstls
Namespace: mynamespace
Labels:
         <none>
Annotations: <none>
API Version: timesten.oracle.com/v4
Kind:
            TimesTenClassic
Metadata:
 Creation Timestamp: 2025-01-16T19:08:03Z
 Generation:
                     1
 Resource Version: 75491472
 Self Link:
/apis/timesten.oracle.com/v4/namespaces/mynamespace/timestenclassics/cstls
 UID:
                     150128b3-10ac-11eb-b019-d681454a288b
Spec:
 Ttspec:
   Db Config Map:
```



```
cs-tls
   Db Secret:
     cs-tls
                      container-registry.oracle.com/timesten/timesten:22.1.1.34.0
   Image:
   Image Pull Policy: Always
   Image Pull Secret:
                      sekret
   Storage Class Name: oci
   Storage Size:
                      250Gi
. . .
Events:
                Age From
 Type Reason
                                 Message
                  ____
                                   ____
      Create
                 4m21s ttclassic Service cstls created
      Create
                 4m21s ttclassic StatefulSet cstls created
      Create
                 4m21s ttclassic Secret tt150128b3-10ac-11eb-b019-d681454a288b
created
    StateChange 3m5s ttclassic Pod cstls-1 Daemon Up
 _
 _
     StateChange 3m5s ttclassic Pod cstls-0 Agent Up
    StateChange 3m5s ttclassic Pod cstls-0 Release 22.1.1.34.0
 _
    StateChange 3m5s ttclassic Pod cstls-1 Agent Up
  _
    StateChange 3m5s ttclassic Pod cstls-1 Release 22.1.1.34.0
  _
    StateChange 3m5s ttclassic Pod cstls-0 Daemon Up
    StateChange 116s ttclassic Pod cstls-0 Database Loaded
  _
     StateChange 116s ttclassic Pod cstls-0 Database Updatable
     StateChange 116s ttclassic Pod cstls-0 CacheAgent Not Running
  _
      StateChange 116s ttclassic Pod cstls-0 RepAgent Not Running
      StateChange 116s ttclassic Pod cstls-0 RepState IDLE
      StateChange 116s ttclassic Pod cstls-0 RepScheme None
 _
      StateChange 115s ttclassic Pod cstls-0 RepAgent Running
 _
      StateChange 115s ttclassic Pod cstls-0 RepScheme Exists
  _
       StateChange 115s ttclassic Pod cstls-0 RepState ACTIVE
  _
       StateChange 96s ttclassic Pod cstls-1 Database Loaded
 _
      StateChange 96s ttclassic Pod cstls-1 Database Not Updatable
      StateChange 96s ttclassic Pod cstls-1 CacheAgent Not Running
      StateChange 96s ttclassic Pod cstls-1 RepAgent Not Running
 _
      StateChange 96s ttclassic Pod cstls-1 RepScheme Exists
 _
      StateChange 96s ttclassic Pod cstls-1 RepState IDLE
      StateChange 90s ttclassic Pod cstls-1 RepAgent Running
 _
       StateChange 84s ttclassic Pod cstls-1 RepState STANDBY
       StateChange 84s ttclassic TimesTenClassic was Initializing, now Normal
```

Your active standby pair of TimesTen databases are successfully deployed (as indicated by Normal.)

Configuration Requirements for the Client

These sections cover the client requirements for TLS.

- Copy a Client Wallet
- Configure Client-Side Attributes

Copy a Client Wallet

When you used the ttCreateCerts utility to create TLS certificates, the cwallet.sso wallet file located in the /scratch/ttuser/instance_dir/instancel/conf/ clientWallet directory was generated. This file must be copied to the application container that is running your TimesTen client instance. See "Create TLS Certificates for Replication and Client/Server" for information on creating the TLS certificates.



This example uses the kubectl cp command to copy the /scratch/ttuser/instance_dir/ instance1/conf/clientWallet/cwallet.sso file from your Linux development host to the application container running your TimesTen client instance.

1. Use the kubectl exec -it command to invoke the shell in the application container that contains your TimesTen client instance. (cstls-0, in this example).

```
% kubectl exec -it cstls-0 -c tt -- /bin/bash
```

2. From the shell just invoked, and from the directory of your choice, create an empty subdirectory. This example creates the clientWallet subdirectory.

```
% mkdir -p clientWallet
```

3. From your Linux development host, use the kubectl cp command to copy the cwallet.sso file from the /scratch/ttuser/instance_dir/instance1/conf/clientWallet directory on your Linux development host to the clientWallet directory that you just created. (This directory is located in the application container that is running your TimesTen client instance.) Recall that the cwallet.sso file was generated when you used the ttCreateCerts utility to create the TLS certificates. See Create TLS Certificates for Replication and Client/Server for information.

% kubectl cp /scratch/ttuser/instance_dir/instance1/conf/clientWallet/ cwallet.sso cstls-0:clientWallet/cwallet.sso -c tt

4. From your shell, verify the cwallet.sso file is located in the clientWallet directory.

```
% ls clientWallet
cwallet.sso
```

You have successfully copied the cwallet.sso client wallet file to the application container that is running your TimesTen client instance.

Configure Client-Side Attributes

You must set client-side attributes for TLS for Client/Server. The attributes can be set in the client DSN definition in *timesten_home/conf/sys.odbc.ini* or in an appropriate Client/Server connection string. See About Using Client/Server Drivers for additional information.

These are the required client-side attributes for TLS for Client/Server:

- wallet: This is the directory that contains the cwallet.sso client wallet file. This directory
 is located in your application container that is running the TimesTen client instance. There
 is no default directory. In this example, recall that the clientWallet directory was created
 to denote this directory. (See Copy a Client Wallet for information.) For purposes of this
 example, the full path to the clientWallet directory is /tt/home/timesten/clientWallet.
 Therefore, in this example, /tt/home/timesten/clientWallet is used to denote this
 directory.
- ciphersuites: This is the cipher suite setting. You can specify more than one value. If you specify more than one value, separate each value by a comma. List the values in order of preference. There is no default value. Values are the following:
 - SSL_ECDHE_ECDSA_WITH_AES_128_GCM_256
 - SSL_ECDHE_ECDSA_WITH_AES_256_GCM_384
 - SSL_RSA_WITH_AES_128_CBC_SHA256

For TLS to be used, the server and the client settings must include at least one common suite. This example specifies SSL_ECDHE_ECDSA_WITH_AES_128_GCM_256. See Configuration for TLS for Client/Server in the Oracle TimesTen In-Memory Database Security Guide.



 encryption: This is the encryption setting for the client. This example specifies the required setting. See Configuration for TLS for Client/Server in the Oracle TimesTen In-Memory Database Security Guide for information on the valid encryption settings.

This example uses a connection string to connect to the <code>cstsl</code> database as the <code>sampleuser</code> user. The <code>sampleuser</code> user was created by the Operator and already exists in the <code>cstsl</code> database. The example then uses the <code>sqlgetconnectattr</code> command from <code>ttlsqlCS</code> on the client to verify TLS is configured correctly on the Server and on the Client and TLS is being used.

1. Connect to the database.

```
% ttIsqlcs -connstr "TTC SERVER1=cstls-0.cstls.mynamespace.svc.cluster.local;
TTC SERVER2=cstls-1.cstls.mynamespace.svc.cluster.local;
TTC SERVER DSN=cstls;UID=sampleuser;PWD=samplepw;
WALLET=tt/home/timesten/clientWallet;
CIPHERSUITES=SSL ECDHE ECDSA WITH AES 128 GCM SHA256;
ENCRYPTION=required";
Copyright (c) 1996, 2025, Oracle and/or its affiliates. All rights reserved.
Type ? or "help" for help, type "exit" to quit ttIsql.
connect "TTC SERVER1=cstls-0.cstls.mynamespace.svc.cluster.local;
TTC SERVER2=cstls-1.cstls.mynamespace.svc.cluster.local;
TTC SERVER DSN=cstls;UID=sampleuser;PWD=******;
WALLET=tt/home/timesten/clientWallet;
CipherSuites=SSL ECDHE ECDSA WITH AES 128 GCM SHA256;
ENCRYPTION=REOUIRED;";
Connection successful:
DSN=;TTC SERVER=cstls-0.cstls.mynamespace.svc.cluster.local;
TTC SERVER DSN=cstls;UID=sampleuser;
DATASTORE=/tt/home/timesten/datastore/cstls;DATABASECHARACTERSET=AL32UTF8;
CONNECTIONCHARACTERSET=AL32UTF8; AUTOCREATE=0; PERMSIZE=200;
DDLREPLICATIONLEVEL=3; FORCEDISCONNECTENABLED=1; (SERVER) ENCRYPTION=Required;
(SERVER) WALLET=file:/tt/home/timesten/csWallet; (client) Encryption=Required;
(client)Wallet=/tt/home/timesten/clientWallet;
(client)CipherSuites=SSL ECDHE ECDSA WITH AES 128 GCM SHA256;
(Default setting AutoCommit=1)
```

2. Use the sqlgetconnectattr command in ttIsqlCS to verify TLS is being used. A return value of 1 indicates TLS is being used.

```
Command> sqlgetconnectattr tt_tls_session;
TT TLS SESSION = 1 (SQL TRUE)
```

You have successfully connected to the database and verified that TLS for Client/Server is being used.



18

Handle Failover and Recovery in TimesTen Classic

Learn about how the TimesTen Operator handles failover and recovery.

- About Node Failure in Kubernetes
- About Handling Failover and Recovery
- Illustrate the Failover and Recovery Process

About Node Failure in Kubernetes

Learn about node failure in Kubernetes and how the TimesTen Operator detects it and takes appropriate action.

Topics:

- How Kubernetes Reports Node Status
- How the TimesTen Kubernetes Operator Handles Node Failure
- About Specifying the .spec.ttspec.deleteDbOnNotReadyNode Datum
- About Kubernetes Events and TimesTen Operator Metrics

How Kubernetes Reports Node Status

The status of a Node in Kubernetes is critical in managing a Kubernetes cluster. Kubernetes provides details about the status of a Node. In particular, the conditions field of a Node's status provides information about the status of running Nodes. For example, the Ready condition provides information about the readiness and health of a Node.

The values for the Ready condition are as follows:

- True: The Node is healthy and is ready to accept Pods.
- False: The Node is not healthy and is not accepting Pods.
- Unknown: The Node controller has not recently heard from the Node.

You can get the status of a Node by using the kubectl get nodes and kubectl describe node commands. For example:

kubectl get nodesAGEVERSIONNAMESTATUSROLESAGEVERSIONNodeAReadycontrol-plane,etcd,master233dv1.25.16+rke2r2

Kubernetes reports that NodeA is Ready.

```
kubectl describe node nodea
...
Conditions:
```



Type LastTransitionTime 	Status		Heartbea eason 	atTime	e Message -
		_			
 Ready 2025 05:01:08 +0000 status	True KubeletR	,	16 Jan	2025	14:04:26 +0000 Thu, 09 Jan kubelet is posting ready

Kubernetes reports that the Ready condition for NodeA is True and NodeA is accepting Pods.

For more information, see https://kubernetes.io/docs/reference/node/node-status/ in the Kubernetes documentation.

How the TimesTen Kubernetes Operator Handles Node Failure

Kubernetes does a good job of detecting and resolving Node and Pod failure. However, there could be cases where Kubernetes cannot resolve failure. Let's look at an example that could affect TimesTen databases that reside in containers in Pods on one or more Nodes in a Kubernetes cluster. Let's then examine what you can do to have the TimesTen Operator detect this situation and take appropriate action.

In a Kubernetes cluster, assume you use the local volume provisioner to make storage on each Kubernetes Node available as persistent storage for Pods running on the Node. A drawback of this approach is that the storage on one Node is not available to other Nodes. Consider the following scenario:

- There are three nodes in the cluster (Node A, Node B, and Node C).
- TimesTen is running on Node A and Node B.
- Node A goes down and is unavailable.

Kubernetes detects the failure of Node A, but cannot automatically create a new Pod on Node C to run TimesTen. This is because the persistent volumes used by TimesTen are local to Node A and therefore Node C cannot access these persistent volumes. As a result, if Node A is down and unavailable, Kubernetes cannot create a new Pod on Node C. The TimesTen Operator can correctly fail over the database to Node B, but cannot bring up a replacement for Node A. Therefore, there is no redundancy in the cluster until Node A comes back up.

You can configure the TimesTen Operator to detect such a situation and take appropriate action to reconfigure and automatically start TimesTen on Node C.

Here's how:

The .spec.ttspec.deleteDbOnNotReadyNode datum of a TimesTenClassic object allows you to direct the TimesTen Operator to detect situations where a Node is not ready (or unknown) for a specific period of time. In such cases, if the .spec.ttspec.deleteDbOnNotReadyNode is specified, the TimesTen Operator takes appropriate action to remedy the situation. For more information about the .spec.ttspec.deleteDbOnNotReadyNode datum, see the deleteDbOnNotReadyNode entry in Table 20-3.

Let's look at this in further detail.

Approximately every .spec.ttspec.pollingInterval seconds, the TimesTen Operator reconciles each TimesTenClassic object. During this reconciliation, the TimesTen Operator examines the state of each Pod associated with a TimesTenClassic object. In addition, the TimesTen Operator also retrieves the state of the Node on which the Pod is running. If a Pod is



scheduled on a Node that is not ready (or unknown), the TimesTen Operator records the time in the TimesTenClassic object's status.

During the next reconciliation (.spec.ttspec.pollingInterval later), if the Pod is assigned to the same Node and the Node is still not ready (or unknown), then the TimesTen Operator checks to see if the .spec.ttspec.deleteDbOnNotReadyNode is specified. If it is specified, the TimesTen Operator checks to see if the Node's not ready condition has existed for more than .spec.ttspec.deleteDbOnNotReadyNode seconds. If so, the TimesTen Operator deletes the Pod and the PVCs associated with the Pod. This causes Kubernetes to create a new Pod and new PVCs on a surviving Node. Once the Pod is scheduled and started by Kubernetes, the TimesTen Operator configures it as usual.

About Specifying the .spec.ttspec.deleteDbOnNotReadyNode Datum

You can specify the .spec.ttspec.deleteDbOnNotReadyNode datum for both replicated and non-replicated configurations.

Note:

Specifying the .spec.ttspec.deleteDbOnNotReadyNode datum could result in the TimesTen Operator deleting PVCs. Deleting PVCs discards the on-disk copy of TimesTen databases. Use caution when specifying the .spec.ttspec.deleteDbOnNotReadyNode datum.

In a replicated environment, depending on your replication setting, deleting PVCs may not cause data loss. For example, assume the following:

- The sample-0 database is active and the sample-1 database is standby.
- The Node on which the sample-0 database is running fails.
- The TimesTen Operator performs the steps to fail over to sample-1, making it the new active.
- Even if the Node on which sample-0 was running comes back up, the TimesTen Operator duplicates the database on sample-1 back to sample-0. In this case, the contents of the PVCs for sample-0 are not relevant.

Similarly, if sample-0 is the standby, when sample-0 comes back up, the TimesTen Operator duplicates the database from sample-1. In this case, the database in the PVC on the failed node is not relevant.

In a non-replicated environment, there may be data loss. For example, there may be data in a database that is not in any other replica database. In this case, if the TimesTen Operator deletes PVCs associated with the database, there could be data loss.

In non-replicated environments where TimesTen is used as a read-only cache, specifying the .spec.ttspec.deleteDbOnNotReadyNode datum may be beneficial. If specified, the TimesTen Operator can re-provision the number of replicas, even if one Node fails.

About Kubernetes Events and TimesTen Operator Metrics

In these situations, the TimesTen Operator generates Kubernetes Events and TimesTen Operator metrics.

The Kubernetes Events are as follows:



- If the TimesTen Operator cannot retrieve one or more Nodes, it generates an appropriate Event, and continues as though .spec.ttspec.deleteDbOnNotReadyNode was not specified.
- If the TimesTen Operator cannot delete Pods or PVCs, it generates an appropriate Event.

The TimesTen Operator metrics are as follows:

- timesten_pods_deleted_unschedulable
- timesten pods unschedulable
- timesten_pvcs_deleted_unschedulable

For more information about these metrics, see TimesTen Kubernetes Operator Metrics.

About Handling Failover and Recovery

The Operator automatically detects failures of the active TimesTen database and the standby TimesTen database and works to fix any failures. When the Operator detects a failure of the active database, it promotes the standby TimesTen database to be the active. Client/server applications that are using the database are automatically reconnected to the new active database. Transactions in flight are rolled back. Prepared statements need to be re-prepared by the applications. The Operator will configure a new standby database.

Illustrate the Failover and Recovery Process

This example simulates a failure of the active TimesTen database. This is for demonstration purposes only. Do not do this in a production environment.

Use the kubectl delete pod command to delete the active database (sample-0 in this case)

% kubectl delete pod sample-0

2. Use the kubectl describe command to observe how the Operator recovers from the failure. The Operator promotes the standby database (sample-1) to be active. Any applications that were connected to the sample-0 database are automatically reconnected to the sample-1 database by TimesTen. After a brief outage, the applications can continue to use the database. See "About the High Level State of TimesTenClassic Objects" for information on the health and states of the active standby pair.

Note: In this example, the text for the Message column displays on two lines for three state changes. However, the actual output displays on one line for each of these three state changes.

```
% kubectl describe ttc sample
Name: sample
. . .
Events:
 Type Reason Age From Message
      _____
                  ----
                        ____
                                  _____
      StateChange 2mls ttclassic TimesTenClassic sample: was Normal, now
ActiveDown
 - StateChange 115s ttclassic Pod sample-1 Database Updatable: Yes
      StateChange 115s ttclassic TimesTenClassic sample:was ActiveDown, now
StandbyDown
 - StateChange 115s ttclassic Pod sample-1 RepState ACTIVE
 _
    StateChange 110s ttclassic Pod sample-0 High Level State Unknown
    StateChange 63s ttclassic Pod sample-0 Pod Phase Running
```



-	StateChange	63s	ttclassic	Pod sample-0 Agent Up
-	StateChange	63s	ttclassic	Pod sample-0 Instance Exists
-	StateChange	63s	ttclassic	Pod sample-0 Daemon Up
-	StateChange	63s	ttclassic	Pod sample-0 Database None
-	StateChange	42s	ttclassic	Pod sample-0 Database Loaded
-	StateChange	42s	ttclassic	Pod sample-0 Database Updatable: No
-	StateChange	42s	ttclassic	Pod sample-0 RepAgent Running
-	StateChange	42s	ttclassic	Pod sample-0 CacheAgent Not Running
-	StateChange	42s	ttclassic	Pod sample-0 RepScheme Exists
-	StateChange	42s	ttclassic	Pod sample-0 RepState IDLE
-	StateChange	36s	ttclassic	Pod sample-0 High Level State Healthy
-	StateChange	36s	ttclassic	Pod sample-0 RepState STANDBY
-	StateChange	36s	ttclassic	TimesTenClassic sample:was StandbyDown,now
Normal				

Kubernetes has automatically respawned a new sample-0 Pod to replace the Pod you deleted. The Operator configured TimesTen within that Pod, bringing the database in the Pod up as the new standby database. The replicated pair of databases are once again functioning normally.



19 Perform Upgrades

The TimesTen Kubernetes Operator provides support for upgrading TimesTen CRDs, the TimesTen Operator, and TimesTen Classic databases to a new patch or patch set.

The TimesTen Operator provides YAML manifest files and Helm charts for upgrading TimesTen CRDs and the TimesTen Operator. You obtain these files from TimesTen container images.

The TimesTen Operator also provides support for upgrading replicated and non-replicated TimesTen Classic databases that are deployed in your Kubernetes namespace. For each replicated or non-replicated configuration, you have the option of choosing a manual or automated upgrade strategy. The processes and procedures differ depending on your configuration and upgrade strategy.

This chapter discusses and details the upgrade processes and procedures using YAML manifest files. If you are using Helm charts, see Use Helm in Your TimesTen Kubernetes Operator Environment.

Topics include:

- About Obtaining Container Images for the Upgrade
- About Upgrading from Previous Releases
- Upgrade the TimesTen CRDs
- About Upgrading the TimesTen Operator
- Upgrade the TimesTen Operator
- About Upgrading TimesTen Classic Databases
- Perform an Automated Upgrade of a Replicated TimesTenClassic Object
- Perform a Manual Upgrade of a Replicated TimesTenClassic Object
- Perform an Automated Upgrade of a Non-Replicated TimesTenClassic Object
- Perform a Manual Upgrade of a Non-Replicated TimesTenClassic Object
- About Upgrading Direct Mode Applications
- About Failures During an Upgrade

About Obtaining Container Images for the Upgrade

The TimesTen Operator provides several options for obtaining container images that contain the release of TimesTen that you want to use for the upgrade. These container images contain the YAML manifest files and Helm charts that are necessary to upgrade the TimesTen CRDs and the TimesTen Operator. For detailed information about your options, see Prepare to Use the TimesTen Kubernetes Operator.

The examples in this chapter assume you have made the following decisions:

Container image: The container image that contains the new release of TimesTen is container-registry.oracle.com/timesten/timesten:22.1.1.35.0.


Download directories: The directories on your development host that contain the YAML manifest files and Helm charts for the upgrade are new_kube_files/deploy and new kube files/helm respectfully.

About Upgrading from Previous Releases

In TimesTen releases 22.1.1.27.0 and greater, the TimesTenClassic Custom Resouce Definition (CRD) defines different schema versions. The TimesTen Operator supports the creation, monitoring, and management of TimesTenClassic objects in these schema versions. For more information about TimesTen CRDs, see About TimesTen CRDs.

In TimesTen releases 22.1.1.27.0 and greater, note the following:

- You can create TimesTenClassic objects in different schema versions. However, to use attributes specific to the latest release of the TimesTen Operator, you must define your object with the v4 schema definitions. For information about these attributes, see TimesTenClassicSpecSpec.
- Kubernetes uses the v4 schema version as the default version. If you use the kubectl get command to fetch a TimesTenClassic object, Kubernetes returns the object in v4 format, unless you explicitly ask for a different format.
- Not only does Kubernetes support multiple versions of a CRD simultaneously, but it can serve an object in a schema version that is different than the one in which it was created. A TimesTenClassic object that you created with one schema version can be fetched with another schema version. For example, you can create a v4 TimesTenClassic object and can fetch it as a different version of the object.
- Kubernetes stores newly created objects using one schema version. Kubernetes stores TimesTenClassic objects in the v4 schema.
- Use the following syntax to define v4 TimesTenClassic objects:

```
apiVersion: timesten.oracle.com/v4
kind: TimesTenClassic
```

Using timesten.oracle.com/v4 for apiVersion instructs Kubernetes to create a v4 object. For more information about apiVersion, see TimesTenClassic.

Upgrade the TimesTen CRDs

You can upgrade TimesTen CRDs that are installed in your Kubernetes cluster. In TimesTen releases 22.1.1.27.0 and greater, the TimesTenClassic CRD contains different schema versions. The v4 schema version is the most current. The v3 and v2 schema versions are available to facilitate compatibility with previous releases of the TimesTen Operator. The v2 schema version is deprecated and will be removed in a future release. For more information, see About Upgrading from Previous Releases.

Note the following:

- You cannot downgrade TimesTen CRDs.
- Since CRDs are cluster-scoped, if you delete TimesTen CRDs from your Kubernetes cluster, all TimesTenClassic objects that are deployed in this cluster are also deleted.

This example uses YAML manifest files for the upgrade. To use Helm charts, see Use Helm in Your TimesTen Kubernetes Operator Environment.



1. On your development host, change to the directory that contains the YAML manifest files for the upgrade. In this example, change to the new_kube_files/deploy directory.

cd new kube files/deploy

2. Replace the TimesTen CRDs that are installed in your Kubernetes cluster.

kubectl replace -f crd.yaml

The output is the following:

```
customresourcedefinition.apiextensions.k8s.io/
timestenclassics.timesten.oracle.com replaced
customresourcedefinition.apiextensions.k8s.io/
timestenscaleouts.timesten.oracle.com replaced
```

Congratulations! You successfully upgraded the TimesTen CRDs.

About Upgrading the TimesTen Operator

Before you begin the upgrade of the TimesTen Operator, ensure you have obtained either the YAML manifest files or the Helm charts necessary for the upgrade. See About Obtaining Container Images for the Upgrade. If you are using Helm charts for the upgrade, see Overview of Helm and TimesTen Helm Charts.

The upgrade process differs depending on whether you are upgrading the TimesTen Operator in one or more namespaces at namespace-scope or in the timesten-operator namespace at cluster-scope. You must upgrade the TimesTen Operator in the same scope as the scope in which you installed the TimesTen Operator. For more information about the TimesTen Operator scopes, see About the TimesTen Operator.

The following sections show you how to perform the upgrade in both scopes using YAML manifest files. The examples assume the files are located in the <code>new_kube_files/deploy</code> directory. See About Obtaining Container Images for the Upgrade.

To upgrade the TimesTen Operator in one or more namespaces in your Kubernetes cluster at namespace-scope, use the following files:

• service_account.yaml: Upgrades the service account, role, and role binding objects associated with the TimesTen Operator.

To perform the upgrade, you use kubectl replace to replace the file in your namespace.

• service_account_cluster.yaml: Upgrades additional privileges and permissions for the TimesTen Operator. If you installed this file, you must upgrade it. See About Installing the TimesTen Operator.

You must modify this file and then upgrade it by using kubectl replace. The modifications you need to make are explained later in the examples.

• operator.yaml: Upgrades the TimesTen Operator in a namespace in your Kubernetes cluster at namespace-scope.

You must modify this file and then upgrade it by using kubectl replace. The modifications you need to make are explained later in the examples.

To upgrade the TimesTen Operator in the timesten-operator namespace in your Kubernetes cluster at cluster-scope, use the following files:



 cluster_config.yaml: Upgrades the timesten-operator namespace as well as the ServiceAccount, Role, RoleBinding, ClusterRole, and ClusterRoleBinding objects necessary to run the TimesTen Operator in the timesten-operator namespace at clusterscope.

To perform the upgrade, you use kubectl replace to replace the file.

• cluster_operator.yaml: Upgrades the TimesTen Operator in the timesten-operator namespace in your cluster at cluster-scope.

You must modify this file and then upgrade it by using <code>kubectl replace</code>. The modifications you need to make are explained later in the examples.

Upgrade the TimesTen Operator

Let's walk through some examples that show you how to upgrade the TimesTen Operator in two namespaces in a Kubernetes cluster at namespace-scope and in the timesten-operator namespace in a Kubernetes cluster at cluster-scope.

- Upgrade the TimesTen Operator at Namespace-Scope
- Upgrade the TimesTen Operator at Cluster-Scope

Upgrade the TimesTen Operator at Namespace-Scope

The examples in this section assume you have obtained the YAML manifest files for the upgrade. See About Obtaining Container Images for the Upgrade. If you are using Helm charts for the upgrade, see Overview of Helm and TimesTen Helm Charts.

There is one example for upgrading the TimesTen Operator in one namespace in your cluster at namespace-scope. There is a second example for upgrading the TimesTen Operator into a second namespace in your cluster at namespace-scope. The examples illustrate that you must upgrade the TimesTen Operator in each namespace. Once upgraded, the TimesTen Operator that you upgraded in namespace one services TimesTenClassic objects in namespace one while the TimesTen Operator that you upgraded in namespace two services TimesTenClassic objects in namespace two.

To complete the upgrade, perform the steps in the following sections:

- Before You Begin
- Upgrade in Namespace One
- Upgrade in Namespace Two

Before You Begin

Let's confirm the namespaces in which the TimesTen Operator is running.

1. Confirm the namespaces.

kubectl get namespaces

The output is similar to the following:

kubectl get namespaces NAME STATUS AGE



mynamespace Active 15d mynamespace2 Active 10d

2. Determine the namespaces in which the TimesTen Operator is running.

```
kubectl get pods -l name=timesten-operator -A
```

The output is similar to the following:

NAMESPACENAMEREADYSTATUSRESTARTSAGEmynamespacetimesten-operator-577f7fbc6f-f75881/1Running010d010d1/1Running

There are two TimesTen Operators. One Operator is running in the mynamespace namespace and a second Operator is running in the mynamespace2 namespace. Each TimesTen Operator is running at namespace-scope.

Upgrade in Namespace One

Let's upgrade the TimesTen Operator in the mynamespace namespace at namespace-scope.

1. Switch to namespace one (mynamespace, in this example).

kubectl config set-context --current --namespace=mynamespace

The output is similar to the following:

Context "default" modified.

2. Confirm a TimesTen Operator is running in this namespace.

kubectl get pods

The output is similar to the following:

```
NAME READY
STATUS RESTARTS AGE
...
timesten-operator-577f7fbc6f-tbr7m 1/1
Running 0 10d
...
```

 Change to the directory that contains the YAML manifest files for the upgrade. In this example, the new_kube_files/deploy directory contains the files.

cd new_kube_files/deploy



4. Upgrade the service account, role, and role binding objects.

```
kubectl replace -f service_account.yaml
```

The output is similar to the following:

```
role.rbac.authorization.k8s.io/timesten-operator replaced
serviceaccount/timesten-operator replaced
rolebinding.rbac.authorization.k8s.io/timesten-operator replaced
```

 Make a copy of the service_account_cluster.yaml file for the first namespace (service account cluster n1.yaml, in this example).

cp service account cluster.yaml service account cluster n1.yaml

- Replace the service account cluster n1.yaml YAML file by doing the following:
 - a. (Optional): Display the contents of the service account cluster n1.yaml file.

cat service account cluster n1.yaml

```
# Copyright (c) 2025, Oracle and/or its affiliates.
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
 name: timesten-operator
  # If running multiple operators on the same cluster:
  #name: timesten-operator-<NAMESPACE>
rules:
- apiGroups:
  _ ""
 resources:
  - nodes
 verbs:
  - get
 - list
  - watch
- apiGroups:
  _ ""
  resources:
  - persistentvolumeclaims
  verbs:
  - get
  - list
 - watch
  - delete
___
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: timesten-operator
  # If running multiple operators on the same cluster:
  #name: timesten-operator-<NAMESPACE>
```



```
subjects:
- kind: ServiceAccount
  name: timesten-operator
  #namespace: <NAMESPACE>
roleRef:
  kind: ClusterRole
  name: timesten-operator
  # If running multiple operators on the same cluster:
  #name: timesten-operator-<NAMESPACE>
  apiGroup: rbac.authorization.k8s.io
```

b. Use a text editor to modify the service account cluster n1.yaml file.

Make the following changes:

- Locate #namespace, remove #, and replace <NAMESPACE> with the name of your namespace (mynamespace, in this example).
- Locate the three occurrences of #name, remove #, and replace <NAMESPACE> with the name of your namespace (mynamespace, in this example).

```
vi service account cluster n1.yaml
# Copyright (c) 2025, Oracle and/or its affiliates.
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
 name: timesten-operator
  # If running multiple operators on the same cluster:
 name: timesten-operator-mynamespace
rules:
- apiGroups:
  _ ""
 resources:
  - nodes
  verbs:
  - get
 - list
  - watch
- apiGroups:
  _ ""
 resources:
  - persistentvolumeclaims
 verbs:
 - get
  - list
  - watch
  - delete
___
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
 name: timesten-operator
  # If running multiple operators on the same cluster:
 name: timesten-operator-mynamespace
subjects:
- kind: ServiceAccount
```

```
name: timesten-operator
namespace: mynamespace
roleRef:
kind: ClusterRole
name: timesten-operator
# If running multiple operators on the same cluster:
name: timesten-operator-mynamespace
apiGroup: rbac.authorization.k8s.io
```

- c. Save and close the service_account_cluster_n1.yaml file.
- d. Replace the service account cluster n1.yaml file.

kubectl replace -f service account cluster n1.yaml

The output is similar to the following:

```
clusterrole.rbac.authorization.k8s.io/timesten-operator-mynamespace
replaced
clusterrolebinding.rbac.authorization.k8s.io/timesten-operator-
mynamespace replaced
```

- 7. Modify the operator.yaml file by doing the following:
 - a. Use a text editor to modify the operator.yaml file.

Replace the following:

- image: Replace container-registry.oracle.com/timesten/timesten:latest with the name of the image for the upgrade. In this example, the name of the image for the upgrade is container-registry.oracle.com/timesten/timesten:22.1.1.35.0.
- If you are running in a multi-architecture environment, modify the affinity section, using the same setting you used during installation. This example uses amd64.

```
vi operator.yaml
```

```
# Copyright (c) 2019 - 2025, Oracle and/or its affiliates.
apiVersion: apps/v1
kind: Deployment
metadata:
 name: timesten-operator
spec:
 replicas: 1
. . .
    spec:
      serviceAccountName: timesten-operator
      imagePullSecrets:
      - name: sekret
      containers:
        - name: timesten-operator
          image: container-registry.oracle.com/timesten/
timesten:22.1.1.35.0
. . .
# An example affinity definition; this pod will only be assigned to a
```

```
node
       # running on amd64 (the default)
       #
             affinity:
               nodeAffinity:
                 requiredDuringSchedulingIgnoredDuringExecution:
                    nodeSelectorTerms:
                      - matchExpressions:
                        - key: "kubernetes.io/arch"
                          operator: In
                          values: ["amd64"]
   b. Save and close the operator.yaml file.
  Upgrade the TimesTen Operator.
8.
   kubectl replace -f operator.yaml
   The output is the following:
   deployment.apps/timesten-operator replaced
9. Verify the TimesTen Operator is running.
   kubectl get pods
   The output is similar to the following:
   NAME
                                                                 READY
   STATUS RESTARTS
                         AGE
   . . .
                                                                  1/1
   timesten-operator-7fb9bddf-xrc8h
   Running 0
                         41s
   . . .
```

10. (Optional) Verify the TimesTen Operator is running the new image.

kubectl describe deployment timesten-operator | grep Image

The output is the following:

Image: container-registry.oracle.com/timesten/timesten:22.1.1.35.0

Congratulations! You successfully upgraded the TimesTen Operator in the mynamespace namespace at namespace-scope. The Operator is using the container-registry.oracle.com/timesten/timesten:22.1.1.35.0 image and is automatically managing any existing TimesTenClassic objects in this namespace.

If you are upgrading the TimesTen Operator in a second namespace at namespace-scope, proceed to Upgrade in Namespace Two. If you are ready to upgrade TimesTenClassic objects in the mynamespace namespace, proceed to About Upgrading TimesTen Classic Databases.



Upgrade in Namespace Two

Let's upgrade the TimesTen Operator in the mynamespace2 namespace at namespace-scope.

1. Switch to namespace two (mynamespace2, in this example).

kubectl config set-context --current --namespace=mynamespace2

The output is similar to the following:

Context "default" modified.

2. Confirm a TimesTen Operator is running in this namespace.

kubectl get pods

The output is similar to the following:

```
NAME READY STATUS RESTARTS AGE
...
timesten-operator-577f7fbc6f-h8hj8 1/1 Running 0 12d
...
```

3. Change to the directory that contains the YAML manifest files for the upgrade. In this example, the new_kube_files/deploy directory contains the files.

cd new_kube_files/deploy

4. Replace the required service account, role, and role binding.

kubectl replace -f service account.yaml

The output is similar to the following:

```
role.rbac.authorization.k8s.io/timesten-operator replaced
serviceaccount/timesten-operator replaced
rolebinding.rbac.authorization.k8s.io/timesten-operator replaced
```

 Make a copy of the service_account_cluster.yaml file for the namespace (service_account_cluster_n2.yaml, in this example).

cp service account cluster.yaml service account cluster n2.yaml

- 6. Replace the service account cluster n2.yaml YAML file by doing the following:
 - a. (Optional): Display the contents of the service account cluster n2.yaml file.

cat service_account_cluster_n2.yaml



The output is similar to the following:

```
# Copyright (c) 2025, Oracle and/or its affiliates.
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
 name: timesten-operator
  # If running multiple operators on the same cluster:
  #name: timesten-operator-<NAMESPACE>
rules:
- apiGroups:
  _ ""
 resources:
  - nodes
 verbs:
  - get
  - list
  - watch
- apiGroups:
  _ ""
 resources:
  - persistentvolumeclaims
 verbs:
 - get
  - list
  - watch
  - delete
___
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
 name: timesten-operator
  # If running multiple operators on the same cluster:
  #name: timesten-operator-<NAMESPACE>
subjects:
- kind: ServiceAccount
 name: timesten-operator
 #namespace: <NAMESPACE>
roleRef:
  kind: ClusterRole
  name: timesten-operator
  # If running multiple operators on the same cluster:
  #name: timesten-operator-<NAMESPACE>
  apiGroup: rbac.authorization.k8s.io
```

b. Use a text editor to modify the service_account_cluster_n2.yaml file.

Make the following changes:

- Locate #namespace, remove #, and replace <NAMESPACE> with the name of your namespace (mynamespace2, in this example).
- Locate the three occurrences of #name, remove #, and replace <NAMESPACE> with the name of your namespace (mynamespace2, in this example).

vi service account cluster n2.yaml



```
# Copyright (c) 2025, Oracle and/or its affiliates.
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: timesten-operator
  # If running multiple operators on the same cluster:
 name: timesten-operator-mynamespace2
rules:
- apiGroups:
  _ ""
  resources:
  - nodes
 verbs:
  - get
  - list
  - watch
- apiGroups:
  _ ""
  resources:
  - persistentvolumeclaims
 verbs:
  - get
  - list
  - watch
  - delete
___
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
 name: timesten-operator
  # If running multiple operators on the same cluster:
 name: timesten-operator-mynamespace2
subjects:
- kind: ServiceAccount
  name: timesten-operator
 namespace: mynamespace2
roleRef:
  kind: ClusterRole
 name: timesten-operator
  # If running multiple operators on the same cluster:
 name: timesten-operator-mynamespace2
  apiGroup: rbac.authorization.k8s.io
```

- c. Save and close the service_account_cluster_n2.yaml file.
- d. Replace the service_account_cluster_n2.yaml file.

kubectl replace -f service account cluster n2.yaml

```
clusterrole.rbac.authorization.k8s.io/timesten-operator-mynamespace2
replaced
clusterrolebinding.rbac.authorization.k8s.io/timesten-operator-
mynamespace2 replaced
```



- 7. Modify the operator.yaml file by doing the following:
 - a. Use a text editor to modify the operator.yaml file.

Replace the following:

- image: Replace container-registry.oracle.com/timesten/timesten:latest with the name of the image for the upgrade. In this example, the name of the image for the upgrade is container-registry.oracle.com/timesten/timesten:22.1.1.35.0.
- If you are running in a multi-architecture environment, modify the <code>affinity</code> section, using the same setting you used during installation. This example uses <code>amd64</code>.

```
vi operator.yaml
# Copyright (c) 2019 - 2025, Oracle and/or its affiliates.
apiVersion: apps/v1
kind: Deployment
metadata:
 name: timesten-operator
spec:
  replicas: 1
. . .
    spec:
     serviceAccountName: timesten-operator
     imagePullSecrets:
      - name: sekret
      containers:
        - name: timesten-operator
          image: container-registry.oracle.com/timesten/
timesten:22.1.1.35.0
. . .
# An example affinity definition; this pod will only be assigned to a
node
# running on amd64 (the default)
#
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
              - matchExpressions:
                - key: "kubernetes.io/arch"
                  operator: In
                  values: ["amd64"]
```

- b. Save and close the operator.yaml file.
- 8. Upgrade the TimesTen Operator.

kubectl replace -f operator.yaml

The output is the following:

deployment.apps/timesten-operator replaced



9. Verify the TimesTen Operator is running.

kubectl get pods

The output is similar to the following:

NAME	READY	STATUS	RESTARTS	AGE
 timesten-operator-c4b99bd8f-smjlm	1/1	Running	0	43s

10. (Optional) Verify the TimesTen Operator is running the new image.

kubectl describe deployment timesten-operator | grep Image

The output is the following:

Image: container-registry.oracle.com/timesten/timesten:22.1.1.35.0

Congratulations! You successfully upgraded the TimesTen Operator in the mynamespace2 namespace at namespace-scope. The Operator is using the containerregistry.oracle.com/timesten/timesten:22.1.1.35.0 image and is automatically managing any existing TimesTenClassic objects in this namespace.

To upgrade TimesTenClassic objects in the mynamespace2 namespace, proceed to About Upgrading TimesTen Classic Databases.

Upgrade the TimesTen Operator at Cluster-Scope

The examples in this section assume you have obtained the YAML manifest files for the upgrade. See About Obtaining Container Images for the Upgrade. If you are using Helm charts for the upgrade, see Overview of Helm and TimesTen Helm Charts.

Let's walk through the steps to upgrade the TimesTen Operator at cluster-scope.

- 1. Confirm the TimesTen Operator is running at cluster-scope.
 - a. Confirm there is a TimesTen Operator running in the timesten-operator namespace.

```
kubectl get pods -l name=timesten-operator -A
```

The output is similar to the following:

NAMESPACE		NAME	READY
STATUS	RESTARTS	AGE	
timesten-	operator	timesten-operator-6bf76dd84b-c8j59	1/1
Running	0	2d4h	

There is a TimesTen Operator running in the timesten-operator namespace.



b. Verify the TimesTen Operator Deployment has the TT_OPERATOR_SCOPE environment variable set to cluster.

```
kubectl describe deployment timesten-operator -n timesten-operator |
grep TT OPERATOR SCOPE
```

The output is the following:

TT OPERATOR SCOPE: cluster

 Change to the directory that contains the YAML manifest files for the upgrade. In this example, the new kube files/deploy directory contains the files.

cd new kube files/deploy

3. Upgrade the service account, cluster role, and cluster rolebinding objects.

kubectl replace -f cluster config.yaml

The output is similar to the following:

```
namespace/timesten-operator replaced
clusterrole.rbac.authorization.k8s.io/timesten-operator replaced
serviceaccount/timesten-operator replaced
clusterrolebinding.rbac.authorization.k8s.io/timesten-operator replaced
```

- 4. Modify the cluster operator.yaml file by doing the following:
 - a. Use a text editor to modify the cluster operator.yaml file.

Replace the following:

- image: Replace container-registry.oracle.com/timesten/timesten:latest with the name of the image for the upgrade. In this example, the name of the image for the upgrade is container-registry.oracle.com/timesten/timesten:22.1.1.35.0.
- If you are running in a multi-architecture environment, modify the affinity section, using the same setting you used during installation. This example uses amd64.

```
vi cluster operator.yaml
```

```
# Copyright (c) 2019 - 2025, Oracle and/or its affiliates.
apiVersion: apps/v1
kind: Deployment
metadata:
    name: timesten-operator
    spec:
        replicas: 1
...
        spec:
            serviceAccountName: timesten-operator
            imagePullSecrets:
            - name: sekret
            containers:
```

```
- name: timesten-operator
          image: container-registry.oracle.com/timesten/
timesten:22.1.1.35.0
. . .
# An example affinity definition; this pod will only be assigned to a
node
# running on amd64 (the default)
#
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
              - matchExpressions:
                - key: "kubernetes.io/arch"
                  operator: In
                  values: ["amd64"]
```

- b. Save and close the cluster operator.yaml file.
- 5. Upgrade the TimesTen Operator.

kubectl replace -f cluster operator.yaml

The output is the following:

deployment.apps/timesten-operator replaced

6. Verify the TimesTen Operator is running in the timesten-operator namespace.

kubectl get pods -n timesten-operator

The output is similar to the following:

NAME	READY	STATUS	RESTARTS	AGE
timesten-operator-5cf4b68d58-5cn6t	1/1	Running	0	54s

7. (Optional) Verify the TimesTen Operator is running the new image.

kubectl describe deployment timesten-operator -n timesten-operator| grep
Image

The output is the following:

Image: container-registry.oracle.com/timesten/timesten:22.1.1.35.0

Congratulations! You successfully upgraded the TimesTen Operator. The Operator is running in the timesten-operator namespace in your Kubernetes cluster at cluster-scope. The Operator is using the container-registry.oracle.com/timesten/timesten:22.1.1.35.0 image.

If you are ready to upgrade TimesTenClassic objects, proceed to About Upgrading TimesTen Classic Databases.



About Upgrading TimesTen Classic Databases

The TimesTen Operator provides support for upgrading TimesTen Classic databases that are deployed in your Kubernetes namespace. The support is applicable to both replicated and non-replicated configurations. For each configuration, the TimesTen Operator supports an automated and a manual image upgrade strategy. The TimesTen Operator determines the upgrade strategy from the .spec.ttspec.imageUpgradeStrategy datum. If the value is Auto (or not specified), the upgrade process is automated. If the value is Manual, the upgrade process is manual. You have the option of specifying

the .spec.ttspec.imageUpgradeStrategy datum when you define the attributes for your TimesTenClassic object. See About Defining TimesTenClassic Objects.

The upgrade process is dependent on the configuration and the image upgrade strategy. The upgrade processes are discussed in the following sections:

- About the Upgrade Process for Replicated Configurations
- About the Upgrade Process for Non-Replicated Configurations

About the Upgrade Process for Replicated Configurations

For replicated configurations where the image upgrade strategy is:

- Auto: The upgrade process is automated and occurs immediately. Kubernetes takes no explicit action to terminate Pods as part of the upgrade process. Instead, the TimesTen Operator automatically terminates Pods in an orchestrated manner to perform the upgrade.
- Manual: The upgrade process is manual. Neither Kubernetes nor the TimesTen Operator takes immediate action. You are responsible for initiating the upgrade.



Note:

The following actions occur during both an automated and a manual upgrade:

- The standby is terminated first. It takes some time for the standby to come back up. When the standby comes back up, it is upgraded to the new release. During the upgrade of the standby, depending on your replication configuration, there may be disruption on the active database.
- Next, the failover from the active to the standby occurs:
 - The active is terminated. It takes some time for the former active to come back up. When the active comes back up, it is upgraded to the new release.
 - The standby database is promoted to the active and the former active becomes the standby.

If you are using AWT cache groups, the standby is normally responsible for transmitting committed transactions from TimesTen to the Oracle Database. While the standby is being upgraded, the active takes on this responsibility. This may increase the load on the active. In addition, part of the upgrade process involves copying the database from the active to the standby. This also increases the workload on the active. These increases may temporarily reduce the performance of the active database.

For more information about how TimesTen performs an upgrade of an active standby pair of TimesTen databases, see Performing an Upgrade with Active Standby Pair Replication in the Oracle TimesTen In-Memory Database Installation, Migration, and Upgrade Guide.

Ensure you perform an upgrade at the appropriate time. We recommend that you do not perform upgrades at the busiest time of a production day. Applications may experience short outages and perhaps reduced performance as a result of the upgrade procedure.

There are examples later in the chapter that show you how to perform an automated and a manual upgrade for replicated TimesTenClassic objects. The examples are in the following sections:

- Perform an Automated Upgrade of a Replicated TimesTenClassic Object
- Perform a Manual Upgrade of a Replicated TimesTenClassic Object

About the Upgrade Process for Non-Replicated Configurations

For non-replicated configurations where the image upgrade strategy is:

• Auto: The upgrade process is automated. The TimesTen Operator does not take immediate action. Rather, Kubernetes takes action and automatically begins terminating Pods and replacing them with new ones. The new Pods run the upgraded TimesTen image.

The number of Pods that Kubernetes terminates and replaces is dependent on the value of the <code>.spec.ttspec.rollingUpdatePartition</code> datum. In particular, Kubernetes upgrades Pods whose ordinal value is greater than or equal to the value

of .spec.ttspec.rollingUpdatePartition. Pods whose ordinal value is less than the value of .spec.ttspec.rollingUpdatePartition are not upgraded. For example, if you have three non-replicated Pods (replicas = 3 where the ordinal value of the Pods are -0,

-1, and -2) and the value of .spec.ttspec.rollingUpdatePartition is 2, the -2 Pod is upgraded, but the -1 and -0 Pods are not. The default for .spec.ttspec.rollingUpdatePartition is 0.

You have the option of specifying the .spec.ttspec.rollingUpdatePartition datum when you define the attributes for your TimesTenClassic object. If you do not specify the datum, the default is 0. See About Defining TimesTenClassic Objects.

You also have the option of changing the value of

the .spec.ttspec.rollingUpdatePartition datum as part of the upgrade procedure. There is an example later in the chapter that illustrates how to do this.

• Manual: The upgrade process is manual. Neither Kubernetes nor the TimesTen Operator takes immediate action. You are responsible for initiating the upgrade by deleting one or more Pods. For manual upgrades, the .spec.ttspec.rollingUpdatePartition datum is ignored.

There are two examples later in the chapter that show you how to perform an automated and a manual upgrade for non-replicated TimesTenClassic objects. The examples are in the following sections:

- Perform an Automated Upgrade of a Non-Replicated TimesTenClassic Object
- Perform a Manual Upgrade of a Non-Replicated TimesTenClassic Object

Perform an Automated Upgrade of a Replicated TimesTenClassic Object

Before starting the automated upgrade, note the following:

Note:

An automated upgrade happens immediately. The TimesTen Operator takes down your databases, restarts them, and then initiates and completes the fail over process. Do not perform this procedure at the busiest time of your production day. Applications see short outages and perhaps reduced performance as a result of the upgrade procedure.

Let's perform an automated upgrade for a replicated TimesTenClassic object.

1. Review the TimesTenClassic objects running in your namespace.

kubectl get ttc

NAME	STATE	ACTIVE	AGE
mannorepsample	AllReplicasReady	N/A	30h
manrepsample	Normal	manrepsample-0	54m
norepsample	AllReplicasReady	N/A	21h
repsample	Normal	repsample-0	50m



There are several TimesTenClassic objects running in your namespace. One of the replicated TimesTenClassic objects is repsample. Its high level state is Normal indicating TimesTen databases in the active standby pair are up and running and functioning properly.

2. Review the image upgrade strategy for the object.

kubectl get ttc repsample -o yaml | grep imageUpgradeStrategy

The output is the following:

imageUpgradeStrategy: Auto

The image upgrade strategy is Auto (also the default), indicating an automated upgrade strategy.

- 3. On your development host, edit the TimesTenClassic object's .spec.ttspec.image datum with the container image you want to use for the upgrade. This example uses container-registry.oracle.com/timesten/timesten:22.1.1.35.0.
 - a. Edit the file, replacing the image value with container-registry.oracle.com/ timesten/timesten:22.1.1.35.0.

kubectl edit ttc repsample

```
# Please edit the object below. Lines beginning with a '#' will be
ignored,
# and an empty file will abort the edit. If an error occurs while
saving this file will be
# reopened with the relevant failures.
#
apiVersion: timesten.oracle.com/v4
kind: TimesTenClassic
...
image: container-registry.oracle.com/timesten/timesten:22.1.1.35.0
...
```

- b. Save the file and exit from the editor.
- 4. Confirm the StatefulSet contains the new image.

kubectl describe statefulset repsample | grep Image

The output is the following:

Image:	container-registry.oracle.com/timesten/timesten:22.1.1.35.0
Image:	<pre>container-registry.oracle.com/timesten/timesten:22.1.1.35.0</pre>
Image:	<pre>container-registry.oracle.com/timesten/timesten:22.1.1.35.0</pre>
Image:	<pre>container-registry.oracle.com/timesten/timesten:22.1.1.35.0</pre>

When the TimesTen Operator detects that there is an update to the TimesTenClassic's .spec.ttspec.image datum, it modifies the StatefulSet with the new image. Since this is an automated upgrade of a replicated TimesTenClassic object, the TimesTen Operator takes action and initiates and orchestrates the upgrade process.

5. Monitor the progress of the upgrade, observing the state transitions.

```
kubectl get events -w
```

```
OBJECT
LAST SEEN TYPE
                  REASON
MESSAGE
. . .
8m4s
         Normal
                   Upgrade
                                          timestenclassic/repsample
Image updated, automatic upgrade started
          Normal
                   Upgrade
                                          timestenclassic/repsample
8m4s
Deleted standby pod repsample-1 during upgrade
7m59s
          Warning Failed
                                          timestenclassic/repsample
Pod repsample-1 was replaced
7m54s Normal StateChange
                                          timestenclassic/repsample
Pod repsample-1 is Not Ready
7m54s Warning StateChange
                                          timestenclassic/repsample
TimesTenClassic was Normal, now ActiveTakeover
7m53s Normal StateChange
                                          timestenclassic/repsample
TimesTenClassic was ActiveTakeover, now StandbyDown
5m28s
          Normal Info
                                          timestenclassic/repsample
Pod repsample-1 Agent Up
5m28s Normal Info
                                          timestenclassic/repsample
Pod repsample-1 Instance Exists
5m28s Normal Info
                                          timestenclassic/repsample
Pod repsample-1 Daemon Down
5m28s Normal Info
                                         timestenclassic/repsample
Pod repsample-1 Daemon Up
5m28s Normal Info
                                         timestenclassic/repsample
Pod repsample-1 Database Unloaded
5m25s Normal Info
                                          timestenclassic/repsample
Pod repsample-1 Database None
5m10s Normal Info
                                         timestenclassic/repsample
Pod repsample-1 Database Loaded
5m10s
         Normal
                                          timestenclassic/repsample
                  Info
Pod repsample-1 RepAgent Not Running
5m10s
        Normal
                  Info
                                          timestenclassic/repsample
Pod repsample-1 RepScheme Exists
5m10s Normal StateChange
                                         timestenclassic/repsample
Pod repsample-1 RepState IDLE
5m4s Normal Info
                                          timestenclassic/repsample
Pod repsample-1 Database Loaded
5m4s
         Normal Info
                                          timestenclassic/repsample
Pod repsample-1 RepAgent Running
         Normal Info
5m4s
                                          timestenclassic/repsample
Pod repsample-1 RepScheme Exists
5m4s Normal StateChange
                                         timestenclassic/repsample
Pod repsample-1 RepState STANDBY
          Normal
                                         timestenclassic/repsample
5m4s
                   StateChange
Pod repsample-1 is Ready
         Normal
                                          timestenclassic/repsample
5m3s
                  Upgrade
Upgrade of standby complete
5m3s
          Normal
                   StateChange
                                          timestenclassic/repsample
TimesTenClassic was StandbyDown, now Normal
```

Normal Upgrade timestenclassic/repsample 4m32s Deleted active pod repsample-0 during upgrade 3m31s Warning Error timestenclassic/repsample Pod repsample-0 Unreachable for 117 seconds 3m31s Normal StateChange timestenclassic/repsample Pod repsample-0 is Not Ready 3m31s Normal StateChange timestenclassic/repsample Pod repsample-0 is Not Active Ready 3m31s Warning StateChange timestenclassic/repsample TimesTenClassic was Normal, now ActiveDown 3m30s Warning Failed timestenclassic/repsample Pod repsample-0 was replaced 3m29s Normal Info timestenclassic/repsample Pod repsample-1 Database Updatable timestenclassic/repsample 3m29s Normal StateChange Pod repsample-1 RepState ACTIVE 3m29s Normal StateChange timestenclassic/repsample Pod repsample-1 is Not Ready 3m29s Normal StateChange timestenclassic/repsample TimesTenClassic was ActiveDown, now ActiveTakeover 3m25s Normal StateChange timestenclassic/repsample Pod repsample-1 is Ready 3m25s Normal StateChange timestenclassic/repsample Pod repsample-1 is Active Ready timestenclassic/repsample 3m25s Normal StateChange TimesTenClassic was ActiveTakeover, now StandbyDown 2m35s Normal Info timestenclassic/repsample Pod repsample-0 Agent Up timestenclassic/repsample 2m35s Normal Info Pod repsample-0 Instance Exists 2m35s Normal Info timestenclassic/repsample Pod repsample-0 Daemon Down 2m35s Normal Info timestenclassic/repsample Pod repsample-0 Daemon Up 2m35s Normal Info timestenclassic/repsample Pod repsample-0 Database Unloaded timestenclassic/repsample 2m32s Normal Info Pod repsample-0 Database None 2m11s Normal Info timestenclassic/repsample Pod repsample-0 Database Loaded 2m11s Normal Info timestenclassic/repsample Pod repsample-0 RepAgent Not Running 2m11s Normal Info timestenclassic/repsample Pod repsample-0 RepScheme Exists 2m11s Normal StateChange timestenclassic/repsample Pod repsample-0 RepState IDLE 2m5s Normal Info timestenclassic/repsample Pod repsample-0 Database Loaded Normal Info timestenclassic/repsample 2m5s Pod repsample-0 RepAgent Running timestenclassic/repsample 2m5s Normal Info Pod repsample-0 RepScheme Exists 2m5s Normal StateChange timestenclassic/repsample Pod repsample-0 RepState STANDBY Normal StateChange timestenclassic/repsample 2m5s Pod repsample-0 is Ready

```
2m5sNormalUpgradetimestenclassic/repsampleUpgrade of active completetimestenclassic/repsample2m5sNormalUpgradetimestenclassic/repsampleUpgrade completed in 359 secstimestenclassic/repsample2m4sNormalStateChangetimestenclassic/repsampleTimesTenClassic was StandbyDown, now Normaltimestenclassic/repsample
```

The automated upgrade process completes. The TimesTenClassic object is in the Normal state. TimesTen databases are up and running and functioning properly. Active standby pair replication is configured between them.

- 6. Verify the active and standby databases are running the correct release.
 - a. Establish a shell in the active Pod.

```
kubectl exec -it repsample-1 -c tt -- /bin/bash
```

b. Run the TimesTen ttVersion utility.

ttVersion

The output is similar to the following:

```
TimesTen Release 22.1.1.35.0 (64 bit Linux/x86_64) (instance1:6624)
2025-01-16T15:16:01Z
Instance admin: timesten
Instance home directory: /tt/home/timesten/instances/instance1
Group owner: timesten
Daemon home directory: /tt/home/timesten/instances/instance1/info
PL/SQL enabled.
```

- c. Exit from the shell.
- d. Establish a shell in the standby Pod.

kubectl exec -it repsample-0 -c tt -- /bin/bash

e. Run the TimesTen ttVersion utility.

ttVersion

The output is similar to the following:

```
TimesTen Release 22.1.1.35.0 (64 bit Linux/x86_64) (instance1:6624)
2025-01-16T15:16:01Z
Instance admin: timesten
Instance home directory: /tt/home/timesten/instances/instance1
Group owner: timesten
Daemon home directory: /tt/home/timesten/instances/instance1/info
PL/SQL enabled.
```

f. Exit from the shell.

The active and standby databases are running the correct release of TimesTen.

Congratulations! You successfully completed an automated upgrade for a replicated TimesTenClassic object. The active and standby databases are upgraded, running, and fully operational.

Perform a Manual Upgrade of a Replicated TimesTenClassic Object

Let's perform an manual upgrade for a replicated TimesTenClassic object by first modifying the TimesTenClassic object, followed by initiating an upgrade of the standby database, and concluding with initiating the fail over from the active database to the standby:

- Modify a Replicated TimesTenClassic Object
- Upgrade the Standby Database
- Fail Over

Modify a Replicated TimesTenClassic Object

Let's modify a replicated TimesTenClassic object.

1. Review the TimesTenClassic objects running in your namespace.

kubectl get ttc

The output is similar to the following:

NAME	STATE	ACTIVE	AGE
mannorepsample	AllReplicasReady	N/A	30h
manrepsample	Normal	manrepsample-0	54m
norepsample	AllReplicasReady	N/A	21h
repsample	Normal	repsample-0	50m

There are several TimesTenClassic objects running in your namespace. One of the replicated TimesTenClassic objects is manrepsample. Its high level state is Normal indicating TimesTen databases in the active standby pair are up and running and functioning properly.

2. Review the image upgrade strategy for the object.

kubectl get ttc manrepsample -o yaml | grep imageUpgradeStrategy

The output is the following:

imageUpgradeStrategy: Manual

The image upgrade strategy is Manual , indicating a manual upgrade strategy.

3. On your development host, edit the TimesTenClassic object's .spec.ttspec.image datum with the container image you want to use for the upgrade. This example uses container-registry.oracle.com/timesten/timesten:22.1.1.35.0.



a. Edit the file, replacing the image value with container-registry.oracle.com/ timesten/timesten:22.1.1.35.0.

kubectl edit ttc manrepsample
Please edit the object below. Lines beginning with a '#' will be
ignored,
and an empty file will abort the edit. If an error occurs while
saving this file will be
reopened with the relevant failures.
#
apiVersion: timesten.oracle.com/v4
kind: TimesTenClassic
...
image: container-registry.oracle.com/timesten/timesten:22.1.1.35.0
...

- b. Save the file and exit from the editor.
- 4. Confirm the StatefulSet contains the new image.

kubectl describe statefulset manrepsample | grep Image

The output is the following:

Image:	<pre>container-registry.oracle.com/timesten/timesten:22.1.1.35.0</pre>
Image:	<pre>container-registry.oracle.com/timesten/timesten:22.1.1.35.0</pre>
Image:	<pre>container-registry.oracle.com/timesten/timesten:22.1.1.35.0</pre>
Image:	<pre>container-registry.oracle.com/timesten/timesten:22.1.1.35.0</pre>

When the TimesTen Operator detects that there is an update to the TimesTenClassic's .spec.ttspec.image datum, it modifies the StatefulSet with the new image. Since this is a manual upgrade of a replicated TimesTenClassic object, the TimesTen Operator takes no further action. This enables you to initiate the manual upgrade.

You successfully modified the TimesTenClassic object. Let's continue the manual upgrade by upgrading the standby database.

Upgrade the Standby Database

Perform these steps to upgrade the standby database.

Note:

Even though you are upgrading the standby database, depending on your replication configuration, this may result in disruption on your active database. This may impact your applications. Perform the upgrade at the appropriate time.

1. Determine which Pod is the standby.

```
kubectl get ttc manrepsample
```

The output is similar to the following.

NAME	STATE	ACTIVE	AGE
manrepsample	Normal	manrepsample-0	160m

Since the manrepsample-0 Pod is the active, the manrepsample-1 Pod is the standby.

2. Delete the standby Pod.

kubectl delete pod manrepsample-1

The output is the following.

pod "manrepsample-1" deleted

Kubernetes automatically creates a new manrepsample-1 Pod to replace the deleted Pod. The TimesTen Operator configures the new manresample-1 Pod as the standby Pod. This new Pod runs the upgraded TimesTen image.

3. Monitor the progress of the upgrade by observing the state transitions.

kubectl get events -w

LAST SEEN	TYPE	REASON			
OBJECT		ME	SSAGE		
2m51s	Warning	g StateChange			timestenclassic/
manrepsampl	e Time	esTenClassic wa	s Normal,	now	ActiveTakeover
2m50s	Normal	StateChange			timestenclassic/
manrepsampl	e Time	esTenClassic wa	s ActiveT a	akeor	ver, now StandbyDown
50s	Normal	Info			timestenclassic/
manrepsampl	e Pod	manrepsample-1	Agent Up		
50s	Normal	Info			timestenclassic/
manrepsampl	e Pod	manrepsample-1	Instance	Exi	sts
50s	Normal	Info			timestenclassic/
manrepsampl	e Pod	manrepsample-1	Daemon Do	own	
50s	Normal	Info			timestenclassic/
manrepsampl	e Pod	manrepsample-1	Daemon U	p	
50s		-			timestenclassic/
manrepsampl	e Pod	manrepsample-1			
43s					timestenclassic/
		manrepsample-1			
					timestenclassic/
		manrepsample-1			
22s					timestenclassic/
		manrepsample-1	RepAgent	Not	
22s	Normal	Info			timestenclassic/
		manrepsample-1			
					timestenclassic/
		manrepsample-1	-		
		Info			timestenclassic/
manrepsampl	e Pod	manrepsample-1	Database	Loa	ded

```
timestenclassic/
16s
         Normal
                   Info
manrepsample Pod manrepsample-1 RepAgent Running
16s Normal Info
                                        timestenclassic/
manrepsample Pod manrepsample-1 RepScheme Exists
16s Normal StateChange
                                timestenclassic/
manrepsample Pod manrepsample-1 RepState STANDBY
16s Normal StateChange
                                       timestenclassic/
manrepsample Pod manrepsample-1 is Ready
16s Normal
                   StateChange
                                        timestenclassic/
manrepsample TimesTenClassic was StandbyDown, now Normal
```

The TimesTen Operator successfully upgraded the standby database. You are now ready to fail over from the active database to the standby.

Fail Over

You are now ready to initiate the fail over from the active database to the standby.

Note:

During fail over, the TimesTen Operator takes down your active database, and **immediately** fails over to the standby. Do not perform this procedure at the busiest time of your production day. It's best to consider performing this operation during a scheduled production outage.

Before failing over, quiesce your applications on the active database. You can use the TimesTen ttAdmin -close and the ttAdmin -disconnect commands. See Opening and Closing the Database for User Connections and Disconnecting from a Database in the Oracle TimesTen In-Memory Database Operations Guide.

To avoid potential data loss, use the TimesTen ttRepAdmin -wait command to wait until replication is caught up, ensuring that transactions that were executed on the active database are replicated to the standby database. See ttRepAdmin in the *Oracle TimesTen In-Memory Database Reference*.

Once the standby is caught up, fail over from the active database to the standby by deleting the active Pod. When you delete the active Pod, the TimesTen Operator automatically detects the failure and promotes the standby database to be the active. Client/server applications that are using the active database are automatically reconnected to the new active database. Transactions in flight are rolled back. Prepared SQL statements do need to be re-prepared by the applications. See About Handling Failover and Recovery for more information.

Let's initiate the fail over.

1. Delete the active Pod.

kubectl delete pod manrepsample-0

The output is the following.

```
pod "manrepsample-0" deleted
```



Kubernetes automatically creates a new manrepsample-0 Pod to replace the deleted Pod. The TimesTen Operator configures the new manresample-0 Pod as the standby Pod. This new Pod runs the upgraded TimesTen image.

2. Monitor the progress of the upgrade, observing the state transitions.

kubectl get events -w

The output is similar to the following:

LAST SEEN TYPE REASON OBJECT MESSAGE . . . 37m Warning Upgrade timestenclassic/ manrepsample Image updated, automatic upgrade disabled 26m Warning Failed timestenclassic/ manrepsample Pod manrepsample-1 was replaced 26m Normal StateChange timestenclassic/ manrepsample Pod manrepsample-1 is Not Ready 2.6m Warning StateChange timestenclassic/ manrepsample TimesTenClassic was Normal, now ActiveTakeover 26m Normal StateChange timestenclassic/ manrepsample TimesTenClassic was ActiveTakeover, now StandbyDown Normal Info timestenclassic/ 24m manrepsample Pod manrepsample-1 Agent Up 24m Normal Info timestenclassic/ manrepsample Pod manrepsample-1 Instance Exists Normal Info timestenclassic/ 24m manrepsample Pod manrepsample-1 Daemon Down 24m Normal Info timestenclassic/ manrepsample Pod manrepsample-1 Daemon Up Normal Info 24m timestenclassic/ manrepsample Pod manrepsample-1 Database Unloaded Normal Info 24m timestenclassic/ manrepsample Pod manrepsample-1 Database None 24m Normal Info timestenclassic/ manrepsample Pod manrepsample-1 Database Loaded 24m Normal Info timestenclassic/ manrepsample Pod manrepsample-1 RepAgent Not Running 24m Normal Info timestenclassic/ manrepsample Pod manrepsample-1 RepScheme Exists Normal StateChange timestenclassic/ 2.4m manrepsample Pod manrepsample-1 RepState IDLE 23m Normal Info timestenclassic/ manrepsample Pod manrepsample-1 Database Loaded Normal Info 23m timestenclassic/ manrepsample Pod manrepsample-1 RepAgent Running 23m Normal Info timestenclassic/ manrepsample Pod manrepsample-1 RepScheme Exists Normal StateChange 23m timestenclassic/ manrepsample Pod manrepsample-1 RepState STANDBY 23m Normal StateChange timestenclassic/ manrepsample Pod manrepsample-1 is Ready 23m Normal StateChange timestenclassic/ manrepsample TimesTenClassic was StandbyDown, now Normal 3m50s Warning Failed timestenclassic/

manrepsample Pod manrepsample-0 was replaced 3m44s Normal StateChange timestenclassic/ manrepsample Pod manrepsample-0 is Not Ready 3m44s Normal StateChange timestenclassic/ manrepsample Pod manrepsample-0 is Not Active Ready 3m44s Warning StateChange timestenclassic/ manrepsample TimesTenClassic was Normal, now ActiveDown 3m43s Normal Info timestenclassic/ manrepsample Pod manrepsample-1 Database Updatable 3m43s Normal StateChange timestenclassic/ manrepsample Pod manrepsample-1 RepState ACTIVE 3m43s Normal StateChange timestenclassic/ manrepsample Pod manrepsample-1 is Not Ready 3m43s Normal StateChange timestenclassic/ manrepsample TimesTenClassic was ActiveDown, now ActiveTakeover 3m38s Normal StateChange timestenclassic/ manrepsample Pod manrepsample-1 is Ready 3m38s Normal StateChange timestenclassic/ manrepsample Pod manrepsample-1 is Active Ready 3m38s Normal StateChange timestenclassic/ manrepsample TimesTenClassic was ActiveTakeover, now StandbyDown 119s Normal Info timestenclassic/ manrepsample Pod manrepsample-0 Agent Up 119s Normal Info timestenclassic/ manrepsample Pod manrepsample-0 Instance Exists 119s Normal Info timestenclassic/ manrepsample Pod manrepsample-0 Daemon Down 118s Normal Info timestenclassic/ manrepsample Pod manrepsample-0 Daemon Up 118s Normal Info timestenclassic/ manrepsample Pod manrepsample-0 Database Unloaded 111s Normal Info timestenclassic/ manrepsample Pod manrepsample-0 Database None 90s Normal Info timestenclassic/ manrepsample Pod manrepsample-0 Database Loaded 90s Normal Info timestenclassic/ manrepsample Pod manrepsample-0 RepAgent Not Running 90s Normal Info timestenclassic/ manrepsample Pod manrepsample-0 RepScheme Exists 90s Normal StateChange timestenclassic/ manrepsample Pod manrepsample-0 RepState IDLE 84s Normal Info timestenclassic/ manrepsample Pod manrepsample-0 Database Loaded 84s Normal Info timestenclassic/ manrepsample Pod manrepsample-0 RepAgent Running Normal Info 84s timestenclassic/ manrepsample Pod manrepsample-0 RepScheme Exists 84s Normal StateChange timestenclassic/ manrepsample Pod manrepsample-0 RepState STANDBY 84s Normal StateChange timestenclassic/ manrepsample Pod manrepsample-0 is Ready 84s Normal StateChange timestenclassic/ manrepsample TimesTenClassic was StandbyDown, now Normal

The upgrade process completes.

3. Confirm which Pod is the active.

```
kubectl get ttc manrepsample
```

The output is similar to the following:

NAME STATE ACTIVE AGE manrepsample Normal manrepsample-1 3h21m

The TimesTen Operator promoted the standby manrepsample-1 database to be the active. Applications that were connected to the manrepsample-0 database are automatically reconnected to the manrepsample-1 database by TimesTen. After a brief outage, the applications can continue to use the database.

- 4. Verify the active and standby databases are running the correct release.
 - a. Establish a shell in the active Pod.

kubectl exec -it manrepsample-1 -c tt -- /bin/bash

b. Run the TimesTen ttVersion utility.

ttVersion

The output is similar to the following:

```
TimesTen Release 22.1.1.35.0 (64 bit Linux/x86_64) (instance1:6624)
2025-01-16T15:16:01Z
Instance admin: timesten
Instance home directory: /tt/home/timesten/instances/instance1
Group owner: timesten
Daemon home directory: /tt/home/timesten/instances/instance1/info
PL/SQL enabled.
```

- c. Exit from the shell.
- d. Establish a shell in the standby Pod.

kubectl exec -it manrepsample-0 -c tt -- /bin/bash

e. Run the TimesTen ttVersion utility.

ttVersion

```
TimesTen Release 22.1.1.35.0 (64 bit Linux/x86_64) (instance1:6624)
2025-01-16T15:16:01Z
Instance admin: timesten
Instance home directory: /tt/home/timesten/instances/instance1
Group owner: timesten
Daemon home directory: /tt/home/timesten/instances/instance1/info
PL/SQL enabled.
```



f. Exit from the shell.

The active and standby databases are running the correct release of TimesTen.

Congratulations! You successfully performed a manual upgrade for a replicated TimesTenClassic object. The active and standby databases are upgraded, running, and fully operational.

Perform an Automated Upgrade of a Non-Replicated TimesTenClassic Object

Let's perform an automated upgrade for a non-replicated TimesTenClassic object consisting of three independent TimesTen databases.

1. Review the TimesTenClassic objects running in your namespace.

kubectl get ttc

The output is similar to the following:

NAME	STATE	ACTIVE	AGE
mannorepsample	AllReplicasReady	N/A	3h47m
norepsample	AllReplicasReady	N/A	4d10s
repsample	Normal	repsample-0	5d23h

One of the non-replicated TimesTenClassic objects is norepsample. Its high level state is AllReplicasReady, indicating all replicas are ready and available. TimesTen databases are up and running and functioning properly.

2. Review the image upgrade strategy and rolling update partition values for the object.

```
kubectl get ttc norepsample -o yaml | grep 'imageUpgradeStrategy\|
rollingUpdatePartition'
```

The output is the following:

```
imageUpgradeStrategy: Auto
rollingUpdatePartition: 2
```

The image upgrade strategy is Auto (also the default), indicating an automated upgrade strategy. The value for rollingUpdatePartition is 2, indicating that Kubernetes upgrades Pods with an ordinal value greater than or equal to 2. For the norepsample object, since the value of replicas is 3, there are three Pods named norepsample-0, norepsample-1, and norepsample-2. Therefore, Kubernetes only upgrades the norepsample-2 Pod. Let's verify this by performing the upgrade.

3. On your development host, edit the TimesTenClassic object's .spec.ttspec.image datum with the container image you want to use for the upgrade. This example uses container-registry.oracle.com/timesten/timesten:22.1.1.35.0.



a. Edit the file, replacing the image value with container-registry.oracle.com/ timesten/timesten:22.1.1.35.0.

kubectl edit ttc norepsample

```
# Please edit the object below. Lines beginning with a '#' will be
ignored,
# and an empty file will abort the edit. If an error occurs while
saving this file will be
# reopened with the relevant failures.
#
apiVersion: timesten.oracle.com/v4
kind: TimesTenClassic
...
image: container-registry.oracle.com/timesten/timesten:22.1.1.35.0
...
```

- b. Save the file and exit from the editor.
- 4. Confirm the StatefulSet contains the new image.

kubectl describe statefulset norepsample | grep Image

The output is the following:

Image:	<pre>container-registry.oracle.com/timesten/timesten:22.1.1.35.0</pre>
Image:	<pre>container-registry.oracle.com/timesten/timesten:22.1.1.35.0</pre>
Image:	<pre>container-registry.oracle.com/timesten/timesten:22.1.1.35.0</pre>
Image:	<pre>container-registry.oracle.com/timesten/timesten:22.1.1.35.0</pre>

When the TimesTen Operator detects that there is an update to the TimesTenClassic's .spec.ttspec.image datum, it modifies the StatefulSet with the new image. Since this is an automated upgrade, the TimesTen Operator takes no further action, but Kubernetes does take action. Kubernetes automatically begins to terminate Pods and replace them with new ones. These new Pods run the new image. Since rollingUpdatePartition is 2, you should expect to see only the norepsample-2 Pod upgraded with the new image.

5. Monitor the progress.

kubectl get ttc norepsample

The output is similar to the following:

NAME	STATE	ACTIVE	AGE
norepsample	SomeReplicasReady	N/A	16m

The object is in the SomeReplicasReady state.

Wait a few minutes. Then, monitor again.

kubectl get ttc norepsample



The output is similar to the following:

NAME	STATE	ACTIVE	AGE
norepsample	AllReplicasReady	N/A	19m

The object is in the AllReplicasReady state. All replicas are ready and available. TimesTen databases are up and running and functioning properly.

Check the image that the norepsample-2 Pod is running.

kubectl describe pod norepsample-2 | grep Image

The output is similar to the following:

```
Image:container-registry.oracle.com/timesten/timesten:22.1.1.35.0Image:container-registry.oracle.com/timesten/timesten:22.1.1.35.0Image:container-registry.oracle.com/timesten/timesten:22.1.1.35.0Image:container-registry.oracle.com/timesten/timesten:22.1.1.35.0
```

The containers in the Pod are running the new container image.

- 7. Check the image for the norepsample-1 and norepsample-0 Pods.
 - a. Check the norepsample-1 Pod.

kubectl describe pod norepsample-1 | grep Image

The output is similar to the following:

```
Image: container-registry.oracle.com/timesten/
timesten:22.1.1.34.0
Image: container-registry.oracle.com/timesten/
timesten:22.1.1.34.0
Image: container-registry.oracle.com/timesten/
timesten:22.1.1.34.0
Image: container-registry.oracle.com/timesten/
timesten:22.1.1.34.0
```

The containers in the Pod are not running the new image. Due to the rollingUpdatePartition value of 2, Kubernetes does not upgrade this Pod with the new image. This is correct behavior.

b. Check the norepsample-0 Pod.

kubectl describe pod norepsample-0 | grep Image

```
Image: container-registry.oracle.com/timesten/
timesten:22.1.1.34.0
Image: container-registry.oracle.com/timesten/
timesten:22.1.1.34.0
Image: container-registry.oracle.com/timesten/
timesten:22.1.1.34.0
```



```
Image: container-registry.oracle.com/timesten/
timesten:22.1.1.34.0
```

The containers in the Pod are not running the new image. Due to the rollingUpdatePartition value of 2, Kubernetes does not upgrade this Pod with the new image. This is correct behavior.

After you confirm the upgrade is working, you can upgrade the remaining Pods.

- 8. Edit the TimesTenClassic object's .spec.ttspec.rollingUpdatePartition datum and change the value from 2 to 0.
 - a. Edit the file, replacing the rollingUpdatePartition value with 2.

kubectl edit ttc norepsample

```
# Please edit the object below. Lines beginning with a '#' will be
ignored,
# and an empty file will abort the edit. If an error occurs while
saving this file will be
# reopened with the relevant failures.
#
apiVersion: timesten.oracle.com/v4
kind: TimesTenClassic
...
rollingUpdatePartition: 0
...
```

b. Save the file and exit from the editor.

Kubernetes automatically begins to terminate Pods and replace them with new ones. These new Pods run the new image. Since <code>rollingUpdatePartition</code> is now 0, you should expect to see the <code>norepsample-1</code> and <code>norepsample-0</code> Pods upgraded with the new image.

- 9. Monitor the progress.
 - a. TimesTenClassic object:

kubectl get ttc norepsample

The output is similar to the following

NAME	STATE	ACTIVE	AGE
norepsample	SomeReplicasReady	N/A	51m

b. Pods:

kubectl get pods

NAME	READY	STATUS	RESTARTS	AGE
 norepsample-0 norepsample-1 norepsample-2	3/3 0/3 3/3	Running Init:0/1 Running	0 0 0	53m 106s 36m



Kubernetes replaces the norepsample-1 Pod first. Wait a few minutes. Then monitor again.

kubectl get pods

The output is similar to the following:

NAME	READY	STATUS	RESTARTS	AGE
norepsample-0	0/3	Init:0/1	0	46s
norepsample-1	3/3	Running	0	3m41s
norepsample-2	3/3	Running	0	38m

Kubernetes replaced the norepsample-1 Pod and is in the process of replacing the norepsample-0 Pod.

Check again.

kubectl get pods

The output is similar to the following:

NAME	READY	STATUS	RESTARTS	AGE
norepsample-0	3/3	Running	0	5m59s
norepsample-1	3/3	Running	0	8m54s
norepsample-2	3/3	Running	0	43m

All Pods are running.

10. Confirm the Pods are running the new container image.

Pod norepsample-1:

kubectl describe pod norepsample-1 | grep Image

The output is similar to the following:

Image:	<pre>container-registry.oracle.com/timesten/timesten:22.1.1.35.0</pre>
Image:	<pre>container-registry.oracle.com/timesten/timesten:22.1.1.35.0</pre>
Image:	<pre>container-registry.oracle.com/timesten/timesten:22.1.1.35.0</pre>
Image:	<pre>container-registry.oracle.com/timesten/timesten:22.1.1.35.0</pre>

Pod norepsample-0:

kubectl describe pod norepsample-0 | grep Image

Image:	container-registry.oracle.com/timesten/timesten:22.1.1.35.0
Image:	<pre>container-registry.oracle.com/timesten/timesten:22.1.1.35.0</pre>



Image: container-registry.oracle.com/timesten/timesten:22.1.1.35.0 Image: container-registry.oracle.com/timesten/timesten:22.1.1.35.0

The Pods are running the new image.

11. (Optional) Confirm the state of the TimesTenClassic object.

kubectl get ttc norepsample

The output is similar to the following:

NAME	STATE	ACTIVE	AGE
norepsample	AllReplicasReady	N/A	62m

- **12.** Verify the databases are running the correct release.
 - a. Establish a shell in the -0 Pod.

kubectl exec -it norepsample-0 -c tt -- /bin/bash

b. Run the TimesTen ttVersion utility.

ttVersion

The output is similar to the following:

```
TimesTen Release 22.1.1.35.0 (64 bit Linux/x86_64) (instance1:6624)
2025-01-16T15:16:01Z
Instance admin: timesten
Instance home directory: /tt/home/timesten/instances/instance1
Group owner: timesten
Daemon home directory: /tt/home/timesten/instances/instance1/info
PL/SQL enabled.
```

- c. Exit from the shell.
- d. Establish a shell in the -1 Pod.

kubectl exec -it norepsample-1 -c tt -- /bin/bash

e. Run the TimesTen ttVersion utility.

ttVersion

```
TimesTen Release 22.1.1.35.0 (64 bit Linux/x86_64) (instance1:6624)
2025-01-16T15:16:01Z
Instance admin: timesten
Instance home directory: /tt/home/timesten/instances/instance1
Group owner: timesten
Daemon home directory: /tt/home/timesten/instances/instance1/info
PL/SQL enabled.
```



- f. Exit from the shell.
- g. Establish a shell in the -2 Pod.

kubectl exec -it norepsample-2 -c tt -- /bin/bash

h. Run the TimesTen ttVersion utility.

ttVersion

The output is similar to the following:

```
TimesTen Release 22.1.1.35.0 (64 bit Linux/x86_64) (instance1:6624)
2025-01-16T15:16:01Z
Instance admin: timesten
Instance home directory: /tt/home/timesten/instances/instance1
Group owner: timesten
Daemon home directory: /tt/home/timesten/instances/instance1/info
PL/SQL enabled.
```

i. Exit from the shell.

The TimesTen databases are running the correct release of TimesTen.

Congratulations! You successfully performed an automated upgrade for a non-replicated TimesTenClassic object. All replicas are ready and available. Pods are running the new TimesTen container image. TimesTen databases are upgraded and fully operational.

Perform a Manual Upgrade of a Non-Replicated TimesTenClassic Object

Let's perform a manual upgrade for a non-replicated TimesTenClassic object consisting of three independent TimesTen databases.

1. Review the TimesTenClassic objects running in your namespace.

kubectl get ttc

The output is similar to the following:

NAME	STATE	ACTIVE	AGE
mannorepsample	AllReplicasReady	N/A	3m32s
norepsample	AllReplicasReady	N/A	11h
repsample	Normal	repsample-0	5d19h

One of the non-replicated TimesTenClassic objects is mannorepsample. Its high level state is AllReplicasReady.

2. Review the image upgrade strategy for this object.

kubectl get ttc mannorepsample -o yaml | grep imageUpgradeStrategy
The output is the following:

```
imageUpgradeStrategy: Manual
```

The image upgrade strategy is Manual. Let's perform an upgrade.

- 3. On your development host, edit the TimesTenClassic object's .spec.ttspec.image datum with the container image you want to use for the upgrade. This example uses container-registry.oracle.com/timesten/timesten:22.1.1.35.0.
 - a. Edit the file, replacing the image value with container-registry.oracle.com/ timesten/timesten:22.1.1.35.0.

kubectl edit ttc mannorepsample

```
# Please edit the object below. Lines beginning with a '#' will be
ignored,
# and an empty file will abort the edit. If an error occurs while
saving this file will be
# reopened with the relevant failures.
#
apiVersion: timesten.oracle.com/v4
kind: TimesTenClassic
...
image: container-registry.oracle.com/timesten/timesten:22.1.1.35.0
...
```

b. Save the file and exit from the editor.

When the TimesTen Operator detects that there is an update to the TimesTenClassic's .spec.ttspec.image datum, it modifies the StatefulSet with the new image. Since this is a manual upgrade of a non-replicated object, the TimesTen Operator and Kubernetes take no further action.

4. (Optional) Confirm the StatefulSet contains the new image.

kubectl describe statefulset mannorepsample | grep Image

The output is the following:

```
Image:container-registry.oracle.com/timesten/timesten:22.1.1.35.0Image:container-registry.oracle.com/timesten/timesten:22.1.1.35.0Image:container-registry.oracle.com/timesten/timesten:22.1.1.35.0Image:container-registry.oracle.com/timesten/timesten:22.1.1.35.0
```

The containers including tt, daemonlog, and exporter contain the new image.

5. Review the Pods for the TimesTenClassic object.

kubectl get pods

The output is similar to the following:

NAME	READY	STATUS	RESTARTS	AGE
mannorepsample-0	3/3	Running	0	26m



mannorepsample-1	3/3	Running	0	26m
mannorepsample-2	3/3	Running	0	26m

There are three Pods associated with the TimesTenClassic object. Each Pod is running a TimesTen database. The databases are independent and have no relationship to each other.

6. Delete the mannorepsample-2 Pod. This action causes Kubernetes to terminate the Pod and replace it with a new one.

kubectl delete pod mannorepsample-2

7. Monitor the progress.

kubectl get ttc mannorepsample

The output is the following:

NAME STATE ACTIVE AGE mannorepsample **SomeReplicasReady** N/A 49m

The object is in the SomeReplicasReady state.

Wait a few minutes. Then monitor again.

kubectl get ttc mannorepsample

The output is the following:

NAME STATE ACTIVE AGE mannorepsample AllReplicasReady N/A 53m

The object is in the AllReplicasReady state. All replicas are ready and available. TimesTen databases are up and running and functioning properly.

8. Check the image that the mannorepsample-2 Pod is running.

kubectl describe pod mannorepsample-2 | grep Image

The output is similar to the following:

Image:	<pre>container-registry.oracle.com/timesten/timesten:22.1.1.35.0</pre>
Image:	<pre>container-registry.oracle.com/timesten/timesten:22.1.1.35.0</pre>
Image:	<pre>container-registry.oracle.com/timesten/timesten:22.1.1.35.0</pre>
Image:	<pre>container-registry.oracle.com/timesten/timesten:22.1.1.35.0</pre>

The containers in the Pod are running the new container image.



9. Delete the remaining Pods.

kubectl delete pod mannorepsample-1

kubectl delete pod mannorepsample-0

10. Monitor the progress.

kubectl get pods

The output is similar to the following:

NAME	READY	STATUS	RESTARTS	AGE
mannorepsample-0	0/3	Init:0/1	0	13s
mannorepsample-1	0/3	Init:0/1	0	60s

Kubernetes starts recreating the Pods.

Wait a few minutes. Then monitor again.

kubectl get pods

The output is the following:

NAME	READY	STATUS	RESTARTS	AGE
mannorepsample-0	3/3	Running	0	1m16s
mannorepsample-1	3/3	Running	0	2m3s
mannorepsample-2	3/3	Running	0	6m32s

The mannorepsample-0 and mannorepsample-1 Pods are now running. The mannorepsample-2 has already been running.

11. Confirm the Pods are running the new container image.

Pod mannorepsample-0:

kubectl describe pod mannorepsample-0 | grep Image

The output is similar to the following:

Image:	<pre>container-registry.oracle.com/timesten/timesten:22.1.1.35.0</pre>
Image:	<pre>container-registry.oracle.com/timesten/timesten:22.1.1.35.0</pre>
Image:	<pre>container-registry.oracle.com/timesten/timesten:22.1.1.35.0</pre>
Image:	<pre>container-registry.oracle.com/timesten/timesten:22.1.1.35.0</pre>

Pod mannorepsample-1:

kubectl describe pod mannorepsample-1 | grep Image



The output is similar to the following:

Image:	<pre>container-registry.oracle.com/timesten/timesten:22.1.1.35.0</pre>
Image:	<pre>container-registry.oracle.com/timesten/timesten:22.1.1.35.0</pre>
Image:	<pre>container-registry.oracle.com/timesten/timesten:22.1.1.35.0</pre>
Image:	<pre>container-registry.oracle.com/timesten/timesten:22.1.1.35.0</pre>

The Pods are running the new image.

12. (Optional) Confirm the state of the TimesTenClassic object.

kubectl get ttc mannorepsample

The output is similar to the following:

NAME	STATE	ACTIVE	AGE
mannorepsample	AllReplicasReady	N/A	79m

- 13. Verify the databases are running the correct release.
 - a. Establish a shell in the -0 Pod.

kubectl exec -it mannorepsample-0 -c tt -- /bin/bash

b. Run the TimesTen ttVersion utility.

ttVersion

The output is similar to the following:

```
TimesTen Release 22.1.1.35.0 (64 bit Linux/x86_64) (instance1:6624)
2025-01-16T15:16:01Z
Instance admin: timesten
Instance home directory: /tt/home/timesten/instances/instance1
Group owner: timesten
Daemon home directory: /tt/home/timesten/instances/instance1/info
PL/SQL enabled.
```

- c. Exit from the shell.
- d. Establish a shell in the -1 Pod.

kubectl exec -it mannorepsample-1 -c tt -- /bin/bash

e. Run the TimesTen ttVersion utility.

ttVersion

The output is similar to the following:

```
TimesTen Release 22.1.1.35.0 (64 bit Linux/x86_64) (instance1:6624)
2025-01-16T15:16:01Z
Instance admin: timesten
Instance home directory: /tt/home/timesten/instances/instance1
```

```
Group owner: timesten
Daemon home directory: /tt/home/timesten/instances/instance1/info
PL/SQL enabled.
```

- f. Exit from the shell.
- g. Establish a shell in the -2 Pod.

kubectl exec -it mannorepsample-2 -c tt -- /bin/bash

h. Run the TimesTen ttVersion utility.

ttVersion

The output is similar to the following:

```
TimesTen Release 22.1.1.35.0 (64 bit Linux/x86_64) (instance1:6624)
2025-01-16T15:16:01Z
Instance admin: timesten
Instance home directory: /tt/home/timesten/instances/instance1
Group owner: timesten
Daemon home directory: /tt/home/timesten/instances/instance1/info
PL/SQL enabled.
```

i. Exit from the shell.

The TimesTen databases are running the correct release of TimesTen.

Congratulations! You successfully performed a manual upgrade for a non-replicated TimesTenClassic object. All replicas are ready and available. Pods are running the new TimesTen container image. TimesTen databases are upgraded and fully operational.

About Upgrading Direct Mode Applications

You cannot use the automated upgrade procedures to upgrade direct mode applications that are running in their own containers. The TimesTen Operator does propagate image changes from a TimesTenClassic object to the associated StatefulSet, but the changes do not initiate the automated upgrade process. You must manually terminate the applications that are running in the containers. In so doing, the StatefulSet spawns new containers to replace the original containers. These new containers run the new TimesTen image. For more information about direct mode applications, see About Using Direct Mode Applications.

About Failures During an Upgrade

If there are failures in any step of the upgrade process, a TimesTenClassic object enters the ManualInterventionRequired state. The remaining steps of the upgrade process are canceled. You must manually fix the TimesTenClassic object. See About the High Level State of TimesTenClassic Objects.



20

TimesTen Kubernetes Operator Object Types

This chapter gives an overview of the TimesTenClassic and the TimesTenScaleout object types. The chapter describes the syntax for each object type.

Topics:

- Overview of the TimesTen Kubernetes Operator Object Types
- About the TimesTenClassic Object Type
- About the TimesTenScaleout Object Type

Overview of the TimesTen Kubernetes Operator Object Types

The TimesTen Kubernetes Operator (TimesTen Operator) installation defines the TimesTenClassic and the TimesTenScaleout object types to the Kubernetes cluster. An object of type TimesTenClassic describes the metadata deploying TimesTen Classic databases. An object of type TimesTenScaleout describes the metadata for TimesTen Scaleout grids and databases. You can create as many TimesTenClassic and TimesTenScaleout objects as you like.

The definition of the TimesTenClassic and the TimesTenScaleout object types use the same basic format that the formal Kubernetes documentation uses to define objects that are built-in to Kubernetes. The facilities available in any given Kubernetes cluster depend on what release of Kubernetes the cluster is using. For information on the Kubernetes API documentation, see:

https://kubernetes.io/docs/reference/kubernetes-api/

The Kubernetes API reference documentation refers to a number of built-in Kubernetes types used in the definition of the TimesTenClassic and the TimesTenScaleout object types. A Kubernetes StatefulSet is of particular importance. The TimesTenClassic and the TimesTenScaleout object types are basically a wrapper around a StatefulSet type. For more information, see:

https://kubernetes.io/docs/reference/kubernetes-api/

About the TimesTenClassic Object Type

The definition of the TimesTenClassic object type defines the set of attributes for deploying TimesTen Classic databases. The TimesTenClassic object type uses the following object definitions:

- TimesTenClassic
- TimesTenClassicSpec
- TimesTenClassicSpecSpec
- TimesTenClassicSpecSpecClientTLS
- TimesTenClassicSpecSpecPrometheus
- TimesTenClassicStatus



TimesTenClassic

An object of type TimesTenClassic describes the metadata for deploying TimesTen Classic databases.

The following table describes the syntax for the TimesTenClassic object type.

Table 20-1 TimesTenClassic syntax

Field	Туре	Description
apiVersion string		Versioned schema of this representation of an object. The TimesTen Kubernetes Operator supports $v4$, $v3$, and $v2$ schema versions.
		The value must be timesten.oracle.com/v4, timesten.oracle.com/v3 or timesten.oracle.com/v2.
		Use the v4 schema version by specifying timesten.oracle.com/v4. The v3 and v2 schema versions are available to facilitate upwards compatibility from earlier releases of the TimesTen Operator. The v2 schema version is deprecated. It is fully supported in this release, but will be removed in a future release.
		For more information about schema versions for TimesTenClassic objects, see About TimesTen CRDs.
kind	string	Type of object. For example, TimesTenClassic.
metadata	ObjectMeta	Metadata about the object, such as its name. All of the object's metadata is passed from the object to the StatefulSet. For information about ObjectMeta, see:
		<pre>https://kubernetes.io/docs/reference/ kubernetes-api/</pre>
spec	TimesTenClassicSpec	Desired configuration of TimesTen Pods and databases.
status	TimesTenClassicStatus	Current status of the Pods in this TimesTenClassic object as well as the status of various TimesTen components within those Pods. This data may be out of date by some window of time.

TimesTenClassicSpec

TimesTenClassicSpec appears in TimesTenClassic. The following table describes the syntax for TimesTenClassicSpec. The third column indicates the schema version in which the field was first introduced. For more information, see About TimesTen CRDs.



Field	Туре	Earliest Schema Version Supported In	Description
ttspec	TimesTenClassicSpecSpec	v2	Attributes specific to TimesTen.
template	PodTemplateSpec	v2	Describes the Pods provisioned for the TimesTenClassic object. In addition to Pods specified by template, there are the tt and daemonlog containers that are automatically included in each Pod. TimesTen runs in the tt container. If you configure and use Prometheus, the exporter container is also included. For information on PodTemplateSpec, see: https://kubernetes.io/docs/ reference/kubernetes-api/
volumeClaimTemplates	PersistentVolumeClaim	v2	TimesTen automatically provisions PersistentVolumeClaims (PVCs) for /tt (and for /ttlog, if specified). If you have applications that are running in containers in the TimesTen Pods, and those applications require additional PVCs, specify them in this field. For information on PersistentVolumeClaim, see: https://kubernetes.io/docs/ reference/kubernetes-api/

Table 20-2 TimesTenClassicSpec syntax

TimesTenClassicSpecSpec

TimesTenClassicSpecSpec appears in TimesTenClassicSpec.

The TimesTen Operator supports v4, v3, and v2 schema versions.

Note:

The v2 schema version is deprecated. It is fully supported in this release, but will be removed in a future release.

For more information about schema versions, see About TimesTen CRDs.

The following table details the fields for TimesTenClassicSpecSpec. Note the following:

- The fields are in alphabetical order.
- The third column indicates the earliest schema version the field was supported in.
- There are some fields of type quantity. The specified value is of the same format as Kubernetes resource limits. For example, 200Gi, 200G, 1000Mi, 1000M, and so on.



• There are fields that are reserved for internal use and are not documented in this table. The names of these fields typically begin with zz. For example, zzTestInfo is reserved for internal use.

Table 20-3 TimesTenClassicSpecSpec syntax

Field	Туре	Earliest Schema Version Supported In	Description
additionalMemoryRequest	quantity	v2	The amount of memory to request in addition to whatever is required for the TimesTen database.
			This memory is used for the TimesTen daemon, subdaemons, agents, and the Client/Server server.
			This value is added to databaseMemorySize that was either specified by you or calculated. The sum is the memory request to Kubernetes.
			The default is 2Gi.

Field	Туре	Earliest Schema Version Supported In	Description
agentAsyncTimeout	integer	v2	At times the TimesTen Operator needs to perform operations on a TimesTen instance or database. When the TimesTen Operator needs to perform such operations, it asks the TimesTen Agent running in the tt container of the appropriate Pod to perform the operation. The Agent issues TimesTen commands and utilities and, using the TimesTen ttIsql utility, runs SQL operations on the appropriate instance or database.
			The Agent runs the operation either synchronously (while the TimesTen Operator waits) or asynchronously (while the TimesTen Operator is not waiting and can perform other operations for other TimesTen objects).
			The only operation that is performed asynchronously is duplicate, where a TimesTen Classic database is copied from the active Pod to the standby Pod. This is done as part of the initial rollout of an active standby pair, and at various times during its lifecycle.
			These duplicate operations can take a long time, and the amount of time increases as the size of the database increases. If an asynchronous operation does not complete within agentAsyncTimeout seconds, the TimesTen Operator decides that it has failed and acts accordingly.
			If your database is large, you may need to increase the default value for agentAsyncTimeout. The default is 600 and is expressed in
			seconds.
agentGetTimeout	integer	v2	Time in seconds that the TimesTen Operator waits for an https GET request to be processed by the TimesTen agent. This includes the TCP and the TLS times as well as the time it takes for the TimesTen agent to implement the GET request.
			The default is 60. A value of 0 indicates that there is no timeout. If the timeout is exceeded, the TimesTen Operator considers the agent to be down.



Field	Туре	Earliest Schema Version Supported In	Description
agentPostTimeout	integer	v2	Time in seconds that the TimesTen Operator waits for an https POST request to be processed by the TimesTen agent. This includes the TCP and the TLS times as well as the time it takes for the TimesTen agent to implement the POST request. The POST requests may take a long time and the time may be proportional to the size of the database. (An example is a POST request to duplicate a database from the active to the standby.) The default is 600. A value of 0 indicates that there is no timeout. If the timeout is exceeded, the TimesTen Operator considers the POST request to have failed.
agentTcpTimeout	integer	v2	Time in seconds that the TimesTen Operator waits for a TCP handshake when communicating with the TimesTen agent. The default is 10. A value of 0 indicates
			that there is no timeout. If the timeout is exceeded, the TimesTen Operator considers the agent to be down.
agentTlsTimeout	integer	v2	Time in seconds that the TimesTen Operator waits for a TLS (https) credential exchange when communicating with the TimesTen agent. The default is 10. A value of 0 indicates that there is no timeout. If the timeout is exceeded, the TimesTen Operator
	hadaar		considers the agent to be down.
automaticMemoryRequests	boolean	v2	 Determines if the TimesTen Operator attempts to set appropriate memory limits and requests for TimesTen Pods. Valid values: true (default): The TimesTen Operator attempts to set memory limits and requests. false: The TimesTen Operator does not set memory limits and requests.

Field	Туре	Earliest Schema Version Supported In	Description
bothDownBehavior	string	v2	If the TimesTenClassic object enters the BothDown state, the TimesTen Operator examines the bothDownBehavior setting to determine what to do. Acceptable values are Best (default) or Manual. See BothDown.
cacheCleanup	boolean	v2	 Determines if the metadata in the Oracle Database should be cleaned up when this TimesTenClassic object is deleted. Use for TimesTen Cache only. Valid values: true (or not specified): The metadata is cleaned up. false: The metadata is not cleaned up. See Dropping Oracle Database Objects Used by Cache Groups with Autorefresh in the Oracle TimesTen In- Memory Database Cache Guide.
clientTLS	TimesTenClassicSpecSpec ClientTLS	v4	If specified, the TimesTen Operator can automatically create self-signed certificates and configure TimesTen to use those certificates for client/server encryption. The fields for the clientTLS object are defined in TimesTenClassicSpecSpecClientTLS. For more information about how the TimesTen Operator automatically generates certificates and configures client/server TLS connections, see Automatically Configure Client/Server TLS.
createASReadinessProbe	boolean	v2	 Determines if the TimesTen Operator creates a readiness probe for a replicated TimesTenClassic object. Valid values: true (default): The TimesTen Operator creates a readiness probe. false: The TimesTen Operator does not create a readiness probe. See About Readiness Probes for TimesTen Containers.
daemonLogCPURequest	quantity	ν2	The amount of CPU requested for the daemonlog container. The default is 200m (one-fifth of a CPU).



Field	Туре	Earliest Schema Version Supported In	Description
daemonLogMemoryRequest	quantity	v2	The amount of memory requested for the daemonlog container.
			The default is 200Mi.
daemonLogSidecar	boolean	v2	Determines if a daemon log container is created in each TimesTen Pod. This container writes the TimesTen daemon logs (from ttmesg.log) to stdout. This causes Kubernetes to record these logs. Valid values:
			 true (or not specified): A daemon log container is created. false: A daemon log container is not created.
databaseCPURequest quantity	quantity	v2	Specify this value to tell the TimesTen Operator how much CPU your tt containers require. This includes CPU used by the TimesTen daemon, subdaemons, replication agents, cache agents, and the Client/Server server.
			There is no default.

Field	Туре	Earliest Schema Version Supported In	Description
databaseMemorySize	quantity	v2	You can specify this value to tell the TimesTen Operator how much shared memory your database requires.
			If you specify a value, that value will be used. If you do not specify a value, the TimesTen Operator attempts to determine the required size from the provided db.ini file.
			If the TimesTen Operator cannot determine the database size, the value 580911104 is used. This is the size required for a default database with a PermSize of 200Mbyte, rounded up to 2Mi. This may be useful for experimentation, but is likely insufficien for production purposes.
			TimesTen recommends that you provide a db.ini file to the TimesTen Operator by using a Configmap or Secret, and that you not specify databaseMemorySize.
			Note: If you provide a db.ini file by using an init container, you must specify database MemorySi ze.
dbConfigMap	array of strings	v2	Name of one or more ConfigMaps to be included in a projected volume. This projected volume is mounted as / ttconfig in the TimesTen containers. If you do not specify dbConfigMap or dbSecret, you must place the metadata files into the /ttconfig directory by using other means. See Populate the /ttconfig Directory.



Field	Туре	Earliest Schema Version Supported In	Description
dbSecret	array of strings	v2	Name of one or more Secrets to be included in a projected volume. This projected volume is mounted as / ttconfig in the TimesTen containers. If you do not specify dbSecret or dbConfigMap, you must place the metadata files into the /ttconfig directory by using other means. See Populate the /ttconfig Directory.
deleteDbOnNotReadyNode	integer	v3	When specified, this datum directs the TimesTen Operator to detect situations where a Node is not ready (or unknown) for a specific period of time. When detected, the TimesTen Operator takes appropriate action to remedy the situation. Specifically, the TimesTen Operator checks to see if the Node's not ready condition has existed for more than deleteDbOnNotReadyNode seconds. If so, the TimesTen Operator deletes the Pod and the PVCs associated with the Pod. This causes Kubernetes to create a new Pod and new PVCs on a surviving Node. Once the Pod is scheduled and started by Kubernetes, the TimesTen Operator configures it as usual. The value is expressed in seconds and must be greater than pollingInterval. By default, this datum is not specified in your TimesTenClassic object definition.
			You must specify it. Note: Use caution when specifying this datum. Specifying this datum could result in the TimesTen Operator deleting PVCs. Deleting PVCs discards the on-disk copy of TimesTen databases.
			For more information, see How the TimesTen Kubernetes Operator Handles Node Failure.
exporterCPURequest	quantity	v2	The amount of CPU requested for the exporter container (if provisioned).



Field	Туре	Earliest Schema Version Supported In	Description
exporterMemoryRequest	quantity	v2	The amount of memory requested for the exporter container (if provisioned).
image	string	v2	The default is 200Mi. Name of the TimesTen image that is executed in the created containers. There is no default. You must specify the name of the image.
imagePullPolicy	string	v2	Determines if and when Kubernetes pulls the TimesTen image from the image repository. Valid values: • Always • IfNotPresent (default) • Never Note: Values are case sensitive.
imagePullSecret	string	v2	Image pull secret that is used to authenticate and give permission to Kubernetes to fetch the specified TimesTen image from its image repository. There is no default. You must specify the name of the image pull secret.
imageUpgradeStrategy	string	v2	 Determines if the TimesTen Operator performs automated upgrades. Valid values: Auto (or not specified): The TimesTen Operator performs automated upgrades. Manual: The TimesTen Operator does not perform an automated upgrade. Values are case sensitive. See Perform Upgrades.
logStorageClassName	string	v2	Name of the storage class that is used to request persistent volumes for the TimesTen database transaction log files. This field is optional.

Туре	Earliest Schema Version Supported In	Description
metav1.LabelSelector	v2	When choosing to use a persistent volume to store the TimesTen transaction logs, the primary determinant of what volumes to use is the logStorageClassName field that you specify. You can optionally specify a label selector by using the logStorageSelector field. This label selector further filters the set of volumes. See:
		<pre>https://kubernetes.io/docs/ concepts/storage/persistent- volumes/#selector</pre>
string	v2	Amount of storage to be provisioned for the TimesTen transaction logs. For information on determining the amount of storage needed for the transaction log files, see Storage Provisioning for TimesTen in the Oracle TimesTen In- Memory Database Operations Guide The default is 50Gi. This default value may be suitable when you are experimenting with the product or using it for demonstration purposes. However, in a production environment, consider choosing a value greater than 50Gi. The examples in this book assume a production environment and use a value of 250Gi.
integer	v2	At runtime, if a container's memory usage is more than its percentage of its limit (both as reported by cgroups), the TimesTen Operator generates Events to inform you of this occurrence.
		The memory usage refers to the container's memory allocation.
	metav1.LabelSelector string	string v2

Field	Туре	Earliest Schema Version Supported In	Description
pollingInterval	integer	v2	Determines how often (expressed in seconds) that the TimesTen Operator checks the status of the TimesTenClassic object. For example, if you set this value to 10, the TimesTen Operator checks the status of the TimesTenClassic object every ten seconds.
			This value interacts with unreachableTimeout. The pollingInterval value should be smaller than the unreachableTimeout value.
			The value must be a positive integer (greater than 0). The default is 5.
prometheus	TimesTenClassicSpecSpec Prometheus	v2	Determines if the TimesTen Exporter is deployed. If specified, the Exporter is deployed. The fields for the prometheus object are defined in TimesTenClassicSpecSpecPrometheus

Field	Туре	Earliest Schema Version Supported In	Description
readOnlyRootFilesystem	boolean	v2	Determines if the TimesTen Operator causes the container image to be mounted read-only in TimesTen containers. In addition, determines if the TimesTen Operator provisions an empty directory and mounts it on top of the /tmp directory in all TimesTen containers in all TimesTen Pods. This includes the tt, daemonlog, and exporter containers in each Pod.
			 Valid values: true (default): The TimesTen Operator provisions an empty directory and mounts it on top of the /tmp directory. In addition, the TimesTen Operator creates these containers with container images mounted as read-only. false: The TimesTen Operator does not provision an empty directory and does not create containers with container images
			mounted as read-only. If you provide your own containers, the TimesTen Operator does not automatically mount your container images as read-only.
			If you upgrade a v1 TimesTenClassic object and if the Pods associated with that TimesTenClassic object are replaced, the replacements do not have readOnlyRootFilesystem set.
reexamine	string	v2	When a TimesTenClassic object is in the ManualInterventionRequired state, the TimesTen Operator examines the reexamine value every pollingInterval seconds. If the value has changed since the last iteration for this object, the TimesTen Operator examines the state of the TimesTen containers for this object. See About the ManualInterventionRequired State for Replicated Objects and About Bringing Up One Database.



Field	Туре	Earliest Schema Version Supported In	Description
repCreateStatement	string	_	<pre>The repReturnServiceAttribute and the repStoreAttribute fields provide some control over the CREATE ACTIVE STANDBY statement that you use to configure your active standby pair replication scheme. However, these fields do not provide a mechanism to set all the replication options. The repCreateStatement field provides more control over the active standby pair replication configuration. If you choose to define a replication scheme, you must choose either the repCreateStatement approach or the repReturnServiceAttribute and the repStoreAttribute approach. You cannot use both approaches simultaneously in a single TimesTenClassic object definition. For example, you cannot use the repReturnServiceAttribute fields in a single TimesTenClassic object definition. However, you can use the repReturnServiceAttribute fields in a single TimesTenClassic object definition. Example of using repCreateStatement: apiVersion: timesten.oracle.com/ v4 kind: TimesTenClassic metadata: name: sample spec: ttspec: repCreateStatement: create active standby pair "{{tt-name}}" on "{{tt- node-0}}" RETURN TWOSAFE store "{{tt-rep-port}} FAILTHRESHOLD 10 TIMEOUT 5 </pre>
			DISABLE RETURN ALL 10 store "{{tt-name}}" on

Field	Туре	Earliest Schema Version Supported In	Description
			"{{tt-node-1}}" PORT {{tt-rep-port}} FAILTHRESHOLD 10 TIMEOUT 5 DISABLE RETURN ALL 10
			The TimesTen Operator does the substitutions for you.
			• {{tt-name}}: The name of the TimesTenClassic object. (For example, sample.)
			 {{tt-node-0}}: The fully qualified DNS name of the -0 Pod for the TimesTenClassic object. (For example, sample-0.sample.mynamespace .svc.cluster.local.)
			 {{tt-node-1}}: The fully qualified DNS name of the -1 Pod for the TimesTenClassic object. (For example, sample-1.sample.mynamespace .svc.cluster.local.)
			 {{tt-rep-port}}: The TCP port either chosen by the TimesTen Operator or specified in the repPort field.
			When you use the repCreateStatement field, you have nearly complete control over the replication configuration. The TimesTen Operator executes the statement you define (after substituting a number of values into it). Since the TimesTen Operator is using the CREATE statement that you define, ensure that the statement you specify is correct and appropriate. If the creation of your active standby pair replication scheme fails, your TimesTenClassic object transitions from the Initializing state to the Failed state. You must then delete the TimesTenClassic object to clean up the resources it holds. See About the High Level State of TimesTenClassic Objects.
			The configuration has the following restrictions:
			• Must be an active standby pair.

Field	Туре	Earliest Schema Version Supported In	Description
			Must not be configured with subscribers. See CREATE ACTIVE STANDBY PAIR in the Oracle TimesTen In-Memory Database SQL Reference and Defining an Active Standby Pair Replication Scheme in the Oracle TimesTen In- Memory Database Replication Guide.
replicas	integer	v2	Valid for non-replicated database configurations.
			Number of TimesTen Pods to provision.
			You can only specify replicas if the value of replicationTopology is none.
			The minimum value is 1. The maximum value is 3. The default is 1.
			You cannot change the value of replicas once the TimesTenClassic object is created.
			For more information, see Create TimesTen Classic Databases.
replicationCipherSuite	string	v2	Determines the encryption algorithm to be used by TimesTen replication. If specified, replication traffic is encrypted.
			You can specify one or more cipher suites. Specify the desired cipher suites, comma-separated and in order of preference. The supported cipher suites are as follows:
			 SSL_ECDHE_ECDSA_WITH_AES_12 8_GCM_SHA256
			 SSL_ECDHE_ECDSA_WITH_AES_25 6_GCM_SHA384
			 SSL_RSA_WITH_AES_128_CBC_SH A256
			See Configuration for TLS for Client/ Server in the Oracle TimesTen In- Memory Database Security Guide.

Field	Туре	Earliest Schema Version Supported In	Description
replicationSSLMandatory	integer	v2	 Determines if SSL encryption is mandatory for replication. Valid values: 0 (or not specified): SSL encryption is not mandatory for replication. 1: SSL encryption is mandatory for replication. This field is only examined if replicationCipherSuite is specified. See About Using Certificates with Client/Server in the Oracle TimesTen In-Memory Database Security Guide.
replicationTopology	string	ν2	 Determines if replication should be configured. Valid values: activeStandbyPair (default): The TimesTen Operator configures an active standby pair replication scheme. none: The TimesTen Operator does not configure replication. Instead, it configures a non-replicated configuration and uses the replicas field to determine how many TimesTen Pods to create. For more information, see Create TimesTen Classic Databases.
repPort	integer	v2	TCP port used for replication. The default is 4444.

Туре	Earliest Schema Version Supported In	Description
string	v2	You can use the repReturnServiceAttribute field to specify the <i>ReturnServiceAttribute</i> clause. This clause is part of the syntax for the CREATE ACTIVE STANDBY PAIR statement. The information you specify is included in your active standby pair's CREATE ACTIVE STANDBY PAIR statement by the TimesTen Operator. Do not specify the repReturnServiceAttribute field if you have specified the repCreateStatement field.
		If you do not specify the repReturnServiceAttribute field (or the repCreateStatement field), the default is NO RETURN.
		See CREATE ACTIVE STANDBY PAIR in the Oracle TimesTen In-Memory Database SQL Reference and Defining an Active Standby Pair Replication Scheme in the Oracle TimesTen In- Memory Database Replication Guide for information on the CREATE ACTIVE STANDBY PAIR statement and the ReturnServiceAttribute clause.
integer	v2	Time in seconds a replicated database remains in the recovering replication state as reported by the TimesTen ttRepStateGet built-in procedure. The recovering replication state indicates the database is in the process of synchronizing updates with the active database after a failure. See ttRepStateGet in the Oracle TimesTen In-Memory Database Reference. The default is 30.
	string	string v2

Field	Туре	Earliest Schema Version Supported In	Description
repStoreAttribute	string	v2	You can use the repStoreAttribute field to specify the <i>StoreAttribute</i> clause. This clause is part of the CREATE ACTIVE STANDBY PAIR statement. The information you specify is included in your active standby pair's CREATE ACTIVE STANDBY PAIR statement by the TimesTen Operator. Do not specify the repStoreAttribute field if you have specified the repCreateStatement field.
			If you do not specify the repStoreAttribute field (or the repCreateStatement field), the default is: PORT repPort FAILTHRESHOLD 0.
			If you specify the repStoreAttribute, you must specify the port. This port is used by replication. The port must match the port provided in the repPort field (or must match the default value if repPort is not specified). If the ports do not match, the TimesTenClassic object enters the Failed state.
			See CREATE ACTIVE STANDBY PAIR in the Oracle TimesTen In-Memory Database SQL Reference and Defining an Active Standby Pair Replication Scheme in the Oracle TimesTen In- Memory Database Replication Guide for information on the CREATE ACTIVE STANDBY PAIR statement and the StoreAttribute clause.
resetUpgradeState	string	v2	The resetUpgradeState field allows an online upgrade to be canceled.

Field	Туре	Earliest Schema Version Supported In	Description
rollingUpdatePartition	integer	v2	Specific to upgrading a non-replicated configuration. Kubernetes upgrades Pods with an ordinal value that is greater than or equal to the rollingUpdatePartition value. Pods with an ordinal value that is less than rollingUpdatePartition are not upgraded.
			For example, if you have three non- replicated Pods (replicas = 3 and Pods are samplerep-0, samplerep-1, and samplerep-2) and you set rollingUpdatePartition to 1, the samplerep_1 and samplerep-2 Pods are upgraded, but the samplerep-0 Pod is not. The default is 0.
			For more information, see Perform Upgrades in this book and Partitioned rolling updates in the Kubernetes documentation.
stopManaging	string	v2	If you change the value of stopManaging for the TimesTenClassic object, the TimesTen Operator places the object in the ManualInterventionRequired state. See About the ManualInterventionRequired State for Replicated Objects and About Bringing Up One Database.
storageClassName	string	v2	Name of the storage class that is used to request persistent volumes for the TimesTen database. There is no default. You must specify
storageSelector	metav1.LabelSelector	v2	the name of the storage class. When choosing to use a persistent volume to store a TimesTen database, the primary determinant of what volumes to use is the StorageClassName that you specify. You can optionally specify a label selector by using the storageSelector field. This label selector further filters the set of volumes. See: https://kubernetes.io/docs/ concepts/storage/persistent- volumes/#selector

Field	Туре	Earliest Schema Version Supported In	Description
storageSize	string	v2	Amount of storage to be provisioned for TimesTen and the database. For information on determining the amount of storage needed for TimesTen, see Storage Provisioning for TimesTen in the Oracle TimesTen In-Memory Database Operations Guide.
			The default is 50Gi. This default value may be suitable when you are experimenting with the product or using it for demonstration purposes. However, in a production environment, consider choosing a value greater than 50Gi. The examples in this book assume a production environment and use a value of 250Gi.
storePassword	string	v4	The TimesTen Operator stores its security certificates in Java Keystores. In order to create and use Java Keystores, they must be secured with a known password (storePassword). The password must be known by both the TimesTen Operator who creates and reads the keystore and the TimesTen Agent who also reads it.
			The default value is a hard-coded string that is not documented. Although not recommended, you can use the storePassword datum to set the password to a different value than the default and control the password setting. Once set, you cannot change storePassword. If it is changed, the TimesTen Operator generates a warning in the form of a Kubernetes Event and ignores the changed value.
terminationGracePeriod	integer	v2	Amount of time (in seconds) that Kubernetes waits for a Pod to gracefully shut down before being forcefully terminated. The default varies depending on your configuration: • Replicated (Active Standby Pair):
			 Non-replicated (Active Standby Pair). Non-replicated: Default is 300 seconds.

Field	Туре	Earliest Schema Version Supported In	Description
unreachableTimeout	integer	v2	Number of seconds that a TimesTen instance or TimesTen database is unavailable before the TimesTen Operator takes action to fail over or otherwise recover from the issue.
			This value interacts with pollingInterval. The pollingInterval value should be smaller than the unreachableTimeout value.
			The value must be a positive integer (greater than 0). The default is 30.
upgradeDownPodTimeout int	integer	v2	Maximum amount of seconds that the TimesTenClassic object remains in the WaitingForActive state. After this period of time, if the TimesTenClassic object is still in the WaitingForActive state, it transitions to the ManualInterventionRequired state.
			The default is 0 (which means there is no timeout. The TimesTenClassic object waits forever, if required).
			For information on the WaitingForActive and the ManualInterventionRequired states, see About the High Level State of TimesTenClassic Objects.

TimesTenClassicSpecSpecClientTLS

TimesTenClassicSpecSpecClientTLS appears in TimesTenClassicSpecSpec.

The following table describes the syntax for TimesTenClassicSpecSpecClientTLS.

Field	Туре	Earliest Schema Version Supported In	Description
auto	boolean	v4	Determines if the TimesTen Operator automatically creates self-signed certificates and configures TimesTen to use those certificates for client/server encryption. Valid values are the following:
			 true: The TimesTen Operator automatically creates and configures self-signed certificates for client/server encryption. false (default): The TimesTen does not automatically create and configure self-signed certificates for client/server encryption.
			If you do not specify the auto datum, and any other field is specified in the .spec.ttspec.clientTLS clause, the default value for the auto datum is false.

Table 20-4 TimeTenClassicSpecSpecClientTLS

Field	Туре	Earliest Schema Version Supported In	Description
ciphersuites	string	v4	Defines the cipher suite(s) used for client/server communication.
			You can specify one or more cipher suites. Specify the desired cipher suites, comma-separated, and in order of preference.
			The supported cipher suites are as follows:
			• SSL_ECDHE_ECDSA_WITH_AES_12 8_GCM_SHA256
			• SSL_ECDHE_ECDSA_WITH_AES_25 6_GCM_SHA384
			• SSL_RSA_WITH_AES_128_CBC_SP A256
			The TimesTen Operator first checks to see if there is a ciphersuites entry in the db.ini file.
			 If there is an entry, the TimesTen Operator uses it.
			 If there is no entry, the TimesTen Operator uses the value specified in this datum.
			 If .spec.ttspec.clientTLS.au to is true, and there is no value specified in either the db.ini file or in this datum, the TimesTen Operator sets the value to SSL_ECDHE_ECDSA_WITH_AES_12 8_GCM_SHA256.
			The TimesTen Operator adds the ciphersuites value to the TimesTen Server's sys.odbc.ini file.
			See Configuration for TLS for Client/ Server in the Oracle TimesTen In- Memory Database Security Guide.
eccurve	string	v4	Defines the size of the elliptical curve.
			The supported values are as follows: • p256
			• p384 (default)
			• p521
			See the -eccurve option of the TimesTen ttCreateCerts utility in the Oracle TimesTen In-Memory Database Reference.

Table 20-4 (Cont.) TimeTenClassicSpecSpecClientTLS



Field	Туре	Earliest Schema Version Supported In	Description
encryption	string	v4	Defines the encryption setting for client/ server access.
			 The supported values are as follows: accepted rejected required The TimesTen Operator first checks to see if there is an encryption entry in the db.ini file. If there is an entry, the TimesTen Operator uses it. If there is no entry, the TimesTen Operator uses the value specified in this datum. If .spec.ttspec.clientTLS.au to is true, and there is no value specified in either the db.ini file or in this datum, the TimesTen Operator sets the value to accepted. The TimesTen Operator adds the encryption value to the TimesTen Server's sys.odbc.ini file.
			See Configuration for TLS for Client/ Server in the Oracle TimesTen In- Memory Database Security Guide.
signAlg	string	v4	Defines the elliptical curve signing algorithm.
			The supported values are as follows: • ecdsasha256
			• ecdsasha384 (default)
			• ecdsasha512
			See the -sign_alg option of the TimesTen ttCreateCerts utility in the Oracle TimesTen In-Memory Database Reference.
validity	integer	v4	Defines the number of days the created certificate is valid.
			The minimum is 30 and the maximum is 9999.
			The default is 3650.
			See the -validity option of the TimesTen ttCreateCerts utility in the Oracle TimesTen In-Memory Database Reference.

Table 20-4 (Cont.) TimeTenClassicSpecSpecClientTLS



TimesTenClassicSpecSpecPrometheus

TimesTenClassicSpecSpecPrometheus appears in TimesTenClassicSpecSpec.

The following table describes the syntax for TimesTenClassicSpecSpecPrometheus.

Table 20-5	TimesTenClassicSpecSpecPrometheus syntax
------------	--

Field	Туре	Earliest Schema Version Supported In	Description
certSecret	string	v2	When using https to serve TimesTen metrics, a Kubernetes Secret is required.
			If you create your own Kubernetes Secret, use the certSecret field to specify the name of this Secret. This Secret contains an Oracle Wallet that holds the necessary certificates used by the TimesTen exporter for serving TimesTen metrics by https. See Create Your Own Oracle Wallet, Certificates, and Secrets for Exposing TimesTen Metrics.
			If you want the TimesTen Operator to automatically create an Oracle Wallet, certificates, and Kubernetes Secrets that are required for https, do not specify the certSecret field. We recommend this approach. See About Transport Layer Security (mutual TLS) Certificates for TimesTen Metrics.
createPodMonitors	boolean	v2	Determines if the TimesTen Kubernetes Operator creates PodMonitors.
			 Valid values: true (or not specified): The TimesTen Operator creates PodMonitors. false: The TimesTen Operator does not create PodMonitors. See Expose TimesTen Metrics with the TimesTen Kubernetes Operator.

Field	Туре	Earliest Schema Version Supported In	Description
insecure	boolean	v2	Determines if the TimesTen exporter is started with no authentication or with Transport Layer Security (mutual TLS). Valid values:
			 true: The TimesTen exporter is started with no authentication and serves data by http. false (or not specified): The TimesTen exporter is started with mutual TLS and serves data by https.
			For more information, see About Using http or https for TimesTen Metrics.
			For information about the TimesTen exporter, see About the TimesTen Exporter in the Oracle TimesTen In- Memory Database Monitoring and Troubleshooting Guide.
limitRate	integer	v2	Determines the limit of GET requests per minute that the TimesTen exporter accepts. The value can be any integer value from 1 to 15.
			The default is 10.
port	integer	v2	Port on which the TimesTen exporter listens.
			The default is 8888.

Field	Туре	Earliest Schema Version Supported In	Description
publish	boolean	Supported	 Determines if the TimesTen Operator provisions a TimesTen exporter container. Valid values: true (or not specified): The TimesTen Operator provisions a TimesTen exporter container. false: The TimesTen Operator does not provision a TimesTen exporter container. If you do not specify the publish field, and any other field is specified in the .spec.ttspec.prometheus clause, the default value for the publish field is true. If you do not specify the .spec.ttspec.prometheus clause and the TimesTen release is 22.1 or greater, the default value for the publish field is dependent on the value of the EXPOSE_METRICS TimesTen Operator environment variable: If EXPOSE_METRICS is "1" (or not
			 specified), the TimesTen Operator treats the publish field as true. If EXPOSE_METRICS is "0", the TimesTen Operator treats the publish field as false. For more information, see Expose TimesTen Metrics with the TimesTen Kubernetes Operator and TimesTen Kubernetes Operator Environment Variables.

TimesTenClassicStatus

TimesTenClassicStatus appears in TimesTenClassic. The Operator stores various persistent information in TimesTenClassicStatus.

The output of the kubectl get and kubectl describe commands display information in TimesTenClassicStatus. This information includes the following:

• awtBehindMb: a field that is present only if AWT (Asynchronous WriteThrough) is in use. The field represents how many megabytes of log is present in TimesTen that has not yet been pushed to Oracle Database. For more information on AWT cache group, see Overview of Cache Groups in the Oracle TimesTen In-Memory Database Cache Guide.



- High Level state of the Active Standby Pair: a string that describes the High Level state of the active standby pair.
- Detailed state of TimesTen in each Pod, which includes the following:
 - Is the TimesTen agent running?
 - Is the TimesTen main daemon running?
 - Is the TimesTen replication agent running?
 - Is the TimesTen cache agent running?
 - Is there a database in the instance?
 - Is the database loaded?
 - Is the database updatable or read only?
 - Is there a replication scheme in the database?
 - What is the replication state of this database?
 - What does this database think the replication state of its peer is?
 - What is the role for TimesTen in this Pod (active or standby)?
 - What is the High Level state of the Pod?

Note:

Unknown values can occur if, for example, the agent is not running or a Pod is unavailable.

About the TimesTenScaleout Object Type

The TimesTenScaleout object type is defined using the following object definitions:

- TimesTen Scaleout
- TimesTenScaleoutSpec
- TimesTenScaleoutSpecSpec
- TimesTenScaleoutSpecSpecPrometheus
- TimesTenScaleoutStatus

TimesTen Scaleout

An object of type TimesTenScaleout describes the metadata for a TimesTen grid and the TimesTen database within the grid in TimesTen Scaleout.

The following table describes the syntax for the TimesTenScaleout object type:



Table 20-6 TimesTenScaleout syntax

Field	Туре	Description	
apiVersion	string	Versioned schema of this representation of an obje	
		The value can be timesten.oracle.com/v1 or timesten.oracle.com/v2.	
kind	string	Type of object (in this example, TimesTenScaleout).	
metadata	ObjectMeta	Metadata about the object, such as its name. For information on ObjectMeta, see:	
		https://kubernetes.io/docs/reference/ kubernetes-api/	
spec	TimesTenScaleoutSpec	Desired configuration of the TimesTen Scaleout grid and the databases within the grid.	
status	TimesTenScaleoutStatus	Current status of the Pods in this TimesTenScaleout object as well as the status of various TimesTen components within those Pods. This data may be out of date by some window of time.	

TimesTenScaleoutSpec

TimesTenScaleoutSpec appears in TimesTenScaleout. The following table describes the syntax for TimesTenScaleoutSpec:

Table 20-7 TimesTenScaleoutSpec syntax

Field	Туре	Description
ttspec	TimesTenScaleoutSpecSpec	Specific TimesTen attributes for deploying a grid in TimesTen Scaleout.
dataTemplate	Array of PodTemplateSpec	Array of specifications for the Pods that contain the TimesTen Scaleout data instances. This field is optional. If specified, there must be k entries in the array. Each entry is the template for the Pods in a different data space. For example, if k is set to 2, then the first entry is for data space 1, and the second entry is for data space 2.
		Use dataTemplate to pass affinity and other settings to Kubernetes. For information about PodTemplateSpec, see:
		<pre>https://kubernetes.io/docs/reference/ kubernetes-api/</pre>
dataVolumeClaimTemplates	Array of arrays of PersistentVolumeClaim	Array of arrays of Volume Claim Templates to be added to just the data instance Pods. There is one list (of lists) per data space.
mgmtTemplate	PodTemplateSpec	Specification for the Pod that contains the TimesTen Scaleout management instance. Use mgmtTemplate to pass affinity and other settings to Kubernetes. For information about PodTemplateSpec, see:
		<pre>https://kubernetes.io/docs/reference/ kubernetes-api/</pre>


Field	Туре	Description
mgmtVolumeClaimTemplates	Array of PersistentVolumeClaim	Array of Volume Claim Templates to be added to just the management instance Pod.
volumeClaimTemplates	PersistentVolumeClaim	TimesTen automatically provisions PersistentVolumeClaims (PVCs) for /tt (and for /ttlog, if specified). If you have applications that are running in containers in the TimesTen Pods, and those applications require additional PVCs, specify them in this field. For information on PersistentVolumeClaim, see:
		<pre>https://kubernetes.io/docs/reference/ kubernetes-api/</pre>
zookeeperTemplate	Array of PodTemplateSpec	Specification for the Pods that contains the TimesTen Scaleout ZooKeeper instances. Use zookeeperTemplate to pass affinity and other settings to Kubernetes. For information about PodTemplateSpec, see:
		<pre>https://kubernetes.io/docs/reference/ kubernetes-api/</pre>
<pre>zookeeperVolumeClaimTempl ates</pre>	Array of PersistentVolumeClaim	Array of Volume Claim Templates to be added to just the ZooKeeper Pods.

Table 20-7 (Cont.) TimesTenScaleoutSpec syntax

TimesTenScaleoutSpecSpec

TimesTenScaleoutSpecSpec appears in TimesTenScaleoutSpec.

The following table describes the syntax for TimesTenScaleoutSpecSpec. There are some fields of type quantity. The specified value is of the same format as Kubernetes resource limits. For example, 200Gi, 200G, 1000Mi, 1000M, and so on.

Note:

There are datum that are reserved for internal use and are not documented in this table. The names of these datum typically begin with zz. For example, zzTestInfo is reserved for internal use.



Field	Туре	Description
additionalMemoryRequest	quantity	The amount of memory to request in addition to whatever is required for an element of the TimesTen database.
		This memory is used for the TimesTen daemon, subdaemons, agents, and the Client/Server server.
		This value is added to databaseMemorySize that was either specified by you or calculated. The sum is the memory request to Kubernetes.
		The default is 2Gi.
agentGetTimeout	integer	Time in seconds that the Operator waits for an https GET request to be processed by the TimesTen agent. This includes the TCP and the TLS times as well as the time it takes for the TimesTen agent to implement the GET request.
		The default is 60. A value of 0 indicates that there is no timeout. If the timeout is exceeded, the Operator considers the agent to be down.
agentPostTimeout	integer	Time in seconds that the Operator waits for an https POST request to be processed by the TimesTen agent. This includes the TCP and the TLS times as well as the time it takes for the TimesTen agent to implement the POST request. The POST requests may take a long time and the time may be proportional to the size of the database. (An example is a POST request to duplicate a database from the active to the standby.)
		The default is 600. A value of 0 indicates that there is no timeout. If the timeout is exceeded, the Operator considers the POST request to have failed.
agentTcpTimeout	integer	Time in seconds that the Operator waits for a TCP handshake when communicating with the TimesTen agent.
		The default is 10. A value of 0 indicates that there is no timeout. If the timeout is exceeded, the Operator considers the agent to be down.
agentTlsTimeout	integer	Time in seconds that the Operator waits for a TLS (https) credential exchange when communicating with the TimesTen agent.
		The default is 10. A value of 0 indicates that there is no timeout. If the timeout is exceeded, the Operator considers the agent to be down.

Table 20-8 TimesTenScaleoutSpecSpec



Field	Туре	Description
automaticMemoryRequests	boolean	Determines if the Operator attempts to set appropriate memory limits and requests for TimesTen Pods.
		 Valid values: true (default): The Operator attempts to set memory limits and requests. false: The Operator does not set memory limits and requests.
cacheCleanup	boolean	Determines if the metadata in the Oracle Database should be cleaned up when this TimesTenClassic object is deleted. Use for TimesTen Cache only.
		Valid values:
		• true (or not specified): The metadata is cleaned up.
		 false: The metadata is not cleaned up.
		See Dropping Oracle Database Objects Used by Cache Groups with Autorefresh in the Oracle TimesTen In-Memory Database Cache Guide.
daemonLogCPURequest	quantity	The amount of CPU requested for the daemonlog container.
		The default is 200m (one-fifth of a CPU).
daemonLogMemoryRequest	quantity	The amount of memory requested for the daemonlog container.
		The default is 200Mi.
daemonLogSidecar	boolean	Determines if a daemon log container is created in each TimesTen Pod. This container writes the TimesTen daemon logs (from ttmesg.log) to stdout. This causes Kubernetes to record these logs. Valid values:
		 true (or not specified): A daemon log container is created. false: A daemon log container is not created.
databaseCPURequest	Quantity	Specify this value to tell the Operator how much CPU your tt containers require. This includes CPU used by the TimesTen daemon, subdaemons, replication agents, cache agents, and the Client/Server server.
		There is no default.

Table 20-8 (Cont.) TimesTenScaleoutSpecSpec



Field	Туре	Description
databaseMemorySize	Quantity	You can specify this value to tell the Operator how much shared memory an element of your database requires.
		If you specify a value, that value will be used. If you do not specify a value, the Operator attempts to determine the required size from the provided db.ini file
		If the Operator cannot determine the database size, the value 580911104 is used. This is the size required for a default database with a PermSize of 200Mbyte, rounded up to 2Mi. This may be useful for experimentation, but is likely insufficient for production purposes.
		TimesTen recommends that you provide a db.ini file to the Operator by using a Configmap or Secret, and that you not specify databaseMemorySize.
		Note:
		If you provide a db.ini file by using an init container, you must specify databaseMem orySize.
dataStorageClassName	string	Name of the storage class that is used to request persistent volumes for the elements of the TimesTen database in the grid. If not specified, the default is the value of storageClassName.
dataStorageSelector	metav1.LabelSelector	When choosing to use a persistent volume to store the elements of a TimesTen database in the grid, the primary determinant of what volumes to use is the dataStorageClassName that you specify. You can optionally specify a label selector by using the dataStorageSelector field. This label selector further filters the set of volumes. See:
		<pre>https://kubernetes.io/docs/ concepts/storage/persistent- volumes/#selector</pre>

Table 20-8 (Cont.) TimesTenScaleoutSpecSpec



Field	Туре	Description
dataStorageSize	string	Amount of storage to be provisioned for each element of the TimesTen database in the grid.
		The default is 50Gi. This default value may be suitable when you are experimenting with the product or using it for demonstration purposes. However, in a production environment, consider choosing a value greater than 50Gi. The examples in this book assume a production environment and use a value of 250Gi.
dbConfigMap	array of strings	Name of one or more ConfigMaps to be included in a projected volume. This projected volume is mounted as / ttconfig in the TimesTen containers. If you do not specify dbConfigMap or dbSecret, you must place the metadata files into the /ttconfig directory by using other means. See Populate the /ttconfig Directory.
dbSecret	array of strings	Name of one or more Secrets to be included in a projected volume. This projected volume is mounted as / ttconfig in the TimesTen containers. If you do not specify dbSecret or dbConfigMap, you must place the metadata files into the /ttconfig directory by using other means. See Populate the / ttconfig Directory.
exporterCPURequest	quantity	The amount of CPU requested for the exporter container (if provisioned).
		The default is 200m (one-fifth of a CPU).
exporterMemoryRequest	quantity	The amount of memory requested for the exporter container (if provisioned).
		The default is 200Mi.
image	string	Name of the TimesTen image that is executed in the created containers.
		There is no default. You must specify the name of the image.
imagePullPolicy	string	Determines if and when Kubernetes pulls the TimesTen image from the image repository.
		Valid values:
		• Always
		• IfNotPresent (default)
		Never
		Note: Values are case sensitive.

Table 20-8 (Cont.) TimesTenScaleoutSpecSpec



Field	Туре	Description
imagePullSecret	string	Image pull secret that is used to authenticate and give permission to Kubernetes to fetch the specified TimesTen image from its image repository. There is no default. You must specify the
		name of the image pull secret.
k	integer	K-Safety value for this TimesTen grid. This value determines the number of copies of data for your TimesTen database. This value also determines the number of StatefulSets that the TimesTen Operator creates. A StatefulSet provides the Pods that are used to implement a single data space in the grid. For example, if you set k to 2, the Operator creates two StatefulSets. One StatefulSet provides the Pods for the data instances in data space one. The second StatefulSet provides the Pods for the data instances in data space two. The Operator also creates a StatefulSet for the management instance and a StatefulSet for ZooKeeper. The default is 2.
		For more information on K-Safety, see K- Safety in the Oracle TimesTen In-Memory Database Scaleout User's Guide.
logStorageClassName	string	Name of the storage class that is used to request persistent volumes for the TimesTen database transaction log files. This field is optional and is valid only for data instances.
logStorageSelector	metav1.LabelSelector	When choosing to use a persistent volume to store the TimesTen transaction log files, the primary determinant of what volumes to use is the logStorageClassName that you specify. You can optionally specify a label selector by using the logStorageSelector field. This label selector further filters the set of volumes. Valid only for data instances. See: https://kubernetes.io/docs/
		<pre>concepts/storage/persistent- volumes/#selector</pre>
logStorageSize	string	Amount of storage to be provisioned for the TimesTen transaction log files. This value is for each element of the TimesTen database in the grid. Valid only for data instances.

Table 20-8 (Cont.) TimesTenScaleoutSpecSpec

Field	Туре	Description
memoryWarningPercent	integer	At runtime, if a container's memory usage is more than its percentage of its limit (both as reported by cgroups), the Operator generates Events to inform you of this occurrence.
		The memory usage refers to the container's memory allocation.
		The default is 90.
mgmtCPURequest	quantity	The amount of cpu requested for the tt container of management instances.
		The default is 1.
mgmtMemoryRequest	quantity	The amount of memory requested for the tt container of management instances.
		The default is 3Gi.
mgmtStorageClassName	string	Name of the storage class that is used to request persistent volumes for the database of the management instance. If not specified, the default is the value of storageClassName.
mgmtStorageSelector	metav1.LabelSelector	When choosing to use a persistent volume to store the database of the management instance, the primary determinant of what volumes to use is the mgmtStorageClassName that you specify. You can optionally specify a label selector by using the mgmtstorageSelector field. This label selector further filters the set of volumes. If you do not specify mgmtstorageSelector, the value is the value of StorageSelector. See:
		https://kubernetes.io/docs/ concepts/storage/persistent- volumes/#selector
mgmtStorageSize	string	Amount of storage to be provisioned for the database of the management instance. The default is 50Gi.

Table 20-8 (Cont.) TimesTenScaleoutSpecSpec



Field	Туре	Description
nReplicaSets	integer	Number of replica sets in the grid. A replica set contains k elements (where each element is an exact copy of the other elements in the replica set). The nReplicaSets value also determines the number of replicas for each StatefulSet. For example, if you set k to 2, the TimesTen Operator creates two StatefulSets for the data instances. If you set nReplicaSets to 3, each StatefulSet contains three replicas, and the total number of replica sets in the grid is three. The default is 1.
		For more information on replica sets, see Understanding Replica Sets in the Oracle TimesTen In-Memory Database Scaleout User's Guide.
nZookeeper	integer	Number of ZooKeeper Pods to provision in a StatefulSet. Valid values:
		 1 3 (default) 5
pollingInterval	integer	Determines how often (expressed in seconds) that the Operator checks the status of the TimesTenScaleout object. For example, if you set this value to 10, the Operator checks the status of the TimesTenScaleout object every ten seconds.
		The value must be a positive integer (greater than 0). The default is 5.
prometheus	TimesTenScaleoutSpecSpecProm etheus	Determines if the TimesTen Exporter is deployed. If specified, the Exporter is deployed. The datum for the prometheus object are defined in TimesTenScaleoutSpecSpecPrometheus.
reexamine	string	When a TimesTenScaleout object is in the ManualInterventionRequired state, the Operator examines the reexamine value every pollingInterval seconds. If the value has changed since the last iteration for this object, the Operator reexamines the TimesTenScaleout object and attempts to resume management of the object.

Table 20-8 (Cont.) TimesTenScaleoutSpecSpec

Field	Туре	Description
replicaSetRecovery	string	Controls the behavior of the Operator when a total replica set failure occurs.
		Valid values:
		 Restart (default): The Operator forcibly unloads and reloads the database when a total replica set failure occurs. Manual: The Operator changes the state of the TimesTenScaleout object to ManualInterventionRequired when a total replica set failure occurs.
stopManaging	string	If you change the value of stopManaging for the TimesTenScaleout object, the Operator places the object in the ManualInterventionRequired state.
storageClassName	string	Name of the storage class that is used to request persistent volumes for the TimesTen database.
		If the value for storageClassName is the same as the value for dataStorageClassName, for mgmtStorageClassName, and for zookeeperStorageClassName, you can just specify storageClassName. In this case, the value for dataStorageClassName, for mgmtStorageClassName, and for zookeeperStorageClassName is set to the value of storageClassName.

Table 20-8 (Cont.) TimesTenScaleoutSpecSpec

Field	Туре	Description
storageSelector	metav1.LabelSelector	When choosing to use a persistent volume to store an element of a TimesTen database in a grid, the primary determinant of what volumes to use is the StorageClassName that you specify. You can optionally specify a label selector by using the storageSelector field. This label selector further filters the set of volumes.
		If the value for storageSelector is the same as the value for dataStorageSelector, for mgmtStorageSelector, and for zookeeperStorageSelector, you can just specify storageSelector. In this case, the value for dataStorageSelector, for mgmtStorageSelector, and for zookeeperStorageSelector is set to the value of storageSelector. See:
		<pre>https://kubernetes.io/docs/ concepts/storage/persistent- volumes/#selector</pre>
storageSize	string	Amount of storage to be used for each element of a TimesTen database in the grid. If the value for storageSize is the same as the value for dataStorageSize, for mgmtStorageSize, and for zookeeperStorageSize. Jour can just specify storageSize. In this case, the value for dataStorageSize, for mgmtStorageSize, and for zookeeperStorageSize is set to the value of storageSize.
		The default is 50Gi. For information on determining the amount of storage needed for TimesTen, see Storage Provisioning for TimesTen in the Oracle TimesTen In-Memory Database Operations Guide.
zookeeperCPURequest	quantity	The amount of cpu requested for the zookeeper container of Zookeeper Pods.
zookeeperMemoryRequest	quantity	The amount of memory requested for the zookeeper container of Zookeeper Pods. The default is 1Gi.

Table 20-8 (Cont.) TimesTenScaleoutSpecSpec



Field	Туре	Description
zookeeperStorageClassName	string	Name of the storage class that is used to request persistent volumes for ZooKeeper's persistent data.
		If not specified, the default is the value of storageClassName.
zookeeperStorageSelector	metav1.LabelSelector	When choosing to use a persistent volume to store ZooKeeper's persistent data, the primary determinant of what volumes to use is the zookeeperStorageClassName that you specify. You can optionally specify a label selector by using the zookeeperStorageSelector field. This label selector further filters the set of volumes. See:
		https://kubernetes.io/docs/ concepts/storage/persistent- volumes/#selector
zookeeperStorageSize	string	Amount of storage to be provisioned for ZooKeeper's persistent data.
		The default is 50Gi.

Table 20-8 (Cont.) TimesTenScaleoutSpecSpec

TimesTenScaleoutSpecSpecPrometheus

TimesTenScaleoutSpecSpecPrometheus appears in TimesTenScaleoutSpecSpec . The following table describes the syntax for TimesTenScaleoutSpecSpecPrometheus.

Table 20-9	TimesTenScaleoutSpecSpecPrometheus syntax
------------	---

Field	Туре	Description
certSecret	string	When using https to serve TimesTen metrics, a Kubernetes Secret is required.
		If you create your own Kubernetes Secret, use the certSecret datum to specify the name of this Secret. This Secret contains an Oracle Wallet that holds the necessary certificates used by the TimesTen exporter for serving TimesTen metrics by https. See Create Your Own Oracle Wallet, Certificates, and Secrets for Exposing TimesTen Metrics.
		If you want the TimesTen Operator to automatically create an Oracle Wallet, certificates, and Kubernetes Secrets that are required for https, do not specify the certSecret datum. We recommend this approach. See About Transport Layer Security (mutual TLS) Certificates for TimesTen Metrics.

Field	Туре	Description
createPodMonitors	boolean	Determines if the TimesTen Kubernetes Operator creates PodMonitors.
		Valid values:
		 true (or not specified): The TimesTen Operator creates PodMonitors. false: The TimesTen Operator does not create PodMonitors.
		See Expose TimesTen Metrics with the TimesTen Kubernetes Operator for details.
insecure	boolean	Determines if the TimesTen exporter is started with no authentication or with Transport Layer Security (mutual TLS).
		Valid values:
		 true: The TimesTen exporter is started with no authentication and serves data by http.
		 false (or not specified): The TimesTen exporter is started with mutual TLS and serves data by https.
		For more information, see About Using http or https for TimesTen Metrics.
		For information about the TimesTen exporter, see About the TimesTen Exporter in the Oracle TimesTen In-Memory Database Monitoring and Troubleshooting Guide.
limitRate	integer	Determines the limit of GET requests per minute that the TimesTen exporter accepts. The value can be any integer value from 1 to 15.
		The default is 10.
port	integer	Port on which the TimesTen exporter listens.
		The default is 8888.

Table 20-9 (Cont.) TimesTenScaleoutSpecSpecPrometheus syntax

Field	Туре	Description
publish	boolean	Determines if the TimesTen Operator provisions a TimesTen exporter container.
		Valid values:
		 true (or not specified): The TimesTen Operator provisions a TimesTen exporter container. false: The TimesTen Operator does
		not provision a TimesTen exporter container.
		If you do not specify the publish datum, and any other datum is specified in the .spec.ttspec.prometheus clause, the default value for the publish datum is true.
		 If you do not specify the .spec.ttspec.prometheus clause and the TimesTen release is 22.1 or greater, the default value for the publish datum is dependent on the value of the EXPOSE_METRICS TimesTen Operator environment variable: If EXPOSE_METRICS is "1" (or not specified), the TimesTen Operator treats the publish datum as true. If EXPOSE_METRICS is "0", the TimesTen Operator treats the publis datum as false.
		For more information, see Expose TimesTen Metrics with the TimesTen Kubernetes Operator and TimesTen Kubernetes Operator Environment Variables.

Table 20-9 (Cont.) TimesTenScaleoutSpecSpecPrometheus syntax

TimesTenScaleoutStatus

TimesTenScaleoutStatus appears in TimesTen Scaleout. The Operator stores various persistent information in TimesTenScaleoutStatus.

The output of the ${\tt kubectl get}$ and ${\tt kubectl describe}$ commands display information in TimesTenScaleoutStatus.

The information includes the following:

- High Level state of the TimesTen grid and the database within the grid.
- Detailed state of TimesTen in each Pod, including:
 - Are the TimesTen agents running?
 - Are the TimesTen daemons running?
 - Is there a database in the instance?
 - Is the database loaded?



- Is the database updatable or read only?
- What is the High Level state of the Pods?

Note:

Unknown values can occur if, for example, the agent is not running or a Pod is unavailable.

21

Helm Charts for the TimesTen Kubernetes Operator

There are four Helm charts that allow you to deploy the TimesTen CRD, the TimesTen Operator, and TimesTenClassic objects. Each chart contains variables and default values that are specific to the chart. The names of the charts are as follows:

- ttcrd: Deploys the TimesTen CRD in your Kubernetes cluster. You cannot change the values of the variables in this chart.
- ttoperator: Deploys the TimesTen Operator in a namespace in your Kubernetes cluster. The TimesTen Operator runs at namespace-scope and services TimesTenClassic objects in this namespace. You have the option of defining your own values for variables in this chart.
- ttclusteroperator: Deploys the TimesTen Operator in the timesten-operator namespace in your Kubernetes cluster. The TimesTen Operator runs at cluster-scope and services TimesTenClassic objects in all namespaces in the cluster. The timestenoperator namespace is created by the TimesTen Operator and cannot be modified. You have the option of defining your own values for variables in this chart.
- ttclassic: Deploys a TimesTenClassic object. The TimesTen Operator creates TimesTen Classic databases based on the object's definition. You have the option of defining your own values for variables in this chart.

The following sections describe the values and defaults values for the ttoperator, ttclusteroperator, and ttclassic charts:

- The ttoperator Chart
- The ttclusteroperator Chart
- The ttclassic Chart

For detailed information about using Helm, see Use Helm in Your TimesTen Kubernetes Operator Environment.

The ttoperator Chart

The ttoperator chart deploys the TimesTen Operator in a namespace in your Kubernetes cluster. The TimesTen Operator runs at namespace-scope and services TimesTenClassic objects in this namespace.

The following table describes the variables and default values for the ttoperator chart. Indentation and case sensitivity must be correct for each variable. There may be some variables that exist in the product, but are not documented.



podAntiAffinity parameters. There is no default. However, if you are using a multi-architecture cluster that consists of amd64 and arm64 nodes, you must instruct Kubernetes to deploy the TimesTen Operator on a specific architecture. top Hee inst Tim ma afi r rec edi top Val Val •	scription
Hei	<pre>finity: nodeAffinity: quiredDuringSchedulingIgnor DuringExecution: nodeSelectorTerms: - matchExpressions: - key: pology.kubernetes.io/zone operator: In values: - antarctica-east1 - antarctica-east1 - antarctica-west1 re is an example that shows how to ruct Kubernetes to deploy the nesTen Operator and the objects it nages on arm64 nodes: finity: nodeAffinity: guiredDuringSchedulingIgnor DuringExecution: nodeSelectorTerms: - matchExpressions: - key: pology.kubernetes.io/zone operator: In values: ["arm64"] id values for values are: "amd64": Use this for TimesTen Operators that you want to run on amd64 nodes. "arm64": Use this for TimesTen Operators that you want to run on arm64": Use this for TimesTen Operators that you want to run on arm64 nodes. "arm64": Use this for TimesTen Operators that you want to run on arm64 nodes.</pre>
afi	e is a complete affinity section with le affinity and pod affinity sections: finity: {}
rec	<pre>nodeAffinity: quiredDuringSchedulingIgnor DuringExecution:</pre>

Variable	Description	Description
		- key:
		"kubernetes.io/arch"
		operator: In
		values: ["amd64"]
		nodeAffinity:
		requiredDuringSchedulingIgnor
		edDuringExecution:
		nodeSelectorTerms:
		- matchExpressions:
		- key:
		topology.kubernetes.io/zone
		operator: In
		values:
		- antarctica-east1
		- antarctica-west1
		preferredDuringSchedulingIgn
		redDuringExecution:
		- weight: 1
		preference:
		matchExpressions:
		- key: another-node-
		label-key
		operator: In
		values:
		- another-node-
		label-value
		podAntiAffinity:
		requiredDuringSchedulingIgno
		edDuringExecution:
		- labelSelector:
		matchExpressions:
		- key: app
		operator: In
		values:
		- exampleApp
		topologyKey:
		"kubernetes.io/hostname"
		podAffinity:
		preferredDuringSchedulingIgno
		redDuringExecution:
		- weight: 50
		podAffinityTerm:
		labelSelector:
		matchExpressions:



Variable	Description	Description
		<pre>- key: app operator: In values: - exampleApp topologyKey:</pre>
		"kubernetes.io/hostname"
annotations	Variable that defines a list of annotations to be applied to a TimesTen Operator Deployment and the Pods that the TimesTen Operator manages.	annotations: x: y
	There is no default.	
connectionManager	Variable that determines if the TimesTen Operator's Connection Manager is used.	connectionManager: true
	 Values are as follows: true (default): The TimesTen Operator's Connection Manager is used. 	
	• false: The TimesTen Operator's Connection Manager is not used.	
	See About the Connection Manager.	
createClusterRole	Variable that determines if cluster level roles are created. Default: true	createClusterRole: true
	By default or if you specify true, the chart creates additional permissions and privileges for the TimesTen Operator.	
	If you specify false, the chart does not create the additional permissions and privileges. You must create these objects by running the kubectl create	
	service_account_cluster.yaml command.The service_account_cluster.yaml YAML manifest file is included in the TimesTen Operator distribution.	



Variable	Description	Description
image	 Variable that defines parameters that affect container images. Default: repository: Location of the container image. For example, container-registry.oracle.com/timesten/timesten. tag: TimesTen release number that contains the relevant Helm charts. For example, "22.1.1.34.0". pullPolicy: Pull policy for the container image. For example, Always. 	<pre>image: repository: phx.ocir.io/ youraccount/tt2211260image tag: "1" pullPolicy: Always</pre>
imagePullSecrets	Variable that defines a list of pull Secrets required to pull container images. There is no default.	imagePullSecrets: sekret
javaHome	Variable that defines the location where Java is installed in the TimesTen container image. Default: "/usr/java/jdk-21"	javaHome: "/usr/java/jdk-21"
labels	Variable that defines a list of labels applied to a TimesTen Operator Deployment and the Pods that the TimesTen Operator manages. There is no default.	labels: x: y
livenessFailureThreshold	Variable that sets the FailureThreshold attribute for the TimesTen Operator's liveness probe. Default: 3	livenessFailureThreshold: 2
livenessInitialDelaySeconds	Variable that sets the InitialDelaySeconds attribute for the TimesTen Operator's liveness probe. Default: 10	
livenessPeriodSeconds	Variable that sets the PeriodSeconds attribute for the TimesTen Operator's liveness probe. Default: 30	livenessPeriodSeconds: 20
livenessSuccessThreshold	Variable that sets the SuccessThreshold attribute for the TimesTen Operator's liveness probe. Default: 1	livenessSuccessThreshold: 2
livenessTimeoutSeconds	Variable that sets the TimeoutSeconds attribute for the TimesTen Operator's liveness probe. Default: 10	livenessTimeoutSeconds: 15



Variable	Description	Description
maxReconciles	Variable that determines how many TimesTenClassic objects the TimesTen Operator processes concurrently.	maxReconciles: 2
	 Valid values are as follows: "2" (default): The TimesTen Operator processes at most two TimesTenClassic objects at one time (in parallel). "1": The TimesTen Operator processes at most one TimesTenClassic object at one time. 	
	For performance purposes, we recommend that you do not change the default value.	
metrics	 Variable that determines if and how TimesTen Operator metrics are exposed. Valid values are as follows: expose: 1 or expose: 0: If you specify expose: 1 (or do not specify it), TimesTen Operator metrics are exposed. If you specify expose: 0, TimesTen Operator metrics are not exposed. scheme: https or scheme: http: If you specify scheme: https (or do not specify it), TimesTen Operator metrics are available by using https/Transport Layer Security (mutual TLS). If you specify scheme: http, TimesTen Operator metrics are available by using http. createServiceMonitor: 1 or createServiceMonitor: 0: If you specify createServiceMonitor: 1 (or do not specify it), the TimesTen Operator attempts to create a ServiceMonitor object. If you specify createServiceMonitor: 0, the TimesTen Operator does not create a ServiceMonitor object. 	<pre>metrics: expose: 0 scheme: http createServiceMonitor: 0</pre>
	See Expose Metrics from the TimesTen Kubernetes Operator.	



Variable	Description	Description
name	Variable for the name of the TimesTen Operator Deployment.	name: timesten-operator
	Default: timesten-operator	
operatorNodePort	Variable that determines the port on which the Connection Manager is available to applications outside the cluster.	operatorNodePort: 32627
	The Connection Manager can be accessed from outside the Kubernetes cluster on any node in the Kubernetes cluster by this port number.	
	The default is 32625. Valid values are between 30000 and 32767.	
operatorSAN	 Variable that adds subject alternate names (SANs) to the TLS certificate that the TimesTen Operator creates to control access to the TimesTen Operator metrics and to the Connection Manager. If you use metrics and the Connection Manager. If you use metrics and the Connection Manager from inside the Kubernetes cluster, you do not need to specify this variable. If you use either metrics or the Connection Manager (or both) from outside the Kubernetes cluster, you must define a SAN for the nodes in the cluster. A SAN value can be a DNS name (potentially wildcarded) or an IP address. The value is a comma delimited list of SAN values. You can specify a single level of wildcards. 	operatorSan: "1.2.3.4,1.2.3.5,*.mycluster. example.org"
podDisruptionBudget	Variable that determines if a	
Poartstartournader	podDisruptionBudget is created . If created, uses the provided values.	<pre>podDisruptionBudget: create: true</pre>
	Default: create: false	pdbName: samplepdb
	A podDisruptionBudget is not created by default.	<pre>maxUnvailable: 1 minAvailable: 1 matchLabels: "x": "y"</pre>

Variable	Description	Description
probes	Variable that determines if TimesTen Operator probes are exposed.	probes:
	If you specify expose: 1 (or do not specify it), TimesTen Operator probes are exposed. If you specify expose: 0, TimesTen Operator probes are not exposed.	expose: 0
	Default:	
	expose: 1	
	See About Readiness and Liveness Probes.	
readinessFailureThreshold	Variable that sets the FailureThreshold attribute for the TimesTen Operator's readiness probe. Default: 1	readinessFailureThreshold: 2
readinessInitialDelaySeconds	Variable that sets the InitialDelaySeconds attribute for the TimesTen Operator's readiness probe. Default: 10	readinessInitialDelaySeconds: 15
readinessPeriodSeconds	Variable that sets the PeriodSeconds attribute for the TimesTen Operator's readiness probe. Default: 10	readinessPeriodSeconds: 15
readinessSuccessThreshold	Variable that sets the SuccessThreshold attribute for the TimesTen Operator's readiness probe. Default: 1	readinessSuccessThreshold: 2
readinessTimeoutSeconds	Variable that sets the TimeoutSeconds attribute for the TimesTen Operator's readiness probe. Default: 10	readinessTimeoutSeconds: 15
replicas	Variable that defines the number of replica Pods in a TimesTen Operator Deployment. Default: 1	replicas: 3



Variable	Description	Description
resources	Variable that defines resource requests	
	and limits.	resources:
	Default:	requests:
		cpu: "300m"
	requests:	memory: "1G"
	cpu: "250m"	limits:
	memory: "1G"	cpu: "300m"
	limits:	memory: "1G"
	cpu: "250m"	
	memory: "1G"	
	2	
serviceAccount	Variable that determines if a service	
	account is created and what annotations	serviceAccount:
	to apply (if any).	create: false
	Default: create: true	
	By default or if you specify create:	
	true, the chart creates a Kubernetes	
	ServiceAccount called timesten-	
	operator as well as a Kubernetes Role	
	object and a Kubernetes RoleBinding	
	object. The Role and RoleBinding	
	objects grant the ServiceAccount the	
	privileges needed to run the TimesTen	
	Operator.	
	If you specify create: false, the	
	chart does not create the	
	ServiceAccount, Role, and RoleBinding	
	objects. You must create these objects	
	by running the kubectl create	
	<pre>service_account.yaml command.</pre>	
	The service_account.yaml YAML	
	manifest file is included in the TimesTen	
	Operator distribution.	



Variable	Description	Description
testAffinity	Variable that defines Kubernetes nodeAffinity, podAffinity, and podAntiAffinity parameters. These affinity configurations are only applied to the Helm test Pod.	<pre>testAffinity: nodeAffinity: requiredDuringSchedulingIgnor</pre>
	There is no default. However, if you are using a multi-architecture cluster that consists of amd64 and arm64 nodes, you must instruct Kubernetes to deploy the TimesTen test Pod on a specific architecture. This must be the same architecture as what the TimesTen Operator and its objects run on. For example, if the TimesTen Operator runs on arm64 nodes, the test Pod must also run on arm64 nodes.	<pre>edDuringExecution: nodeSelectorTerms: - matchExpressions: - key: topology.kubernetes.io/zone operator: In values: - antarctica-east1 - antarctica-west1</pre>
		Here is an example that shows how to instruct Kubernetes to run the test Poo on arm64 nodes:
		<pre>testAffinity: nodeAffinity:</pre>
		<pre>requiredDuringSchedulingIgnor edDuringExecution: nodeSelectorTerms: - matchExpressions: - key: topology.kubernetes.io/zone operator: In</pre>
		<pre>values: ["arm64"]</pre>
		 Valid values for values are: "amd64": Use this if you want the test Pod to run on amd64 nodes. "arm64": Use this if you want the test Pod to run on arm64 nodes.
testAnnotations	Variable that defines a list of annotations to be applied to a Helm test Pod. This Pod is created when you run the helm test command.	testAnnotations: x: y
	Default: helm.sh/hook-delete-policy: hook-succeeded	

Variable	Description	Description
testLabels	Variable that defines a list of labels to b applied to a Helm test Pod. This Pod i created when you run the helm test command.	
	There is no default.	

The ttclusteroperator Chart

The ttclusteroperator chart deploys the TimesTen Operator in the timesten-operator namespace in your Kubernetes cluster. The TimesTen Operator runs at cluster-scope and services TimesTenClassic objects in all namespaces in the cluster. The timesten-operator namespace is created by the TimesTen Operator and cannot be modified.

The following table describes the variables and default values for the ttclusteroperator chart. Indentation and case sensitivity must be correct for each variable. There may be some variables that exist in the product, but are not documented.



Table 21-2 Th	e ttclusteroperator Chart
---------------	---------------------------

Variable	Description	Description
affinity	Variable that defines Kubernetes nodeAffinity, podAffinity, and podAntiAffinity parameters. There is no default. However, if you are using a multi-architecture cluster that consists of amd64 and arm64 nodes, you must instruct Kubernetes to deploy the TimesTen Operator on a specific architecture.	affinity: nodeAffinity: requiredDuringSchedulingIgnor edDuringExecution: nodeSelectorTerms: - matchExpressions: - key: topology.kubernetes.io/zone operator: In values: - antarctica-east1 - antarctica-west1
		Here is an example that shows how to instruct Kubernetes to deploy the TimesTen Operator and the objects it manages on arm64 nodes:
		affinity: nodeAffinity:
		<pre>requiredDuringSchedulingIgnor edDuringExecution:</pre>
		topology.kubernetes.io/zone operator: In values: ["arm64"]
		Valid values for values are:
		• "amd64": Use this for TimesTen Operators that you want to run on amd64 nodes.
		• "arm64": Use this for TimesTen Operators that you want to run on arm64 nodes.
		Here is a complete affinity section with node affinity and pod affinity sections:
		<pre>affinity: {} nodeAffinity:</pre>
		<pre>requiredDuringSchedulingIgnor edDuringExecution: nodeSelectorTerms: - matchExpressions:</pre>

Variable	Description	Description
		- key:
		"kubernetes.io/arch"
		operator: In
		values: ["amd64"]
		nodeAffinity:
		requiredDuringSchedulingIgnor
		edDuringExecution:
		nodeSelectorTerms:
		- matchExpressions:
		- key:
		topology.kubernetes.io/zone
		operator: In
		values:
		- antarctica-east1
		- antarctica-easti - antarctica-westi
		preferredDuringSchedulingIgno
		redDuringExecution:
		- weight: 1
		preference:
		matchExpressions:
		- key: another-node-
		label-key
		operator: In
		values:
		- another-node-
		label-value
		podAntiAffinity:
		requiredDuringSchedulingIgnor
		edDuringExecution:
		- labelSelector:
		matchExpressions:
		- key: app
		operator: In
		values:
		- exampleApp
		topologyKey:
		"kubernetes.io/hostname"
		podAffinity:
		preferredDuringSchedulingIgno
		redDuringExecution:
		- weight: 50
		podAffinityTerm:
		labelSelector:
		matchExpressions:

Variable	Description	Description
		<pre>- key: app operator: In values: - exampleApp topologyKey: "kubernetes.io/hostname"</pre>
annotations	Variable that defines a list of annotations to be applied to a TimesTen Operator Deployment and the Pods that the TimesTen Operator manages. There is no default.	annotations: x: y
connectionManager	Variable that determines if the TimesTen Operator's Connection Manager is used.	connectionManager: true
	 Values are as follows: true (default): The TimesTen Operator's Connection Manager is used. false: The TimesTen Operator's Connection Manager is not used. 	
	See About the Connection Manager.	
createClusterRole	Variable that determines if cluster level roles are created. Default: true By default or if you specify true, the chart creates additional permissions and privileges for the TimesTen Operator.	createClusterRole: true
	If you specify false, the chart does not create the additional permissions and privileges. You must create these objects by running the kubectl create service_account_cluster.yaml command. The service_account_cluster.yaml YAML manifest file is included in the TimesTen Operator distribution.	



Variable	Description	Description
image	 Variable that defines parameters that affect container images. Default: repository: Location of the container image. For example, container-registry.oracle.com/timesten/timesten. tag: TimesTen release number that contains the relevant Helm charts. For example, "22.1.1.34.0". pullPolicy: Pull policy for the container image. For example, Always. 	<pre>image: repository: phx.ocir.io/ youraccount/tt2211260image tag: "1" pullPolicy: Always</pre>
imagePullSecrets	Variable that defines a list of pull Secrets required to pull container images. There is no default.	imagePullSecrets: sekret
javaHome	Variable that defines the location where Java is installed in the TimesTen container image. Default: "/usr/java/jdk-21"	javaHome: "/usr/java/jdk-21"
labels	Variable that defines a list of labels applied to a TimesTen Operator Deployment and the Pods that the TimesTen Operator manages. There is no default.	labels: x: y
livenessFailureThreshold	Variable that sets the FailureThreshold attribute for the TimesTen Operator's liveness probe. Default: 3	livenessFailureThreshold: 2
livenessInitialDelaySeconds	Variable that sets the InitialDelaySeconds attribute for the TimesTen Operator's liveness probe. Default: 10	livenessInitialDelaySeconds: 15
livenessPeriodSeconds	Variable that sets the PeriodSeconds attribute for the TimesTen Operator's liveness probe. Default: 30	livenessPeriodSeconds: 20
livenessSuccessThreshold	Variable that sets the SuccessThreshold attribute for the TimesTen Operator's liveness probe. Default: 1	livenessSuccessThreshold: 2
livenessTimeoutSeconds	Variable that sets the TimeoutSeconds attribute for the TimesTen Operator's liveness probe. Default: 10	livenessTimeoutSeconds: 15



Variable	Description	Description
maxReconciles	 Variable that determines how many TimesTenClassic objects the TimesTen Operator processes concurrently. Valid values are as follows: "2" (default): The TimesTen Operator processes at most two TimesTenClassic objects at one time (in parallel). "1": The TimesTen Operator processes at most one TimesTenClassic object at one time. For performance purposes, we recommend that you do not change the default value. 	maxReconciles: 2
metrics	 Variable that determines if and how TimesTen Operator metrics are exposed. Valid values are as follows: expose: 1 or expose: 0: If you specify expose: 1 (or do not specify it), TimesTen Operator metrics are exposed. If you specify expose: 0, TimesTen Operator metrics are not exposed. scheme: https or scheme: http: If you specify scheme: https (or do not specify it), TimesTen Operator metrics are available by using https/Transport Layer Security (mutual TLS). If you specify scheme: http: (or do not specify it), TimesTen Operator metrics are available by using https/Transport Layer Security (mutual TLS). If you specify scheme: http; TimesTen Operator metrics are available by using http. createServiceMonitor: 1 or createServiceMonitor: 0: If you specify createServiceMonitor: 1 (or do not specify it), the TimesTen Operator attempts to create a ServiceMonitor: 0, the TimesTen Operator does not create a ServiceMonitor object. If you specify createServiceMonitor object. Default: expose: 1 scheme: https createServiceMonitor: 1 	<pre>metrics: expose: 0 scheme: http createServiceMonitor: 0</pre>
	See Expose Metrics from the TimesTen Kubernetes Operator.	

Variable	Description	Description
name	Variable for the name of the TimesTen Operator Deployment.	name: timesten-operator
	Default: timesten-operator	
operatorNodePort	Variable that determines the port on which the Connection Manager is available to applications outside the cluster.	operatorNodePort: 32627
	The Connection Manager can be accessed from outside the Kubernetes cluster on any node in the Kubernetes cluster by this port number.	
	The default is 32625. Valid values are between 30000 and 32767.	
operatorSAN	 Variable that adds subject alternate names (SANs) to the TLS certificate that the TimesTen Operator creates to control access to the TimesTen Operator metrics and to the Connection Manager. If you use metrics and the Connection Manager from inside the Kubernetes cluster, you do not need to specify this variable. If you use either metrics or the Connection Manager (or both) from outside the Kubernetes cluster, you must define a SAN for the nodes in the cluster. A SAN value can be a DNS name (potentially wildcarded) or an IP address. The value is a comma delimited list of SAN values. You can specify a single level of wildcards. 	operatorSan: "1.2.3.4,1.2.3.5,*.mycluster example.org"
podDisruptionBudget	Variable that determines if a podDisruptionBudget is created . If created, uses the provided values.	<pre>podDisruptionBudget: create: true</pre>
	Default : create: false	pdbName: samplepdb
	A podDisruptionBudget is not created by default.	<pre>maxUnvailable: 1 minAvailable: 1 matchLabels: "x": "y"</pre>

Variable	Description	Description
probes	Variable that determines if TimesTen Operator probes are exposed. If you specify expose: 1 (or do not specify it), TimesTen Operator probes are exposed. If you specify expose: 0, TimesTen Operator probes are not exposed. Default: expose: 1	probes: expose: 0
	See About Readiness and Liveness Probes.	
readinessFailureThreshold	Variable that sets the FailureThreshold attribute for the TimesTen Operator's readiness probe. Default: 1	readinessFailureThreshold: 2
readinessInitialDelaySeconds	Variable that sets the InitialDelaySeconds attribute for the TimesTen Operator's readiness probe. Default: 10	readinessInitialDelaySeconds: 15
readinessPeriodSeconds	Variable that sets the PeriodSeconds attribute for the TimesTen Operator's readiness probe. Default: 10	readinessPeriodSeconds: 15
readinessSuccessThreshold	Variable that sets the SuccessThreshold attribute for the TimesTen Operator's readiness probe. Default: 1	readinessSuccessThreshold: 2
readinessTimeoutSeconds	Variable that sets the TimeoutSeconds attribute for the TimesTen Operator's readiness probe. Default: 10	readinessTimeoutSeconds: 15
replicas	Variable that defines the number of replica Pods in a TimesTen Operator Deployment. Default: 1	replicas: 3



Variable	Description	Description
resources	Variable that defines resource requests	
	and limits.	resources:
	Default:	requests:
		cpu: "300m"
	requests:	memory: "1G"
	cpu: "250m"	limits:
	memory: "1G"	cpu: "300m"
	limits:	memory: "1G"
	cpu: "250m"	-
	memory: "1G"	
serviceAccount	Variable that determines if a service	
	account is created and what annotations	
	to apply (if any).	create: false
	Default: create: true	
	By default or if you specify create:	
	true, the chart creates a Kubernetes	
	ServiceAccount called timesten-	
	operator as well as a Kubernetes Role	
	object and a Kubernetes RoleBinding	
	object. The Role and RoleBinding	
	objects grant the ServiceAccount the	
	privileges needed to run the TimesTen	
	Operator.	
	If you specify create: false, the	
	chart does not create the	
	ServiceAccount, Role, and RoleBinding	
	objects. You must create these objects	
	by running the kubectl create	
	service_account.yaml command.	
	The service_account.yaml YAML	
	manifest file is included in the TimesTen	
	Operator distribution.	



Variable	Description	Description
testAffinity	Variable that defines Kubernetes nodeAffinity, podAffinity, and podAntiAffinity parameters. These affinity configurations are only applied to the Helm test Pod.	<pre>testAffinity: nodeAffinity: requiredDuringSchedulingIgnor</pre>
	There is no default. However, if you are using a multi-architecture cluster that consists of amd64 and arm64 nodes, you must instruct Kubernetes to deploy the TimesTen test Pod on a specific architecture. This must be the same architecture as what the TimesTen Operator and its objects run on. For example, if the TimesTen Operator runs on arm64 nodes, the test Pod must also run on arm64 nodes.	<pre>edDuringExecution: nodeSelectorTerms: - matchExpressions: - key: topology.kubernetes.io/zone operator: In values: - antarctica-east1 - antarctica-west1</pre>
		Here is an example that shows how to instruct Kubernetes to run the test Poo on arm64 nodes:
		<pre>testAffinity: nodeAffinity:</pre>
		<pre>requiredDuringSchedulingIgnor edDuringExecution:</pre>
		<pre>values: ["arm64"]</pre>
		 Valid values for values are: "amd64": Use this if you want the test Pod to run on amd64 nodes. "arm64": Use this if you want the test Pod to run on arm64 nodes.
testAnnotations	Variable that defines a list of annotations to be applied to a Helm test Pod. This Pod is created when you run the helm test command. Default:	testAnnotations: x: y
	helm.sh/hook-delete-policy: hook-succeeded	



Variable	Description	Description
testLabels	Variable that defines a list of la applied to a Helm test Pod. T created when you run the hel command.	This Pod is testLabels:
	There is no default.	

The ttclassic Chart

The ttclassic chart creates a TimesTen Classic object.

The following table describes the variables and default values for the ttclassic chart. Indentation and case sensitivity must be correct for each variable. The fields are in alphabetical order.

Table 21-3 Variables for the ttclassic Chart

Variable	Default	Example
additionalMemoryRequest	Variable that defines the amount of memory to request in addition to what is required for the TimesTen database. This memory is used for the TimesTen daemon, subdaemons, replication agents, cache agents, and the server in a client/server environment. Default: 2Gi	additionalMemoryRequest: 3Gi
	See TimesTenClassicSpecSpec.	
affinity	Variable to define Kubernetes nodeAffinity, podAffinity, and podAntiAffinity parameters.	affinity: nodeAffinity:
	There is no default.	<pre>requiredDuringSchedulingIgnor edDuringExecution: nodeSelectorTerms: - matchExpressions: - key: topology.kubernetes.io/zone operator: In values: - antarctica-east1 - antarctica-west1</pre>

Variable	Default	Example
agentTcpTimeout	Variable that determines the time in seconds that the TimesTen Operator waits for a TCP handshake when communicating with the TimesTen agent.	agentTcpTimeout: 20
	The default is 10. A value of 0 indicates that there is no timeout. If the timeout is exceeded, the TimesTen Operator considers the agent to be down.	
	See TimesTenClassicSpecSpec.	
agentTlsTimeout	Variable that determines the time in seconds that the Operator waits for a TLS (https) credential exchange when communicating with the TimesTen agent.	agentTlsTimeout: 20
	The default is 10. A value of 0 indicates that there is no timeout. If the timeout is exceeded, the TimesTen Operator considers the agent to be down.	
	See TimesTenClassicSpecSpec.	
annotations	A list of annotations to be applied to a TimesTenClassic object.	annotations:
	There is no default.	х: у
clientTLS	Optional variable used to instruct the TimesTen Operator to automatically generate certificates and automatically configure client/server TLS connections.	clientTLS: auto: true validity: 9999
	For the TimesTen Operator to automatically perform these operations, you must specify clientTLS and you must set clientTLS.auto to true.	encryption: required
	For a list of options and defaults for clientTLS, see TimesTenClassicSpecSpecClientTLS.	
	For more information about how the TimesTen Operator automatically generates certificates and configures client/server connections, see Automatically Configure Client/Server TLS.	
containers	Variable to define a custom container. This container is created in each Pod in a TimesTenClassic object. The provided YAML is copied directly to	containers: - name: sampleapp image: sample.com/
	the .spec.template.spec.containe rs section of a TimesTenClassic object YAML manifest.	<pre>sample:latest command: - /bin/sample</pre>
	Default: There is no default.	
	See TimesTenClassicSpec.	

Table 21-3 (Cont.) Variables for the ttclassic Chart


Variable	Default Variable that determines if the TimesTen Operator creates a readiness probe for a replicated TimesTenClassic. Valid values: • true (default): The TimesTen Operator creates a readiness probe. • false: The TimesTen Operator does not create a readiness probe. • See About Readiness Probes for TimesTen Containers.		Example	
createASReadinessProbe			createASReadinessProbe: true	
customerService[1-5]			<pre>customService1: apiVersion: v1 kind: Service metadata: name: sampleexternal spec: type: NodePort selector: database.timesten.oracle.com:</pre>	
		Note: The YAML that you provide for a customSe rvice definition is used without modificatio n.	<pre>payroll ports: - port: 6625 targetPort: 6625 nodePort: 31625 customService2: apiVersion: v1 kind: Service metadata: name: sampleinteral spec: type: NodePort selector:</pre>	
	Default: There is no d	lefault.	<pre>database.timesten.oracle.com: payroll ports: - port: 6626 targetPort: 6626 nodePort: 31626</pre>	
daemonLogCPURequest	Amount of CPU requested for the daemonlog container. Default: 200m		daemonlogCPURequest: 400m	



Variable	Default	Example	
daemonLogMemoryRequest	Amount of memory requested for the daemonlog container.	daemonLogMemoryRequest: 300Mi	
	Default: 200Mi		
	See TimesTenClassicSpecSpec.		
databaseCPURequest	Amount of CPU your tt containers require. This includes the CPU used by the TimesTen daemon, subdaemons, replication agents, cache agents, and the server in a client/server environment.	databaseCPURequest: 2	
	Default: There is no default.		
	See TimesTenClassicSpecSpec.		
databaseMemorySize	Amount of shared memory your database requires.	databaseMemorySize: 20Gi	
	Default: There is no default.		
	See TimesTenClassicSpecSpec.		
dbConfigMap	Variable that creates a Kubernetes ConfigMap based on the provided values and adds a dbConfigMap entry to a TimesTenClassic object definition. Default: There is no default. By default, a ConfigMap is not created by a ttclassic chart. See TimesTenClassicSpecSpec.	<pre>dbConfigMap: - name: sample directory: cm Note: You must create a directory in the /ttclassic directory of the ttclassic chart. All files in the directory are added to the generated ConfigMap. This example assumes you have created a cm directory in the /</pre>	
		ttclassic directory tree.	
dbSecret	Variable that creates a Kubernetes Secret based on the provided values and adds a dbSecret entry to a TimesTenClassic object definition.	dbSecret: - name: sample directory: seekret	
	Default: There is no default. By default, a Secret is not created by a ttclassic chart. See TimesTenClassicSpecSpec.	Note: You must create a directory in the /ttclassic directory of the ttclassic chart. All files in the directory are added to the generated Secret. This example assumes you have created a seekret directory in the /ttclassic directory tree.	

Variable	Default	Example
deleteDbOnNotReadyNode	When specified, this variable directs the TimesTen Operator to detect situations where a Node is not ready (or unknown) for a specific period of time. When detected, the TimesTen Operator takes appropriate action to remedy the situation. Specifically, the TimesTen Operator checks to see if the Node's not ready condition has existed for more than deleteDbOnNotReadyNode seconds. If so, the TimesTen Operator deletes the Pod and the PVCs associated with the Pod. This causes Kubernetes to create a new Pod and new PVCs on a surviving Node. Once the Pod is scheduled and started by Kubernetes, the TimesTen Operator configures it as usual. The value is expressed in seconds and must be greater than	deleteDbOnNotReadyNode: 30
	pollingInterval. By default, this variable is not specified. You must specify it.	
	Note: Use caution when specifying this datum. Specifying this datum could result in the TimesTen Operator deleting PVCs. Deleting PVCs discards the on-disk copy of TimesTen databases.	
	For more information, see How the TimesTen Kubernetes Operator Handles Node Failure.	
exporterCPURequest	Amount of CPU requested for the exporter container (if provisioned). Default: 200m See TimesTenClassicSpecSpec.	exporterCPURequest: 400m
exporterMemoryRequest	Amount of memory requested for the exporter container (if provisioned).	exporterMemoryRequest: 400Mi



Variable	Default	Example
image	 Parameters that affect container images. Default: repository: Location of the container image. For example, container-registry.oracle.com/timesten/timesten. tag: TimesTen release number that contains the relevant Helm charts. For example, "22.1.1.34.0". pullPolicy: Pull policy for the container image. For example, Always. 	<pre>image: repository: phx.ocir.io/ youraccount/tt2211260image tag: "1" pullPolicy: Always</pre>
imagePullSecret	Pull Secret required to pull container images. Default: There is no default. See TimesTenClassicSpecSpec.	imagePullSecret: sekret
imageUpgradeStrategy	Variable to determine if the TimesTen Operator performs automated upgrades. If set to Manual, Helm cannot automatically upgrade or rollback a release. Default: Auto See TimesTenClassicSpecSpec.	imageUpgradeStrategy: Manual
initContainers	Variable used to define an init container. This init container is created in each Pod in a TimesTenClassic object. The provided YAML is copied directly to the .spec.template.spec section of a TimesTenClassic object YAML manifest. Default: There is no default.	<pre>initContainers: - name: sample image: command: - sh - "-c" - /bin/bash <<'EOF' echo test/user > / ttconfig/testUser ls -1 /ttconfig EOF volumeMounts: - name: tt-config mountPath: /ttconfig</pre>
labels	A list of labels to be applied to a TimesTenClassic object. Default: There is no default.	labels: x: y



Variable	riable Default	
logStorageClassName	Name of the storage class that is used to request persistent volumes for the TimesTen database transaction log files.	logStorageClassName: fast
	Default: There is no default. If you do not specify logStorageClassName, a	
	log volume is not allocated. See TimesTenClassicSpecSpec.	
logStorageSize	Amount of storage to be provisioned for the TimesTen transaction log files.	logStorageSize: 100Gi
	Default: 50Gi	
	See TimesTenClassicSpecSpec.	
name	Name of the TimesTenClassic object to be created.	name: sample
	Default: releaseName	
podDisruptionBudget	Variable that determines if a podDisruptionBudget is created . If created, uses the provided values.	<pre>podDisruptionBudget: create: true</pre>
	Default: create: false	pdbName: samplepdb
	A podDisruptionBudget is not created by default.	<pre>maxUnvailable: 1 minAvailable: 1 matchLabels: "x": "y"</pre>
pollingInterval	Variable that determines how often	pollingInterval: 6
	(expressed in seconds) that the TimesTen Operator checks the status of a TimesTenClassic object.	
	Default: 5	

 Table 21-3
 (Cont.) Variables for the ttclassic Chart

Variable	Default	Example
readOnlyRootFilesystem	Variable that determines if the TimesTen Operator causes the container image to be mounted read only in TimesTen containers. The TimesTen Operator provisions an empty directory and mounts it on top of the /tmp directory in all TimesTen containers in all TimesTen Pods. This includes the tt, daemonlog, and exporter containers in each Pod.	readOnlyRootFilesystem: true
	Valid values:	
	 true (default): The TimesTen Operator provisions an empty directory and mounts it on top of the /tmp directory. In addition, the TimesTen Operator creates these containers with container images mounted as read-only. false: The TimesTen Operator does not provision an empty directory and does not create containers with container images mounted as read-only. 	
	If you provide your own containers, the TimesTen Operator does not automatically mount your container images as read-only.	
	If you upgrade a v1 TimesTenClassic object and if the Pods associated with that TimesTenClassic object are replaced, the replacements do not have readOnlyRootFilesystem set.	
replicas	Variable that determines the number of Pods to provision for a non-replicated TimesTenClassic object. The default is 1. The maximum value is 3.	replicas: 2
replicationTopology	 Variable that determines the configuration for a TimesTen Classic database. Valid values: activeStandbyPair (default): The TimesTen Operator configures an active standby pair replication scheme. none: The TimesTen Operator does not configure replication. Instead, it configures a non-replicated configuration and uses the replicas variable to determine how many TimesTen Pods to 	replicationTopology: activeStandbyPair or replicationTopology: none

 Table 21-3
 (Cont.) Variables for the ttclassic Chart

Variable	Default	Example
repStateTimeout	Variable that indicates the time in seconds a replicated database remains in the recovering replication state as reported by the TimesTen ttRepStateGet built-in procedure.	repStateTimeout: 40
	The recovering replication state indicates that the database is in the process of synchronizing updates with the active database after a failure. For more information about the TimesTen ttRepStateGet built-in procedure, see ttRepStateGet in the Oracle TimesTen In-Memory Database Reference.	
	Default: 30 (expressed in seconds).	
	For more information about repStateTimeout, see TimesTenClassicSpecSpec.	
rollingUpdatePartition	Variable that is specific to upgrading a non-replicated configuration. Kubernetes upgrades Pods with an ordinal value that is greater than or equal to the rollingUpdatePartition value.	rollingUpdatePartition: 2
	For example, if you have three non- replicated Pods (replicas = 3 and Pods are samplerep-0, samplerep-1, and samplerep-2) and you set rollingUpdatePartition to 1, the samplerep_1 and samplerep-2 Pods are upgraded, but the samplerep-0 Pod is not.	
	The default is 0.	
	For more information, see Perform Upgrades in this book and Partitioned rolling updates in the Kubernetes documentation.	
securityContext	Variable used to create an optional securityContext definition. This securityContext definition is applied to TimesTenClassic at the Pod level.	securityContext runAsNonRoot: true
	The provided YAML is copied directly to the .spec.template.spec section of a TimesTenClassic object YAML manifest.	
	Default: There is no default.	



Variable	Default	Example	
serviceAccountName	Name of the service account that is assigned to the Pods that are created by the TimesTen Operator.	serviceAccountName: xyz	
	By default, when the TimesTen Operator creates StatefulSets (which in turn create Pods that run TimesTen), the StatefulSets and Pods have no service account. Specifying serviceAccountName lets you define a		
	Kubernetes ServiceAccount that the TimesTen Pods run under. Default: There is no default.		
storageClassName	Name of the storage class that is used to request persistent volumes for a TimesTen database.	storageClassName: oci-bv	
	Default: There is no default. However, you must specify a value for storageClassName.		
	See TimesTenClassicSpecSpec.		
storageSize	Amount of storage to be provisioned for TimesTen and the TimesTen database.	storageSize: 250Gi	
	Default: 50Gi		
storePassword	The TimesTen Operator stores its security certificates in Java Keystores. In order to create and use Java Keystores, they must be secured with a known password (storePassword). The password must be known by both the TimesTen Operator who creates and reads the keystore and the TimesTen Agent who also reads it.	storePassword: pseudoRandomString	
	The default value is a hard-coded string that is not documented.		
	Although not recommended, you can use the storePassword variable to set the password to a different value than the default and control the password setting. Once set, you cannot change storePassword. If it is changed, the TimesTen Operator generates a warning in the form of a Kubernetes Event and ignores the changed value.		
templateSpec	Variable used to insert arbitrary YAML into the .spec.template.spec section of a TimesTenClassic object. Default: There is no default.	templateSpec: preemptionPolicy: Never	
	See TimesTenClassicSpec.		

Variable	iable Default		
terminationGracePeriod	 Variable that determines the amount of time (in seconds) that Kubernetes waits for a Pod to gracefully shut down before being forcefully terminated. The default varies depending on your configuration: Replicated (Active Standby Pair: Default) is 10 seconds. Non-replicated: Default is 300 seconds. 	terminationGracePeriod: 30	
testAffinity	Variable to define Kubernetes nodeAffinity, podAffinity, and podAntiAffinity parameters. These affinity configurations are only applied to the Helm test Pod. There is no default.	<pre>testAffinity: nodeAffinity: requiredDuringSchedulingIgnor edDuringExecution: nodeSelectorTerms: - matchExpressions: - key: topology.kubernetes.io/zone operator: In values: - antarctica-east1 - antarctica-west1</pre>	
testAnnotations	A list of annotations to be applied to a Helm test Pod. This Pod is created when you run the helm test command. Default: helm.sh/hook-delete-policy: hook-succeeded	testAnnotations: x: y	
testLabels	A list of labels to be applied to a Helm test Pod. This Pod is created when you run the helm test command. Default: There is no default.	testLabels: x: y	
tolerations	Variable to define tolerations to be applied to the Pods in a TimesTenClassic object. The provided YAML is copied directly to the .spec.template.spec section of a TimesTenClassic object YAML manifest. Default: There is no default.	<pre>tolerations: - key: "key1" operator: "Equal" value: "value1" effect: "NoSchedule"</pre>	

Variable	Default	Example
ttspec	Variable to insert arbitrary YAML into the .spec.ttspec section of a TimesTenClassic object. Use this variable to specify additional datum that are supported by the TimesTenClassic object type, but are not explicitly supported by the ttclassic chart.	ttspec: agentGetTimeout: 10 prometheus: port: 7777
	Default: There is no default.	
	See TimesTenClassicSpecSpec.	
volumes	Variable used to create optional volume definitions. These definitions are applied to TimesTenClassic objects.	volumes: - name: tt-config
	The provided YAML is copied directly to the .spec.template.spec section of a TimesTenClassic object YAML manifest.	emptyDir: {}
	Default: There is no default.	

 Table 21-3
 (Cont.) Variables for the ttclassic Chart



22 TimesTen Kubernetes Operator Metrics

There are several TimesTen Kubernetes Operator (TimesTen Operator) metrics that can be exposed and then published by Prometheus or some other scraper. These metrics include information about the functionality of the TimesTen Operator as well as the overall status of TimesTenClassic and TimesTenScaleout objects that the TimesTen Operator manages.

The following table details the TimesTen Operator metrics:

Table 22-1	TimesTen	Operator	Metrics
------------	----------	----------	---------

Metric	Туре	Description
timesten_agent_tt_mem_usage	Gauge	Amount of memory used by the TimesTen Agent and TimesTen within a Pod. Reports the memory usage by each Pod as reported by Linux cgroups in the memory.usage_in_bytes value. Reports the value in bytes.
timesten_agent_rss	Gauge	Resident Set Size (rss) for the TimesTen Agent within each Pod running TimesTen. Reports the value as reported by the VmRSS field in the / proc/self/status file, which is managed by the Linux kernel for the TimesTen agent process within each tt container in each Pod that is running TimesTen. Reports the value in Kilobytes.
timesten_classic_state	Gauge	State of each TimesTenClassic object. The specific state is reported in the state label.
<pre>timesten_classic_state_allrepli casready</pre>	Gauge	Number of non-replicated TimesTenClassic objects in the AllReplicasReady state.
<pre>timesten_classic_state_noreplic asready</pre>	Gauge	Number of non-replicated TimesTenClassic objects in the NoReplicasReady state.
timesten_classic_state_normal	Gauge	Number of replicated TimesTenClassic objects in the Normal state.
<pre>timesten_classic_state_not_allr eplicasready</pre>	Gauge	Number of non-replicated TimesTenClassic objects in states other than AllReplicasReady or Initializing.
<pre>timesten_classic_state_not_norm al</pre>	Gauge	Number of replicated TimesTenClassic objects in states other than Normal or Initializing.
timesten_classic_state_somerepl icasready	Gauge	Number of non-replicated TimesTenClassic objects in the SomeReplicasReady state.



Table 22-1 (Cont.) TimesTen Operator Metrics

Metric	Туре	Description
<pre>timesten_pods_deleted_unschedul able</pre>	Gauge	Total count of Pods that the TimesTen Operator has deleted. The value increases each time a Pod is deleted.
		For more information, see the deleteDbOnNotReadyNode datum in Table 20-3.
timesten_pods_unschedulable	Gauge	Number of Pods that are not currently scheduled by Kubernetes. The value increases and decreases over time.
<pre>timesten_pvcs_deleted_unschedul able</pre>	Gauge	Total count of PVCs that the TimesTen Operator has deleted. The value increases each time a PVC is deleted.
		For more information, see the deleteDbOnNotReadyNode datum in Table 20-3.
timesten_scaleout_state	Gauge	State of each TimesTenScaleout object. The specific state is reported in the state label.
timesten_scaleout_state_normal	Gauge	Number of TimesTenScaleout objects in the Normal state.
<pre>timesten_scaleout_state_not_nor mal</pre>	Gauge	Number of TimesTenScaleout objects in states other than Normal or Initializing.
timesten_operator_mem_usage	Gauge	Amount of memory used by the TimesTen Kubernetes Operator. Reports the memory usage by the TimesTen Operator Pod as reported by Linux cgroups in the memory.usage_in_bytes value. Reports the value in bytes.
timesten_operator_rss	Gauge	Resident Set Size (rss) for the TimesTen Kubernetes Operator. Reports the value as reported by the VmRSS field in the / proc/self/status file, which is managed by the Linux kernel for the TimesTen Kubernetes Operator process. Reports the value in Kilobytes.

Each metric has these labels:

- namespace: The namespace of the TimesTen object that the metric describes.
- name:
 - For the timesten_operator_mem_usage and timesten_operator_rss metrics, the name label is the name of the TimesTen Operator Pod.
 - For the timesten_agent_tt_mem_usage and timesten_agent_rss metrics, the name label is the name of the TimesTen Pod.
 - For all other metrics, the name label is the name of the TimesTen object that the metric describes.

In addition there are several other labels of secondary importance:

- container: The name of the container that generated the metric. For example, timestenoperator.
- endpoint: The name of the endpoint that the metric came from. For example, metrics.
- instance: The IP address and port of the TimesTen Operator Pod that generated the metric. For example, 10.244.1.111:8080
- job: The name of the job. For example, timesten-operator.
- pod: The name of the TimesTen Operator Pod that generated the metric. For example, timesten-operator-5f4f4c69f6-z9h2k.
- service. The name of the Service. For example, timesten-operator.

Here is an example of a sample timesten classic state normal metric:

```
timesten_classic_state_normal{container="timesten-operator",
endpoint="metrics", instance="10.244.1.111:8080", job="timesten-operator",
name="sample", namespace="mynamespace", pod="timesten-operator-5f4f4c69f6-
z9h2k",
service="timesten-operator"}
```

The metric has a value of 1 or 0, depending on whether the TimesTenClassic object (sample, in this case) is healthy or not.

Here is an example of a sample timesten_classic_state metric:

```
timesten_classic_state{container="timesten-operator",
endpoint="metrics", instance="10.244.1.111:8080", job="timesten-operator",
name="sample", namespace="mynamespace", pod="timesten-operator-5f4f4c69f6-
z9h2k",
service="timesten-operator", state="Normal"}
```

The metric has a value of 1 or 0, depending on whether the TimesTenClassic object (sample, in this case) is in the Normal state or not.

For details on how the TimesTen Operator exposes metrics, see Expose Metrics from the TimesTen Kubernetes Operator.

Note:

The TimesTen Operator automatically exposes many other additional metrics. We cannot guarantee these additional metrics will exist from release to release. In addition, these metrics may not be specifically pertinent or useful for the TimesTen Operator.



TimesTen Kubernetes Operator Environment Variables

There are environment variables for the TimesTen Operator. The <code>operator.yaml</code> and <code>cluster_operator.yaml</code> files contain these environment variables. You can change the default setting for these environment variables depending on your needs. For more information about the TimesTen Operator, see About the TimesTen Operator.

The following table describes these variables:

Table 23-1	TimesTen	Operator	Environment	Variables
------------	----------	----------	-------------	-----------

Environment Variable	Description		
CREATE_SERVICEMONITOR	Determines if the TimesTen Operator creates a Kubernetes ServiceMonitor object.		
	 Valid values are as follows: "1" (default): The TimesTen Operator should create a ServiceMonitor object. "0": The TimesTen Operator should not create a ServiceMonitor object. 		
	If EXPOSE_METRICS is set to "0" and CREATE_SERVICEMONITOR is set to "1" (default), CREATE_SERVICEMONITOR is treated as though it was set to "0".		
	See Expose Metrics from the TimesTen Kubernetes Operator.		
EXPOSE_METRICS	Determines if TimesTen Operator metrics are exposed outside of the TimesTen Operator's Pods.		
	 Valid values are as follows: "1" (default): Metrics should be exposed outside of the TimesTen Operator's Pods. "0": Metrics should not be exposed outside of the TimesTen Operator's Pods. 		
	If EXPOSE_METRICS is set to "0" and CREATE_SERVICEMONITOR is set to "1" (default), CREATE_SERVICEMONITOR is treated as though it was set to "0".		
	In addition, if EXPOSE_METRICS is set to "0", the value of METRICS_SCHEME is ignored and http is always used.		
	See Expose Metrics from the TimesTen Kubernetes Operator.		
EXPOSE_PROBES	Determines if liveness probes should be exposed outside of the TimesTen Operator's Pods.		
	 Valid values are as follows: "1" (default): Probes should be exposed. "0": Probes should not be exposed. 		

Environment Variable	Description			
JAVA_HOME	Specifies the location where Java is installed in the TimesTen container image.			
	Default: /usr/java/jdk-21.			
	If you are using a different location, change the value for this environment variable.			
METRICS_SCHEME	Determines if TimesTen Operator metrics should be made available by https or http.			
	Valid values are as follows:"https" (default): https is used.			
	• "http": http is used.			
	If EXPOSE_METRICS is set to "0", the value of METRICS_SCHEME is ignored and http is always used.			
	See Expose Metrics from the TimesTen Kubernetes Operator.			
TT_CONNECTION_MANAGER	Determines if the TimesTen Operator's Connection Manager is enabled.			
	Valid values are as follows:			
	• "1" (default): Enable the Connection Manager.			
	• "0": Do not enable the Connection Manager.			
	If you disable metrics for the TimesTen Operator or you do not use https for metrics, the Connection Manager is not enabled. For example, if you set METRICS_SCHEME to "http" or EXPOSE_METRICS to "0", then TT CONNECTION MANAGER is ignored and treated as "0".			
	See About the Connection Manager.			
TT_CONNECTION_MANAGER_NODEPORT	Determines the port number on which the Connection Manager is available to applications outside of the Kubernetes cluster.			
	The Connection Manager can be accessed from outside the Kubernetes cluster on any node in the Kubernetes cluster by this port number.			
	The default is 32625. Valid values are between 30000 and 32767.			
	Use this environment variable to change the port number.			
TT_MAX_RECONCILES	Determines how many TimesTenClassic objects the TimesTen Operator processes concurrently.			
	 Valid values are as follows: "2" (default): The TimesTen Operator processes at most two TimesTenClassic objects at one time (in parallel). "1": The TimesTen Operator processes at most one TimesTenClassic object at one time. 			
	For performance purposes, we recommend that you do not change the default value.			

Table 23-1 (Cont.) TimesTen Operator Environment Variables

Environment Variable	Description
Environment Variable TT_OPERATOR_SAN	 Adds subject alternate names (SANs) to the TLS certificate that the TimesTen Operator creates to control access to the TimesTen Operator metrics and to the Connection Manager. If you use metrics and the Connection Manager from inside the Kubernetes cluster, you do not need to set this environment variable. If you use either metrics or the Connection Manager (or both) from outside the Kubernetes cluster, you must define a SAN for the nodes in the cluster. A SAN value can be a DNS name (potentially wildcarded) or an IP address. The value is a comma delimited list of SAN values, such as "1.2.3.4, 1.2.3.5, *.mycluster.example.org". You can specify a single level of wildcards.
	By default, there is no SAN specified.

Table 23-1 (Cont.) TimesTen Operator Environment Variables

24 Dockerfile ARGs

You have several options for obtaining TimesTen container images. One of these options is to build your own TimesTen container image. TimesTen provides a Dockerfile for this purpose. For more information about TimesTen container images, see About TimesTen Container Images and Container Registry Options.

The Dockerfile supports a number of ARGs. These ARGs let you override the attributes of the Dockerfile (and its resultant image). You supply these ARGs on the docker build command line.

The following table describes the supported ARGs:

ARG name	Default value	Description
TT_BASE	container- registry.oracle.com/os/ oraclelinux:8	Name of the base image.
UNZIP_BASE	<pre>container- registry.oracle.com/os/ oraclelinux:8</pre>	Name of the image that is used to unzip the TimesTen distribution.
TT_DISTRO	timesten22.1.1.34.0.server.linu x8664.zip or timesten22.1.1.34.0.server.linu	you are including in the container
	xarm64.zip depending on whether you are using amd64 or arm64 nodes .	For information about deploying the TimesTen Kubernetes Operator in a multi-architecture Kubernetes cluster, see About Deploying in a Multi- Architecture Kubernetes Cluster.
TT_RELEASE	22.1.1.34.0	Release number (in dotted decimal format) of the TimesTen release that is included in \$TT_DISTRO.
TT_USER	timesten	Name of the Linux user that is created in the container image. This is the user who runs TimesTen.
		To define a different user to run TimesTen, supply this ARG on the docker build command line.
TT_UID	3429	The numeric UID of \$TT_USER.
		To define a different UID for <i>STT_USER</i> , supply this ARG on the docker build command line.

Table 24-1 Dockerfile ARGs



Table 24-1 (Cont.) Dockerfile ARGs

ARG name	Default value	Description	
TT_GROUP	timesten	Name of the Linux group that is create in the container image. This is the nam of the TimesTen users group.	
		To define a different name for the TimesTen group, supply this ARG on the docker build command line.	
TT_GID	3429	The numeric GID of \$TT_GROUP.	
		To define a different GID for \$TT_GROUP, supply this ARG on the docker build command line.	
TT_EXTRA_LINUX_PACKAGES	There is no default.	Additional Linux packages that you want installed in the container image by the yum install command.	

A Active Standby Pair Example

This appendix provides an example showing you the complete process for deploying and running your active standby pair of TimesTen databases in the Kubernetes cluster. After the databases are up and running, the example demonstrates how the Operator controls and manages the databases. If the active database fails, the Operator performs the necessary tasks to failover to the standby database, making that standby database the active one. The example concludes with procedures to delete the TimesTen databases and to stop the Operator.

Note:

You have the option of using Helm to deploy your active standby pair of TimesTen databases. For more information about using Helm, see Use Helm in Your TimesTen Kubernetes Operator Environment.

- Before You Begin
- Create a ConfigMap Object
- Create a TimesTenClassic Object
- Monitor Deployment
- Verify Existence of Underlying Objects
- Verify Connection to the Active TimesTen Database
- Recover from Failure
- Clean Up

Before You Begin

Review the following sections and complete if you have not already done so.

- 1. Before You Begin
- 2. Prepare to Use the TimesTen Kubernetes Operator
- 3. Install TimesTen Custom Resource Definitions
- 4. Learn About and Install the TimesTen Kubernetes Operator

Create a ConfigMap Object

This section creates the sample ConfigMap. This ConfigMap contains the db.ini, the adminUser, and the schema.sql metadata files. This ConfigMap will be referenced when you define the TimesTenClassic object. See "Overview of Configuration Metadata and Kubernetes Facilities" for information on the configuration files and the ConfigMap facility.

On your Linux development host:



From the directory of your choice, create an empty subdirectory for the metadata files. This
example creates the cm_sample subdirectory. (The cm_sample directory is used in the
remainder of this example to denote this directory.)

```
% mkdir -p cm_sample
```

2. Create the db.ini file in this ConfigMap directory (cm_sample, in this example). In this db.ini file, define the PermSize and DatabaseCharacterSet connection attributes.

```
vi cm/db.ini
PermSize=200
DatabaseCharacterSet=AL32UTF8
```

3. Create the adminUser file in this ConfigMap directory (cm_sample in this example). In this adminUser file, create the sampleuser user with the samplepw password.

vi cm/adminUser

vi cm/schema.sql

sampleuser/samplepw

4. Create the schema.sql file in this ConfigMap directory (cm_sample in this example). In this schema.sql file, define the s sequence and the emp table for the sampleuser user. The Operator will automatically initialize your database with these object definitions.

```
create sequence sampleuser.s;
create table sampleuser.emp (
  id number not null primary key,
  name char(32)
);
```

5. Create the ConfigMap. The files in the cm_sample directory are included in the ConfigMap and, later, will be available in the TimesTen containers.

In this example:

- The name of the ConfigMap is sample. Replace sample with a name of your choosing. (sample is represented in **bold** in this example.)
- This example uses cm_sample as the directory where the files that will be copied into the ConfigMap reside. If you use a different directory, replace cm_sample with the name of your directory. (cm_sample is represented in **bold** in this example.)

Use the kubectl create command to create the ConfigMap:

```
% kubectl create configmap sample --from-file=cm_sample
configmap/sample created
```

You successfully created and deployed the sample ConfigMap.

6. Use the kubectl describe command to verify the contents of the ConfigMap. (sample, in this example.)

```
% kubectl describe configmap sample
Name: sample
Namespace: mynamespace
Labels: <none>
Annotations: <none>
Data
====
adminUser:
----
```



```
sampleuser/samplepw
db.ini:
----
PermSize=200
DatabaseCharacterSet=AL32UTF8
schema.sql:
----
create sequence sampleuser.s;
create table sampleuser.emp (
   id number not null primary key,
   name char(32)
);
```

Create a TimesTenClassic Object

Events: <none>

This section creates the TimesTenClassic object. For detailed information about the TimesTenClassic object type, see About the TimesTenClassic Object Type.

Perform these steps:

 Create an empty YAML file. You can choose any name, but you may want to use the same name you used for the name of the TimesTenClassic object. (In this example, sample.) The YAML file contains the definitions for the TimesTenClassic object. See "TimesTenClassicSpecSpec" for information on the fields that you must specify in this YAML file as well as the fields that are optional.

In this example, replace the following. (The values you can replace are represented in **bold**.)

- name: Replace sample with the name of your TimesTenClassic object.
- storageClassName: Replace oci-bv with the name of the storage class used to allocate PersistentVolumes to hold TimesTen.
- storageSize: Replace 250Gi with the amount of storage that should be requested for each Pod to hold TimesTen. Note: This example assumes a production environment and uses a value of 250Gi for storageSize. For demonstration purposes, a value of 50Gi is adequate. See the storageSize and the logStorageSize entries in "Table 20-3" for information.
- image: Replace container-registry.oracle.com/timesten/timesten:22.1.1.34.0 with the location and name of your image.
- imagePullSecret: Replace sekret with the image pull secret that Kubernetes should use to fetch the TimesTen image.
- dbConfigMap: This example uses one ConfigMap (called sample) for the db.ini, the adminUser, and the schema.sql metadata files. This ConfigMap will be included in the ProjectedVolume. This volume is mounted as /ttconfig in the TimesTen containers. See "Using ConfigMaps and Secrets" and "Example Using One ConfigMap" for information on ConfigMaps.

% vi sample.yaml

```
apiVersion: timesten.oracle.com/v4
kind: TimesTenClassic
```



```
metadata:
  name: sample
spec:
  ttspec:
   storageClassName: oci-bv
   storageSize: 250Gi
   image: container-registry.oracle.com/timesten/timesten:22.1.1.34.0
   imagePullSecret: sekret
   dbConfigMap:
   - sample
```

 Use the kubectl create command to create the TimesTenClassic object from the contents of the YAML file (in this example, sample.yaml). Doing so begins the process of deploying your active standby pair of TimesTen databases in the Kubernetes cluster.

```
% kubectl create -f sample.yaml
configmap/sample created
timestenclassic.timesten.oracle.com/sample created
```

You successfully created the TimesTenClassic object in the Kubernetes cluster. The process of deploying your TimesTen databases begins, but is not yet complete.

Monitor Deployment

Use the kubectl get and the kubectl describe commands to monitor the progress of the active standby pair as it is provisioned.

Note:

For the kubectl get timestenclassic and kubectl describe timestenclassic commands, you can alternatively specify kubectl get ttc and kubectl describe ttc respectively. timestenclassic and ttc are synonymous when used in these commands, and return the same results. The first kubectl get and the first kubectl describe examples in this appendix use timestenclassic. The remaining examples in this appendix use timestenclassic.

1. Use the kubectl get command and review the STATE field. Observe the value is Initializing. The active standby pair provisioning has begun, but is not yet complete.

```
% kubectl get timestenclassic sample
NAME STATE ACTIVE AGE
sample Initializing None 11s
```

2. Use the kubectl describe command to view the initial provisioning in detail.

```
% kubectl describe timestenclassic sample
Name: sample
Namespace: mynamespace
           <none>
Labels:
Annotations: <none>
API Version: timesten.oracle.com/v4
            TimesTenClassic
Kind:
Metadata:
 Creation Timestamp: 2025-01-16T15:35:12Z
  Generation:
                     1
  Resource Version: 20251755
  Self Link:
/apis/timesten.oracle.com/v4/namespaces/mynamespace/timestenclassics/sample
```



UID: 517a8646-a354-11ea-a9fb-0a580aed5e4a Spec: Ttspec: Db Config Map: sample Image: container-registry.oracle.com/timesten/timesten:22.1.1.34.0 Image Pull Policy: Always Image Pull Secret: sekret Storage Class Name: oci-bv Storage Size: 250Gi Status: Active Pods: None High Level State: Initializing Last Event: 3 Pod Status: Cache Status: Cache Agent: Down Cache UID Pwd Set: false N Cache Groups: 0 Db Status: Db: Unknown Db Id: 0 Db Updatable: Unknown Initialized: true Pod Status: Agent: Down Last Time Reachable: 0 Pod IP: Pod Phase: Pending Replication Status: Last Time Rep State Changed: 0 Rep Agent: Down Rep Peer P State: Unknown Rep Scheme: Unknown Rep State: Unknown Times Ten Status: Daemon: Down Instance: Unknown Release: Unknown Admin User File: false Cache User File: false Cg File: false High Level State: Down Intended State: Active sample-0 Name: Schema File: false Cache Status: Cache Agent: Down Cache UID Pwd Set: false N Cache Groups: 0 Db Status: Db: Unknown Db Id: 0 Db Updatable: Unknown Initialized: true Pod Status: Agent: Down Last Time Reachable: 0 Pod IP: Pod Phase: Pending Replication Status: Last Time Rep State Changed: 0

```
Rep Agent:
                                Down
     Rep Peer P State:
                                Unknown
     Rep Scheme:
                                Unknown
     Rep State:
                                Unknown
   Times Ten Status:
                     Down
     Daemon:
     Instance: Unknown
   Admin User File: false
   Cache User File: false
   Cg File:
                      false
   High Level State: Unknown
   Intended State: Standby
                    sample-1
   Name ·
   Name:
Schema File:
                     false
 Rep Create Statement: create active standby pair "sample" on
 "sample-0.sample.mynamespace.svc.cluster.local", "sample" on
 "sample-1.sample.mynamespace.svc.cluster.local" NO RETURN store "sample" on
 "sample-0.sample.mynamespace.svc.cluster.local" PORT 4444 FAILTHRESHOLD 0
 store "sample" on "sample-1.sample.mynamespace.svc.cluster.local" PORT 4444
FAILTHRESHOLD 0
 Rep Port:
                      4444
 Status Version:
                    1.0
Events:
 Type Reason Age From
                            Message
  _____ ____
                              _____
 -
      Create 50s ttclassic Secret tt517a8646-a354-11ea-a9fb-0a580aed5e4a
created
 - Create 50s ttclassic Service sample created
       Create 50s ttclassic StatefulSet sample created
```

 Use the kubectl get command again to see if value of the STATE field has changed. In this example, the value is Normal, indicating the active standby pair of databases are now provisioned and the process is complete.

% kubect	l get ttc	sample	
NAME	STATE	ACTIVE	AGE
sample	Normal	sample-0	3m5s

4. Use the kubectl describe command again to view the active standby pair provisioning in detail.

Note: In this example, the now Normal line displays on its own line. In the actual output, this line does not display as its own line, but at the end of the StateChange previous line.

```
% kubectl describe ttc sample
Name: sample
Namespace: mynamespace
Labels:
            <none>
Annotations: <none>
API Version: timesten.oracle.com/v4
Kind:
            TimesTenClassic
Metadata:
 Creation Timestamp: 2025-01-16T15:35:12Z
 Generation: 1
 Resource Version: 20252668
  Self Link:
/apis/timesten.oracle.com/v4/namespaces/mynamespace/timestenclassics/sample
 UID:
                     517a8646-a354-11ea-a9fb-0a580aed5e4a
Spec:
  Ttspec:
   Db Config Map:
     sample
```



```
Image:
                       container-registry.oracle.com/timesten/timesten:22.1.1.34.0
   Image Pull Policy: Always
   Image Pull Secret: sekret
   Storage Class Name: oci-bv
   Storage Size:
                       250Gi
Status:
 Active Pods:
                  sample-0
 High Level State: Normal
 Last Event:
                   35
 Pod Status:
   Cache Status:
     Cache Agent: Not Running
     Cache UID Pwd Set: true
     N Cache Groups:
                        Ο
   Db Status:
     Db:
                   Loaded
     Db Id:
                   26
     Db Updatable: Yes
   Initialized: true
   Pod Status:
     Agent:
                          Up
     Last Time Reachable: 1590939597
     Pod IP:
                         192.0.2.1
     Pod Phase:
                         Running
   Replication Status:
     Last Time Rep State Changed: 0
     Rep Agent:
                                 Running
     Rep Peer P State:
                                 start
     Rep Scheme:
                                 Exists
     Rep State:
                                 ACTIVE
   Times Ten Status:
     Daemon:
                     Up
     Instance:
                    Exists
                    22.1.1.34.0
     Release:
   Admin User File: true
   Cache User File: false
   Cg File:
                     false
   High Level State: Healthy
   Intended State: Active
   Name:
                     sample-0
   Schema File:
                    true
   Cache Status:
                    Not Running
     Cache Agent:
     Cache UID Pwd Set: true
     N Cache Groups:
                        0
   Db Status:
                   Loaded
     Db:
     Db Id:
                   26
     Db Updatable: No
   Initialized:
                   true
   Pod Status:
     Agent:
                          Up
     Last Time Reachable: 1590939597
     Pod IP:
                          192.0.2.2
     Pod Phase:
                          Running
   Replication Status:
     Last Time Rep State Changed: 1590939496
     Rep Agent:
                                Running
     Rep Peer P State:
                                 start
     Rep Scheme:
                                 Exists
     Rep State:
                                  STANDBY
   Times Ten Status:
```

```
Daemon:
                       Up
     Instance:
                     Exists
                      22.1.1.34.0
     Release:
   Admin User File:
                      true
   Cache User File:
                      false
   Cg File:
                       false
   High Level State: Healthy
   Intended State:
                      Standby
                       sample-1
   Name:
   Schema File:
                       true
 Rep Create Statement: create active standby pair "sample" on
"sample-0.sample.mynamespace.svc.cluster.local", "sample" on
"sample-1.sample.mynamespace.svc.cluster.local" NO RETURN store "sample" on
"sample-0.sample.mynamespace.svc.cluster.local" PORT 4444 FAILTHRESHOLD 0
store "sample" on "sample-1.sample.mynamespace.svc.cluster.local" PORT 4444
FAILTHRESHOLD 0
                       4444
 Rep Port:
 Status Version:
                     1.0
Events:
 Type Reason
                 Age From
                                  Message
  _____
                  ____ ___
                                    _____
 -
      Create
                  4m43s ttclassic Secret tt517a8646-a354-11ea-a9fb-0a580aed5e4a
created
                  4m43s ttclassic Service sample created
      Create
      Create 4m43s ttclassic service sample created
Create 4m43s ttclassic StatefulSet sample created
      StateChange 3m47s ttclassic Pod sample-0 Daemon Unknown
      StateChange 3m47s ttclassic Pod sample-0 CacheAgent Unknown
      StateChange 3m47s ttclassic Pod sample-0 RepAgent Unknown
      StateChange 3m47s ttclassic Pod sample-1 Daemon Unknown
 _
      StateChange 3m47s ttclassic Pod sample-1 CacheAgent Unknown
  _
  _
       StateChange 3m47s ttclassic Pod sample-1 RepAgent Unknown
       StateChange 3m26s ttclassic Pod sample-0 Agent Up
 _
       StateChange 3m26s ttclassic Pod sample-0 Release 22.1.1.34.0
 _
      StateChange 3m26s ttclassic Pod sample-0 Daemon Down
      StateChange 3m26s ttclassic Pod sample-1 Agent Up
  _
      StateChange 3m26s ttclassic Pod sample-1 Release 22.1.1.34.0
 _
      StateChange 3m26s ttclassic Pod sample-1 Daemon Down
 _
      StateChange 3m26s ttclassic Pod sample-0 Daemon Up
 _
 _
      StateChange 3m25s ttclassic Pod sample-1 Daemon Up
 _
      StateChange 2m13s ttclassic Pod sample-0 RepState IDLE
 _
      StateChange 2m13s ttclassic Pod sample-0 Database Updatable
 _
      StateChange 2m13s ttclassic Pod sample-0 CacheAgent Not Running
 _
      StateChange 2m13s ttclassic Pod sample-0 RepAgent Not Running
      StateChange 2m13s ttclassic Pod sample-0 RepScheme None
  _
  _
      StateChange 2m13s ttclassic Pod sample-0 Database Loaded
      StateChange 2m11s ttclassic Pod sample-0 RepAgent Running
  _
      StateChange 2m10s ttclassic Pod sample-0 RepScheme Exists
      StateChange 2m10s ttclassic Pod sample-0 RepState ACTIVE
  _
      StateChange 113s ttclassic Pod sample-1 Database Loaded
  _
      StateChange 113s ttclassic Pod sample-1 Database Not Updatable
 _
       StateChange 113s ttclassic Pod sample-1 CacheAgent Not Running
 _
                         ttclassic Pod sample-1 RepAgent Not Running
       StateChange 113s
 _
       StateChange 113s
                          ttclassic Pod sample-1 RepScheme Exists
                         ttclassic Pod sample-1 RepState IDLE
       StateChange 113s
       StateChange 106s ttclassic Pod sample-1 RepAgent Running
       StateChange 101s ttclassic Pod sample-1 RepState STANDBY
       StateChange 101s
                         ttclassic TimesTenClassic was Initializing, now Normal
```

Your active standby pair of TimesTen databases are successfully deployed (as indicated by Normal.) There are two TimesTen databases, configured as an active standby pair. One database is active. (In this example, sample-0 is the active database, as indicated by Rep State ACTIVE). The other database is standby. (In this example, sample-1 is the standby

database as indicated by Rep State STANDBY). The active database can be modified and queried. Changes made on the active database are replicated to the standby database. If the active database fails, the Operator automatically promotes the standby database to be the active. The formerly active database will be repaired or replaced, and will then become the standby.

Verify Existence of Underlying Objects

Use the kubectl describe commands to verify the underlying objects.

1. StatefulSet:

```
% kubectl get statefulset sample
NAME READY AGE
sample 2/2 8m21s
```

2. Service:

% kubect	l get servio	ce sample			
NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
sample	ClusterIP	None	<none></none>	6625/TCP	9m28s

Pods:

% kubectl get pods				
NAME	READY	STATUS	RESTARTS	AGE
sample-0	2/2	Running	0	10m
sample-1	2/2	Running	0	10m
timesten-operator-5d7dcc7948-8mnz4	1/1	Running	0	11h

PersistentVolumeClaims (PVCs):

```
% kubectl get pvc
NAME
                           STATUS VOLUME
CAPACITY ACCESS MODES STORAGECLASS AGE
tt-persistent-sample-0
                         Bound
ocidl.volume.oc1.phx.abyhqljrbxcgzyixa4pmmcwiqxgqclc7gxvdnoty367w2qn26tij6kfpx
6qq
250Gi
           RWO
                         oci-by
                                        10m
tt-persistent-sample-1
                          Bound
ocid1.volume.oc1.phx.abyhqljtt4qxxoj5jqiskriskh66hakaw326rbza4uigmuaezdnu53qhh
oaa
250Gi
           RWO
                          oci-bv
                                        10m
```

Verify Connection to the Active TimesTen Database

You can run the kubectl exec command to invoke shells in your Pods and control TimesTen, which is running in those Pods. By default, TimesTen runs in the Pods as the timesten user. Once you have established a shell in the Pod, verify you can connect to the sample database, and that the information from the metadata files is correct. You can optionally run queries against the database or any other operations.

1. Establish a shell in the Pod.

% kubectl exec -it sample-0 -c tt -- /bin/bash

 Connect to the sample database. Verify the information in the metadata files is in the database correctly. For example, attempt to connect to the database as the sampleuser user. Check that the PermSize value of 200 is correct. Check that the sampleuser.emp table exists.



```
% ttIsql sample
Copyright (c) 1996, 2025, Oracle and/or its affiliates. All rights reserved.
Type ? or "help" for help, type "exit" to quit ttIsql.
connect "DSN=sample";
Connection successful:
DSN=sample;UID=timesten;DataStore=/tt/home/timesten/datastore/sample;
DatabaseCharacterSet=AL32UTF8; ConnectionCharacterSet=AL32UTF8; PermSize=200;
DDLReplicationLevel=3;
(Default setting AutoCommit=1)
Command> connect adding "uid=sampleuser;pwd=samplepw" as sampleuser;
Connection successful:
DSN=sample;UID=sampleuser;DataStore=/tt/home/timesten/datastore/sample;
DatabaseCharacterSet=AL32UTF8;ConnectionCharacterSet=AL32UTF8;PermSize=200;
DDLReplicationLevel=3;
(Default setting AutoCommit=1)
sampleuser: Command> tables;
  SAMPLEUSER.EMP
1 table found.
```

Recover from Failure

This example simulates a failure of the active TimesTen database. This is for demonstration purposes only. Do not do this in a production environment.

Use the kubectl delete pod command to delete the active database (sample-0 in this case)

```
% kubectl delete pod sample-0
```

2. Use the kubectl describe command to observe how the Operator recovers from the failure. The Operator promotes the standby database (sample-1) to be active. Any applications that were connected to the sample-0 database are automatically reconnected to the sample-1 database by TimesTen. After a brief outage, the applications can continue to use the database. See "About the High Level State of TimesTenClassic Objects" for information on the health and states of the active standby pair.

Note: In this example, the text for the Message column displays on two lines for three state changes. However, the actual output displays on one line for each of these three state changes.

```
% kubectl describe ttc sample
Name: sample
. . .
Events:
 Type Reason Age From Message
 -----
                 ____
                       ----
                                 _____
 - StateChange 2mls ttclassic TimesTenClassic sample: was Normal, now
ActiveDown
      StateChange 115s ttclassic Pod sample-1 Database Updatable: Yes
 _
      StateChange 115s ttclassic TimesTenClassic sample:was ActiveDown, now
StandbyDown
     StateChange 115s ttclassic Pod sample-1 RepState ACTIVE
      StateChange 110s ttclassic Pod sample-0 High Level State Unknown
 _
     StateChange 63s ttclassic Pod sample-0 Pod Phase Running
     StateChange 63s ttclassic Pod sample-0 Agent Up
      StateChange 63s ttclassic Pod sample-0 Instance Exists
```



```
StateChange 63s ttclassic Pod sample-0 Daemon Up
      StateChange 63s ttclassic Pod sample-0 Database None
      StateChange 42s ttclassic Pod sample-0 Database Loaded
 _
      StateChange 42s ttclassic Pod sample-0 Database Updatable: No
 _
      StateChange 42s ttclassic Pod sample-0 RepAgent Running
       StateChange 42s ttclassic Pod sample-0 CacheAgent Not Running
 _
       StateChange 42s ttclassic Pod sample-0 RepScheme Exists
 _
       StateChange 42s ttclassic Pod sample-0 RepState IDLE
 _
 _
       StateChange 36s ttclassic Pod sample-O High Level State Healthy
       StateChange 36s ttclassic Pod sample-0 RepState STANDBY
 _
       StateChange 36s ttclassic TimesTenClassic sample:was StandbyDown, now
Normal
```

Kubernetes has automatically respawned a new sample-0 Pod to replace the Pod you deleted. The Operator configured TimesTen inside of that Pod, bringing the database in the Pod up as the new standby database. The replicated pair of databases are once again functionally normally.

Clean Up

This example concludes with deleting the databases and all objects associated with TimesTenClassic. These steps are used for example purposes only. Doing these steps results in the termination of the Pods that are running the TimesTen databases as well as the deletion of the TimesTen databases themselves.

1. Delete the ConfigMap object. (sample, in this example.)

```
% kubectl delete configmap sample
configmap "sample" deleted
```

Delete the TimesTenClassic object and the underlying objects.

```
% kubectl delete -f sample.yaml
timestenclassic.timesten.oracle.com "sample" deleted
```

Verify the Pods that were running the TimesTen databases no longer exist.

% kubectl get pods				
NAME	READY	STATUS	RESTARTS	AGE
timesten-operator-5d7dcc7948-8mnz4	1/1	Running	0	5d23h

4. Delete the persistent storage used to hold your databases. You have to do this manually.

```
% kubectl get pvc
NAME STATUS VOLUME
CAPACITY ACCESS MODES STORAGECLASS AGE
tt-persistent-sample-0 Bound
...
tt-persistent-sample-1 Bound
...
% kubectl delete pvc tt-persistent-sample-0
persistentvolumeclaim "tt-persistent-sample-0" deleted
% kubectl delete pvc tt-persistent-sample-1
persistentvolumeclaim "tt-persistent-sample-1
```

 If you no longer want to run the Operator, you can stop it. Navigate to the /deploy directory (kube_files/deploy, in this example) and use the kubectl delete command to stop the operator.

```
% cd kube_files/deploy
% kubectl delete -f operator.yaml
deployment.apps "timesten-operator" deleted
```



B

TimesTen Cache in TimesTen Classic Example

This appendix provides a working example for using TimesTen Cache with active standby pair of TimesTen databases in your Kubernetes environment. This example should not be used for production purposes. It assumes a test environment. Your Oracle Database should be customized with the settings specific to your environment.

Topics:

- Set Up the Oracle Database to Cache Data
- Create Metadata Files and a Kubernetes Facility
- Create a TimesTenClassic Object
- Monitor Deployment of a TimesTenClassic Object
- Verify TimesTen Cache Configuration
- Perform Operations On Cache Group Tables
- Clean Up Cache Metadata on the Oracle Database

Set Up the Oracle Database to Cache Data

The following sections describe the tasks that must be performed in the Oracle Database:

- Create the Oracle Database Users
- Grant Privileges to the Cache Administration User
- Create the Oracle Database Tables to Be Cached

Create the Oracle Database Users

Before you can use TimesTen Cache, you must create the following users in your Oracle database:

- A cache administration user. This user creates and maintains Oracle Database objects that store information about the cache environment. This user also enforces predefined behaviors of cache group types.
- One or more schema users who owns Oracle Database tables that are cached in a TimesTen database.

See Create the Oracle Database Users and Default Tablespace in the Oracle TimesTen In-Memory Database Cache Guide for information.

This example creates the cacheuser2 cache administration user and the oratt schema user in the Oracle Database.

 Create a shell from which you can access your Oracle Database and then use SQL*Plus to connect to the Oracle Database as the sys user. Then, create a default tablespace to store the TimesTen Cache management objects. See Create the Oracle Database Users and Default Tablespace in the Oracle TimesTen In-Memory Database Cache Guide for information.



This example creates the cachetablespace2 tablespace.

% sqlplus sys/syspwd@oracache as sysdba

SQL> CREATE TABLESPACE cachetablespace2 DATAFILE 'datatt2.dbf' SIZE 100M;

Tablespace created.

Use SQL*Plus to create the schema user. Grant this schema user the minimum privileges
required to create tables in the Oracle Database to be cached in your TimesTen database.

This example creates the oratt schema user.

SQL> CREATE USER oratt IDENTIFIED BY oraclepwd;

User created.

SQL> GRANT CREATE SESSION, RESOURCE TO oratt;

Grant succeeded.

3. Use SQL*Plus to create the cache administration user. Assign the cachetablespace2 tablespace to this user. You will later use the same name of this Oracle cache administration user in the cacheUser metadata file.

This example creates the cacheuser2 user.

```
SQL> CREATE USER cacheuser2 IDENTIFIED BY oraclepwd
DEFAULT TABLESPACE cachetablespace2
QUOTA UNLIMITED ON cachetablespace2;
User created.
SQL> commit;
Commit complete.
SQL> exit
```

Grant Privileges to the Cache Administration User

The cache administration user must be granted a specific set of privileges depending on the cache group types that will be created in the TimesTen databases and the operations performed on those cache groups. TimesTen provides the grantCacheAdminPrivileges.sql SQL*Plus script that you can run in your Oracle Database to grant the cache administration user the minimum set of privileges required to perform cache operations. See Grant Privileges to the Oracle Cache Administration User and see Required Privileges for Cache Administration User for Cache Operations in the Oracle TimesTen In-Memory Database Cache Guide.

Perform these steps to run the grantCacheAdminPrivileges.sql script:

 Create a shell from which you can access your Oracle Database, and then from the directory of your choice, create an empty subdirectory. This example creates the oraclescripts subdirectory.

% mkdir -p oraclescripts

2. From your Linux development host, use the kubectl cp command to copy the grantCacheAdminPrivileges.sql Script from the *installation_dir/*oraclescripts directory on your Linux development host to the oraclescripts directory that you just created. Recall that the *installation_dir* directory was created when you unpacked the TimesTen distribution.



% cp /installation_dir/oraclescripts/grantCacheAdminPrivileges.sql database-oracle:oraclescripts/grantCacheAdminPrivileges.sql

3. From your shell, verify the script is located in the oraclescripts directory.

```
% ls oraclescripts
grantCacheAdminPrivileges.sql
```

4. Use SQL*Plus to connect to the Oracle Database as the sys user. Then, run the oraclescripts/grantCacheAdminPrivileges.sql script. This script grants the cacheuser2 cache administration user the minimum set of privileges required to perform cache group operations. See Grant Privileges to the Oracle Cache Administration User in the Oracle TimesTen In-Memory Database Cache Guide.

```
% sqlplus sys/syspwd@oracache as sysdba
SQL> @grantCacheAdminPrivileges "cacheuser2";
Please enter the administrator user id
The value chosen for administrator user id is cacheuser2
TT CACHE ADMIN ROLE role already exists
0. Granting the CREATE SESSION privilege to CACHEUSER2
1. Granting the TT CACHE ADMIN ROLE to CACHEUSER2
2. Granting the DBMS LOCK package privilege to CACHEUSER2
3. Granting the DBMS DDL package privilege to CACHEUSER2
4. Granting the DBMS FLASHBACK package privilege to CACHEUSER2
5. Granting the CREATE SEQUENCE privilege to CACHEUSER2
6. Granting the CREATE CLUSTER privilege to CACHEUSER2
7. Granting the CREATE OPERATOR privilege to CACHEUSER2
8. Granting the CREATE INDEXTYPE privilege to CACHEUSER2
9. Granting the CREATE TABLE privilege to CACHEUSER2
10. Granting the CREATE PROCEDURE privilege to CACHEUSER2
11. Granting the CREATE ANY TRIGGER privilege to CACHEUSER2
12. Granting the GRANT UNLIMITED TABLESPACE privilege to CACHEUSER2
13. Granting the DBMS LOB package privilege to CACHEUSER2
14. Granting the SELECT on SYS.ALL OBJECTS privilege to CACHEUSER2
15. Granting the SELECT on SYS.ALL SYNONYMS privilege to CACHEUSER2
16. Checking if the cache administrator user has permissions on the
   default tablespace
   Permission exists
18. Granting the CREATE TYPE privilege to CACHEUSER2
19. Granting the SELECT on SYS.GV$LOCK privilege to CACHEUSER2
20. Granting the SELECT on SYS.GV$SESSION privilege to CACHEUSER2
21. Granting the SELECT on SYS.DBA DATA FILES privilege to CACHEUSER2
22. Granting the SELECT on SYS.USER USERS privilege to CACHEUSER2
23. Granting the SELECT on SYS.USER FREE SPACE privilege to CACHEUSER2
24. Granting the SELECT on SYS.USER TS QUOTAS privilege to CACHEUSER2
25. Granting the SELECT on SYS.USER SYS PRIVS privilege to CACHEUSER2
26. Granting the SELECT on SYS.V$DATABASE privilege to CACHEUSER2 (optional)
27. Granting the SELECT ANY TRANSACTION privilege to CACHEUSER2
******** Initialization for cache admin user done successfully ********
```

You have successfully run the grantCacheAdminPrivileges.sql script in the Oracle Database.

Create the Oracle Database Tables to Be Cached

This example creates two tables in the oratt user schema. See "Create the Oracle Database Users" for information on this user.

• readtab: This table will later be cached in a read-only cache group.



- writetab: This table will later be cached in an AWT cache group.
- Create a shell from which you can access your Oracle Database and then use SQL*Plus to connect to the Oracle Database as the sys user. Then create the oratt.readtab and the oratt.writetab tables.

```
% sqlplus sys/syspwd@oracache as sysdba
SQL> CREATE TABLE oratt.readtab (keyval NUMBER NOT NULL PRIMARY KEY,
    str VARCHAR2(32));
Table created.
SQL> CREATE TABLE oratt.writetab (pk NUMBER NOT NULL PRIMARY KEY,
    attr VARCHAR2(40));
```

Table created.

 Use SQL*Plus to insert rows into the oratt.readtab and the oratt.writetab tables. Then verify the rows have been inserted.

```
SQL> INSERT INTO oratt.readtab VALUES (1,'Hello');
1 row created.
SQL> INSERT INTO oratt.readtab VALUES (2,'World');
1 row created.
SQL> INSERT INTO oratt.writetab VALUES (100, 'TimesTen');
1 row created.
SQL> INSERT INTO oratt.writetab VALUES (101, 'Cache');
1 row created.
```

```
SOL> commit;
```

Commit complete.

Verify the rows have been inserted into the tables.

```
SQL> SELECT * FROM oratt.readtab;
```

KEYVAL STR 1 Hello 2 World SQL> SELECT * FROM oratt.writetab; PK ATTR 100 TimesTen 101 Cache

3. Use SQL*Plus to grant the SELECT privilege on the oratt.readtab table and the SELECT, INSERT, UPDATE, and DELETE privileges on the oratt.writetab table to the cache administration user (cacheuser2, in this example).

SQL> GRANT SELECT ON oratt.readtab TO cacheuser2;



Grant succeeded.
SQL> GRANT SELECT ON oratt.writetab TO cacheuser2;
Grant succeeded.
SQL> GRANT INSERT ON oratt.writetab TO cacheuser2;
Grant succeeded.
SQL> GRANT UPDATE ON oratt.writetab TO cacheuser2;
Grant succeeded.
SQL> GRANT DELETE ON oratt.writetab TO cacheuser2;
Grant succeeded.

4. Use SQL*Plus to query the nls_database_parameters system view to determine the Oracle Database database character set. The Oracle Database database character set must match the TimesTen database character set. (The TimesTen database character set will be set later. See "Create Metadata Files and a Kubernetes Facility" for details.)

In this example, the query returns the AL32UTF8 database character set.

```
SQL> SELECT value FROM nls_database_parameters WHERE
    parameter='NLS_CHARACTERSET';
VALUE
AL32UTF8
```

You have successfully created the Oracle Database tables that will be cached in the TimesTen cache group tables.

Create Metadata Files and a Kubernetes Facility

There are metadata files that are specific to using TimesTen Cache:

- cacheUser: This file is required. The user in this file is created in the TimesTen databases and serves as the cache manager. The name of this user must match the name of the cache administration user that you created in the Oracle Database. See "Create the Oracle Database Users" for information on the cache administration user in the Oracle Database. Also see "cacheUser" for more information on the cacheUser metadata file.
- cachegroups.sql: This file is required. The contents of this file contain the CREATE CACHE GROUP definitions. The file can also contain the LOAD CACHE GROUP statement and the built-in procedures to update statistics on the cache group tables (such as, ttOptEstimateStats and ttOptUpdateStats). See "cachegroups.sql" for more information on this file.
- tnsnames.ora: This file is required. It defines Oracle Net Services to which applications connect. For TimesTen Cache, this file configures the connectivity between TimesTen and the Oracle Database (from which data is being cached). In this context, TimesTen is the application that is the connection to the Oracle Database. See "tnsnames.ora" for more information on this file.
- sqlnet.ora: This file may be required. It may be necessary depending on your Oracle
 Database configuration. The file defines options for how client applications communicate
 with the Oracle Database. In this context, TimesTen is the application. The tnsnames.ora
 and sqlnet.ora files together define how an application communicates with the Oracle
 Database. See "sqlnet.ora" for information on this file.

- db.ini: This file is required if you are using TimesTen Cache. The contents of this file contain TimesTen connection attributes for your TimesTen databases, which will be included in TimesTen's sys.odbc.ini file. For TimesTen Cache, you must specify the OracleNetServiceName and the DatabaseCharacterSet connection attributes. The DatabaseCharacterSet connection attribute must match the Oracle database character set. See "db.ini" for more information on this file.
- schema.sql: The contents of this file contain database objects, such as tables, sequences, and users. The instance administrator uses the ttIsql utility to run this file immediately after the database is created. This file is run before the Operator configures TimesTen Cache or replication, so ensure there are no cache definitions in this file.

In TimesTen Cache, one or more cache table users own the cache tables. If this cache table user is not the cache manager user, then you must specify the schema.sql file and in it you must include the schema user and assign the appropriate privileges to this schema user. For example, if the oratt schema user was created in the Oracle Database, and this user is not the TimesTen cache manager user, you must create the TimesTen oratt user in this file. See "Create the Oracle Database Users" for more information on the schema users in the Oracle Database. Also see "schema.sql" for more information on the schema schema.sql file.

In addition, you can use these other supported metadata files:

- adminUser: The user in this file is created in the TimesTen databases and is granted ADMIN privileges. See "adminUser" for more information on this file.
- epilog.sql: The contents of this file contain operations that must be performed after the Operator configures replication. For example, if you are using XLA, you could create replicated bookmarks for XLA in this file. This file is run after cache and replication have been configured. See "epilog.sql" for more information on this file.

You can include these metadata files in one or more Kubernetes facilities (for example, in a Kubernetes Secret, in a ConfigMap, or in an init container). This ensures the metadata files are populated in the /ttconfig directory of the TimesTen containers. Note that there is no requirement as to how to get the metadata files into this /ttconfig directory. See "Populate the /ttconfig Directory" for more information.

This example uses the ConfigMap facility to populate the /ttconfig directory in your TimesTen containers. The adminUser, db.ini, schema.sql, cacheUser, cachegroups.sql, tnsnames.ora, and sqlnet.ora metadata files are used in this example.

On your Linux development host:

1. From the directory of your choice, create an empty subdirectory for the metadata files. This example creates the cm_cachetest subdirectory. (The cm_cachetest directory is used in the remainder of this example to denote this directory.)

```
% mkdir -p cm cachetest
```

2. Navigate to the ConfigMap directory.

% cd cm_cachetest

3. Create the adminUser file in this ConfigMap directory (cm_cachetest, in this example). In this adminUser file, create the sampleuser user with the samplepw password.

vi adminUser

sampleuser/samplepw

4. Create the db.ini file in this ConfigMap directory (cm_cachetest, in this example). In this db.ini file, define the PermSize, DatabaseCharacterSet, and the OracleNetServiceName



connection attributes. The DatabaseCharacterSet value must match the database character set value in the Oracle Database. See "Create the Oracle Database Tables to Be Cached" for information on how to query the nls_database_parameters system view to determine the Oracle Database database character set. In this example, the value is AL32UTF8.

```
vi db.ini
PermSize=200
```

DatabaseCharacterSet=AL32UTF8 OracleNetServiceName=Oracache

5. Create the schema.sql file in this ConfigMap directory (cm_cachetest, in this example). In this example, create the oratt user. Recall that this user was previously created in the Oracle Database. See "Create the Oracle Database Users" for information on the oratt user in the Oracle Database.

```
vi schema.sql
```

create user oratt identified by ttpwd; grant admin to oratt;

6. Create the cacheUser metadata file in this ConfigMap directory (cm_cachetest, in this example). The cacheUser file must contain one line of the form cacheUser/ttpassword/ orapassword, where cacheUser is the user you wish to designate as the cache manager in the TimesTen database, ttpassword is the TimesTen password you wish to assign to this user, and orapassword is the Oracle Database password that has already been assigned to the Oracle Database cache administration user. Note that the cacheUser name in this file must match the Oracle Database cache administration user that you previously created. See "Create the Oracle Database Users" for more information on the Oracle Database cache administration user.

In this example, the cacheuser2 user with password of oraclepwd was already created in the Oracle Database. Therefore, supply cacheuser2 as the TimesTen cache manager user. You can assign any TimesTen password to this TimesTen cache manager user. This example assigns ttpwd.

vi cacheuser

cacheuser2/ttpwd/oraclepwd

7. Create the cachegroups.sql metadata file in this ConfigMap directory (cm_cachetest, in this example). The cachegroups.sql file contains the cache group definitions. In this example, a dynamic AWT cache group and a read-only cache group are created. In addition, the LOAD CACHE GROUP statement is included to load rows from the oratt.readtab cached table in the Oracle Database into the oratt.readtab cache table in the TimesTen database.

vi cachegroups.sql

CREATE DYNAMIC ASYNCHRONOUS WRITETHROUGH CACHE GROUP writecache FROM oratt.writetab (pk NUMBER NOT NULL PRIMARY KEY,attr VARCHAR2(40)); CREATE READONLY CACHE GROUP readcache AUTOREFRESH INTERVAL 5 SECONDS FROM oratt.readtab (keyval NUMBER NOT NULL PRIMARY KEY,str VARCHAR2(32));

LOAD CACHE GROUP readcache COMMIT EVERY 256 ROWS;


Create the tnsnames.ora metadata file in this ConfigMap directory (cm_cachetest, in this example).

```
vi tnsnames.ora
OraTest =
 (DESCRIPTION =
 (ADDRESS = (PROTOCOL = TCP)(HOST = database.myhost.svc.cluster.local)
  (PORT = 1521))
 (CONNECT_DATA =
  (SERVER = DEDICATED)
  (SERVICE_NAME = OraTest.my.sample.com)))
OraCache =
 (DESCRIPTION =
  (ADDRESS = (PROTOCOL = TCP)(HOST = database.myhost.svc.cluster.local)
  (PORT = 1521))
 (CONNECT_DATA =
  (SERVER = DEDICATED)
  (SERVICE_NAME = OraCache.my.sample.com)))
```

Create the sqlnet.ora metadata file in this ConfigMap directory (cm_cachetest, in this example).

```
vi sqlnet.ora
NAME.DIRECTORY_PATH= {TNSNAMES, EZCONNECT, HOSTNAME}
SQLNET.EXPIRE_TIME = 10
SSL VERSION = 1.2
```

10. Use the Linux ls command to verify the metadata files are in the ConfigMap directory (cm cachetest, in this example).

```
% ls
adminUser cacheUser schema.sql tnsnames.ora
cachegroups.sql db.ini sqlnet.ora
```

11. Create the ConfigMap. The files in the cm_cachetest directory are included in the ConfigMap and, later, will be available in the TimesTen containers.

In this example:

- The name of the ConfigMap is cachetest. Replace cachetest with a name of your choosing. (cachetest is represented in **bold** in this example.)
- This example uses cm_cachetest as the directory where the files that will be copied into the ConfigMap reside. If you use a different directory, replace cm_cachetest with the name of your directory. (cm cachetest is represented in **bold** in this example.)

Use the kubectl create command to create the ConfigMap:

```
% kubectl create configmap cachetest --from-file=cm_cachetest
configmap/cachetest created
```

12. Use the kubectl describe command to verify the contents of the ConfigMap. (cachetest, in this example.) The metadata files are represented in **bold**.

```
% kubectl describe configmap cachetest;
Name: cachetest
Namespace: mynamespace
Labels: <none>
Annotations: <none>
Data
====
tnsnames.ora:
```



```
OraTest =
 (DESCRIPTION =
 (ADDRESS = (PROTOCOL = TCP)(HOST = database.myhost.svc.cluster.local)
 (PORT = 1521))
 (CONNECT_DATA =
 (SERVICE_NAME = OraTest.my.sample.com)))
OraCache =
 (DESCRIPTION =
 (ADDRESS = (PROTOCOL = TCP)(HOST = database.myhost.svc.cluster.local)
 (PORT = 1521))
 (CONNECT_DATA =
 (SERVER = DEDICATED)
 (SERVICE_NAME = OraCache.my.sample.com)))
```

adminUser:

---sampleuser/samplepw

cacheUser:

---cacheuser2/ttpwd/oraclepwd

cachegroups.sql:

```
CREATE DYNAMIC ASYNCHRONOUS WRITETHROUGH CACHE GROUP writecache
FROM oratt.writetab (
    pk NUMBER NOT NULL PRIMARY KEY,
    attr VARCHAR2(40)
);
```

```
CREATE READONLY CACHE GROUP readcache
AUTOREFRESH
INTERVAL 5 SECONDS
FROM oratt.readtab (
keyval NUMBER NOT NULL PRIMARY KEY,
str VARCHAR2(32)
);
```

LOAD CACHE GROUP readcache COMMIT EVERY 256 ROWS;

db.ini:

```
permSize=200
databaseCharacterSet=AL32UTF8
oracleNetServiceName=Oracache
```

schema.sql:

```
create user oratt identified by ttpwd;
grant admin to oratt;
```

sqlnet.ora:

```
NAME.DIRECTORY_PATH= {TNSNAMES, EZCONNECT, HOSTNAME}
SQLNET.EXPIRE_TIME = 10
SSL VERSION = 1.2
```

Events: <none>



You have successfully created and deployed the cachetest ConfigMap.

Create a TimesTenClassic Object

This section creates a TimesTenClassic object.

Perform these steps:

 Create an empty YAML file. You can choose any name, but you may want to use the same name you used for the name of the TimesTenClassic object. (In this example, cachetest.) The YAML file contains the definitions for the TimesTenClassic object. See TimesTenClassicSpecSpec for information on the fields that you must specify in this YAML file as well as the fields that are optional.

In this example, note these fields:

- name: Replace cachetest with the name of your TimesTenClassic object (represented in **bold**).
- storageClassName: Replace oci-bv with the name of the storage class used to allocate PersistentVolumes to hold TimesTen.
- storageSize: Replace 250Gi with the amount of storage that should be requested for each Pod to hold TimesTen. Note: This example assumes a production environment and uses a value of 250Gi for storageSize. For demonstration purposes, a value of 50Gi is adequate.
- image: Replace container-registry.oracle.com/timesten/timesten:22.1.1.34.0 with the location and name of your image.
- imagePullSecret: Replace sekret with the image pull secret that Kubernetes should use to fetch the TimesTen image.
- dbConfigMap: This example uses one ConfigMap (called cachetest) for the metadata files (represented in bold).

% vi cachetest.yaml

```
apiVersion: timesten.oracle.com/v4
kind: TimesTenClassic
metadata:
    name: cachetest
spec:
    ttspec:
    storageClassName: oci-bv
    storageSize: 250Gi
    image: container-registry.oracle.com/timesten/timesten:22.1.1.34.0
    imagePullSecret: sekret
    dbConfigMap:
    - cachetest
```

2. Use the kubectl create command to create the TimesTenClassic object from the contents of the YAML file (in this example, cachetest.yaml). Doing so begins the process of deploying your active standby pair of TimesTen databases in the Kubernetes cluster.

```
% kubectl create -f cachetest.yaml
timestenclassic.timesten.oracle.com/cachetest created
```

You have successfully created the TimesTenClassic object in the Kubernetes cluster. The process of deploying your TimesTen databases begins, but is not yet complete.



Monitor Deployment of a TimesTenClassic Object

Use the kubectl get and the kubectl describe commands to monitor the progress of the active standby pair as it is provisioned.

1. Use the kubectl get command and review the STATE field. Observe the value is Initializing. The active standby pair provisioning has begun, but is not yet complete.

% kubectl get ttc cachetest NAME STATE ACTIVE AGE cachetest Initializing None 41s

2. Use the kubectl get command again to see if value of the STATE field has changed. In this example, the value is Normal, indicating the active standby pair of databases are now provisioned and the process is complete.

```
% kubectl get ttc cachetest
NAME STATE ACTIVE AGE
cachetest Normal cachetest-0 3m58s
```

3. Use the kubectl describe command to view the active standby pair provisioning in detail.

Note the following:

- The cachetest Configmap has been correctly referenced in the dbConfigMap field (represented in **bold**).
- The cache agent is running in the active and the standby Pods (represented in **bold**).
- The cache administration user UID and password have been set in the active and the standby Pods (represented in **bold**).
- Two cache groups have been created in the active and the standby Pods (represented in **bold**).
- The replication agent is running in the active and standby Pods (represented in **bold**).

```
% kubectl describe ttc cachetest
Name: cachetest
Namespace: mynamespace
           <none>
Labels:
Annotations: <none>
API Version: timesten.oracle.com/v4
Kind:
            TimesTenClassic
Metadata:
 Creation Timestamp: 2025-01-16T03:29:48Z
 Generation: 1
 Resource Version: 78390500
 Self Link: /apis/timesten.oracle.com/v4/namespaces/mynamespace/
timestenclassics/cachetest
 UID:
                    2b18d81d-15a9-11eb-b999-be712d29a81e
Spec:
 Ttspec:
   Db Config Map:
     cachetest
   Image:
                      container-registry.oracle.com/timesten/timesten:22.1.1.34.0
   Image Pull Policy: Always
   Image Pull Secret: sekret
   Storage Class Name: oci-bv
   Storage Size:
                      250Gi
Status:
 Active Pods: cachetest-0
 High Level State: Normal
```

```
Last Event:
                 28
Pod Status:
 Cache Status:
   Cache Agent:
                      Running
   Cache UID Pwd Set: true
   N Cache Groups:
                      2
 Db Status:
   Db:
                 Loaded
   Db Id:
                 30
   Db Updatable: Yes
 Initialized:
                 true
 Pod Status:
   Agent:
                        Up
   Last Time Reachable: 1603510527
   Pod IP:
                       10.244.7.92
   Pod Phase:
                       Running
 Replication Status:
   Last Time Rep State Changed: 0
   Rep Agent:
                               Running
   Rep Peer P State:
                               start
   Rep Scheme:
                               Exists
   Rep State:
                               ACTIVE
 Times Ten Status:
   Daemon: Up
                  Exists
   Instance:
                  22.1.1.34.0
   Release:
 Admin User File: true
 Cache User File: true
 Cg File:
                   true
 High Level State: Healthy
 Intended State:
                   Active
 Name:
                   cachetest-0
 Schema File:
                  true
 Cache Status:
   Cache Agent:
                      Running
   Cache UID Pwd Set: true
   N Cache Groups:
                      2
 Db Status:
   Db:
                 Loaded
                 30
   Db Id:
   Db Updatable: No
 Initialized:
                true
 Pod Status:
   Agent:
                        Up
   Last Time Reachable: 1603510527
   Pod IP:
                        10.244.8.170
   Pod Phase:
                       Running
 Replication Status:
   Last Time Rep State Changed: 1603510411
   Rep Agent:
                                Running
   Rep Peer P State:
                                start
   Rep Scheme:
                                Exists
   Rep State:
                                STANDBY
 Times Ten Status:
                  Up
   Daemon:
   Instance:
                   Exists
                    22.1.1.34.0
   Release:
 Admin User File:
                   true
 Cache User File:
                    true
 Cg File:
                     true
 High Level State:
                     Healthy
 Intended State:
                     Standby
```

```
Name:
                       cachetest-1
   Schema File:
                      true
 Rep Create Statement: create active standby pair "cachetest" on
 "cachetest-0.cachetest.mynamespace.svc.cluster.local", "cachetest" on
 "cachetest-1.cachetest.mynamespace.svc.cluster.local" NO RETURN store
 "cachetest" on "cachetest-0.cachetest.mynamespace.svc.cluster.local"
 PORT 4444 FAILTHRESHOLD 0 store "cachetest" on
 "cachetest-1.cachetest.mynamespace.svc.cluster.local" PORT 4444 FAILTHRESHOLD 0
                      4444
 Rep Port:
 Status Version:
                     1.0
Events:
 Type Reason Age From Message
                 5m40s ttclassic Secret tt2b18d81d-15a9-11eb-b999-be712d29a81e
      Create
created
                 5m40s ttclassic Service cachetest created
 -
    Create
      Create 5m40s ttclassic StatefulSet cachetest created
 _
 _
     StateChange 4m28s ttclassic Pod cachetest-0 Agent Up
     StateChange 4m28s ttclassic Pod cachetest-0 Release 22.1.1.34.0
 _
    StateChange 4m28s ttclassic Pod cachetest-0 Daemon Up
 _
    StateChange 3m18s ttclassic Pod cachetest-0 RepScheme None
 _
     StateChange 3m18s ttclassic Pod cachetest-0 RepAgent Not Running
     StateChange 3m18s ttclassic Pod cachetest-0 RepState IDLE
     StateChange 3m18s ttclassic Pod cachetest-0 Database Loaded
      StateChange 3m18s ttclassic Pod cachetest-0 Database Updatable
      StateChange 3m18s ttclassic Pod cachetest-0 CacheAgent Not Running
      StateChange 2m57s ttclassic Pod cachetest-0 CacheAgent Running
      StateChange 2m47s ttclassic Pod cachetest-1 Agent Up
       StateChange 2m47s ttclassic Pod cachetest-1 Release 22.1.1.34.0
 _
       StateChange 2m46s ttclassic Pod cachetest-0 RepAgent Running
       StateChange 2m46s ttclassic Pod cachetest-0 RepScheme Exists
       StateChange 2m46s ttclassic Pod cachetest-0 RepState ACTIVE
 _
       StateChange 2m46s ttclassic Pod cachetest-1 Daemon Up
      StateChange 2m9s ttclassic Pod cachetest-1 CacheAgent Running
      StateChange 2m9s ttclassic Pod cachetest-1 Database Not Updatable
 _
      StateChange 2m9s ttclassic Pod cachetest-1 Database Loaded
 _
      StateChange 2m9s ttclassic Pod cachetest-1 RepAgent Not Running
 _
      StateChange 2m9s ttclassic Pod cachetest-1 RepScheme Exists
 _
      StateChange 2m9s ttclassic Pod cachetest-1 RepState IDLE
 _
      StateChange 2m3s ttclassic Pod cachetest-1 RepAgent Running
       StateChange 118s ttclassic Pod cachetest-1 RepState STANDBY
 _
       StateChange 118s ttclassic TimesTenClassic was Initializing, now Normal
```

Your active standby pair of TimesTen databases are successfully deployed (as indicated by Normal.) You are now ready to verify that TimesTen Cache is configured correctly and is working properly.

Verify TimesTen Cache Configuration

To verify that TimesTen Cache is configured correctly and is working properly, perform the following steps:

1. Review the active (cachetest-0, in this example) Pod and the standby Pod (cachetest-1, in this example).

% kubectl get pods				
NAME	READY	STATUS	RESTARTS	AGE
cachetest-0	2/2	Running	0	8m16s
cachetest-1	2/2	Running	0	8m15s
timesten-operator-f84766548-tch7s	1/1	Running	0	36d



 Use the kubectl exec -it command to invoke the shell in the active Pod (cachetest-0, in this example).

```
% kubectl exec -it cachetest-0 -c tt -- /bin/bash
```

 Use ttIsql to connect to the cachetest database. Confirm the TimesTen connection attributes are correct. In particular, note that the OracleNetServiceName connection attribute is correctly set to Oracache (represented in **bold**).

```
% ttIsql cachetest;
Copyright (c) 1996, 2025, Oracle and/or its affiliates. All rights reserved.
Type ? or "help" for help, type "exit" to quit ttIsql.
connect "DSN=cachetest";
Connection successful: DSN=cachetest;UID=timesten;DataStore=/tt/home/timesten/
datastore/cachetest;
```

```
DatabaseCharacterSet=AL32UTF8;ConnectionCharacterSet=AL32UTF8;AutoCreate=0;
PermSize=200;OracleNetServiceName=Oracache;DDLReplicationLevel=3;
ForceDisconnectEnabled=1;
(Default setting AutoCommit=1)
```

4. Use the ttIsql cachegroups to view the definition of the cacheuser2.readcache and the cacheuser2.writecache cache groups.

```
Command> cachegroups;
```

Cache Group CACHEUSER2.READCACHE:

Cache Group Type: Read Only Autorefresh: Yes Autorefresh Mode: Incremental Autorefresh State: On Autorefresh Interval: 5 Seconds Autorefresh Status: ok Aging: No aging defined

```
Root Table: ORATT.READTAB
Table Type: Read Only
```

Cache Group CACHEUSER2.WRITECACHE:

Cache Group Type: Asynchronous Writethrough (Dynamic) Autorefresh: No Aging: LRU on

Root Table: ORATT.WRITETAB Table Type: Propagate

2 cache groups found.

5. Use ttIsql to query the oratt.readtab cache table. Note that the data from the oratt.readtab cached table in the Oracle Database is correctly loaded in the oratt.readcache cache table in the TimesTen database. Recall that you specified the LOAD CACHE GROUP statement in the cachegroups.sql metadata file. See Create Metadata Files and a Kubernetes Facility for information on this cachegroups.sql metadata file.

Command> SELECT * FROM oratt.readtab; < 1, Hello > < 2, World > 2 rows found.



You have verified that the cache groups were created and data was correctly loaded in the oratt.readtab table.

Perform Operations On Cache Group Tables

The examples in this section perform operations on the oratt.readtab and the oratt.writetab tables to verify that TimesTen Cache is working properly.

- Perform Operations on the oratt.readtab Table
- Perform Operations on the oratt.writetab Table

Perform Operations on the oratt.readtab Table

This section performs operations on the oratt.readtab table.

 Create a shell from which you can access your Oracle Database and then use SQL*Plus to connect to the Oracle Database as the schema user (oratt, in this example). Then, insert a new row, delete an existing row, and update an existing row in the oratt.readtab table of the Oracle Database and commit the changes.

```
% sqlplus oratt/oraclepwd@oracache;
SQL> INSERT INTO oratt.readtab VALUES (3,'Welcome');
1 row created.
SQL> DELETE FROM oratt.readtab WHERE keyval=2;
1 row deleted.
SQL> UPDATE oratt.readtab SET str='Hi' WHERE keyval=1;
1 row updated.
SQL> COMMIT;
Commit complete.
```

Since the read-only cache group was created with an autorefresh interval of 5 seconds, the TimesTen oratt.readtab cache table in the readcache cache group is automatically refreshed after 5 seconds with the committed updates from the cached oratt.readtab table of the Oracle Database. The next step is to test that the data was correctly propagated from the Oracle Database to the TimesTen database.

2. Use the kubectl exec -it command to invoke the shell in the container of the Pod that is running the TimesTen active database (cachetest-0, in this example).

% kubectl exec -it cachetest-0 -c tt -- /bin/bash

3. Use the TimesTen ttIsql utility to connect to the cachetest database. Query the TimesTen oratt.readtab table to verify that the table has been updated with the committed updates from the Oracle Database.

% ttIsql cachetest;

Copyright (c) 1996, 2025, Oracle and/or its affiliates. All rights reserved. Type ? or "help" for help, type "exit" to quit ttIsql.



```
connect "DSN=cachetest";
Connection successful: DSN=cachetest;UID=timesten;DataStore=/tt/home/timesten/
datastore/cachetest;
DatabaseCharacterSet=AL32UTF8;ConnectionCharacterSet=AL32UTF8;AutoCreate=0;
PermSize=200;OracleNetServiceName=Oracache;DDLReplicationLevel=3;
ForceDisconnectEnabled=1;
(Default setting AutoCommit=1)
Command> SELECT * FROM oratt.readtab;
< 1, Hi >
< 3, Welcome >
2 rows found.
```

You have verified that TimesTen Cache is working correctly for the oratt.readtab table and the readcache cachegroup.

Perform Operations on the oratt.writetab Table

This example performs operations on the oratt.writetab table.

1. Use the kubectl exec -it command to invoke the shell in the container of the Pod that is running the TimesTen active database (cachetest-0, in this example).

```
% kubectl exec -it cachetest-0 -c tt -- /bin/bash
```

2. Use the TimesTen ttIsql utility to connect to the cachetest database as the cache manager user (cacheuser2, in this example). Issue a SELECT statement on the TimesTen oratt.writetab table. Recall that the writecache cache group is a dynamic cache group. Thus by issuing the SELECT statement, the cache instance is automatically loaded from the cached Oracle Database table, if the data is not found in the TimeTen cache table.

```
% ttIsql "DSN=cachetest;UID=cacheuser2;PWD=ttpwd;OraclePWD=oraclepwd";
```

```
Copyright (c) 1996, 2025, Oracle and/or its affiliates. All rights reserved. Type ? or "help" for help, type "exit" to quit ttIsql.
```

```
connect "DSN=cachetest;UID=cacheuser2;PWD=*******;OraclePWD=*******";
Connection successful: DSN=cachetest;UID=cacheuser2;DataStore=/tt/home/timesten/
datastore/cachetest;
DatabaseCharacterSet=AL32UTF8;ConnectionCharacterSet=AL32UTF8;AutoCreate=0;
PermSize=200;OracleNetServiceName=Oracache;DDLReplicationLevel=3;
ForceDisconnectEnabled=1;
(Default setting AutoCommit=1)
Command> SELECT * FROM oratt.writetab WHERE pk=100;
```

```
< 100, TimesTen >
```

```
1 row found.
```

3. Use ttIsql to insert a new row, delete an existing row, and update an existing row in the TimesTen oratt.writetab cache table, and commit the changes.

```
Command> INSERT INTO oratt.writetab VALUES (102,'Cache');
1 row inserted.
Command> DELETE FROM oratt.writetab WHERE pk=101;
1 row deleted.
Command> UPDATE oratt.writetab SET attr='Oracle' WHERE pk=100;
1 row updated.
Command> COMMIT;
```



The committed updates on the TimesTen <code>oratt.writetab</code> cache table in the <code>writecache</code> cache group should automatically be propagated to the <code>oratt.writetab</code> table in the Oracle Database.

4. Create a shell from which you can access your Oracle Database and then use SQL*Plus to connect to the Oracle database as the schema user (oratt, in this example). Then query the contents of the oratt.writetab table in the Oracle Database to verify the committed updates from the TimesTen database have been propagated to the oratt.writetab table of the Oracle Database.

You have verified that TimesTen Cache is working correctly for the oratt.writetab table and the writecache cachegroup.

Clean Up Cache Metadata on the Oracle Database

When you create certain types of cache groups in a TimesTen database, TimesTen stores metadata about that cache group in the Oracle Database. If you later delete that TimesTen database, TimesTen does not automatically delete the metadata in the Oracle Database. As a result, metadata can accumulate on the Oracle Database. See Dropping Oracle Database Objects Used by Cache Groups with Autorefresh in the *Oracle TimesTen In-Memory Database Cache Guide* for more information.

However, in a Kubernetes environment, if you provide a cacheUser metadata file and a cachegroups.sql metadata file when you initially create the TimesTenClassic object, then, by default, the Operator automatically cleans up the Oracle Database metadata if you delete that TimesTenClassic object.

If you do not want the Operator to automatically clean up the Oracle Database, you set the cacheCleanup field in the TimesTenClassic object definition to false. See the cacheCleanup entry in "Table 20-3" for more information. Also see "About Configuration Metadata Details" for information on the cacheUser and the cachegroups.sql files.



Create Your Own Oracle Wallet, Certificates, and Secrets for Exposing TimesTen Metrics

By default, the TimesTen Operator automatically exposes TimesTen metrics and uses Transport Layer Security (mutual TLS)/https to serve these metrics. It exposes these metrics to Prometheus or any other scrape target.

The TimesTen Operator works with the TimesTen exporter to expose these metrics. For more information about the TimesTen exporter, see Overview of the TimesTen Kubernetes Operator and the TimesTen Exporter.

If TimesTen metrics are served by https, then by default the TimesTen Operator automatically creates an Oracle Wallet, all necessary certificates, and Kubernetes Secrets so that https can be used to expose TimesTen metrics securely.

For more information about how the TimesTen Operator exposes TimesTen metrics, see Expose TimesTen Metrics with the TimesTen Kubernetes Operator.

Although not recommended, you have the option of creating your own Oracle Wallet, certificates, and Kubernetes Secrets for use in exporting these metrics.

This appendix shows you how.

Topics:

- About Creating Your Own Oracle Wallet, Certificates, and Kubernetes Secrets
- Create Your Own Oracle Wallet, Certificates, and Kubernetes Secrets

About Creating Your Own Oracle Wallet, Certificates, and Kubernetes Secrets

You have the option of creating your own Oracle Wallet, certificates, and Kubernetes Secrets to serve TimesTen metrics by https. If you choose this option, the TimesTen Operator cannot create a PodMonitor object with sufficient information to allow Prometheus to access TimesTen. You can create a PodMonitor object yourself or otherwise edit the Prometheus configuration files to cause Prometheus to scrape TimesTen metrics.

After you create a Kubernetes Secret containing an Oracle Wallet (that contains the necessary certificates), you must include this Secret in your TimesTenClassic or TimesTenClassic object YAML manifest file. You do this by specifying the .spec.ttspec.prometheus.certSecret datum in your object definition.

Here is a code snippet of a TimesTenClassic object YAML manifest file:

```
apiVersion: timesten.oracle.com/v4
kind: TimesTenClassic
metadata:
   name: samplecertsecret
spec:
   ttspec:
```



```
prometheus:
    certSecret: prometheuscert
    port: 7777
```

Note the following:

- The .spec.ttspec.prometheus datum is specified in the TimesTen Classic object YAML manifest file. This causes the TimesTen Operator to automatically deploy the TimesTen exporter in a separate container within each Pod running TimesTen.
- The certSecret datum is specified in the .spec.ttspec.prometheus clause of the TimesTenClassic object. The wallet contained in the prometheusecert Kubernetes Secret is used for Transport Layer Security (mutual TLS)/https.

The port datum is specified. This is the port on which the TimesTen exporter listens. The causes the TimesTen Operator to set up the http server on TCP port 7777 in each TimesTen Pod.

Here is a summary of the tasks you need to complete to create your own Oracle Wallet, certificates, and Kubernetes Secret. The summary also includes the tasks to include the appropriate Kubernetes Secret in a TimesTenClassic or TimesTenScaleout object YAML manifest file. There is a complete example in Create Your Own Oracle Wallet, Certificates, and Kubernetes Secrets.

- Create a TimesTen instance. See Before You Begin.
- Use the TimesTen ttExporter utility to generate the certificates. One of the certificates that is created is the self-signed server certificate. This certificate is placed in an Oracle Wallet. See Create Certificates.
- Place the Oracle Wallet into a Kubernetes Secret. See Create a Kubernetes Secret Containing an Oracle Wallet.
- Specify the name of the Secret in the spec.ttspec.prometheus.certSecret datum of a TimesTenClassic or TimesTenScaleout object YAML manifest file. See Define and Deploy a TimesTenClassic Object.
- Save the PEM formatted file containing the server certificate, the client certificate, and the client private key that were created when you ran the TimesTen ttExporter utility. You need these later to configure the Prometheus server. See Create Certificates.

Here are additional references:

- For information about the spec.ttspec.prometheus.certSecret datum, see TimesTenClassicSpecSpecPrometheus and TimesTenScaleoutSpecSpecPrometheus.
- For information about the command line options for the ttExporter utility, see ttExporter in the Oracle TimesTen In-Memory Database Reference.

Create Your Own Oracle Wallet, Certificates, and Kubernetes Secrets

Let's look at an example that shows you how to create your own Oracle Wallet, certificates, and Kubernetes Secrets. The example also shows you how to specify a Kubernetes Secret that contains an Oracle Wallet in a TimesTenClassic object YAML manifest file.



Note:

The steps are the same for a TimesTenScaleout object.

- Before You Begin
- Create Certificates
- Create a Kubernetes Secret Containing an Oracle Wallet
- Define and Deploy a TimesTenClassic Object

Before You Begin

The TimesTen ttExporter utility is located in the /bin directory of a TimesTen instance. Since the ttExporter utility is located in the TimesTen instance, you are required to create a TimesTen instance on your development host so that you have access to the ttExporter utility. You create a TimesTen instance from a TimesTen installation. A TimesTen installation is created when you unzip the TimesTen distribution.

You must download the TimesTen distribution and unzip it to create a TimesTen installation before beginning these steps. You may have already completed this process if you chose to build the TimesTen container image.

- If you have not already done so, download and unzip the TimesTen distribution into a directory on your development host.
- On your development host from a directory of your choice, create a directory for the TimesTen instance. This example assumes you have previously created the /scratch/ ttuser directory. The example creates the /scratch/ttuser/instance1_exporter_dir directory.

mkdir /scratch/ttuser/instance1_exporter_dir

- Create the TimesTen instance located in the TimesTen installation directory. Replace the following:
 - installation_dir: Name of the TimesTen installation directory. This is the directory where you unzipped the TimesTen distribution.
 - tt22.1.1.34.0: TimesTen release number in ttdottedrelease format, where dottedrelease is 22.1.1.34.0 in this example.
 - instance1 exporter: Name of the TimesTen instance.
 - /scratch/ttuser/instance1_exporter_dir: Location of the TimesTen instance. You created this directory in the previous step.

```
./installation_dir/tt22.1.1.34.0/bin/ttInstanceCreate -name
instance1 exporter -location /scratch/ttuser/instance1 exporter dir
```

The output is similar to the following:

```
Creating instance in /scratch/ttuser/instance1_exporter_dir/
instance1 exporter ...
```

NOTE: The TimesTen daemon startup/shutdown scripts have not been installed.

4. Set the TIMESTEN_HOME environment variable. You must set this variable before you run the ttExporter utility. This example uses the bash Bourne-type shell.

. /scratch/ttuser/instance1 exporter dir/instance1 exporter/bin/ttenv.sh

The output is similar to the following, with not all output shown:

```
LD_LIBRARY_PATH set to ...

PATH set to ...

CLASSPATH set to ...

TIMESTEN_HOME set to /scratch/ttuser/instance1_exporter_dir/

instance1_exporter
```

You successfully created the TimesTen instance on your development host. You are now ready to use the ttExporter utility to create the certificates.

Create Certificates

There are certificates that are necessary in order to use Transport Layer Security (mutual TLS)/https. They are as follows:

- Server certificate: A self-signed certificate that is stored in an Oracle Wallet. This certificate is used by the TimesTen exporter. The name of the Oracle Wallet is cwallet.sso.
- Exported server certificate: The self-signed server certificate in PEM format. This certificate is required for your Prometheus configuration.
- Client certificate and client private key: The client certificate and the client private key
 required for your Prometheus configuration.

The following steps show you how to create these certificates:

1. Check that the TIMESTEN_HOME environment variable is set. You set this environment variable in a previous step. See Before You Begin.

echo \$TIMESTEN HOME



The output is the following:

/scratch/ttuser/instance1 exporter dir/instance1 exporter

2. On your development host, from a directory of your choice, create a subdirectory to store an Oracle Wallet. This example creates the exportercertdir directory.

mkdir -p exportercertdir

3. Create the self-signed server certificate. This certificate is stored as an Oracle Wallet. The name of the file that contains the Oracle Wallet is cwallet.sso. It contains the certificate information required by the TimesTen exporter. Later, you will use a Kubernetes Secret to place the cwallet.sso Oracle Wallet file into the /ttconfig/exporterWallet location of the exporter container.

```
ttExporter -create-server-certificate -certificate-common-name
*.samplecertsecret.mynamespace.svc.cluster.local -certificate-alt-names
*.samplecertsecret.mynamespace.svc.cluster.local -certificate-directory
exportercertdir
```

The -certificate-common-name and -certificate-alt-names ttExporter options are required. For detailed information on these options, see ttExporter in the Oracle TimesTen In-Memory Database Reference.

The -certificate-common-name option is the Common Name (CN) that is included in the certificate. It matches the DNS name where the certificate is installed. This CN can contain only one name. Single-level wildcards are acceptable. In this example, the CN name is *.samplecertsecret.mynamespace.svc.cluster.local, where:

- * is a single level wildcard.
- samplecertsecret is the name of your TimesTenClassic or your TimesTenScaleout object.
- *mynamespace* is the name of your namespace.
- svc.cluster.local completes the required format for the DNS name.

The -certificate-alt-names option is the Subject Alternative Name (SAN) that is included in the certificate. This name includes the CN mentioned previously as well as any other DNS names that need access to the TimesTen Exporter. Single level wildcards are acceptable. In this example, the SAN name includes only the CN name. Specifically, the SAN name is *.samplecertsecret.mynamespace.svc.cluster.local, where:

- * is a single level wildcard.
- samplecertsecret is the name of your TimesTenClassic or your TimesTenScaleout object.
- mynamespace is the name of your namespace.
- svc.cluster.local completes the required format for the DNS name.

Since these options require you to specify the name of the TimesTenClassic (or the TimesTenScaleout) object and the name of your namespace, you must know these names before completing this step. In addition, you must use these same names when defining your TimesTen Classic or your TimesTenScaleout object.



4. Export the server certificate.

```
ttExporter -export-server-certificate exportercertdir/server.crt - certificate-directory exportercertdir
```

This command exports the server certificate in PEM format. In this example, the name of the file that contains the certificate is server.crt. Save this file. You need it later when configuring Prometheus.

5. Create and export the client certificate and the client private key.

```
ttExporter -export-client-certificate exportercertdir/client.crt -export-
client-private-key exportercertdir/key.crt -certificate-directory
exportercertdir
```

This command creates the client certificate. In this example, the contents of the client certificate is stored in the client.crt file. The example also creates the client private key and stores its contents in the key.crt file. Save these files. You need them later when configuring Prometheus.

6. (Optional): Verify the certificates are created.

ls -a exportercertdir

The output is similar to the following:

```
. client.crt server.crt
.. key.crt .ttwallet.BA0F2D86-B6D2-4095-A4D0-CDF1FF89E9BF
```

Verify the ttExporter utility has created the Oracle Wallet.

```
ls -a exportercertdir/.ttwallet*
```

The output is the following:

.. cwallet.sso

You have successfully created the server certificate, the client certificate, and the client private key. Make a note of these files and their location. You need them later. Specifically, you need to specify the cwallet.sso Oracle Wallet file when you create the Kubernetes Secret. See Create a Kubernetes Secret Containing an Oracle Wallet. In addition, you need to specify the server.crt, the client.crt, and the key.crt files later when you configure Prometheus.

Note:

Configuring Prometheus is outside the scope of this book. For information on configuring Prometheus, see About Configuring the TimesTen Exporter and Prometheus with Client Certificate Authentication in the *Oracle TimesTen In-Memory Database Monitoring and Troubleshooting Guide*.



Create a Kubernetes Secret Containing an Oracle Wallet

The following steps show you how to create a Kubernetes Secret for an Oracle Wallet. This Oracle Wallet contains the self-signed server certificate. You created the Oracle Wallet in Create Certificates.

 On your development host, from a directory of your choice, create an empty subdirectory for the Oracle Wallet (the cwallet.sso file). This example creates the walletdir subdirectory.

mkdir -p walletdir

2. Copy the cwallet.sso Oracle Wallet to the directory.

cp exportercertdir/.ttwallet*/cwallet.sso walletdir/cwallet.sso

In this example, the Oracle Wallet is located in the exportercertdir/.ttwallet*/ cwallet.sso walletdir directory. You created this directory in Create Certificates.

 Create the Kubernetes Secret for the Oracle Wallet. Ensure to specify the / exporterWallet directory.

```
kubectl create secret generic prometheuscert --from-
file=exporterWallet=walletdir/cwallet.sso
```

The kubectl create generic secret command does the following:

- Creates the prometheuscert Kubernetes Secret.
- Includes the exporterWallet metadata file. This file is required when including the cwallet.sso file in the Secret.
- Defines walletdir as the location for the cwallet.sso file.
- Defines the cwallet.sso file as the name of the Oracle Wallet file.

The output is the following:

secret/prometheuscert created

You have successfully created the Kubernetes Secret. Make a note of the name of the Secret. You use it later when you create your TimesTenClassic or TimesTenScaleout object.

Define and Deploy a TimesTenClassic Object

Let's define a TimesTenClassic object with the appropriate information such that the TimesTen Operator automatically provisions the TimesTen exporter in a separate container within each Pod that is running TimesTen. Let's use the .spec.ttspec.prometheus.certSecret datum to instruct the TimesTen Operator to use the Oracle Wallet located in the Kubernetes Secret that you previously created. (You created this Secert in Create a Kubernetes Secret Containing an Oracle Wallet).



1. Define a TimesTenClassic object.

```
vi samplecertsecret.yaml
apiVersion: timesten.oracle.com/v4
kind: TimesTenClassic
metadata:
 name: samplecertsecret
spec:
  ttspec:
    storageClassName: oci-bv
    storageSize: 250G
    image: container-registry.oracle.com/timesten/timesten:22.1.1.34.0
    imagePullSecret: sekret
    prometheus:
     certSecret: prometheuscert
     port: 7777
    dbConfigMap:
    - samplecertsecret
```

Note the following:

- The .spec.ttspec.prometheus clause is specified. The TimesTen Operator provisions a TimesTen exporter container in each TimesTen Pod.
- The .spec.ttspec.prometheus.certSecret datum is specified. The value of this datum is the name of the Kubernetes Secret containing the Oracle Wallet you previously created.
- The TimesTen exporter is listening on port 7777.
- 2. Create the TimesTenClassic object from the contents of the YAML file.

kubectl create -f samplecertsecret.yaml

The output is the following:

```
configmap/samplecertsecret created
timestenclassic.timesten.oracle.com/samplecertsecret created
```

3. Wait a few minutes then confirm the TimesTenClassic object is in the Normal state. Confirm also that the TimesTen Operator provisioned a TimesTen exporter container in each TimesTen Pod.

kubectl get ttc samplecertsecret

Output.

NAME	STATE	ACTIVE	AGE
samplecertsecret	Normal	samplecertsecret-0	6m19s

Confirm there is a TimesTen exporter container.

kubectl get pods



Output.

NAME	READY	STATUS	RESTARTS	AGE
samplecertsecret-0	3/3	Running	0	2m59s
samplecertsecret-1	3/3	Running	0	2m59s
timesten-operator-7f77c749fd-lkhtr	1/1	Running	0	60m

The TimesTen Operator provisioned three containers for each TimesTen Pod. One container is running the TimesTen exporter. The TimesTen exporter is listening on port 7777 and functions as an https server.

Your next step is to edit the appropriate Prometheus configuration files to cause Prometheus to scrape TimesTen metrics. For more information about configuring Prometheus, see https:// prometheus.io/docs/prometheus/latest/configuration/configuration/.

