

Oracle® Communications

Cloud Native Core, cnDBTier User Guide



Release 25.2.100
G39115-03
November 2025

ORACLE®

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

1	Introduction	
2	cnDBTier Architecture	
2.1	cnDBTier Setups	3
2.2	Ingress and Egress Communication Over External IPs	6
3	cnDBTier Features	
3.1	Monitoring Cluster Events to Determine Data Loss in Clusters	1
3.2	Storing NDB Logs in PVC	2
3.3	Enhanced Georeplication Recovery using Parallel Backup Transfer and Restore	2
3.4	Transparent Data Encryption (TDE)	4
3.5	Support for CNE Cloud Native Load Balancer (CNLB)	5
3.6	Multithreaded Applier (MTA)	6
3.7	Support for TLS	7
3.8	Network Policies	8
3.9	REST APIs for CNC Console	8
3.10	PVC Health Monitoring	12
3.11	cnDBTier Automated Backup	12
3.12	cnDBTier Password Encryption	13
3.13	Database Backup Encryption and Secure Transfer	13
3.14	Node Selector and Node Affinity	15
3.15	cnDBTier Scaling	16
3.16	Application Service Mesh (ASM) for External Communication	17
3.17	Support for Automated Certificate Lifecycle Management	18
4	cnDBTier APIs	
4.1	cnDBTier Cluster Events APIs	1
4.2	cnDBTier APIs for CNC Console	13
4.3	cnDBTier Backup APIs	43
4.4	cnDBTier Replication Service APIs	50
4.4.1	cnDBTier Switchover APIs	50

4.4.2	cnDBTier Stop Replica APIs	58
4.4.3	cnDBTier Listbackups API	62
4.4.4	cnDBTier Replication Service Heartbeat API	66
4.5	cnDBTier Status APIs	70

5 cnDBTier Metrics

5.1	cnDBTier API Node Read and Write Metrics	2
5.2	cnDBTier Remote Server Backup Transfer Status Metrics	3
5.3	cnDBTier Backup Transfer Status Metrics	4
5.4	cnDBTier Parallel Backup Transfer Progress Metrics	4
5.5	cnDBTier Heartbeat Metrics	5
5.6	cnDBTier Node Status Metrics	5
5.7	cnDBTier Table Read Write Metrics	6
5.8	cnDBTier CPU Usage Metrics	10
5.9	cnDBTier Memory Usage Metrics	11
5.10	cnDBTier Bin Log Usage Metrics	11
5.11	cnDBTier Replication Metrics	11
5.12	cnDBTier Automated Backup Metrics	12
5.13	cnDBTier Fault Recovery State Metrics	13
5.14	cnDBTier Replica Status Metrics	14
5.15	cnDBTier ndbinfo Transporters Metrics	15
5.16	cnDBTier ndbinfo Threadstat Metrics	16
5.17	cnDBTier ndbinfo Operations Per Fragment Metrics	18
5.18	cnDBTier ndbinfo Locks Per Fragment Metrics	21
5.19	cnDBTier ndbinfo Disk Write Speed Aggregate Metrics	24
5.20	cnDBTier Replication Error Skip Info Metrics	26
5.21	cnDBTier BinLog Injector Thread Info Metrics	27
5.22	cnDBTier PVC Health Monitoring Metrics	27
5.23	cnDBTier PVC Capacity Monitoring Metrics	28
5.24	cnDBTier Cert Expiration Time Metrics	29
5.25	DBTier Tc Time Track Metrics	29
5.26	cnDBTier Replication Switchover Status Metrics	33
5.27	JVM Memory Metrics	33

6 cnDBTier Alerts

6.1	cnDBTier Remote Server Backup Transfer Status Alerts	1
6.2	cnDBTier Backup Transfer Status Alerts	3
6.3	cnDBTier Heartbeat Alerts	9
6.4	cnDBTier BinLog Injector Thread Alerts	14
6.5	cnDBTier Replication Error Skip Alerts	16

6.6	cnDBTier Georeplication Recovery Status Alerts	30
6.7	cnDBTier Cluster Status Alerts	33
6.8	cnDBTier Automated Backup Alerts	38
6.9	cnDBTier Bin Log Usage Alerts	52
6.10	cnDBTier Replication Alerts	59
6.11	cnDBTier Memory Usage Alerts	69
6.12	cnDBTier CPU Usage Alerts	74
6.13	cnDBTier Node Status Alerts	76
6.14	cnDBTier Node Data Volume Alerts	79
6.15	cnDBTier Certificate Expiry Alerts	86
6.16	cnDBTier PVC Health Alerts	90
6.17	cnDBTier Backup Manager Svc Down Alerts	95
6.18	cnDBTier Forced Switchover Disabled Alerts	98

7 Maintenance Procedures

7.1	Starting or Stopping cnDBTier	1
7.1.1	Starting cnDBTier	1
7.1.2	Stopping cnDBTier	4
7.2	Starting or Stopping cnDBTier Georeplication Service	5
7.2.1	Starting or Stopping georeplication using dbtreplmgr script	6
7.2.2	Starting cnDBTier Georeplication Between Sites Manually	8
7.2.3	Stopping cnDBTier Georeplication Between Sites Manually	13
7.3	Scaling cnDBTier Pods	20
7.3.1	Horizontal Scaling	20
7.3.1.1	Scaling ndbappmysqld Pods	21
7.3.1.2	Scaling ndbmttd Pods	23
7.3.2	Vertical Scaling	35
7.3.2.1	Scaling ndbmttd Pods	35
7.3.2.2	Scaling ndbappmysqld Pods	46
7.3.2.3	Scaling ndbmysqld Pods	53
7.3.2.4	Vertical Scaling of db-replication-svc Pods	61
7.3.2.5	Scaling of ndbmgmd Pods	69
7.4	Managing TLS	77
7.4.1	Managing TLS for Georeplication	77
7.4.1.1	Enabling TLS for Georeplication	77
7.4.1.2	Modifying cnDBTier Certificates to Establish TLS Between Georeplication Sites	78
7.4.2	Managing TLS for Communication with NFs	85
7.4.2.1	Enabling TLS for Communication with NFs	85
7.4.2.2	Modifying cnDBTier Certificates to Establish TLS for Communication with NFs	86
7.5	Adding a Georedundant cnDBTier Cluster	90

7.5.1	Adding cnDBTier Georedundant Cluster to Single Site cnDBTier Cluster	91
7.5.2	Adding cnDBTier Georedundant Cluster to Two-Site cnDBTier Clusters	100
7.5.3	Adding cnDBTier Georedundant Cluster to Three-Site cnDBTier Clusters	109
7.6	Removing a Georedundant cnDBTier Cluster	125
7.6.1	Extracting cnDBTier Cluster Script	125
7.6.2	Removing cnDBTier Cluster 4	126
7.6.3	Removing cnDBTier Cluster 3	129
7.6.4	Removing cnDBTier Cluster 2	132
7.6.5	Removing cnDBTier Cluster 1	134
7.7	Adding and Removing Replication Channel Group	137
7.7.1	Converting Single Replication Channel Group to Multiple Replication Channel Groups	138
7.7.2	Converting Multiple Replication Channel Groups to Single Replication Channel Group	147
7.8	Managing Passwords and Secrets	158
7.8.1	Modifying cnDBTier Password Encryption Key	158
7.8.2	Changing cnDBTier Passwords	161
7.8.2.1	Prerequisites	162
7.8.2.2	Installing and Using dbtpasswd Script	162
7.8.2.3	Configuration Options	163
7.8.2.4	Changing All cnDBTier Passwords in a Site	165
7.8.2.5	Changing All Passwords in a Stand-Alone or Georeplication Site	168
7.8.2.6	Changing All cnDBTier Passwords in Phases	170
7.8.2.7	Changing an NF Password	173
7.8.3	Modifying cnDBTier Backup Encryption Password	174
7.8.4	Modifying SSH Keys for Transferring Backups	176
7.8.5	Modifying Transparent Data Encryption Password	180
7.9	Modifying HTTPS Certificates	181
7.10	Modifying Remote Server Configurations for Secure Transfer of Backups	184
7.11	Checking the Georeplication Status Between Clusters	186
7.12	Changing Authentication Plugin on cnDBTier Sites	210
7.13	LCM based Automation	212
7.14	Support for Dual Stack	213
7.15	Migrating cnDBTier to Use ASM for External Communication	226

Preface

- [Documentation Accessibility](#)
- [Diversity and Inclusion](#)
- [Conventions](#)

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 customer access to and use of Oracle support services will be pursuant to the terms and conditions specified in their Oracle order for the applicable services.

Diversity and Inclusion

Oracle is fully committed to diversity and inclusion. Oracle respects and values having a diverse workforce that increases thought leadership and innovation. As part of our initiative to build a more inclusive culture that positively impacts our employees, customers, and partners, we are working to remove insensitive terms from our products and documentation. We are also mindful of the necessity to maintain compatibility with our customers' existing technologies and the need to ensure continuity of service as Oracle's offerings and industry standards evolve. Because of these technical constraints, our effort to remove insensitive terms is ongoing and will take time and external cooperation.

Conventions

The following text conventions are used in this document:

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

My Oracle Support

My Oracle Support (<https://support.oracle.com>) is your initial point of contact for all product support and training needs. A representative at Customer Access Support can assist you with My Oracle Support registration.

Call the Customer Access Support main number at 1-800-223-1711 (toll-free in the US), or call the Oracle Support hotline for your local country from the list at <http://www.oracle.com/us/support/contact/index.html>. When calling, make the selections in the sequence shown below on the Support telephone menu:

- For Technical issues such as creating a new Service Request (SR), select **1**.
- For Non-technical issues such as registration or assistance with My Oracle Support, select **2**.
- For Hardware, Networking and Solaris Operating System Support, select **3**.

You are connected to a live agent who can assist you with My Oracle Support registration and opening a support ticket.

My Oracle Support is available 24 hours a day, 7 days a week, 365 days a year.

Acronyms

The following table lists the acronyms and the terminologies used in the document.

Table Acronyms and Terminologies

Term	Definition
AES	Advanced Encryption Standard
API	Application Programming Interface
ASM	Aspen Service Mesh
cnDBTier	Oracle Communications Cloud Native Core, cnDBTier
CNE	Oracle Communications Cloud Native Core, Cloud Native Environment
CNLB	Cloud Native Load Balancer
CSAR	Cloud Service Archive
DB	Database
FQDN	Fully Qualified Domain Name
GR	Georeplication
GRR	Georeplication Recovery
IPv4	IP addresses Version 4
JVM	Java Virtual Machine
MOS	My Oracle Support
MTA	MultiThreaded Appliers
NDB	Network Database
NF	Network Function
PDB	Pod Disruption Budget
PEM	Privacy Enhanced Mail
PSP	Pod Security Policies
PV	Persistent Volume
PVC	Persistent Volume Claim
SSH	Secure Shell
SFTP	Secure File Transfer Protocol
SQL	Structured Query Language
TDE	Transparent Data Encryption
TLS	Transport Layer Security

What's New in This Guide

This section introduces the documentation updates for release 25.2.1xx.

Release 25.2.100 - G39115-03, November 2025

Metrics

Added the following built-in cnDBTier JVM heap metrics as these metrics can be accessed from cnDBTier:

- `jvm_memory_used_bytes`
- `jvm_memory_committed_bytes`
- `jvm_memory_max_bytes`

Release 25.2.100 - G39115-02, November 2025

APIs

Added the following APIs in the [cnDBTier Status APIs](#) section.

- `http://base-uri/db-tier/replication/status/realtime`
- `http://base-uri/db-tier/replication/status/realtime/sitename/{siteName}`
- `http://base-uri/db-tier/replication/status/realtime/sitename/{siteName}/remotesitename/{remoteSiteName}`
- `http://base-uri/db-tier/replication/status/realtime/sitename/{siteName}/remotesitename/{remoteSiteName}/replgroupid/{replGroupId}`

Release 25.2.100 - G39115-01, November 2025

Feature Updates

- New Features:
 - **Automatic Management of TLS Certificates**
Added the [Support for Automated Certificate Lifecycle Management](#) section that explains how to integrate cnDBTier with Oracle Communications Cloud Native Core, Certificate Management (OCCM) to support automation of certificate lifecycle management.
 - **Application Service Mesh for External Communication**
Added the [Application Service Mesh \(ASM\) for External Communication](#) section that explains how to secure external communication through ASM (Application Service Mesh) by selectively applying Istio sidecar injection only to pods where external communication is involved.

APIs

Added the following APIs in the [cnDBTier Status APIs](#)

- `http://base-uri/db-tier/replication/status/realtime`
- `http://base-uri/db-tier/replication/status/realtime/sitename/{siteName}`
- `http://base-uri/db-tier/replication/status/realtime/sitename/{siteName}/remotesitename/{remoteSiteName}`
- `http://base-uri/db-tier/replication/status/realtime/sitename/{siteName}/remotesitename/{remoteSiteName}/replgroupid/{replGroupId}`

Alerts

The "Recommended Actions" procedure for each of the alerts is elaborated for clear understanding. See [cnDBTier Alerts](#).

Metrics

The following metrics are added in the [cnDBTier Metrics](#):

- Added the following Transaction Coordinator (TC) time track metrics to monitor the average event time:
 - db_tier_tc_time_track_stats_total_index_count
 - db_tier_tc_time_track_stats_total_index_time
 - db_tier_tc_time_track_stats_total_read_count
 - db_tier_tc_time_track_stats_total_read_time
 - db_tier_tc_time_track_stats_total_scan_count
 - db_tier_tc_time_track_stats_total_scan_time
 - db_tier_tc_time_track_stats_total_scan_fragments_count
 - db_tier_tc_time_track_stats_total_scan_fragments_time
 - db_tier_tc_time_track_stats_total_transaction_count
 - db_tier_tc_time_track_stats_total_transaction_time
 - db_tier_tc_time_track_stats_total_write_count
 - db_tier_tc_time_track_stats_total_write_time
- Added the following metric to monitor the status of replication switchover:
 - DBTier Replication Switchover Status

Maintenance Procedures

- Added the [Starting or Stopping georeplication using dbtrep1mgr script](#) section that explains how to use dbtrep1mgr script to start or stop georeplication between sites.
- Added the [Migrating cnDBTier to Use ASM for External Communication](#) section that explains how to remove ASM for internal communication and retain ASM only for external communication.

1

Introduction

Oracle Communications Cloud Native Core, cnDBTier (cnDBTier) is the geodiverse database layer that provides persistence storage for state and subscriber data on a cloud environment such as Oracle Communications Cloud Native Core, Cloud Native Environment (CNE) or Oracle Cloud Infrastructure (OCI). This document provides information about the architecture, features, APIs, observability metrics and alerts, and the manual procedures for configuring cnDBTier.

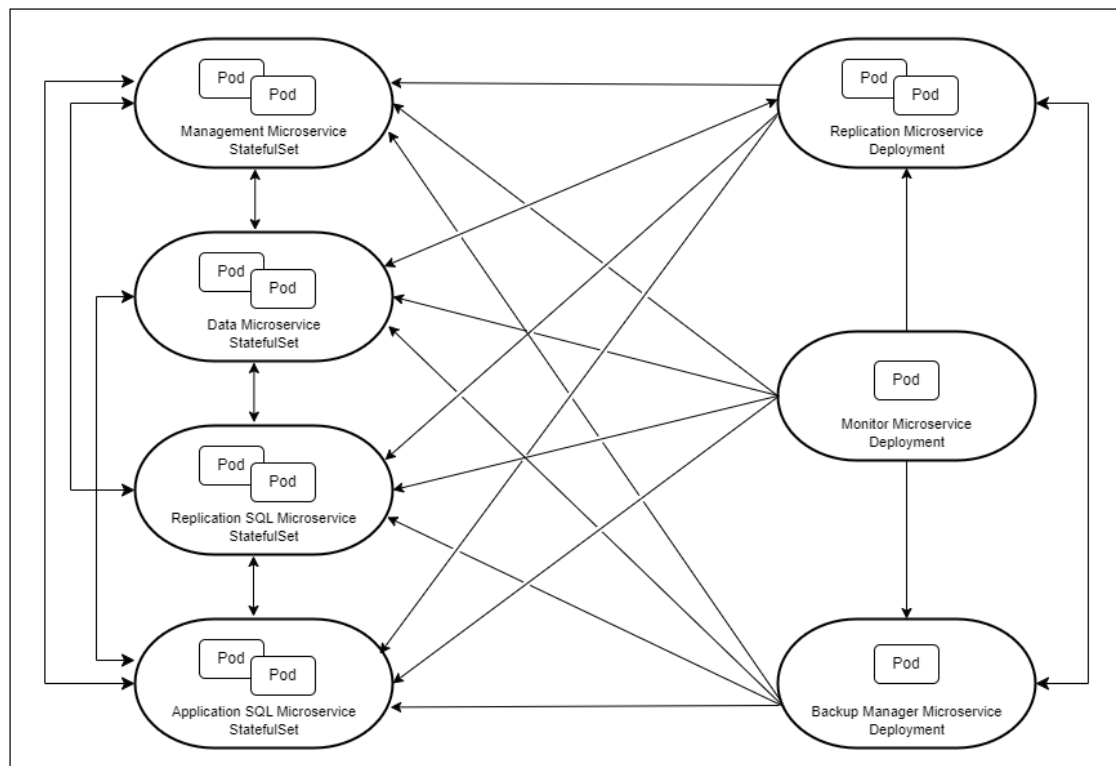
2

cnDBTier Architecture

cnDBTier comprises various microservices deployed in Kubernetes-based cloud native environments, for example: Oracle Communications Cloud Native Core, Cloud Native Environment (CNE) or Oracle Cloud Infrastructure (OCI). Cloud native environments provide various common services such as logs, metric data collection, analysis, graphs, and chart visualization. cnDBTier microservices integrate with these common services and provide them with the necessary data such as logs and metrics.

The following diagram depicts the overall architecture of the cnDBTier:

Figure 2-1 cnDBTier Architecture



The following are the microservices of cnDBTier and their usages:

1. Management Microservice:

- The Management microservice is responsible for managing other nodes within the NDB Cluster. It performs functions such as providing configuration data and running backups.
- This microservice node is started first before any other node as it manages the configurations of other nodes in the cluster.
- This microservice manages and reads the cluster configuration file and distributes the information to all nodes in the cluster that request for it.

- This microservice maintains a log of cluster activities. Management clients can connect to the management server to check the status of the cluster.
- The management server manages the cluster logs. The data nodes transfer information about any events that occur on the data nodes to the management server. The management server in turn writes these information to the cluster log.

2. Data Microservice:

- The Data microservice stores all the cluster data in key-value pairs.
- cnDBTier supports two replicas, that is a single node group can have up to two data nodes. For example, if there are four data nodes, then there will be two node groups, where the first group consists of the first two data nodes and the second group consists of the last two data nodes. All the data nodes of a single node group store the same data for redundancy and high availability. Therefore, at least one data node from each node group must be alive all the time to process or support CRUD (Create, Retrieve, Update, Delete) operations.
- The Data microservice is used to handle all the data in the tables using the NDB Cluster storage engine.
- This microservice empowers a data node to accomplish many activities, but not limited to the following:
 - Distributed transaction handling
 - Node recovery
 - Checkpointing to disk
 - Online backup

3. Replication SQL (API) Microservice:

- The Replication SQL microservice hosts a MySQL server that connects to the data pods and provides bidirectional georeplication support to the remote sites.
- This microservice receives the events from the data nodes, maintains binlogs on PVC, and uses the binlogs to perform replication.
- cnDBTier uses a set of two Replication SQL microservices to establish replication with the mate site. Where, one microservice is hosted as an active replication channel and the other hosted as a standby replication channel to handle georeplication failures.

4. Application SQL (APP) Microservice:

- The Application SQL microservice is used by the Network Functions (NF) for performing CRUD (Create, Retrieve, Update, Delete) operations on cnDBTier.
- This microservice is responsible for performing cnDBTier initializations such as database creation, table creation, and user creation.

5. Monitor Microservice:

- The Monitor microservice stores information about the current behavior of cnDBTier in the form of metrics. These metrics are sent to Prometheus or a similar monitoring tool when the tool requests for these metrics or data.
- The monitor microservice hosts multiple REST APIs. These REST APIs are used for handling many specific use cases such as creating on-demand backups. It also hosts other REST APIs that are used by CNC Console to integrate cnDBTier on CNC Console GUI, thereby making cnDBTier more user friendly. For more information about cnDBTier APIs, see the [cnDBTier APIs](#) section.

6. Backup Manager Microservice:

- This Backup Manager microservice is responsible for coordinating all kinds of backup creations—scheduled backups by CronJob, on-demand backups by end user, and Fault recovery backups.
- The Backup Manager microservice uses the Backup Executor service to run the mentioned or instructed commands on the respective data microservice. The Backup Manager microservice is also responsible for coordinating with the Backup Executor service to perform the following operations:
 - Transferring the backups to local replication service when a fault recovery is triggered.
 - Purging the old backups to save space on data microservice and prevent the data microservice from crashing.

7. Replication Microservice:

- The Replication microservice is responsible for setting up the replication between cnDBTier sites. It also manages the replication channels between cnDBTier sites.
- This microservice has the following optional features:
 - Skip replication errors when there is an irrecoverable error on the replication channels. This helps to resume the replication channels post an error.
 - Transfer backups to remote server over a secure channel using SFTP.
- This microservice handles replication fault recovery to recover the failed sites. When there is a replication failure, this microservice performs the following tasks:
 - a. Fetches data backup from a healthy site and uses the backup in a failed site to recover the data.
 - b. Reestablishes the replication channel thereby recovering the failed sites.
- This microservice checks the growth in Binlogs and purges the binlogs according to the configuration. This helps to save the space in replication SQL microservice and prevent it from crashing.

2.1 cnDBTier Setups

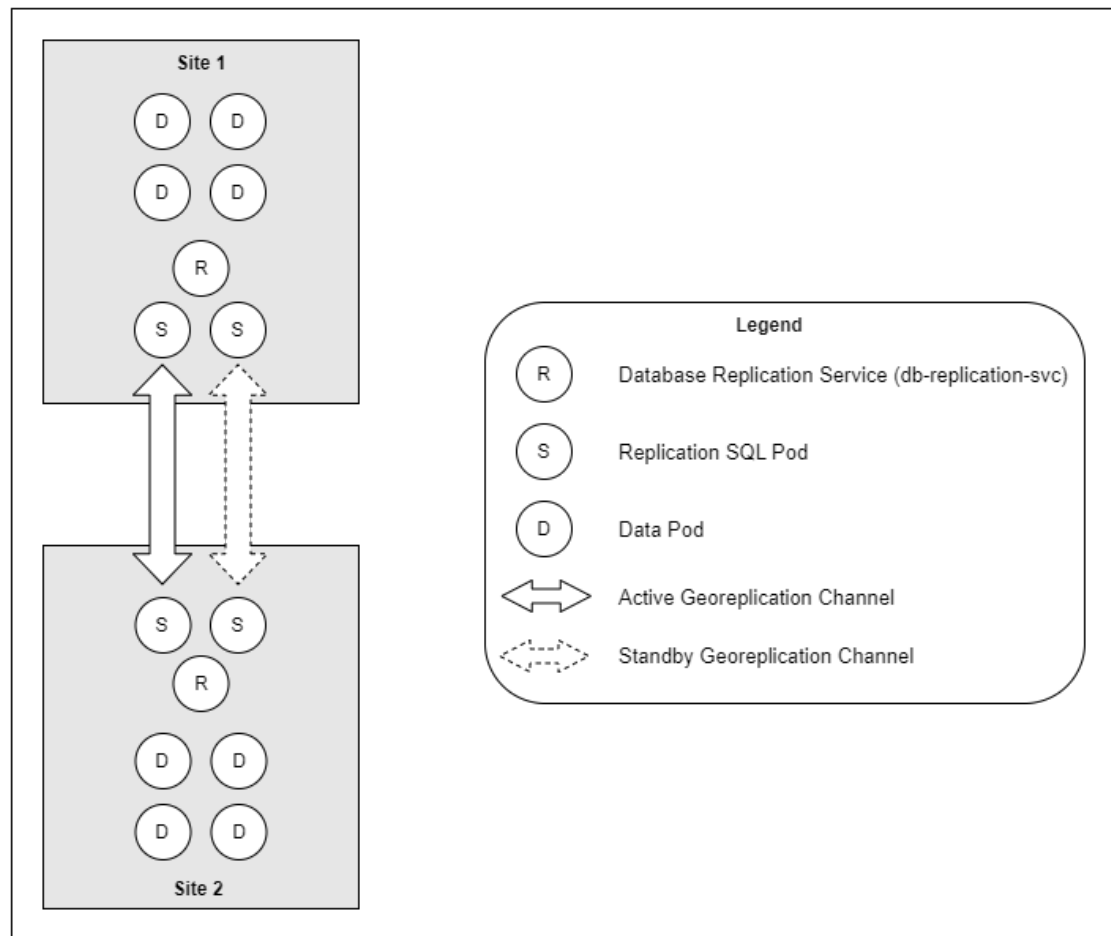
cnDBTier supports two-site, three-site, and four-site georeplication setups. This section provides information and figures to understand the replication setups between cnDBTier sites based on their topology.

Note

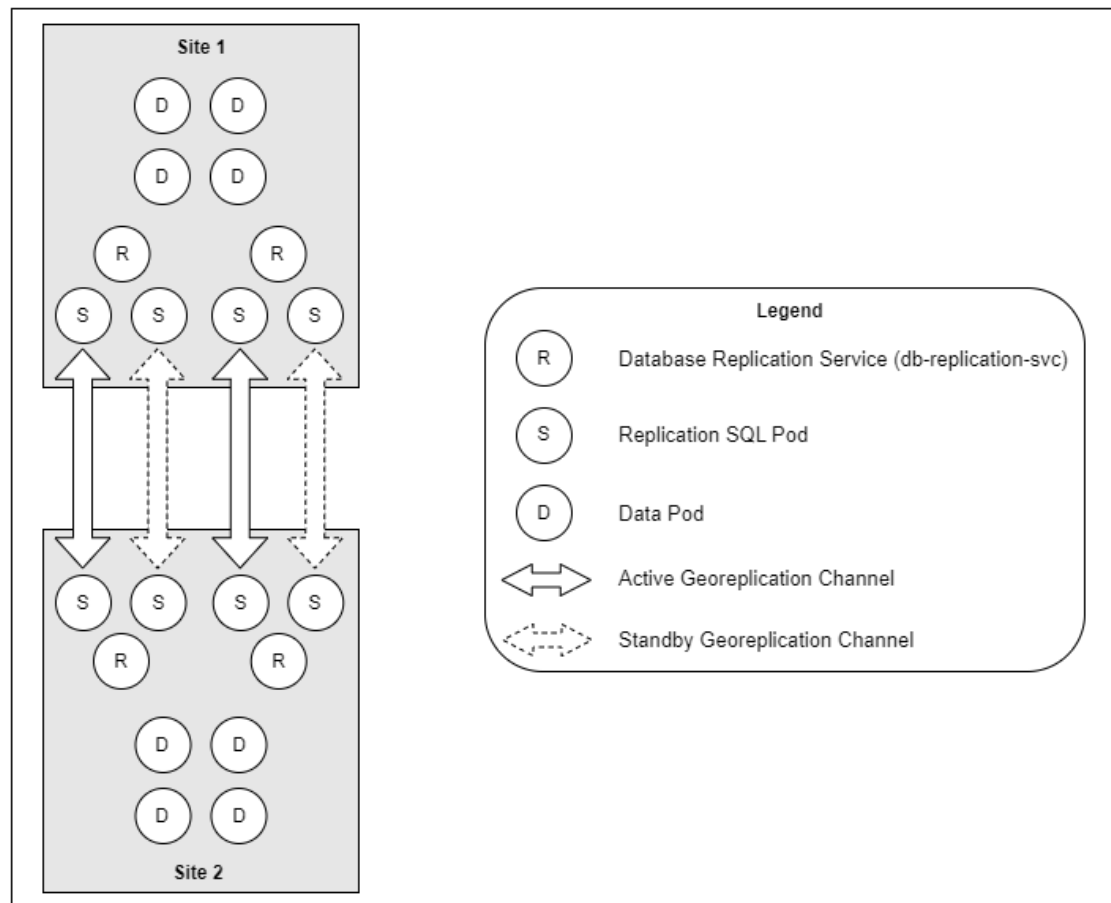
The setups detailed in this section are for reference only. You must choose your cnDBTier setups depending on your requirements and traffic intensity analysis.

Two Site Replication

The following images depict the replication SQL pod connection between two cnDBTier sites with single or multiple replication channel groups. They also depict how replication service pods communicate with each other to set up the replication.

Figure 2-2 Two-Site Setup with Single Replication Channel Group

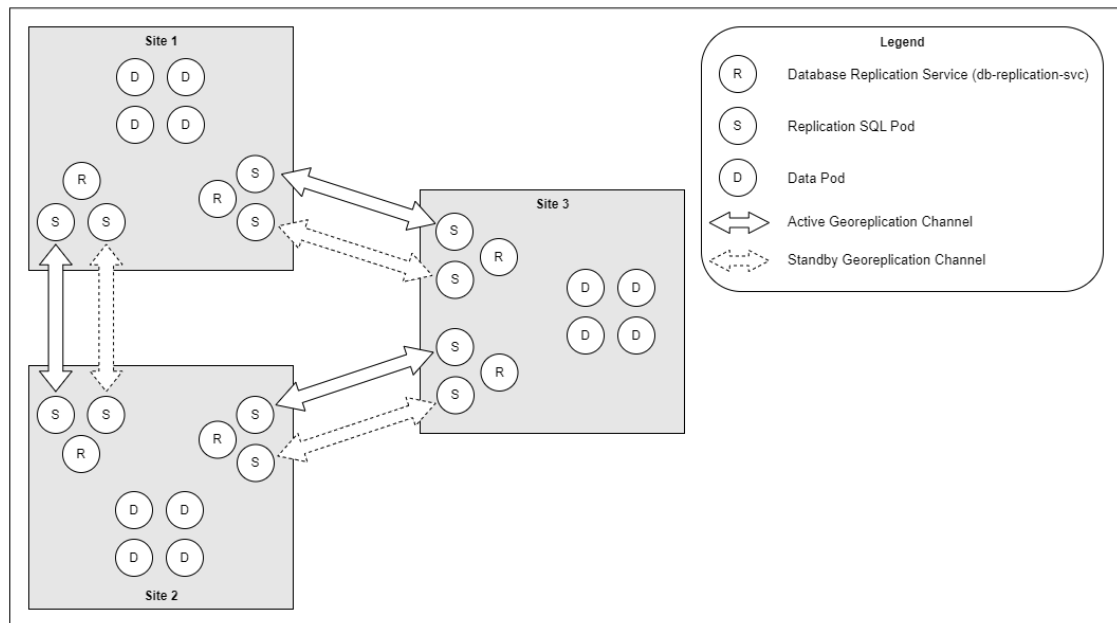
This sample setup contains two cnDBTier sites (Site 1 and Site 2). Each site has a single replication channel group consisting of a database replication service pod and two replication SQL pods. The replication channel group on both the sites communicate with each other for replication through a georeplication channel. The replication channel group consists of one active and one standby channel where the active channel is responsible for georeplication. When the active replication channel fails, the standby channel switches as the active channel for replication.

Figure 2-3 Two-Site Setup with Multiple Replication Channel Groups

This sample setup contains two cnDBTier sites (Site 1 and Site 2). Each site has two replication channel groups and each replication channel group contains a database replication service pod and two replication SQL pods. One replication channel group on one site (Site a) communicates with one replication group on the other site (Site 2) through a georeplication channel. Each replication channel group consists of one active and one standby channel, where the active channel is responsible for georeplication. When the active replication channel fails, the standby channel switches as the active channel for replication.

Three Site Replication

The following diagram depicts the replication SQL pod connection between three cnDBTier sites for replication. It also depicts how replication service pods communicate with each other to set up the replication:

Figure 2-4 Three-Site Replication

This sample setup contains three cnDBTier sites (Site 1, Site 2, and Site 3). Each site is configured with two replication channel groups. Each replication channel group contains a database replication service pod and two replication SQL pods. One replication channel group on one site (For example: Site a) communicates with one replication group on the other sites (For example: Site 2 or Site 3) through a georeplication channel. Each replication channel group consists of one active and one standby channel, where the active channel is responsible for georeplication. When the active replication channel fails, the standby channel switches as the active channel for replication.

Note

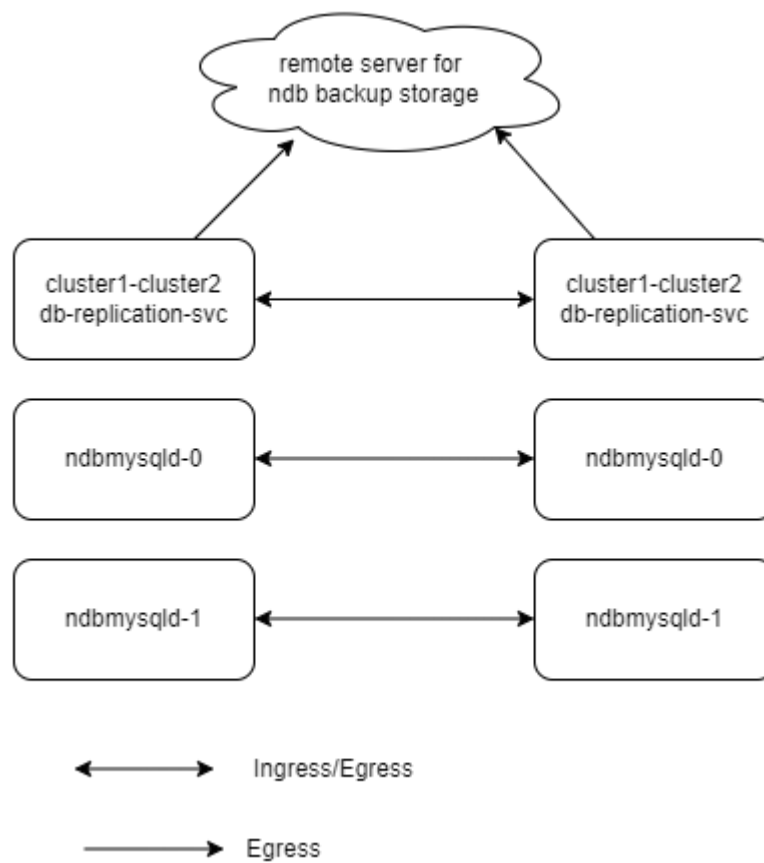
You can extend the three site setup for creating a four-site georeplication setup.

2.2 Ingress and Egress Communication Over External IPs

If cnDBTier is deployed with two or more clusters, it requires external Load Balancer IPs to facilitate communication between these clusters. The db-replication-svc pods in one cluster must be able to communicate (Ingress or Egress) with the db-replication-svc pods in all the other clusters to establish and monitor replication channels. Similarly, each cluster's active and standby ndbmysqld pods must communicate (Ingress or Egress) with the active and standby ndbmysqld pods of the other clusters to perform controller-controller replication.

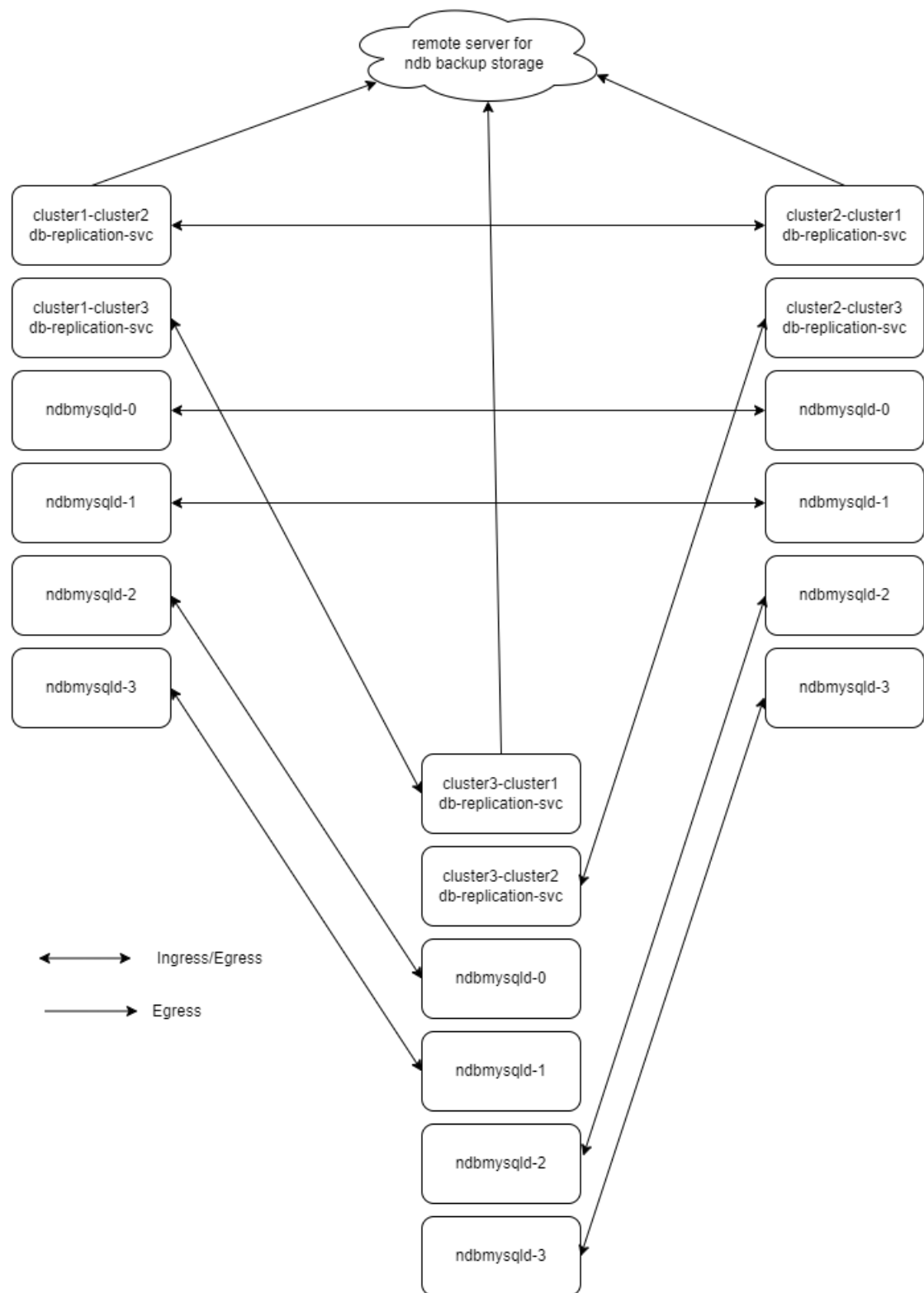
The following diagram depicts the ingress or egress routes in a two-cluster setup:

Figure 2-5 Two_Cluster_Ingress_Egress_Routes



The following diagram depicts the ingress or egress routes in a three-cluster setup:

Figure 2-6 Three_Cluster_Ingress_Egress_Routes



3

cnDBTier Features

This chapter describes the key features of cnDBTier.

3.1 Monitoring Cluster Events to Determine Data Loss in Clusters

Cluster disconnections due to network issues can lead to loss of data in the clusters. However, there were no means to identify the loss of data. With this feature, cnDBTier provides the following REST APIs to monitor cnDBTier cluster events to determine any data loss:

- http://<base-uri>/db-tier/reset/parameter/cluster_restart_disconnect
- http://<base-uri>/db-tier/reset/cluster/{cluster-name}/parameter/cluster_restart_disconnect
- <http://<base-uri>/db-tier/cluster/status>
- <http://base-uri/db-tier/cluster/status/events/{numberOfLastEvents}>
- <http://<base-uri>/db-tier/all/cluster/status/>
- <http://<base-uri>/db-tier/all/cluster/status/events/{numberOfLastEvents}>

For more information about these APIs, see [cnDBTier Cluster Events APIs](#).

cnDBTier monitors the following cluster events and records them each time they occur:

- Cluster install start
- Cluster restart
- Cluster checkpoint
- Cluster restore
- Cluster re-sync
- Cluster failure

cnDBTier uses the records of these events to determine a data loss in the cluster due to cluster disconnect and triggers an alert when it identifies a data loss. On identifying a data loss in a cluster, NFs can decide if they have to divert the traffic to other sites and then start a georeplication recovery on the identified cnDBTier cluster to restore and re-sync the database.

Metrics

cnDBTier uses the existing `db_tier_cluster_disconnect` metric to record the number of times a cluster disconnects. For more information about this metric, see [cnDBTier Node Status Metrics](#).

Alerts

cnDBTier uses the existing `MYSQL_NDB_CLUSTER_DISCONNECT` alert to inform about the data loss in the cluster. For more information about this alert, see [cnDBTier Cluster Status Alerts](#).

3.2 Storing NDB Logs in PVC

cnDBTier stores all NDB pod (ndbmcmd, ndbmtid, ndbmysqld, ndbappmysqld) logs in PVC. These logs remain persistent even when the pods restart and they can be used to debug any data node related issues. This feature is enabled in cnDBTier by default and doesn't require any configuration.

ndbmtid pod logs:

ndbmtid pod logs are stored in the `data_node_<node_id>.log` file in the `/var/occnedb/dbdata/data` directory. When the size of a log file exceeds 200 MB, the log file is archived and stored in the same directory in the following file name format: `data_node_<node_id>_<file_creation_timestamp>.log.gz`. The system archives a maximum of five such log files and deletes the older ones.

ndbmysqld or ndbappmysqld server logs:

mysqld server logs are stored in the `mysqld.log` file in the `/var/occnedb/mysql` directory. When the size of a log file exceeds 200 MB, the log file is archived and stored in the same directory in the following file name format: `mysqld.log.old.<serial_number>` (for example, `mysqld.log.old.1`). The system archives a maximum of five such log files and deletes the older ones.

ndbmysqld general logs:

If `/global/api/general_log` is set to `true`, ndbmysqld pod logs are stored in the `ndbmysqld_general_log.log` file. When the size of a log file exceeds 200 MB, the log file is archived and stored in the same directory in the following file name format: `ndbmysqld_general_log.log.old.<serial_number>` (for example, `ndbmysqld_general_log.log.old.1`). The system archives a maximum of five such log files and deletes the older ones.

ndbmcmd cluster logs:

ndbmcmd pod logs are stored in the `ndbmcmd_cluster.log` file in the `/var/occnedb/mysqlndbcluster` directory. When the size of a log file exceeds 10 M, the log file is archived and stored in the same directory in the following file name format: `cluster.log.<serial_number>` (for example, `cluster.log.1`). The system archives a maximum of ten such log files and deletes the older ones.

3.3 Enhanced Georeplication Recovery using Parallel Backup Transfer and Restore

Earlier, transferring and restoring backup during a georeplication recovery involved additional data compression at the data node and serial backup transfer to the remote sites. This process consumed more time and impacted the performance of georeplication recovery. With this feature, cnDBTier implements the following enhancements in backup transfer and restore thereby improving the performance and efficiency of georeplication recovery:

- Avoids additional backup compression.
- Supports parallel backup transfer from healthy cluster to georeplication recovery cluster.
- Restores data node backups in parallel, as and when the backups are transferred from the healthy cluster to the georeplication recovery cluster.

Testing this feature using sample data showed considerable improvement in the performance of georeplication recovery. The following table provides information about the cluster configuration used to test this feature:

Table 3-1 Cluster Configuration

Cluster Detail	Configuration
Number of data nodes in each cluster	8
CPU count of backup manager service	1
RAM size of backup manager service	1Gi
CPU count of each node in backup executor service	1
RAM size of each node in backup executor service	1Gi
CPU count of replication service	2
RAM size of replication service	2Gi

The following table provides information about the time taken to complete georeplication recovery when parallel backup transfer is enabled or disabled, considering different data sizes in each data node:

Table 3-2 Georeplication Recovery Time when Parallel Backup is Enabled or Disabled

Data Size Per Node (GB)	Parallel Backup Transfer and Restore (Enabled or Disabled)	Time Taken for Georeplication Recovery (Seconds)	Improvement in Performance (%)
5	Disabled	1255	NA
5	Enabled	1125	10.92
10	Disabled	1858	NA
10	Enabled	1565	17.12
15	Disabled	2503	NA
15	Enabled	1997	22.49
20	Disabled	3074	NA
20	Enabled	2439	23.04
25	Disabled	3723	NA
25	Enabled	2859	26.25

Note

The performance measures provided in the table are as per the sample data used for testing the feature on a sample cnDBTier environment, and are not the promised values. The actual performance varies from system to system depending on various factors including cluster configuration and data load.

Managing Parallel Backup Transfer and Restore

The following sections provide details about managing parallel backup transfer and restore:

Enabling or Disabling Parallel Backup Transfer and Restore

You can enable or disable parallel backup transfer and restore for georeplication recovery by setting the `/global/parallelbackuptransferandrestore/enable` parameter to *true* or *false* respectively in the `custom_values.yaml` file. For more information about enabling and configuring this feature, see the "customizing cnDBTier" section in *Oracle Communications Cloud Native Core, Cloud Native Environment Installation, Upgrade, and Fault Recovery Guide*.

Note

If you want to enable or disable parallel backup transfer and restore after installing cnDBTier, update the parameter in the `custom_values.yaml` file and perform an upgrade with the updated `custom_values.yaml` file. For the procedure to upgrade cnDBTier, see *Oracle Communications Cloud Native Core, Cloud Native Environment Installation, Upgrade, and Fault Recovery Guide*.

Metrics

The following metrics are added to observe the progress of parallel backup and restore during georeplication recovery:

- `db_tier_local_backup_transfer_progress`
- `db_tier_remote_backup_transfer_progress`

For more information about these metrics, see [cnDBTier Parallel Backup Transfer Progress Metrics](#).

3.4 Transparent Data Encryption (TDE)

Transparent Data Encryption (TDE) encrypts data at the storage layer, that is the files stored in the disk or PVC of the data nodes. TDE encrypts and decrypts data dynamically as it is written to or read from the storage, without requiring any modifications to the application's code. This guarantees that the sensitive data stored in the database files on disk remains encrypted while at rest, offering a crucial security layer against unauthorized access, particularly in situations where physical security controls fail.

Enabling TDE slightly increases the restart time of the data nodes as TDE handles encryption and decryption dynamically. The following table provides information about the time taken for the data nodes to restart when TDE is enabled or disabled, considering different data sizes in each data node:

Table 3-3 Data Node Restart Time When TDE is Enabled or Disabled

Data Size Per Node (in GB)	TDE Enabled or Disabled	Average Restart Time (in seconds)	Increase in Restart Time when TDE is Enabled (in percentage)
10	Disabled	166.5	NA
10	Enabled	181.5	9%
20.7	Disabled	216	NA
20.7	Enabled	242.5	12.27%
30	Disabled	329	NA
30	Enabled	349.75	6.32%
37	Disabled	376	NA

Table 3-3 (Cont.) Data Node Restart Time When TDE is Enabled or Disabled

Data Size Per Node (in GB)	TDE Enabled or Disabled	Average Restart Time (in seconds)	Increase in Restart Time when TDE is Enabled (in percentage)
37	Enabled	397.5	5.72%

Note

Analysis and testing of this feature with sample data reveal that enabling TDE increases the restart time of data nodes by an average of 8.33%. However, the performance vary depending on your cnDBTier setup and data size. Ensure that you test and understand the performance impact while using TDE for data encryption.

Managing Transparent Data Encryption (TDE)

The following sections provide details about managing Transparent Data Encryption (TDE):

Enabling Transparent Data Encryption (TDE)

You can enable or disable the TDE feature using the `/global/ndb/EncryptedFileSystem` parameter in the `custom_values.yaml` file. For more information about enabling and disabling this feature, see the "Customizing cnDBTier" section in *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide*.

Configuring Transparent Data Encryption (TDE)

Before enabling TDE, you must create the necessary secrets that are used to encrypt the data in data nodes. For information about creating secrets for TDE, see the "Creating Secret" section in *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide*.

3.5 Support for CNE Cloud Native Load Balancer (CNLB)

cnDBTier supports network segregation using CNE Cloud Native Load Balancer (CNLB) to effectively manage ingress and egress traffic flows in the system. CNLB is an alternate solution to the existing LBVM solution. To leverage this feature, you must perform the following configurations:

1. Configure the ingress and egress destination subnet addresses in the `cnlb.ini` file before installing CNE. For more information about ingress and egress destination subnet addresses, see the [Ingress and Egress Communication Over External IPs](#) section.
2. Configure the required traffic segregation details in the `custom_values.yaml` file before installing cnDBTier.

Managing Cloud Native Load Balancer (CNLB)

The following sections provide details on managing the CNLB feature:

Enabling or Disabling CNLB

As CNLB is powered by CNE, you can enable or disable CNLB while installing CNE. For more information about enabling and configuring CNLB, see *Oracle Communications Cloud Native Core, Cloud Native Environment Installation, Upgrade, and Fault Recovery Guide*.

Configuring CNLB for cnDBTier Traffic Segregation

Depending on your requirement, you must perform the following configurations to use CNLB network segregation:

- To use network segregation for active and standby replication channels, configure the required CNLB annotations for ndbmysql pods in the `custom_values.yaml` file.
- To allow communication between local and remote site replication pods over a separate network, configure CNLB ingress or egress annotations for the db-replication-svc pods in the `custom_values.yaml` file.

For more information about configuring CNLB for traffic segregation, see the "Configuring Cloud Native Load Balancer" section in *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide*.

3.6 Multithreaded Applier (MTA)

The Multithreaded Applier (MTA) feature provides the functionality allows independent binlog transactions to be run in parallel at a replica thereby increasing the peak replication throughput. NDB replication is modified to support the use of generic MySQL server MTA mechanism.

Managing Multithreaded Applier (MTA)

The following sections provide details on managing the MTA feature:

Enabling or Disabling MTA

You can enable or disable the MTA feature using the `replica_parallel_workers` parameter in the `custom_values.yaml` file. This feature is disabled when `replica_parallel_workers` is set to 1 and enabled when the parameter is set to a value greater than 1. This feature is disabled by default. For more information on this parameter, see the "Customizing cnDBTier" section in *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide*.

You can refer to the following sample configurations to enable or disable MTA in the `custom_values.yaml` file:

Sample configuration to enable MTA:

```
global:
  additionalndbconfigurations:
    mysql:
      replica_parallel_workers: 4
      binlog_transaction_dependency_tracking: "WRITESET"
      ndb_log_transaction_dependency: 'ON'
```

Sample configuration to disable MTA:

```
global:
  additionalndbconfigurations:
    mysql:
      replica_parallel_workers: 1
      binlog_transaction_dependency_tracking: "COMMIT_ORDER"
      ndb_log_transaction_dependency: 'OFF'
```

For more information on the MTA feature, see [MySQL documentation](#).

Configuring MTA

Additional configurations are not applicable to this feature.

3.7 Support for TLS

Transport Layer Security (TLS) is a cryptographic protocol designed to safeguard the communication between a client and a server. This protocol ensures that private and sensitive information such as passwords are transferred over encrypted communication channels. cnDBTier supports both TLS 1.2 and TLS 1.3.

cnDBTier uses TLS to secure data transfer in the following scenarios:

- **Georeplication between cnDBTier Sites:**

Earlier, cnDBTier provided support to configure the replication SQL pods manually to establish Transport Layer Security (TLS) for georeplication between cnDBTier sites. With this feature, cnDBTier automates the process of configuring TLS for georeplication between cnDBTier sites.

When TLS feature is enabled for replication, cnDBTier performs or supports the following operations during replication:

- The replication SQL pod uses the certificates provided or configured to establish an encrypted connection for georeplication. This encrypted connection remains throughout the life cycle of the replication. When the replication breaks and a switchover occurs, the standby replication SQL pod establishes a new replication channel using the given certificates and provides the encrypted connection.
- cnDBTier reestablishes the TLS connection after a georeplication recovery. That is, when a georeplication recovery completes successfully, the system reestablishes the encrypted connection between the replication channels. This ensures that the TLS is kept intact even after a georeplication recovery.
- cnDBTier supports maintaining separate certificates for each replication channel groups and each site replicating to other site.
- cnDBTier supports maintaining list of chipers used for replication.

- **Communication between cnDBTier and Network Functions:**

cnDBTier supports TLS for application SQL pods to establish secure connection for communication with Network Functions (NFs).

When TLS feature is enabled, cnDBTier performs or supports the following operations during communication with NFs:

- The application SQL pod uses the certificates provided or configured to establish an encrypted connection for communication with NFs. This encrypted connection remains throughout the life cycle of the connection between NF and cnDBTier.
- cnDBTier reestablishes the TLS connection between NFs and cnDBTier after a georeplication recovery. That is, when a georeplication recovery completes successfully, the system reestablishes the encrypted connection between the NFs and cnDBTier. This ensures that the TLS is kept intact even after a georeplication recovery.

Managing TLS

The following sections provide details about managing TLS:

Enabling TLS

You can enable or disable TLS for georeplication, NF communication, or both using the `custom_values.yaml` file as follows:

- To enable TLS for secure georeplication, use the `/global/tls/enable` parameter in the `custom_values.yaml` file. You must also configure the other TLS parameters to provide the

necessary certificates for the `ndbmysqld` pods in the `custom_values.yaml` file. For the procedure to enable TLS for georeplication, see [Enabling TLS for Georeplication](#). For more information about the TLS parameters, see the "Customizing `cnDBTier`" section in *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide*.

- To enable TLS for a secure connection between `cnDBTier` and NFs, use the `/global/ndbappTLS/enable` parameter in the `custom_values.yaml` file. You must also configure the other TLS parameters to provide the necessary certificates for the `ndbappmysqld` pods in the `custom_values.yaml` file. For the procedure to enable TLS for communication between `cnDBTier` and NFs, see [Enabling TLS for Communication with NFs](#). For more information about the TLS parameters, see the "Customizing `cnDBTier`" section in *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide*.

Modifying `cnDBTier` Certificates for TLS

You can modify the `cnDBTier` certificates that the system uses to establish encrypted connections for georeplication and communication with NFs. For procedures, see [Modifying `cnDBTier` Certificates to Establish TLS Between Georeplication Sites](#) and [Modifying `cnDBTier` Certificates to Establish TLS for Communication with NFs](#).

3.8 Network Policies

Network Policies is an application-centric construct that allows `cnDBTier` pods to control or restrict the incoming and outgoing network traffic. This ensures security and isolation of services within a Kubernetes cluster.

Network Policies creates pod-level rules to specify how pods (Containers) in a Kubernetes cluster communicate with each other and with external resources. `cnDBTier` uses Network Policies on the pods to enhance the security and control both incoming and outgoing traffic effectively. For more information about Network Policies, see [Kubernetes Network Policies](#) documentation.

Enabling and Configuring Network Policies

You can enable or disable network policies for different pods using the network policy parameters in the `custom_values.yaml` file. For more information about enabling or disabling this feature for `cnDBTier` pods, see the "Customizing `cnDBTier`" section in *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide*.

3.9 REST APIs for CNC Console

`cnDBTier` supports CNC Console to integrate `cnDBTier` functionalists on CNC Console GUI. To enable this feature, `cnDBTier` exposes REST APIs. CNC Console uses these REST APIs to integrate `cnDBTier` data on CNC console GUI.

For detailed information about the REST APIs, see [cnDBTier APIs for CNC Console](#).

Network Functions (NFs) can integrate `cnDBTier` on CNC Console to perform the following (but not limited to) operations:

- Checking the status of `cnDBTier` cluster or georeplication.
- Checking the list of completed backups.
- Checking `cnDBTier` version.
- Checking the health status of `cnDBTier` services such as replication service, monitor service, data service, and backup manager service.

- Initiating on-demand backup.
- Performing georeplication recovery.

To integrate and customize cnDBTier on CNC Console, NFs must configure the menuCncc.json file as shown in the following example:

```
{
  "menu": {
    "routeConfig": {
      "home": {
        "label": "Home",
        "value": "home",
        "isDefault": true
      },
      "commonServices": {
        "label": "Common Services",
        "value": "commonServices",
        "isDefault": true
      },
      "services/{nf}/{service}": {
        "label": "Services",
        "value": "services"
      },
      "configurations/{nf}/{configType}": {
        "label": "Configurations",
        "value": "configurations"
      }
    },
    "menuItems": [
      {
        "attr": {
          "id": "cnDBTier",
          "name": "cnDBTier",
          "sequence": 10
        },
        "children": [
          {
            "attr": {
              "id": "services/<NF Name>/dbconfig-backupList",
              "name": "Backup List",
              "sequence": 10
            }
          },
          {
            "attr": {
              "id": "cnDBTierHealth",
              "name": "cnDBTier Health",
              "sequence": 20
            },
            "children": [
              {
                "attr": {
                  "id": "services/<NF Name>/dbconfig-backupHealthStatus",
                  "name": "Backup Manager Health Status",
                  "sequence": 100
                }
              }
            ]
          }
        ]
      }
    ]
  }
}
```

```

    },
    {
      "attr": {
        "id": "services/<NF Name>/dbconfig-monitorHealthStatus",
        "name": "Monitor Health Status",
        "sequence": 110
      }
    },
    {
      "attr": {
        "id": "services/<NF Name>/dbconfig-ndbHealthStatus",
        "name": "NDB Health Status",
        "sequence": 120
      }
    },
    {
      "attr": {
        "id": "services/<NF Name>/dbconfig-replicationHealthStatus",
        "name": "Replication Health Status",
        "sequence": 130
      }
    }
  ]
},
{
  "attr": {
    "id": "services/<NF Name>/dbconfig-version",
    "name": "cnDBTier Version",
    "sequence": 30
  }
},
{
  "attr": {
    "id": "services/<NF Name>/dbconfig-databaseStatisticsReport",
    "name": "Database Statistics Report",
    "sequence": 40
  }
},
{
  "attr": {
    "id": "GeoreplicationRecovery",
    "name": "Georeplication Recovery",
    "sequence": 50
  },
  "children": [
    {
      "attr": {
        "id": "services/<NF Name>/dbconfig-updateClusterAsFailed",
        "name": "Update Cluster As Failed",
        "sequence": 100
      }
    },
    {
      "attr": {
        "id": "services/<NF Name>/dbconfig-
startGeoreplicationRecovery",

```

```

        "name": "Start Georeplication Recovery",
        "sequence": 110
    },
    {
        "attr": {
            "id": "services/<NF Name>/dbconfig-georeplicationRecoveryStatus",
            "name": "Georeplication Recovery Status",
            "sequence": 120
        }
    }
],
{
    "attr": {
        "id": "configurations/<NF Name>/dbconfig-geoReplicationStatus",
        "name": "Georeplication Status",
        "sequence": 60
    }
},
{
    "attr": {
        "id": "services/<NF Name>/dbconfig-clusterStatus",
        "name": "Local Cluster Status",
        "sequence": 70
    }
},
{
    "attr": {
        "id": "services/<NF Name>/dbconfig-onDemandBackup",
        "name": "On Demand Backup",
        "sequence": 80
    }
},
{
    "attr": {
        "id": "services/<NF Name>/dbconfig-heartBeatStatus",
        "name": "Replication HeartBeat Status",
        "sequence": 90
    }
}
]
}
}
}

```

References:

- For procedures to perform georeplication recovery using CNC Console, see the "Restoring Georeplication (GR) Failure" section in *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide*.
- For information about CNC Console GUI, see *Oracle Communications Cloud Native Configuration Console User Guide*.

- For more information about integrating cnDBTier APIs on CNC Console for a specific NF, see the respective NF documents.

3.10 PVC Health Monitoring

The platform-level issues and hardware failures impact the PVC health that in turn impacts the cnDBTier health. Before implementing this feature, to identify any PVC issues, you must wait for the MySQL Cluster processes to log an error. With this feature, cnDBTier provides options to monitor the health of PVCs that are mounted on cnDBTier pods.

Managing PVC Health Monitoring

The following sections provide details about managing PVC Health Monitoring:

Enabling and Configuring PVC Health Monitoring

You can enable or disable PVC Health Monitoring using the PVC health global parameters available in the `custom_values.yaml` file. For more information about enabling or disabling this feature, see the "Customizing cnDBTier" section in *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide*.

Monitoring PVC Health Through Metrics and Alerts

PVC Health Monitoring facilitates each pod to monitor its PVC and pass the PVC health condition to the monitor service. The monitor service combines this data and produces metrics for each PVC's health. For information about PVC Health Monitoring metrics, see [cnDBTier Metrics](#).

3.11 cnDBTier Automated Backup

The cnDBTier backup service creates and safely stores database backups periodically so that the user can always have a relatively recent copy of the data, to recover from cnDBTier fault scenarios. cnDBTier backup service works on each MySQL cluster data node. Each data node creates a backup of the hosted data and maintains a copy of the backup.

The automatic backup runs when all the database nodes are running. The retention period and frequency of the backup are configurable. The status of the database backups taken is stored in the database nodes and is available to database monitor service to include in metrics. The following table describes these parameters and their default values.

Table 3-4 DB Backup Service

Parameter	Units	Default	Description
backup_period	integer	1	The number of days between each backup.
num_backups_to_retain	integer	3	The number of backups to retain. Recommended number of backups is at least 3. This ensures at least one backup remains, in case the automated backup system force purges additional backups, due to backup size growth.

Monitoring Automated Backups

The status of the database backups is stored in the database nodes and is accessible to the database Monitor service. The system maintains metrics and alerts to monitor and track automatic backups. Additionally, it triggers an alert when the current NDB cluster performs a backup. For information about automatic backup metrics, see [cnDBTier Automated Backup](#)

[Metrics](#). For information about automatic backup alerts, see [cnDBTier Automated Backup Alerts](#).

REST APIs

cnDBTier provides the `http://<base-uri>/db-tier/status/cluster/local/backup` REST API to fetch the details of the data node backups that are in progress in the NDB cluster. For more information about this REST API, see [cnDBTier Backup APIs](#).

Note

MySQL doesn't allow schema change in a cluster when a backup is in progress. Therefore, ensure that you don't make any schema changes in a cluster when a data node backup is in progress. You can use the `http://<base-uri>/db-tier/status/cluster/local/backup` API to:

- check if the current cluster is performing a database back up.
- retrieve the timestamp of the next scheduled routine backup.
- determine the time window for NF upgrades.

3.12 cnDBTier Password Encryption

cnDBTier provides the password encryption feature to encrypt for replication username and password stored in the database to ensure that the passwords are secure and are not exposed. When the password encryption feature is enabled, the replication username and password are encrypted throughout the life cycle of cnDBTier unless the feature is disabled.

Managing cnDBTier Password Encryption

The following sections provide details about managing cnDBTier password encryption:

Enabling cnDBTier Password Encryption

You can enable or disable the cnDBTier password encryption feature using the `/global/encryption/enable` parameter in the `custom_values.yaml` file. For more information about enabling and disabling this feature, see the "Customizing cnDBTier" section in *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide*.

Configuring cnDBTier Password Encryption

Before enabling cnDBTier password encryption, you must create the necessary secrets that are used to encrypt the passwords. For information about creating secrets for password encryption, see the "Creating Secret" section in *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide*.

You can modify the encryption key used to encrypt the replication username and password. For the procedure to modify the password encryption key, see the [Modifying cnDBTier Password Encryption Key](#) section.

3.13 Database Backup Encryption and Secure Transfer

cnDBTier supports creation of database backups at configured intervals which are used to restore database during a georeplication recovery. These backups contain sensitive subscriber data that are transferred to remote servers and sites for performing fault recovery. Therefore,

it's important to encrypt the backups to protect the sensitive data and avoid any misuse. cnDBTier uses the encrypt and decrypt options provided by MySQL NDB cluster to encrypt the on-demand or periodic backups at the cluster level.

cnDBTier provides the functionality to securely transfer the backups stored in the data nodes to remote sites using Secure File Transfer Protocol (SFTP). This ensures that there is no loss of backups due to purging in cnDBTier as the backups are stored in remote servers. The backups in the remote server are saved under the `<remote server path>/<cnDBTier site name> path` where,

- `<remote server path>`, is the path of the remote server configured in the `/global/remotetransfer/remoteserverpath` parameter in the `custom_values.yaml` file.
- `<cnDBTier site name>`, is the name of the cnDBTier site.

The backups in the remote server are saved in the following formats:

- `backup_<backup_id>_Encrypted.zip`, if backup encryption is enabled.
- `backup_<backup_id>_Unencrypted.zip`, if backup encryption is disabled.

Managing Database Backup Encryption and Secure Transfer

The following sections provide details about managing database backup encryption and secure backup transfer:

Enabling Database Backup Encryption

Each cnDBTier cluster has an option to enable or disable the backup encryption independently. You can enable or disable the database encryption using the `/global/backupencryption/enable` parameter in the `custom_values.yaml` file. For more information about enabling and disabling this feature, see the "Customizing cnDBTier" section in *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide*.

When this feature is enabled, the backup data record and log files written by each data node are encrypted using the password present in the "occne-backup-encryption-secret" secret and a Salt that is randomly generated using a key derivation function (KDF) that employs the PBKDF2-SHA256 algorithm to generate a symmetric encryption key for that file.

Configuring Database Backup Encryption

You must provide the required credentials to encrypt the backups when the backup is initiated (on-demand backups, periodic backups, or backups initiated during restore georeplication). cnDBTier uses secret keys to encrypt database backups. You can configure these keys using Kubernetes secret during the installation. When the `START BACKUP` command is initiated, the DB backup manager service reads these keys and initiates the backup using the `encrypt password(ENCRYPT PASSWORD=password)`. For information about creating secrets for encrypting backups, see the "Creating Secret" section in *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide*.

cnDBTier provides an option to modify the backup encryption password when required. For information about modifying cnDBTier backup encryption password, see [Modifying cnDBTier Backup Encryption Password](#).

Note

The password used for encryption must follow the cnDBTier password requirement or standard.

Enabling Secure Backup Transfer

You can enable or disable secure backup transfer using the `/global/remotetransfer/enable` parameter in the `custom_values.yaml` file. For more information about enabling and disabling this secure backup transfer, see the "Customizing cnDBTier" section in *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide*.

Configuring Secure transfer of backups to Remote Server

cnDBTier requires the remote server configurations to securely transfer backups. You can use the `custom_values.yaml` file to configure the details about remote servers such as remote server IP, port, and path. These configurations can be done during an installation or upgrade. For more information about configuring the remote server details, see the "Customizing cnDBTier" section in *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide*.

cnDBTier uses SSH key and username to securely transfer backups to remote servers. You can configure these keys and username using Kubernetes secret during the installation or upgrade. For information about creating secrets for secure backup transfer, see the "Creating Secret" section in *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide*.

cnDBTier also provides an option to modify the remote server configurations, such as IP, port, path, username, and SSH key in cnDBTier. For more information about modifying remote server configurations in cnDBTier, see [Modifying Remote Server Configurations for Secure Transfer of Backups](#).

Note

Currently, cnDBTier doesn't support using the remote server backups for performing any restore or recovery procedures.

3.14 Node Selector and Node Affinity

The Node Selector and Node Affinity feature allows the Kubernetes scheduler to determine the type of nodes in which the cnDBTier pods are to be scheduled, depending on the predefined node labels or constraints. cnDBTier uses `nodeSelector` and `nodeAffinity` options to schedule cnDBTier pods to specific worker nodes.

`nodeSelector` is the basic form of cluster node selection constraint. It allows you to define the node labels (constraints) in the form of key-value pairs. When the `nodeSelector` feature is used, Kubernetes schedules the pods to only the nodes that match each of the node labels you specify.

`nodeAffinity` allows you to define advanced constraints to limit the pod scheduling on specific nodes. There are two types of node affinity:

- `requiredDuringSchedulingIgnoredDuringExecution` (Hard type): In this type, the scheduler doesn't schedule the pods unless the defined rules are met.
- `preferredDuringSchedulingIgnoredDuringExecution` (Soft type): In this type, the scheduler tries to find a node that meets the defined rules. However, even if a matching node is not available, the scheduler still schedules the pod.

Managing Node Selector and Node Affinity Feature

The following sections provide details on managing the Node Selector and Node Affinity feature:

Enabling Node Selector and Node Affinity

You can enable or disable the `nodeSelector` and `nodeAffinity` options for the pods using the `custom_values.yaml` file. For more information on enabling and disabling parameters, see the "Customizing cnDBTier" section in *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide*.

Note

- You can enable either `nodeSelector` or `nodeAffinity` option to schedule cnDBTier pods to specific worker nodes.
- By default, both `nodeSelector` and `nodeAffinity` options are disabled.

Configuring Node Selector and Node Affinity

Depending on the `nodeSelector` or `nodeAffinity` option enabled, you can configure its parameters in the `custom_values.yaml` file. For more information on the node selector and node affinity configuration parameters, see the "Customizing cnDBTier" section in *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide*.

Note

Configure the node selector or node affinity parameters only when either of the options is enabled.

3.15 cnDBTier Scaling

cnDBTier provides the ability to expand or contract the capacity of your resources to support the changing usage of your application. This includes scaling out and in, or scaling up and down the resources. cnDBTier supports two types of scaling:

- **Horizontal Scaling:** Horizontal scaling refers to adding additional nodes to the pool of resources to share the load. cnDBTier supports horizontal scaling for the following pods:
 - `ndbappmysqld`: cnDBTier supports both manual and auto-scaling of `ndbappmysqld` pods. When horizontal auto-scaling is enabled, `ndbappmysqld` scales automatically depending on the CPU and RAM usage. In this case, the higher and lower limits for scaling are defined in the `custom_values.yaml` file. When auto-scaling is disabled, cnDBTier provides an option to manually scale the `ndbappmysqld` pods as per your requirement.
 - `ndbmttd`: cnDBTier supports only manual scaling up of `ndbmttd` data pods, and doesn't support scaling down.
- **Vertical Scaling:** Vertical scaling refers to increasing the processing power (CPU, memory, and PVC) of a cluster. cnDBTier supports manual vertical scaling for the following pods:
 - `ndbmttd`
 - `ndbappmysqld`
 - `ndbmysqld`
 - `db-replication-svc`
 - `ndbmcmd`

Additionally, cnDBTier provides the `dbtscale_vertical_pvc` script to automatically update the PVC values of the pods.

For horizontal and vertical scaling procedures, see [Scaling cnDBTier Pods](#).

Managing cnDBTier Scaling

The following sections provide details on managing the cnDBTier scaling feature:

Enabling Horizontal Pod Auto-Scaler

You can enable or disable horizontal auto-scaling for `ndbappmysql` pods using the `/global/autoscaling/ndbapp/enabled` parameter in the `custom_values.yaml` file. You can enable or disable horizontal auto-scaling on the basis of memory consumption, CPU consumption, or both using the following parameters:

- `/api/ndbapp/horizontalPodAutoscaler/memory/enabled`
- `/api/ndbapp/horizontalPodAutoscaler/cpu/enabled`

For more information about these parameters, see the "Customizing cnDBTier" section in *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide*.

Configuring Horizontal Pod Auto-Scaler

You can use the following parameters to configure the average memory utilization and CPU utilization using the following parameters which triggers the auto-scaling when the average memory or CPU utilization reaches the specified percentage:

- `/api/ndbapp/horizontalPodAutoscaler/memory/averageUtilization`
- `/api/ndbapp/horizontalPodAutoscaler/cpu/averageUtilization`

For more information about these parameters, see the "Customizing cnDBTier" section in *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide*.

3.16 Application Service Mesh (ASM) for External Communication

The cnDBTier now supports secure external communication through ASM (Application Service Mesh) by selectively applying Istio sidecar injection only to pods where external communication is involved. By selectively enforcing ASM policies, cnDBTier ensures the security and compliance of its external interfaces, while preserving high performance and simplifying manageability within the cnDBTier ecosystem.

To enable ASM for both internal and external communication, the `global.istioSidecarInject.mode` parameter must be configured in the `custom_values.yaml` file. For more information on how to configure the parameter, see the "Global Parameters" section in *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide*.

Following is a sample configuration:

```
global:
  istioSidecarInject:
    mode: <none|external|all>
```

Where:

- none - Do not inject sidecar
- external - Inject only for external communication
- all - Inject for all communication (both internal and external)

3.17 Support for Automated Certificate Lifecycle Management

In cnDBTier 25.1.2xx and earlier, the Transport Layer Security (TLS) certificates were managed manually. When multiple instances of cnDBTier were deployed in a 5G network, certificate management, such as certificate creation, renewal, removal, and so on, became tedious and error-prone.

Starting with cnDBTier 25.2.1xx, you can integrate cnDBTier with Oracle Communications Cloud Native Core, Certificate Management (OCCM) to support automation of certificate lifecycle management. OCCM manages TLS certificates stored in Kubernetes secrets by integrating with Certificate Authority (CA) using the Certificate Management Protocol Version 2 (CMPv2) protocol in the Kubernetes secret. OCCM obtains and signs TLS certificates within the cnDBTier namespace. For more information about OCCM, see *Oracle Communications Cloud Native Core, Certificate Management User Guide*.

This feature enables automated TLS certificate lifecycle management for HTTPS, MySQL replication SQL pods and MySQL application SQL pods within the cnDBTier environment, ensuring secure and uninterrupted communication during certificate updates.

Certificate Monitoring for HTTPS and TLS Certificates

A certificate monitoring service is responsible for ensuring that TLS/HTTPS certificates are continuously observed for changes and updated in a controlled, zero-downtime manner.

- **Continuous Monitoring:** The monitoring service continuously watches TLS/HTTPS certificates for any updates.
- **Change detection:** The monitoring services detects the change whenever a TLS/HTTPS certificate or secret is updated, and appropriate reload actions are initiated based on the pod type and connection behavior.
- **No Service restart:** The monitoring service does not force an immediate restart or a full service disruption is performed, in case of a change in the certificate alone.

Replication SQL Pods for Monitoring the TLS Certificate

The Replication SQL pods are responsible for MySQL replication between the cnDBTier sites. TLS is used for securing MySQL replication traffic.

When a Certificate change is detected, the following is the workflow:

1. **Monitoring for Changes:** A monitoring service detects for any changes in the certificates. Upon certificate update, the change is detected and validated.
2. **Reloading Certificate on Detection:** The updated certificates are reloaded into the replication SQL pod. However, no restart or switchover is triggered immediately.
3. **Deferred Usage of New Certificates:** The new certificates are loaded, but not actively used in ongoing replication sessions. The active replication sessions continue using the old TLS session until a natural replication session switchover event occurs, for example new TLS handshake or replication session restart, to begin using the new certificate. This avoids disruption to ongoing replication sessions.

APP SQL Pods for Monitoring the TLS Certificate

The APP SQL pods handle application-related SQL traffic, primarily servicing Network Function (NF) workloads. These workloads often involve high-throughput, low-latency, and long-lived connections, making seamless TLS handling crucial for availability and stability.

On Certificate update, following is the workflow:

1. The pod reloads the TLS so that APP SQL pod is loaded with new certificate.
2. Existing NF connections remain active and active sessions continue to use the old certificate to prevent disruption.
3. New connections are automatically established using the new Certificate.

Scheduled Switchover via Cron Job for Replication SQL and APP SQL Pods

A cron job is scheduled to run at predefined intervals to enforce a switchover or reload when a Certificate change is detected. This ensures updated Certificates are applied promptly, avoiding the risks associated with outdated or expired Certificates.

In case of Replication SQL pods, if the Certificate change has occurred but no natural switchover has happened, then the cron job triggers a forced reload or a switchover, applying the new Certificate.

In case of APP SQL pods:

1. The pod reloads the TLS so that APP SQL pod is loaded with new certificate.
2. Existing NF connections remain active and active sessions continue to use the old Certificate to prevent disruption.
3. New connections are automatically established using the new Certificate.

4

cnDBTier APIs

This chapter describes the APIs used in cnDBTier and their parameters.

Note

The "occne-cndbtier" namespace name used in this section is only an example. Ensure that you configure the namespace name according to your environment.

4.1 cnDBTier Cluster Events APIs

cnDBTier cluster events APIs are used to retrieve or update the following details about a cnDBTier cluster:

- Retrieve information about the events that occur in a cluster. The following table provides information about the cluster events:

Table 4-1 Cluster Event Types

Cluster Event	Description
START	This event is recorded when the cluster is installed.
RESTART	This event is recorded when the cluster is restarted after the cluster is disconnected and started.
RESTORE	This event is recorded when the cluster is restored from the backup during Georeplication.
RE_SYNC	This event is recorded when the cluster is synced after the cluster is restored and georeplication channels are established, and after all the data is synced from remote sites after the restore is performed.
FAILURE	This event is recorded when the cluster is in failed state.
CHECKPOINT	This event is recorded when the database is completely restored from the remote site, from this timestamp, it means that the database is in sync with remote sites until other restarts.
UNAVAILABLE_GRR_RESTORE_IN_PROGRESS	This event is returned when the Geo Replication Recovery is in progress in this cluster.
UNAVAILABLE_GRR_RESTORE_FAILED	This event is returned when the Geo Replication Recovery is Failed in this cluster.
UNAVAILABLE_UNKNOWN	This event is returned when the Site is marked as Failed and Geo Replication Recovery is not started in the cluster.

- Reset the counter value of `cluster_restart_disconnect` to 0 to indicate that the cluster is in healthy state currently.

These details can be used to determine if there is a data loss due to the events in a cluster. These APIs are hosted by the DB monitor service.

Table 4-2 cnDBTier Cluster Events APIs

Rest API URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
http://<base-uri>/db-tier/reset/parameter/cluster_restart_disconnect	This API is used to reset the counter value of cluster_restart_disconnect to zero in the current cluster to indicate that the cluster is currently healthy. Service Type: Realtime. This API provides real time data.	POST	NA	200 OK Successfully Updated cluster_restart_disconnect to 0. 500 INTERNAL SERVER ERROR - failed to reset cluster_restart_disconnect.	NA
http://<base-uri>/db-tier/reset/cluster/{cluster-name}/parameter/cluster_restart_disconnect	This API is used to reset the counter value of cluster_restart_disconnect to zero in the specified cluster to indicate that the cluster is currently healthy. Service Type: Realtime. This API provides real time data.	POST	NA	200 OK Successfully Updated cluster_restart_disconnect to 0. 404 Not Found - The cluster name has not been configured. 500 INTERNAL SERVER ERROR - failed to reset cluster_restart_disconnect.	NA

Table 4-2 (Cont.) cnDBTier Cluster Events APIs

Rest API URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
http://<base-uri>/db-tier/cluster/status	This API is used to retrieve the timestamps of the events from the current cluster along with details of the latest event that occurred in the cluster. Service Type: Realtime. This API provides real time data.	GET	NA	200 OK - Successfully fetched the cluster status with the last event occurred. 500 INTERNAL SERVER ERROR - Unable to fetch cluster status with the last event occurred.	<pre>{ "clusterName": "<cluster name>", "currentTS": "<current timestamp in UTC format>", "lastStartTS": "<last start Time Stamp if present else blank>", "lastRestoreTS": "<last restore Time Stamp if present else blank>", "lastDBResyncTS": "<last DB Resync Time Stamp if present else blank>", "lastRestartTS": "<last Restart Time Stamp if present else blank>", "lastFailureTS": "<last Failure Time Stamp if present else blank>", "lastCheckpointTS": "<last Checkpoint Time Stamp if present else blank>", "lastMonitorPollTS": "<last Monitor Poll Time Stamp if present else blank>", "clusterDisconnectCount": "<Cluster Disconnect Count>", "eventsList": [{ "eventType": "<event Type>", "eventTS": "<Time Stamp at which event occurred>" }], "remoteSitesMonitorTS": [{ "clusterName": "<cluster name>",</pre>

Table 4-2 (Cont.) cnDBTier Cluster Events APIs

Rest API URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
					<pre> "monitorTS": "<Last Monitor Time Stamp for the cluster If present else blank>" }, { "clusterName": "<cluster name>", "monitorTS": "<Last Monitor Time Stamp for the cluster If present else blank>" }]</pre>

Table 4-2 (Cont.) cnDBTier Cluster Events APIs

Rest API URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
http://base-uri/db-tier/cluster/status/events/{numberOfLastEvents}	This API is used to retrieve the timestamps of the events from the current cluster along with details of the specified number of latest events that occurred in the cluster. Service Type: Realtime. This API provides real time data.	GET	NA	200 OK: Successfully fetched the cluster status with the last the last <numberOfLastEvents> events occurred. 400 BAD REQUEST - If {numberOfLastEvents} exceeds the configured max count which is by default 10 500 INTERNAL SERVER ERROR: Unable to fetch cluster status with the last event occurred.	<pre>{ "clusterName": "<cluster name>", "currentTS": "<current timestamp in UTC format>", "lastStartTS": "<last start Time Stamp if present else blank>", "lastRestoreTS": "<last restore Time Stamp if present else blank>", "lastDBResyncTS": "<last DB Resync Time Stamp if present else blank>", "lastRestartTS": "<last Restart Time Stamp if present else blank>", "lastFailureTS": "<last Failure Time Stamp if present else blank>", "lastCheckpointTS": "<last Checkpoint Time Stamp if present else blank>", "lastMonitorPollTS": "<last Monitor Poll Time Stamp if present else blank>", "clusterDisconnectCount": "<Cluster Disconnect Count>", "eventsList": [{ "eventType": "<event Type>", "eventTS": "<Time Stamp at which event occurred>" }, { "eventType": "<event Type>", "eventTS": "<Time Stamp at which event</pre>

Table 4-2 (Cont.) cnDBTier Cluster Events APIs

Rest API URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
					<pre> occurred>" }], "remoteSitesMonitorTS": [{ "clusterName": "<cluster name>", "monitorTS": "<Last Monitor Time Stamp for the cluster If present else blank>" }, { "clusterName": "<cluster name>", "monitorTS": "<Last Monitor Time Stamp for the cluster If present else blank>" }] } </pre>

Table 4-2 (Cont.) cnDBTier Cluster Events APIs

Rest API URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
http://<base-uri>/db-tier/all/cluster/status/	This API is used to retrieve the timestamps of events from all the clusters along with details of the latest event occurred in the clusters. Service Type: Realtime. This API provides real time data.	GET	NA	200 OK - Successfully fetched the cluster status with the last event occurred. 500 INTERNAL SERVER ERROR - Unable to fetch cluster status with the last event occurred.	<pre>[{ "clusterName": "<cluster name>", "currentTS": "<current timestamp in UTC format>", "lastStartTS": "<last start Time Stamp if present else blank>", "lastRestoreTS": "<last restore Time Stamp if present else blank>", "lastDBResyncTS": "<last DB Resync Time Stamp if present else blank>", "lastRestartTS": "<last Restart Time Stamp if present else blank>", "lastFailureTS": "<last Failure Time Stamp if present else blank>", "lastCheckpointTS": "<last Checkpoint Time Stamp if present else blank>", "lastMonitorPollTS": "<last Monitor Poll Time Stamp if present else blank>", "clusterDisconnectCount": "<Cluster Disconnect Count>", "eventsList": [{ "eventType": "<event Type>", "eventTS": "<Time Stamp at which event occurred>" }], }]</pre>

Table 4-2 (Cont.) cnDBTier Cluster Events APIs

Rest API URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
					<pre> "remoteSitesMonitorTS": [{ "clusterName": "<cluster name>", "monitorTS": "<Last Monitor Time Stamp for the cluster If present else blank>" }, { "clusterName": "<cluster name>", "monitorTS": "<Last Monitor Time Stamp for the cluster If present else blank>" }], { "clusterName": "<cluster name>", "currentTS": "<current timestamp in UTC format>", "lastStartTS": "<last start Time Stamp if present else blank>", "lastRestoreTS": "<last restore Time Stamp if present else blank>", "lastDBResyncTS": "<last DB Resync Time Stamp if present else blank>", "lastRestartTS": "<last Restart Time Stamp if present else blank>", "lastFailureTS": "<last Failure Time Stamp if present else blank>", "lastCheckpointTS": "<last Checkpoint Time Stamp if present else blank>", </pre>

Table 4-2 (Cont.) cnDBTier Cluster Events APIs

Rest API URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
					<pre> "lastMonitorPollTS": "<last Monitor Poll Time Stamp if present else blank>", "clusterDisconnectCount": "<Cluster Disconnect Count>", "eventsList": [{ "eventType": "<event Type>", "eventTS": "<Time Stamp at which event occurred>" }], "remoteSitesMonitorTS": [{ "clusterName": "<cluster name>", "monitorTS": "<Last Monitor Time Stamp for the cluster If present else blank>" }, { "clusterName": "<cluster name>", "monitorTS": "<Last Monitor Time Stamp for the cluster If present else blank>" }] </pre>

Table 4-2 (Cont.) cnDBTier Cluster Events APIs

Rest API URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
http://<base-uri>/db-tier/all/cluster/status/events/{numberOfLastEvents}	This API is used to retrieve the timestamps of events from all the clusters along with details of the specified number of latest events that occurred in the clusters. Service Type: Realtime. This API provides real time data.	GET	NA	200 OK: Successfully fetched the cluster status with the last the last <numberOfLastEvents> events occurred. 400 BAD REQUEST - If {numberOfLastEvents} exceeds the configured max count which is by default 10. 500 INTERNAL SERVER ERROR: Unable to fetch cluster status with the last event occurred.	<pre>[{ "clusterName": "<cluster name>", "currentTS": "<current timestamp in UTC format>", "lastStartTS": "<last start Time Stamp if present else blank>", "lastRestoreTS": "<last restore Time Stamp if present else blank>", "lastDBResyncTS": "<last DB Resync Time Stamp if present else blank>", "lastRestartTS": "<last Restart Time Stamp if present else blank>", "lastFailureTS": "<last Failure Time Stamp if present else blank>", "lastCheckpointTS": "<last Checkpoint Time Stamp if present else blank>", "lastMonitorPollTS": "<last Monitor Poll Time Stamp if present else blank>", "clusterDisconnectCount": "<Cluster Disconnect Count>", "eventsList": [{ "eventType": "<event Type>", "eventTS": "<Time Stamp at which event occurred>" }, {</pre>

Table 4-2 (Cont.) cnDBTier Cluster Events APIs

Rest API URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
					<pre> "eventType": "<event Type>", "eventTS": "<Time Stamp at which event occurred>" }], "remoteSitesMonitorTS": [{ "clusterName": "<cluster name>", "monitorTS": "<Last Monitor Time Stamp for the cluster If present else blank>" }, { "clusterName": "<cluster name>", "monitorTS": "<Last Monitor Time Stamp for the cluster If present else blank>" }] }, { "clusterName": "<cluster name>", "currentTS": "<current timestamp in UTC format>", "lastStartTS": "<last start Time Stamp if present else blank>", "lastRestoreTS": "<last restore Time Stamp if present else blank>", "lastDBResyncTS": "<last DB Resync Time Stamp if present else blank>", "lastRestartTS": "<last Restart Time Stamp if present else blank>," </pre>

Table 4-2 (Cont.) cnDBTier Cluster Events APIs

Rest API URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
					<pre> "lastFailureTS": "<last Failure Time Stamp if present else blank>", "lastCheckpointTS": "<last Checkpoint Time Stamp if present else blank>", "lastMonitorPollTS": "<last Monitor Poll Time Stamp if present else blank>", "clusterDisconnectCount": "<Cluster Disconnect Count>", "eventsList": [{ "eventType": "<event Type>", "eventTS": "<Time Stamp at which event occurred>" }, { "eventType": "<event Type>", "eventTS": "<Time Stamp at which event occurred>" }], "remoteSitesMonitorTS": [{ "clusterName": "<cluster name>", "monitorTS": "<Last Monitor Time Stamp for the cluster If present else blank>" }, { "clusterName": "<cluster name>", "monitorTS": </pre>

Table 4-2 (Cont.) cnDBTier Cluster Events APIs

Rest API URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
					"<Last Monitor Time Stamp for the cluster If present else blank>" <pre> }] }]</pre>

Note

Replace the value of <base-uri> in the REST URL with <db-monitor-svc service name>.<namespace>:8080.

Refer to the following examples to use these REST APIs to update or retrieve the cluster details:

```

$ curl -X POST http://<base-uri>/db-tier/reset/parameter/
cluster_restart_disconnect
$ curl -X POST http://<base-uri>/db-tier/reset/cluster/{cluster-name}/
parameter/cluster_restart_disconnect
$ curl -X GET http://<base-uri>/db-tier/cluster/status
$ curl -X GET http://<base-uri>/db-tier/cluster/status/events/
{numberoflastevents}
$ curl -X GET http://<base-uri>/db-tier/all/cluster/status
$ curl -X GET http://<base-uri>/db-tier/all/cluster/status/events/
{numberoflastevents}
```

4.2 cnDBTier APIs for CNC Console

cnDBTier APIs facilitate CNC Console to determine the status of cnDBTier and the associated cross-site replication. The DB replication service hosts these APIs.

Table 4-3 cnDBTier APIs for CNC Console

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
http://<base-uri>/ocdbtier/status/replication/realtime/{remoteSiteName}	This API provides site-specific replication status. Service Type: Realtime. This API provides real time data.	GET	NA	200 OK - Fetched the list of backups present in cnDBTier. 400 Bad Request 404 Not Found 500 Internal Server Error. 503 Service Unavailable - Cross-site replication is not enabled.	200 OK: <pre> { "localSiteName": "<current-site-name>", "remoteSiteName": "<remote-site-name>", "replicationStatus": "UP/DOWN/INITIALIZING" "secondsBehindRemote": "<greater_secondsBehindRemote_of_multiplegroups_withremotesite>" "replicationGroupDelay": [{ "replchannel_group_id": "1", "secondsBehindRemote": <seconds>, "channelDetails": [{ "remote_replication_ip": "127.0.0.1", "role": "ACTIVE" }, { "remote_replication_ip": "127.0.0.2", "role": "STANDBY" }] }, { </pre>

Table 4-3 (Cont.) cnDBTier APIs for CNC Console

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
					<pre> "replchannel_group_id" : "2", "secondsBehindRemote": <seconds>, "channelDetails": [{ "remote_replication_ip ": "127.0.0.3", "role": "ACTIVE" }, { "remote_replication_ip ": "127.0.0.4", "role": "STANDBY" }] } </pre> 400 Bad Request: <pre> { "title": "Bad Request", "status": 400, "detail": "<Exception details>", "cause": <cause>, "invalidParams": <params> } </pre> 404 Not Found: <pre> { "title": "Not Found", "status": 404, "detail": "<Exception details>", "cause": <cause>, "invalidParams": </pre>

Table 4-3 (Cont.) cnDBTier APIs for CNC Console

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
					<pre><params> }</pre> 500 Internal Server Error: <pre>{ "title": "Internal Server Error", "status": 500, "detail": "<Exception details>", "cause": <cause>, "invalidParams": <params> }</pre> 503 Service Unavailable - Cross-site replication is not enabled: <pre>{ "title": "Service Unavailable", "status": 503, "detail": "<Exception details>", "cause": <cause>, "invalidParams": <params> }</pre>

Table 4-3 (Cont.) cnDBTier APIs for CNC Console

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
http://<base-uri>/ocdbtier/status/replication/realtime	This API provides the overall replication status. Service Type: Realtime. This API provides real time data.	GET	NA	200 OK - Cross-site replication is active 500 Internal Server Error 503 Service Unavailable - Cross-site replication is not enabled	200 OK - Cross-site replication is active: <pre> { [{ "localSiteName": "<current-site-name>", "remoteSiteName": "<remote-site-name>", "replicationStatus": "UP/DOWN/INITIALIZING" "secondsBehindRemote": "<greater_secondsBehindRemote_of_multiplegroups_withremotesite>" "replicationGroupDelay": [{ replchannel_group_id: 1, secondsBehindRemote: <seconds> }, { replchannel_group_id: 2, secondsBehindRemote: <seconds> }], { "localSiteName": "<current-site-name>", "remoteSiteName": "<remote-site-name>", </pre>

Table 4-3 (Cont.) cnDBTier APIs for CNC Console

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
					<pre> "replicationStatus":"U P/DOWN/INITIALIZING" "secondsBehindRemote": "<greater_secondsBehin dRemote_of_multiplegro ups_withremotesite>" "replicationGroupDelay ": [{ replchannel_group_id: 1, secondsBehindRemote: <seconds> }, { replchannel_group_id: 2, secondsBehindRemote: <seconds> }]] } } </pre> 500 Internal Server Error: <pre> { "title": "Internal Server Error", "status": 500, "detail": "<Exception details>", "cause": <cause>, "invalidParams": <params> } </pre>

Table 4-3 (Cont.) cnDBTier APIs for CNC Console

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
					503 Service Unavailable - Cross-site replication is not enabled: <pre>{ "title": "Service Unavailable", "status": 503, "detail": "<Exception details>", "cause": <cause>, "invalidParams": <params> }</pre>
http://<base-uri>/ocdbtier/status/cluster/local/realtime	This API provides the local cluster status. Service Type: Realtime. This API provides real time data.	GET	NA	200 OK- The local NDB cluster is UP or DOWN 500 Internal Server Error	200 OK- The local NDB cluster is UP or DOWN: <pre>{ "clusterName": "<current-cluster-name>", "clusterStatus": "UP/DOWN" }</pre> 500 Internal Server Error: <pre>{ "title": "Internal Server Error", "status": 500, "detail": "<Exception details>", "cause": <cause>, "invalidParams": <params> }</pre>

Table 4-3 (Cont.) cnDBTier APIs for CNC Console

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
http://<base-uri>/ocdbtier/version	This API provides the cnDBTier version. Service Type: Realtime. This API provides real time data.	GET	NA	200 OK 500 Internal Server Error	200 OK: <pre>{ "cndbtier_version": "25.2.100", "ndb_version": "ndb-8.4.3" }</pre> 500 Internal Server Error: <pre>{ "title": "Internal Server Error", "status": 500, "detail": "<Exception details>", "cause": <cause>, "invalidParams": <params> }</pre>

Table 4-3 (Cont.) cnDBTier APIs for CNC Console

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
http://<base-uri>/ocdbtier/list/backups	This API provides the list of backups completed in the current site. Service Type: Realtime. This API provides real time data.	GET	NA	200 OK - Fetched the Heart Beat Status of every replication service running on the current cluster 404 NOT FOUND Table Missing 500 Internal Server Error	200 OK: <pre>{ "localSiteName": "<current_site_name>", "backupDetails": [{ "backupId": "<backup identifier>", "backupSize": "<size of backup>", "creationTimeStamp": "<timestamp at which backup was initiated>" }, { "backupId": "<backup identifier>", "backupSize": "<size of backup>", "creationTimeStamp": "<timestamp at which backup was initiated>" }, { "backupId": "<backup identifier>", "backupSize": "<size of backup>", "creationTimeStamp": "<timestamp at which backup was initiated>" }] }</pre> 404 NOT FOUND Table Missing: <pre>{ "title": "Not Found", "status": 404, "detail": "DBTIER_BACKUP_INFO table doesn't exist"</pre>

Table 4-3 (Cont.) cnDBTier APIs for CNC Console

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
					in the current site so cannot list the backup", "cause": null, "invalidParams": null } • 500 Internal Server Error: <pre>{ "title": "Internal Server Error", "status": 500, "detail": "Could not list the backups", "cause": null, "invalidParams": null }</pre>

Table 4-3 (Cont.) cnDBTier APIs for CNC Console

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
http://<base-uri>/ocdbtier/statistics/report/dbinfo	This API provides database statistic reports. Service Type: Realtime. This API provides real time data.	GET	NA	200 OK 500 INTERNAL_SERVER_ERROR 503 SERVICE_UNAVAILABLE	200 OK: <pre>{ "databaseCount": <database count>, "databaseTablesCount": [{ "dbName": "<database name>", "tableCount": <table count> }, { "dbName": "<database name>", "tableCount": <table count> }], "databaseTableRowsCount": [{ "dbName": "<database name>", "tables": [{ "tableName": "<table name>", "rowCount": <row count> }], { "dbName": "<database name>", "tables": [{ "tableName": "<table name>", "rowCount": <row count> }] }] }] }</pre>

Table 4-3 (Cont.) cnDBTier APIs for CNC Console

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
					<pre> }, { "tableName": "<table name>", "rowCount": <row count> }] } </pre> 500 INTERNAL_SERVER_ERROR: <pre> { "title": "Internal Server Error", "status": 500, "detail": "<Exception Details>", "cause": null, "invalidParams": null } </pre> 503 SERVICE_UNAVAILABLE: <pre> { "title": "Service Unavailable", "status": 503, "detail": "<Exception details>", "cause": null, "invalidParams": null } </pre>

Table 4-3 (Cont.) cnDBTier APIs for CNC Console

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
http://<base-uri>/ocdbtier/heartbeat/status	This API provides the overall HeartBeat Status. Service Type: Realtime. This API provides real time data.	GET	NA	200 OK 503 SERVICE_UNAVAILABLE - Not able to query the Database 500 INTERNAL_SERVER_ERROR - Could Not Check The HeartBeat Status	200 OK: <pre>{ "heartBeatDetails": [{ "remoteSiteName": "<remote_site_name>", "heartBeatStatus": "<SUCCESS/FAILURE>", "heartBeatLag": "<Heart Beat Lag in Seconds>", "replicationChannelGroupId": <replication channel group id> }, { "remoteSiteName": "<remote_site_name>", "heartBeatStatus": "<SUCCESS/FAILURE>", "heartBeatLag": "<Heart Beat Lag in Seconds>", "replicationChannelGroupId": <replication channel group id> }], "siteName": "<current_site_name>", }</pre> 503 SERVICE_UNAVAILABLE - Not able to query the Data Base: <pre>{ "title": "Service Unavailable", "status": 503, "detail": "Not able</pre>

Table 4-3 (Cont.) cnDBTier APIs for CNC Console

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
					to query the Data Base", "cause": null, "invalidParams": null } 500 INTERNAL_SERVER_ERROR - Could Not Check The HeartBeat Status: <pre>{ "title": "Internal Server Error", "status": 500, "detail": "Could Not Check The HeartBeat Status", "cause": null, "invalidParams": null }</pre>

Table 4-3 (Cont.) cnDBTier APIs for CNC Console

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
http://<base-uri>/ocdbtier/on-demand/backup/initiate	This API initiates an on-demand backup. Service Type: Realtime. This API provides real time data.	GET	NA	200 OK 404 NOT FOUND 500 INTERNAL_SERVER_ERROR	200 OK: <pre>{ "siteName": "<current_site_name>", "drStatus": "<drStatus>", "backupId": "<backupId>", "backupStatus": "<backupStatus>", "remoteTransferStatus" : "<remoteTransferStatus>", "initiateBackup": false/true, "transferStatus": "<transferStatus>" }</pre> 404 NOT FOUND: <pre>{ "title": "Not Found", "status": 404, "detail": "<Exception message>", "cause": null, "invalidParams": null }</pre> 500 INTERNAL_SERVER_ERROR: <pre>{ "title": "Internal Server Error", "status": 500, "detail": "<Exception message>", "cause": null, "invalidParams": null }</pre>

Table 4-3 (Cont.) cnDBTier APIs for CNC Console

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
http://<base-uri>/ocdbtier/on-demand/backup/initiate	This API initiates on-demand backup as per the provided requirements. Service Type: Realtime. This API provides real time data.	POST	<pre>{ "siteName": "<current_site_name>", "drStatus": "<drStatus>", "backupId": "<backupId>", "backupStatus": "<backupStatus>", "remoteTransferStatus": "<remoteTransferStatus>", "initiateBackup": false/true, "transferStatus": "<transferStatus>" }</pre>	200 OK 400 BAD REQUEST 404 NOT FOUND 500 INTERNAL_SERVER_ERROR	200 OK: <pre>{ "siteName": "<current_site_name>", "drStatus": "<drStatus>", "backupId": "<backupId>", "backupStatus": "<backupStatus>", "remoteTransferStatus": "<remoteTransferStatus>", "initiateBackup": false/true, "transferStatus": "<transferStatus>" }</pre> 404 BAD REQUEST: <pre>{ "title": "Bad Request", "status": 400, "detail": "Couldn't initiate the backup when initiateBackup flag is set to false", "cause": null, "invalidParams": [{ "param": "initiateBackup", "reason": "Couldn't initiate the backup when initiateBackup flag is set to false" }] }</pre>

Table 4-3 (Cont.) cnDBTier APIs for CNC Console

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
					404 NOT FOUND: <pre>{ "title": "Not Found", "status": 404, "detail": "<Exception message>", "cause": null, "invalidParams": null }</pre> 500 INTERNAL_SERVER_ERROR: <pre>{ "title": "Internal Server Error", "status": 500, "detail": "BACKUP_INITIATION_FAILED", "cause": null, "invalidParams": null }</pre>

Table 4-3 (Cont.) cnDBTier APIs for CNC Console

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
http://<base-uri>/ocdbtier/health-info/replication-svc/status	This API provides the health status of the replication service. Service Type: Realtime. This API provides real time data.	GET	NA	200 OK 503 SERVICE_UNAVAILABLE - Replication Service IP is empty 500 INTERNAL_SERVER_ERROR - Could Not Check The Replication Health Status	200 OK: <pre>{ "localSiteName": "<siteName>", "healthStatusDetails": [{ "serviceName": "<serviceName>", "serviceStatus": "UP/ DOWN", "connectionToDbStatus" : "UP/DOWN", "overAllReplicationSer viceHealth": "UP/DOWN" }, { "serviceName": "<serviceName>", "serviceStatus": "UP/ DOWN", "connectionToDbStatus" : "UP/DOWN", "overAllReplicationSer viceHealth": "UP/DOWN" }.....] }</pre> 503 SERVICE_UNAVAILABLE - Replication Service IP is empty: <pre>{ "title": "Service Unavailable", "status": 503, "detail": "Replication Service</pre>

Table 4-3 (Cont.) cnDBTier APIs for CNC Console

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
					<pre>IP is empty", "cause": null, "invalidParams": null }</pre> 500 INTERNAL_SERVER_ERROR - Could Not Check The Replication Health Status: <pre>{ "title": "Internal Server Error", "status": 500, "detail": "Could Not Check The Health Status", "cause": null, "invalidParams": null }</pre>
http://<base-uri>/ocdbtier/health-info/monitor-svc/status	This API provides the health status of monitor service. Service Type: Realtime. This API provides real time data.	GET	NA	200 OK (Monitor Service is healthy)	<pre>200 OK: { "serviceName": "mysql- cluster-db-monitor-svc", "connectionToDbStatus": "UP", "metricScrapeStatus": "UP", "overAllMonitorServiceHeal th": "UP" }</pre>

Table 4-3 (Cont.) cnDBTier APIs for CNC Console

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
http://<base-uri>/ocdbtier/health-info/ndb-svc/status	This API provides the overall health status of NDB data service. Service Type: Realtime. This API provides real time data.	GET	NA	200 OK 404 NOT FOUND - sitename not found 500 INTERNAL_SERVER_ERROR - Could Not Check The Health Status	200 OK: <pre>{ "localSiteName": "<siteName>", "ndbHealthStatusDetails": [{ "serviceName": "<serviceName>", "serviceStatus": "UP/DOWN", "pvcHealthStatus": "UP/DOWN" }, { "serviceName": "<serviceName>", "serviceStatus": "UP/DOWN", "pvcHealthStatus": "UP/DOWN" }] }</pre> 404 NOT FOUND - sitename not found: <pre>{ "title": "Not Found", "status": 404, "detail": "sitename not found", "cause": null, "invalidParams": null }</pre> 500 INTERNAL_SERVER_ERROR

Table 4-3 (Cont.) cnDBTier APIs for CNC Console

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
					- Could Not Check The Health Status: <pre>{ "title": "Internal Server Error", "status": 500, "detail": "Could Not Check The Health Status", "cause": null, "invalidParams": null }</pre>

Table 4-3 (Cont.) cnDBTier APIs for CNC Console

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
http://<base-uri>/ocdbtier/health-info/backup-mgr-svc/status	This API provides the health status of the backup manager service. Service Type: Realtime. This API provides real time data.	GET	NA	200 OK 500 INTERNAL_SERVER_ERROR - Could Not Check The Health Status	200 OK: <pre>{ "serviceName": "mysql-cluster-db-backup-manager-svc", "connectionToDbStatus" : "UP", "serviceStatus": "UP", "backupExecutorHealthStatus": [{ "nodeId": 2, "connectionToDbStatus" : "UP" }, { "nodeId": 1, "connectionToDbStatus" : "UP" }], "overAllBackupManagerHealth": "UP" }</pre> 500 INTERNAL_SERVER_ERROR - Could Not Check The Health Status: <pre>{ "title": "Internal Server Error", "status": 500, "detail": "Could Not Check The Health Status", "cause": null, "invalidParams": null }</pre>

Table 4-3 (Cont.) cnDBTier APIs for CNC Console

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
http://<base-uri>/ocdbtier/list/clusterdetails	This API provides the cnDBTier cluster details. Service Type: Realtime. This API provides real time data.	GET	NA	200 OK 404 NOT FOUND 500 INTERNAL SERVER ERROR	200 OK: <pre>[{ "clusterName": "<clusterName>", "clusterId": "1", "status": "ACTIVE/ FAILED" }, { "clusterName": "<clusterName>", "clusterId": "2", "status": "ACTIVE/ FAILED" }, { "clusterName": "<clusterName>", "clusterId": "3", "status": "ACTIVE/ FAILED" }]</pre> 404 NOT FOUND: <pre>{ "title": "Not Found", "status": 404, "detail": "<Error message>", "cause": null, "invalidParams": null }</pre> 500 INTERNAL SERVER ERROR: <pre>{ "title": "INTERNAL_SERVER_ERROR ", "status": 500, "detail": "<Error message>",</pre>

Table 4-3 (Cont.) cnDBTier APIs for CNC Console

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
					<pre>"cause": null, "invalidParams": null }</pre>
http://<base-uri>/ocdbtier/markcluster/failed	This API fetches information about failed cnDBTier clusters. Service Type: Realtime. This API provides real time data.	GET	NA	200 OK 404 NOT FOUND 500 INTERNAL SERVER ERROR	200 OK: <pre>{ "clusterNameTobeMarkedAsFailed": "", "failedClusterNames": ["<failedcluster1>", "<failedcluster2>"] }</pre> 404 NOT FOUND: <pre>{ "title": "Not Found", "status": 404, "detail": "<Error message>", "cause": null, "invalidParams": null }</pre> 500 INTERNAL SERVER ERROR: <pre>{ "title": "INTERNAL_SERVER_ERROR", "status": 500, "detail": "<Error message>", "cause": null, "invalidParams": null }</pre>

Table 4-3 (Cont.) cnDBTier APIs for CNC Console

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
http://<base-uri>/ocdbtier/markcluster/failed	This API is used to mark cnDBTier clusters as Failed. Service Type: Realtime. This API provides real time data.	PUT	<pre>{ "clusterNameToBeMarkedAsFailed": "<failedcluster>", "failedClusterNames": [[]] }</pre>	200 OK 400 BAD REQUEST 404 NOT FOUND 500 INTERNAL SERVER ERROR 503 SERVICE_UNAVAILABLE	200 OK: <pre>{ "clusterNameToBeMarkedAsFailed": "", "failedClusterNames": ["<failedcluster1>", "<failedcluster2>"] }</pre> 400 BAD REQUEST: <pre>{ "title": "BAD_REQUEST", "status": 400, "detail": "<ErrorMessage>", "cause": null, "invalidParams": null }</pre> 404 NOT FOUND: <pre>{ "title": "NotFound", "status": 404, "detail": "<ErrorMessage>", "cause": null, "invalidParams": null }</pre> 500 INTERNAL SERVER ERROR: <pre>{ "title": "INTERNAL_SERVER_ERROR", "status": 500, "detail": "<ErrorMessage>" }</pre>

Table 4-3 (Cont.) cnDBTier APIs for CNC Console

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
					<pre>message>", "cause": null, "invalidParams": null }</pre> 503 SERVICE_UNAVAILABLE: <pre>{ "title": "Service Unavailable", "status": 503, "detail": "<Error message>", "cause": null, "invalidParams": null }</pre>

Table 4-3 (Cont.) cnDBTier APIs for CNC Console

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
http://<base-uri>/ocdbtier/faultrecovery/status	This API is used to monitor the status of georeplication recovery. Service Type: Realtime. This API provides real time data.	GET	NA	200 OK 404 NOT FOUND 500 INTERNAL SERVER ERROR	200 OK: <pre>{ "localClusterName": "<local cluster name>", "faultRecoverStatus": [{ "clusterName": "<clusterName>", "faultRecoverState": "<faultRecoverState>" }, { "clusterName": "<clusterName>", "faultRecoverState": "<faultRecoverState>" }, { "clusterName": "<clusterName>", "faultRecoverState": "<faultRecoverState>" }] }</pre> 404 NOT FOUND: <pre>{ "title": "Not Found", "status": 404, "detail": "<Error message>", "cause": null, "invalidParams": null }</pre>

Table 4-3 (Cont.) cnDBTier APIs for CNC Console

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
					500 INTERNAL SERVER ERROR: <pre>{ "title": "INTERNAL_SERVER_ERROR ", "status": 500, "detail": "<Error message>", "cause": null, "invalidParams": null }</pre>

Table 4-3 (Cont.) cnDBTier APIs for CNC Console

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
http://<base-uri>/ocdbtier/faultrecovery/start	This API is used to start georeplication recovery. Service Type: Realtime. This API provides real time data.	PUT	{ "backupClusterName": "<backup cluster name>", "failedClusterName": "<failed cluster name>" }	200 OK 400 BAD REQUEST 404 NOT FOUND 409 CONFLICT 500 INTERNAL SERVER ERROR 503 SERVICE_UNAVAILABLE	200 OK: <pre>{ "backupClusterName": "<backup cluster name>", "failedClusterName": "<failed cluster name>" }</pre> 400 BAD REQUEST: <pre>{ "title": "INVALID INPUT", "status": 400, "detail": "<Error message>", "cause": null, "invalidParams": null }</pre> 404 NOT FOUND: <pre>{ "title": "Not Found", "status": 404, "detail": "<Error message>", "cause": null, "invalidParams": null }</pre> 409 CONFLICT: <pre>{ "title": "Conflict", "status": 409, "detail": "<Error message>", "cause": null, "invalidParams": null }</pre>

Table 4-3 (Cont.) cnDBTier APIs for CNC Console

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
					500 INTERNAL SERVER ERROR: <pre>{ "title": "INTERNAL_SERVER_ERROR ", "status": 500, "detail": "<Error message>", "cause": null, "invalidParams": null }</pre> 503 SERVICE_UNAVAILABLE: <pre>{ "title": "Service Unavailable", "status": 503, "detail": "<Error message>", "cause": null, "invalidParams": null }</pre>

Table 4-3 (Cont.) cnDBTier APIs for CNC Console

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
http://base-uri/ ocdbtier/backup/ status	This API is used to check the status of NDB backup. Service Type: Realtime. This API provides real time data.	GET	Not Applicable	200 OK 500 INTERNAL SERVER ERROR	200 OK <pre>{ "currentTimestamp": "<current timestamp>", "isNdbBackupInProgress": true/false, "nextBackupScheduleTime": "<next backup scheduled time>" }</pre> 500 INTERNAL SERVER ERROR <pre>{ "title": "Internal Server Error", "status": 500, "detail": "<Exception details>", "cause": <cause>, "invalidParams": <params> }</pre>

Note

Replace the value of <base-uri> in the URL with <db-monitor-svc service name>.<namespace>:8080.
For example: curl http://mysql-cluster-db-monitor-svc.occne-cndbtier:8080/ocdbtier/status/replication/realtime

4.3 cnDBTier Backup APIs

cnDBTier backup APIs are used to create on-demand database backups and check the status of the backups in cnDBTier. The cnDBTier backup API is hosted by the DB backup manager service.

Table 4-4 cnDBTier Backup API

Rest URL		HTTP Method	Request Payload	Response Code	Response Payload
http://<base-uri>/db-tier/on-demand/backup/initiate	This API is used to initiate site-specific on-demand backup creation. Service Type: Realtime. This API provides real time data.	POST	NA	200 OK - Successfully started the on-demand backup 400 BAD REQUEST request is invalid 409 CONFLICT backup is already going on 500 INTERNAL SERVER ERROR - Could not initiate on-demand backup. 503 SERVICE UNAVAILABLE data node is down	200 OK: <pre>{ "backup_id": "<Backup id>", "status": "<Backup Initiation Status>" "backup_encryption_flag": "<backup encryption enabled/disabled>" }</pre> 400 BAD REQUEST request is invalid: <pre>{ "backup_id": "", "status": "BACKUP_INITIATION_FAILED" "backup_encryption_flag": "true/false" }</pre> 409 CONFLICT backup is already going on: <pre>{ "backup_id": "", "status": "BACKUP_INITIATION_FAILED" "backup_encryption_flag": "true/false" }</pre> 500 INTERNAL SERVER ERROR: <pre>{ "error_type": "<error type>", "error_message": "<erro</pre>

Table 4-4 (Cont.) cnDBTier Backup API

Rest URL		HTTP Method	Request Payload	Response Code	Response Payload
					<pre>r message>" }</pre> • 503 SERVICE UNAVAILABLE data node is down: <pre>{ "backup_id": "", "status": "BACKUP_INITIATION_FAILED" "backup_encryption_flag": "true/false" }</pre>

Table 4-4 (Cont.) cnDBTier Backup API

Rest URL		HTTP Method	Request Payload	Response Code	Response Payload
http://<base-uri>/db-tier/on-demand/backup/<BACKUP_ID_ID>/status	This API is used to get site-specific on-demand backup status. Service Type: Realtime. This API provides real time data.	GET	NA	200 OK - Successfully started the on-demand backup 400 BAD REQUEST Request is invalid 404 NOT FOUND backup_id does not exist 500 INTERNAL SERVER ERROR - Could not check the status for the specified on-demand backup ID	200 OK: Response payload when <code>.Values.global.remoteTransfer.enable</code> is set to <i>true</i>: <pre>{ "backup_status": "<Backup status>", "transfer_status": "<Transfer Status>" }</pre> Note: The <code>"transfer_status"</code> parameter in the response payload indicates the backup transfer status from db-backup-manager-svc to db-replication-svc and not to the remote server. Response payload when <code>.Values.global.remoteTransfer.enable</code> is set to <i>false</i>: <pre>{ "status": "<Backup status>" }</pre> 400 BAD REQUEST Request is invalid: <pre>{ "error_description": "<error_description>", "error_type": "<error type>" }</pre> 404 NOT FOUND backup_id does not exist: <pre>{</pre>

Table 4-4 (Cont.) cnDBTier Backup API

Rest URL		HTTP Method	Request Payload	Response Code	Response Payload
					<pre>"error_description": "<error_description>", "error_type": "<error type>" } • 500 INTERNAL SERVER ERROR: { "error_type": "<error type>", "error_message": "<erro r message>" }</pre>

Table 4-4 (Cont.) cnDBTier Backup API

Rest URL		HTTP Method	Request Payload	Response Code	Response Payload
http://<base-uri>/db-tier/health	This API is used to perform a cnDBTier health check. Service Type: Realtime. This API provides real time data.	GET	NA	200 OK 500 INTERNAL SERVER ERROR	200 OK: <pre>{ "is_connected_to_db": "<true false>", "overall_status": "<true false>", "service_name": "<mysql-cluster-db-backup-manager-svc>" "executor_status_list" : [{ "_node_id": "<node_id>", "_is_connected_to_db": "<true false>", "_is_running": "<true false>" }, ...] } </pre> 500 INTERNAL SERVER ERROR: <pre>{ "error_type": "<error type>", "error_message": "<error message>" }</pre>

Table 4-4 (Cont.) cnDBTier Backup API

Rest URL		HTTP Method	Request Payload	Response Code	Response Payload
This API is used to get the cnDBTier NDB backup status. Service Type: Realtime. This API provides real time data.	GET	NA	200 OK 500 INTERNAL SERVER ERROR	200 OK: <pre>{ "isNdbBackupInProgress": "<true if an NDB backup is in progress; otherwise, false>", "nextBackupScheduleTime": "<UTC timestamp for the next scheduled routine backup>", "currentTimestamp": "<Current UTC timestamp from the NDB cluster>" }</pre> 500 INTERNAL SERVER ERROR: <pre>Not able to fetch Current NDB Backup Status</pre>	

Note

Replace the value of <base-uri> in the URL with <db-backup-manager-svc svc>: 8080.

For example,

- Run the following command to get the db-backup-manager-svc pod name from cnDBTier cluster:

```
$ kubectl -n <namespace of cnDBTier cluster> get pods | grep "db-backup-manager-svc" | awk '{print $1}'
```

- Run the following command to get the db-backup-manager-svc svc name from cnDBTier cluster:

```
$ kubectl -n <namespace of cnDBTier cluster> get svc | grep "db-backup-manager-svc" | awk '{print $1}'
```

- Run the following commands to login to the db-backup-manager-svc pod, use the REST APIs to create an on-demand backup, and check the backup status:

```
$ kubectl -n <namespace of cnDBTier cluster> exec -it <db-backup-manager-svc pod> -- bash
$ curl -i -X POST http://<db-backup-manager-svc svc>:8080/db-tier/on-demand/backup/initiate
$ curl -i -X GET http://<db-backup-manager-svc svc>:8080/db-tier/on-demand/backup/<BACKUP_ID_INITIATED>/status
```

4.4 cnDBTier Replication Service APIs

This section provides details about cnDBTier replication service APIs.

4.4.1 cnDBTier Switchover APIs

cnDBTier Switchover APIs are used to start or stop switchovers on replication services.

Table 4-5 cnDBTier Switchover APIs

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
http://<base-uri>/ocdbtier/georeplication/switchover/start/siteName/{siteName}/remotesitenam/{remoteSiteName}/replchannelgroupid/{replChannelGroupid}	This API is used to start replication switchover for a site with respect to remote site and group ID. Service Type: Realtime. This API provides real time data.	PUT	NA	200 OK 404 Not Found 500 Internal Server Error	200 OK: <pre>{ "replicationSwitchOver": "start" }</pre> 404 Not Found: <pre>{ "title": "Not Found", "status": 404, "detail": "<Exception details>", "cause": <cause>, "invalidParams": <params> }</pre> 500 Internal Server Error: <pre>{ "title": "Internal Server Error", "status": 500, "detail": "<Exception details>", "cause": <cause>, "invalidParams": <params> }</pre>

Table 4-5 (Cont.) cnDBTier Switchover APIs

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
http://<base-uri>/ocdbtier/georeplication/switchover/stop/{siteName}/remotesitenam/{remoteSiteName}/replchannelgroupid/{replChannelGroupid}	This API is used to stop replication switchover for a site with respect to remote site and group ID. Service Type: Realtime. This API provides real time data.	PUT	NA	200 OK 404 Not Found 500 Internal Server Error	200 OK: <pre>{ "replicationSwitchOver": "stop" }</pre> 404 Not Found: <pre>{ "title": "Not Found", "status": 404, "detail": "<Exception details>", "cause": <cause>, "invalidParams": <params> }</pre> 500 Internal Server Error: <pre>{ "title": "Internal Server Error", "status": 500, "detail": "<Exception details>", "cause": <cause>, "invalidParams": <params> }</pre>

Table 4-5 (Cont.) cnDBTier Switchover APIs

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
http://<base-uri>/ocdbtier/georeplication/switchover/start/{siteName}/remotesitenam/{remoteSiteName}	This API is used to start replication switchover for a site with respect to remote site name. Service Type: Realtime. This API provides real time data.	PUT	NA	200 OK 404 Not Found 500 Internal Server Error	200 OK: <pre>{ "replicationSwitchOver": "start" }</pre> 404 Not Found: <pre>{ "title": "Not Found", "status": 404, "detail": "<Exception details>", "cause": <cause>, "invalidParams": <params> }</pre> 500 Internal Server Error: <pre>{ "title": "Internal Server Error", "status": 500, "detail": "<Exception details>", "cause": <cause>, "invalidParams": <params> }</pre>

Table 4-5 (Cont.) cnDBTier Switchover APIs

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
http://<base-uri>/ocdbtier/georeplication/switchover/stop/{siteName}/remotesitenam/{remoteSiteName}	This API is used to stop replication switchover for a site with respect to remote site name. Service Type: Realtime. This API provides real time data.	PUT	NA	200 OK 404 Not Found 500 Internal Server Error	200 OK: <pre>{ "replicationSwitchOver": "stop" }</pre> 404 Not Found: <pre>{ "title": "Not Found", "status": 404, "detail": "<Exception details>", "cause": <cause>, "invalidParams": <params> }</pre> 500 Internal Server Error: <pre>{ "title": "Internal Server Error", "status": 500, "detail": "<Exception details>", "cause": <cause>, "invalidParams": <params> }</pre>

Table 4-5 (Cont.) cnDBTier Switchover APIs

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
http://<base-uri>/ocdbtier/georeplication/switchover/start/siteName/{siteName}	This API is used to start replication switchover with respect to site name. Service Type: Realtime. This API provides real time data.	PUT	NA	200 OK 404 Not Found 500 Internal Server Error	200 OK: <pre>{ "replicationSwitchOver": "start" }</pre> 404 Not Found: <pre>{ "title": "Not Found", "status": 404, "detail": "<Exception details>", "cause": <cause>, "invalidParams": <params> }</pre> 500 Internal Server Error: <pre>{ "title": "Internal Server Error", "status": 500, "detail": "<Exception details>", "cause": <cause>, "invalidParams": <params> }</pre>

Table 4-5 (Cont.) cnDBTier Switchover APIs

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
http://<base-uri>/ocdbtier/georeplication/switchover/stop/{sitename}/{siteName}	This API is used to stop replication switchover for a site with respect to site name. Service Type: Realtime. This API provides real time data.	PUT	NA	200 OK 404 Not Found 500 Internal Server Error	200 OK: <pre>{ "replicationSwitchOver": "stop" }</pre> 404 Not Found: <pre>{ "title": "Not Found", "status": 404, "detail": "<Exception details>", "cause": <cause>, "invalidParams": <params> }</pre> 500 Internal Server Error: <pre>{ "title": "Internal Server Error", "status": 500, "detail": "<Exception details>", "cause": <cause>, "invalidParams": <params> }</pre>

Note

The value of <base-uri> in the REST URL is <db-replication-svc Loadbalancer IP>: <db-replication-svc Loadbalancer PORT>.

Depending on your scenario, choose one of the following methods to obtain the LoadBalancer IP and LoadBalancer Port of the replication service in a cnDBTier cluster:

- When HTTP is enabled in a cnDBTier cluster:

- Run the following command to get the LoadBalancer IP of the replication service for cluster1:

```
$ IP=$(kubectl get svc -n cluster1 | grep repl | awk
'{print $4}' | head -n 1 )
```

- Run the following commands to get the LoadBalancer Port of the replication service for cluster1:

```
$ PORT=$(kubectl get svc -n cluster1 | grep repl | awk
'{print $5}' | cut -d '/' -f 1 | cut -d ':' -f 1 | head -n 1)
```

- Sample command to call the REST API:

```
$ curl -X PUT http://$IP:$PORT/ocdbtier/georeplication/
switchover/start/siteName/{siteName}
```

- When HTTPS is enabled in a cnDBTier cluster:

- Obtain the client certificate which is present in the secret configured in "/global/https/secrets/clientSecret". For example, client-cert.pem and client-key.pem.

- Obtain the CA certificate which is present in the secret configured by user in "/global/https/secrets/caSecret" for example, combine-ca.pem.

- Run the following command to get the LoadBalancer IP of the replication service for cluster1:

```
$ IP=$(kubectl get svc -n cluster1 | grep repl | awk
'{print $4}' | head -n 1 )
```

- Run the following commands to get the LoadBalancer Port of the replication service for cluster1:

```
$ PORT=$(kubectl get svc -n cluster1 | grep repl | awk
'{print $5}' | cut -d '/' -f 1 | cut -d ':' -f 1 | head -n 1)
```

- Sample command to call the REST API:

```
$ curl --cert client-cert.pem --cert-type PEM --key
client.key.pem --key-type PEM --cacert combine-ca.pem -X
https://$IP:$PORT/ocdbtier/georeplication/switchover/start/
siteName/{siteName}
```


4.4.2 cnDBTier Stop Replica APIs

cnDBTier Stop Replica APIs are used to stop a replication channel.

Table 4-6 cnDBTier Stop Replica APIs

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
http://<base-uri>/ocdbtier/georeplication/stopreplica/{siteName}/remotesitename/{remoteSiteName}/replchannelgroupid/{replChannelGroupid}	This API is used to stop replication on a site with respect to remote site name and replication channel group ID. Service Type: Realtime. This API provides real time data.	PUT	NA	200 OK 404 Not Found 500 Internal Server Error	200 OK: <pre>{ "stopReplica": "stop" }</pre> 404 Not Found: <pre>{ "title": "Not Found", "status": 404, "detail": "<Exception details>", "cause": <cause>, "invalidParams": <params> }</pre> 500 Internal Server Error: <pre>{ "title": "Internal Server Error", "status": 500, "detail": "<Exception details>", "cause": <cause>, "invalidParams": <params> }</pre>

Table 4-6 (Cont.) cnDBTier Stop Replica APIs

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
http://<base-uri>/ocdbtier/georeplication/stopreplica/siteName/{siteName}/remotesitenam/{remoteSiteName}	This API is used to stop replication on a site with respect to remote site names. Service Type: Realtime. This API provides real time data.	PUT	NA	200 OK 404 Not Found 500 Internal Server Error	200 OK: <pre>{ "stopReplica": "stop" }</pre> 404 Not Found: <pre>{ "title": "Not Found", "status": 404, "detail": "<Exception details>", "cause": <cause>, "invalidParams": <params> }</pre> 500 Internal Server Error: <pre>{ "title": "Internal Server Error", "status": 500, "detail": "<Exception details>", "cause": <cause>, "invalidParams": <params> }</pre>

Table 4-6 (Cont.) cnDBTier Stop Replica APIs

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
http://<base-uri>/ocdbtier/georeplication/stopreplica/siteName/{siteName}	This API is used to stop replica on a site with respect to site name. Service Type: Realtime. This API provides real time data.	PUT	NA	200 OK 404 Not Found 500 Internal Server Error	200 OK: <pre>{ "stopReplica": "stop" }</pre> 404 Not Found: <pre>{ "title": "Not Found", "status": 404, "detail": "<Exception details>", "cause": <cause>, "invalidParams": <params> }</pre> 500 Internal Server Error: <pre>{ "title": "Internal Server Error", "status": 500, "detail": "<Exception details>", "cause": <cause>, "invalidParams": <params> }</pre>

Note

- This section provides only the definition of the Stop Replica API. If you want to stop the replication between the sites, follow the procedure given in the "Stopping cnDBTier Georeplication Service" section in *Oracle Communications Cloud Native Core, cnDBTier User Guide*.
- The value of <base-uri> in the REST URL is <db-replication-svc Loadbalancer IP>: <db-replication-svc Loadbalancer PORT>.

Depending on your scenario, choose one of the following methods to obtain the LoadBalancer IP and LoadBalancer Port of the replication service in a cnDBTier cluster:

- When HTTP is enabled in a cnDBTier cluster:
 - Run the following command to get the LoadBalancer IP of the replication service for cluster1:


```
$ IP=$(kubectl get svc -n cluster1 | grep repl | awk '{print $4}' | head -n 1 )
```
 - Run the following commands to get the LoadBalancer Port of the replication service for cluster1:


```
$ PORT=$(kubectl get svc -n cluster1 | grep repl | awk '{print $5}' | cut -d '/' -f 1 | cut -d ':' -f 1 | head -n 1)
```
 - Sample command to call the REST API:


```
$ curl -X PUT http://$IP:$PORT/ocdbtier/georeplication/stopreplica/siteName/{siteName}
```
- When HTTPS is enabled in a cnDBTier cluster:
 - Obtain the client certificate which is present in the secret configured in "/global/https/secrets/clientSecret". For example, client-cert.pem and client-key.pem.
 - Obtain the CA certificate which is present in the secret configured in "/global/https/secrets/caSecret". For example, combine-ca.pem.
 - Run the following command to get the LoadBalancer IP of the replication service for cluster1:


```
$ IP=$(kubectl get svc -n cluster1 | grep repl | awk '{print $4}' | head -n 1 )
```
 - Run the following commands to get the LoadBalancer Port of the replication service for cluster1:


```
$ PORT=$(kubectl get svc -n cluster1 | grep repl | awk '{print $5}' | cut -d '/' -f 1 | cut -d ':' -f 1 | head -n 1)
```
 - Sample command to call the REST API:


```
$ curl --cert client-cert.pem --cert-type PEM --key client.key.pem --key-type PEM -cacert combine-ca.pem -X PUT
```

```
https://$IP:$PORT/ocdbtier/georeplication/stopreplica/sitename/  
{siteName}
```

4.4.3 cnDBTier Listbackups API

cnDBTier Listbackups API is used to determine the backups present in the cnDBTier. The cnDBTier Listbackups API is hosted by the DB Replication service.

Table 4-7 cnDBTier Listbackups API

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
http://<base-uri>/db-tier/backup/site/{siteName}/listbackups	This API is used to list site-specific backups. Service Type: Realtime. This API provides real time data.	GET	NA	200 OK - Fetched the list of backups present in cnDBTier. 404 Not Found - Improper Site details or missing table. 503 INTERNAL SERVER ERROR - Could not list the backups.	200 OK: <pre> { "localSiteName": "<current_site_name>", "backupDetails": [{ "backupId": "<backup>", "backupSize": "<size of backup>", "creationTimeStamp": "<timestamp at which backup was initiated>" }, { "backupId": "<backup_id>", "backupSize": "<size of backup>", "creationTimeStamp": "<timestamp at which backup was initiated>" }, { "backupId": "<backup_id>", "backupSize": "<size of backup>", "creationTimeStamp": "<timestamp at which backup was initiated>" }] } </pre> 404 Not Found: The response payload can be one of the following: - Site Name is null so cannot list the backup

Table 4-7 (Cont.) cnDBTier Listbackups API

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
					– Site: <siteName> doesn't match with the current site Name so cannot list the backup – DBTIER_BACKUP_INFO table doesn't exist in the current site so cannot list the backup • 503 INTERNAL SERVER ERROR: Could not list the backups, <Exception Message>

Note

The value of <base-uri> in the REST URL is <db-replication-svc Loadbalancer IP>: <db-replication-svc Loadbalancer PORT>.

Depending on your scenario, refer to the following points to obtain the LoadBalancer IP and LoadBalancer Port of a cnDBTier cluster:

- When HTTP is enabled in a cnDBTier cluster:
 - Run the following command to get the replication service LoadBalancer IP for cluster1:

```
$ IP=$(kubectl get svc -n cluster1 | grep repl | awk
'{print $4}' | head -n 1 )
```

- Run the following commands to get the replication service LoadBalancer Port for cluster1:

```
$ PORT=$(kubectl get svc -n cluster1 | grep repl | awk
'{print $5}' | cut -d '/' -f 1 | cut -d ':' -f 1 | head -n 1)
```

- Sample command to call the REST API:

```
$ curl -i -X GET http://$IP:$PORT/db-tier/backup/site/cluster1/
listbackups
```

- When HTTPS is enabled in a cnDBTier cluster:
 - Obtain the client certificate which is present in the secret configured in "/global/https/secrets/clientSecret". For example, client-cert.pem and client-key.pem.
 - Obtain the CA certificate which is present in the secret configured in "/global/https/secrets/caSecret". For example, combine-ca.pem.
 - Run the following command to get the replication service LoadBalancer IP for cluster1:

```
$ IP=$(kubectl get svc -n cluster1 | grep repl | awk
'{print $4}' | head -n 1 )
```

- Run the following commands to get the replication service LoadBalancer Port for cluster1:

```
$ PORT=$(kubectl get svc -n cluster1 | grep repl | awk
'{print $5}' | cut -d '/' -f 1 | cut -d ':' -f 1 | head -n 1)
```

- Sample command to call the REST API:

```
$ curl --cert client-cert.pem --cert-type PEM --key
client.key.pem --key-type PEM -cacert combine-ca.pem -X PUT
https://$IP:$PORT/ocdbtier/georeplication/stopreplica/sitename/
{siteName}
```


4.4.4 cnDBTier Replication Service Heartbeat API

cnDBTier Replication Service Heartbeat API is used to provide the Heartbeat statuses of all replication services running on the current cluster.

Table 4-8 cnDBTier Replication Service Heartbeat API

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
http://<base-uri>/db-tier/status/db-replication-svc/realtime	This API is used to list site-specific heartbeat status. Service Type: Realtime. This API provides real time data.	GET	NA	200 OK - Fetched the heartbeat status of each replication service running on the current cluster. 503 SERVICE_UNAVAILABLE - Not able to query the database 503 INTERNAL SERVER ERROR - Could not check the replication service status.	200 OK: <pre> { "localSiteName": "<current_site_name>", "heartBeatDetails": [{ "remoteSiteName": "<remote_site_name>", "heartBeatStatus": "<SUCCESS/FAILURE>", "heartBeatLag": "<Heart Beat Lag in Seconds>", "replicationChannelGroupId": "<replication channel group id>" }, { "remoteSiteName": "<remote_site_name>", "heartBeatStatus": "<SUCCESS/FAILURE>", "heartBeatLag": "<Heart Beat Lag in Seconds>", "replicationChannelGroupId": "<replication channel group id>" }] } </pre> 503 SERVICE_UNAVAILABLE: Not able to query the Data Base 500 INTERNAL_SERVER_ERROR : Could not check the replication service

Table 4-8 (Cont.) cnDBTier Replication Service Heartbeat API

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
					status, <Exception Message>

Note

The value of <base-uri> in the REST URL is <db-replication-svc Loadbalancer IP>: <db-replication-svc Loadbalancer PORT>.

Depending on your scenario, refer to the following points to obtain the LoadBalancer IP and LoadBalancer Port of a cnDBTier cluster:

- When HTTP is enabled in a cnDBTier cluster:
 - Run the following command to get the replication service LoadBalancer IP for cluster1:

```
$ IP=$(kubectl get svc -n cluster1 | grep repl | awk
'{print $4}' | head -n 1 )
```

- Run the following commands to get the replication service LoadBalancer Port for cluster1:

```
$ PORT=$(kubectl get svc -n cluster1 | grep repl | awk
'{print $5}' | cut -d '/' -f 1 | cut -d ':' -f 1 | head -n 1)
```

- Sample command to call the REST API:

```
$ curl -i -X GET http://$IP:$PORT/db-tier/status/db-replication-
svc/realtime
```

- When HTTPS is enabled in a cnDBTier cluster:
 - Obtain the Client certificate which is present in the secret configured by user in "/global/https/secrets/clientSecret". For Ex - client-cert.pem and client-key.pem
 - Obtain the CA certificate which is present in the secret configured in "/global/https/secrets/caSecret". For example, combine-ca.pem.
 - Run the following command to get the replication service LoadBalancer IP for cluster1:

```
$ IP=$(kubectl get svc -n cluster1 | grep repl | awk
'{print $4}' | head -n 1 )
```

- Run the following commands to get the replication service LoadBalancer Port for cluster1:

```
$ PORT=$(kubectl get svc -n cluster1 | grep repl | awk
'{print $5}' | cut -d '/' -f 1 | cut -d ':' -f 1 | head -n 1)
```

- Sample command to call the REST API:

```
$ curl --cert client-cert.pem --cert-type PEM --key
client.key.pem --key-type PEM -cacert combine-ca.pem -X PUT
https://$IP:$PORT/ocdbtier/georeplication/stopreplica/siteName/
{siteName}
```

4.5 cnDBTier Status APIs

cnDBTier Status APIs enable the NF applications to determine the status of cnDBTier and the associated cross-site replication. The DB Monitor service hosts the cnDBTier Status APIs.

Table 4-9 cnDBTier Status API

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
http://<base-uri>/db-tier/status/replication/{mate-site-name}/realtime	This API is used to get site-specific replication status. Service Type: Realtime. This API provides real time data.	GET	NA	200 OK - Cross-site replication is enabled. 400 Bad Request. 404 Not Found - The indicated mate site has not been configured. 503 Service Unavailable - Cross-site replication is not enabled and/or DOWN.	200 OK: <pre> { "replicationStatus": "Up" "secondsBehindRemote": "<greater_secondsBehindRemote_of_multiplegroups_withremotesite>" "replicationGroupDelay": [{ replchannel_group_id: 1, secondsBehindRemote: <seconds> }, { replchannel_group_id: 2, secondsBehindRemote: <seconds> }] "siteDetails": [{ remote_replication_ip: 127.0.0.1, role: ACTIVE }, { remote_replication_ip: 127.0.0.2, role: STANDBY }, { remote_replication_ip: 127.0.0.2,</pre>

Table 4-9 (Cont.) cnDBTier Status API

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
					<pre> role: ACTIVE }, { remote_replication_ip: 127.0.0.3, role: STANDBY }] } </pre> • 404 Not Found: NA • 503 Service Unavailable: NA

Table 4-9 (Cont.) cnDBTier Status API

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
http://<base-uri>/db-tier/status/replication/realtime	This API is used to get the overall replication status. Service Type: Realtime. This API provides real time data.	GET	NA	200 OK - Cross-site replication is enabled. 503 Service Unavailable - Cross-site replication is not enabled and/or DOWN.	200 OK: <pre> { [{ "localSiteName": "<current-site-name>", "remoteSiteName": "<remote-site-name>", "replicationStatus": "UP/DOWN/INITIALIZING" "secondsBehindRemote": "<greater_secondsBehindRemote_of_multiplegroups_withremotesite>" "replicationGroupDelay": [{ replchannel_group_id: 1, secondsBehindRemote: <seconds> }, { replchannel_group_id: 2, secondsBehindRemote: <seconds> }], { "localSiteName": "<current-site-name>", "remoteSiteName": "<remote-site-name>" }] }] } </pre>

Table 4-9 (Cont.) cnDBTier Status API

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
					<pre> "replicationStatus": "UP/DOWN/INITIALIZING" "secondsBehindRemote": "<greater_secondsBehindRemote_of_multiplegroups_withremotesite>" "replicationGroupDelay": [{ replchannel_group_id: 1, secondsBehindRemote: <seconds> }, { replchannel_group_id: 2, secondsBehindRemote: <seconds> }] } </pre> 503 Service Unavailable: NA
http://<base-uri>/db-tier/status/local	This API is used to get the local cluster status. Service Type: Cached. This API doesn't provide real time data. This API is deprecated and will be removed in a future release.	GET	NA	200 OK - The local NDB cluster is available. 503 Service Unavailable - The local NDB cluster is not available.	NA

Table 4-9 (Cont.) cnDBTier Status API

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
http://<base-uri>/db-tier/status/replication/{mate-site-name}	This API is used to get site-specific replication status. Service Type: Cached. This API doesn't provide real time data. This API is deprecated and will be removed in a future release.	GET	NA	200 OK - Cross-site replication is enabled. 400 Bad Request. 404 Not Found - The indicated mate site has not been configured. 503 Service Unavailable - Cross-site replication is not enabled and/or DOWN.	200 OK: <pre> { "replicationStatus": "UP/DOWN/INITIALIZING", "siteDetails":[{ "remote_replication_ip": "<ip>", "role": "ACTIVE/STANDBY/FAILED" }, { "remote_replication_ip": "<ip>", "role": "STANDBY/STANDBY/FAILED" }] } </pre> 404 Not Found: NA 503 Service Unavailable: NA

Table 4-9 (Cont.) cnDBTier Status API

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
http://<base-uri>/db-tier/status/replication	This API is used to get the overall replication status. Service Type: Cached. This API doesn't provide real time data. This API is deprecated and will be removed in a future release.	GET	NA	200 OK - Cross-site replication is enabled. 503 Service Unavailable - Cross-site replication is not enabled and/or DOWN.	200 OK: <pre> { [{ "localSiteName": "<current-site-name>", "remoteSiteName": "<remote-site-name>", "replicationStatus": "UP/DOWN/INITIALIZING" }, { "localSiteName": "<current-site-name>", "remoteSiteName": "<remote-site-name>", "replicationStatus": "UP/DOWN/INITIALIZING" }] }</pre> 503 Service Unavailable: NA
http://<base-uri>/db-tier/status/cluster/local/realtime	This API is used to get the local cluster status. Service Type: Realtime. This API provides real time data.	GET	NA	200 OK - The local NDB cluster is available. 503 Service Unavailable - The local NDB cluster is not available.	NA

Table 4-9 (Cont.) cnDBTier Status API

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
http://<base-uri>/db-tier/statistics/dbinfo	This API is used to get database statistics reports. Service Type: Realtime. This API provides real time data.	GET	NA	200 OK - Data is available. 503 Service Unavailable - DB statistics report is empty. 500 INTERNAL SERVER ERROR - Could not list DB statistics reports	200 OK: <pre> { "databaseCount": <database count>, "databaseTablesCount": [{ "dbName": "<database name>", "tableCount": <table count> }, { "dbName": "<database name>", "tableCount": <table count> }], "databaseTableRowsCount": [{ "dbName": "<database name>", "tables": [{ "tableName": "<table name>", "rowCount": <row count> }], { "dbName": "<database name>", "tables": [{ "tableName": "<table name>", "rowCount": <row count> }] }] </pre>

Table 4-9 (Cont.) cnDBTier Status API

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
					<pre> }, { "tableName": "<table name>", "rowCount": <row count> }] }] } </pre> 503 Service Unavailable: NA 500 INTERNAL SERVER ERROR: NA
http://<base-uri>/db-tier/version	This API is used to get the cnDBTier version. Service Type: Realtime. This API provides real time data.	GET	NA	200 OK - cnDBTier version retrieved successfully. 500 INTERNAL SERVER ERROR - failed to get the cnDBTier version.	200 OK: <pre> { "cndbtier_version": "23.2.0", "ndb_version": "ndb-8.4.3" } </pre> 500 INTERNAL SERVER ERROR: "failed to get version"

Table 4-9 (Cont.) cnDBTier Status API

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
<code>http://base-uri/db-tier/replication/status/realtime</code>	This API is used to get realtime replication status across all sites. Service Type: Realtime. This API provides real time data.	GET	NA	200 OK - Cross-site replication is enabled. 400 Bad Request. 404 Not Found - The indicated mate site has not been configured. 503 Service Unavailable - Cross-site replication is not enabled and/or DOWN.	200 OK: <pre>[{ "siteNameAndRemoteSiteName": "ses3,ses1", "siteName": "ses3", "remoteSiteName": "ses1", "status": "UP", "replicationDetailsPerGroupId": [{ "replicationGroupId": 1, "status": "UP", "allSqlStatusDetails": [{ "sqlNode": "ndbmysqld-0", "sourceHost": "10.233.29.41", "externalIp": "10.233.35.195", "replicaIORunning": "Yes", "replicaSQLRunning": "Yes", "serverId": 1002, "role": "ACTIVE", "lastErrno": 0, "lastError": "",</pre>

Table 4-9 (Cont.) cnDBTier Status API

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
					<pre> "lastIOErrno": 0, "lastIOError": "", "lastSQLErrno": 0, "lastSQLError": "", "sqlRemainingDelay": null, "sourceSqlStatusDetail s": { "sqlNode": "ndbmysqld-2", "sourceHost": "10.233.35.195", "externalIp": "10.233.29.41", "replicaIORunning": "Yes", "replicaSQLRunning": "Yes", "serverId": 3000, "role": "ACTIVE", "lastErrno": 0, "lastError": "", "lastIOErrno": 0, "lastIOError": "", "lastSQLErrno": 0, "lastSQLError": "", "sqlRemainingDelay": null </pre>

Table 4-9 (Cont.) cnDBTier Status API

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
					<pre> } }, { "sqlNode": "ndbmysqld-1", "sourceHost": "10.233.50.242", "externalIp": "10.233.35.129", "replicaIORunning": "No", "replicaSQLRunning": "No", "serverId": 1003, "role": "STANDBY", "lastErrno": 0, "lastError": "", "lastIOErrno": 0, "lastIOError": "", "lastSQLErrno": 0, "lastSQLError": "", "sqlRemainingDelay": null, "sourceSqlStatusDetails": { "sqlNode": "ndbmysqld-3", "sourceHost": "10.233.35.129", "externalIp": </pre>

Table 4-9 (Cont.) cnDBTier Status API

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
					<pre> "10.233.50.242", "replicaIORunning": "No", "replicaSQLRunning": "No", "serverId": 3001, "role": "STANDBY", "lastErrno": 0, "lastError": "", "lastIOErrno": 0, "lastIOError": "", "lastSQLErrno": 0, "lastSQLError": "", "sqlRemainingDelay": null } }] }, { "siteNameAndRemoteSite Name": "ses3,ses2", "siteName": "ses3", "remoteSiteName": "ses2", "status": "UP", "replicationDetailsPer GroupId": [{ "replicationGroupId": 1, </pre>

Table 4-9 (Cont.) cnDBTier Status API

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
					<pre> "status": "UP", "allSqlStatusDetails": [{ "sqlNode": "ndbmysqld-2", "sourceHost": "10.233.20.84", "externalIp": "10.233.9.23", "replicaIORunning": "Yes", "replicaSQLRunning": "Yes", "serverId": 2002, "role": "ACTIVE", "lastErrno": 0, "lastError": "", "lastIOErrno": 0, "lastIOError": "", "lastSQLErrno": 0, "lastSQLError": "", "sqlRemainingDelay": null, "sourceSqlStatusDetails": { "sqlNode": "ndbmysqld-2", "sourceHost": </pre>

Table 4-9 (Cont.) cnDBTier Status API

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
					<pre> "10.233.9.23", "externalIp": "10.233.20.84", "replicaIORunning": "Yes", "replicaSQLRunning": "Yes", "serverId": 3002, "role": "ACTIVE", "lastErrno": 0, "lastError": "", "lastIOErrno": 0, "lastIOError": "", "lastSQLErrno": 0, "lastSQLError": "", "sqlRemainingDelay": null } }, { "sqlNode": "ndbmysqld-3", "sourceHost": "10.233.29.205", "externalIp": "10.233.50.79", "replicaIORunning": "No", "replicaSQLRunning": "No", </pre>

Table 4-9 (Cont.) cnDBTier Status API

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
					<pre> "serverId": 2003, "role": "STANDBY", "lastErrno": 0, "lastError": "", "lastIOErrno": 0, "lastIOError": "", "lastSQLErrno": 0, "lastSQLError": "", "sqlRemainingDelay": null, "sourceSqlStatusDetail s": { "sqlNode": "ndbmysqld-3", "sourceHost": "10.233.50.79", "externalIp": "10.233.29.205", "replicaIORunning": "No", "replicaSQLRunning": "No", "serverId": 3003, "role": "STANDBY", "lastErrno": 0, "lastError": "", "lastIOErrno": 0, "lastIOError": "", </pre>

Table 4-9 (Cont.) cnDBTier Status API

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
					<pre> "lastSQLErrno": 0, "lastSQLError": "", "sqlRemainingDelay": null }] } }, { "siteNameAndRemoteSite Name": "ses2,ses3", "siteName": "ses2", "remoteSiteName": "ses3", "status": "UP", "replicationDetailsPer GroupId": [{ "replicationGroupId": 1, "status": "UP", "allSqlStatusDetails": [{ "sqlNode": "ndbmysqld-2", "sourceHost": "10.233.9.23", "externalIp": "10.233.20.84", "replicaIORunning": "Yes", </pre>

Table 4-9 (Cont.) cnDBTier Status API

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
					<pre> "replicaSQLRunning": "Yes", "serverId": 3002, "role": "ACTIVE", "lastErrno": 0, "lastError": "", "lastIOErrno": 0, "lastIOError": "", "lastSQLErrno": 0, "lastSQLError": "", "sqlRemainingDelay": null, "sourceSqlStatusDetail s": { "sqlNode": "ndbmysqld-2", "sourceHost": "10.233.20.84", "externalIp": "10.233.9.23", "replicaIORunning": "Yes", "replicaSQLRunning": "Yes", "serverId": 2002, "role": "ACTIVE", "lastErrno": 0, "lastError": "", </pre>

Table 4-9 (Cont.) cnDBTier Status API

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
					<pre> "lastIOErrno": 0, "lastIOError": "", "lastSQLErrno": 0, "lastSQLError": "", "sqlRemainingDelay": null } }, { "sqlNode": "ndbmysqld-3", "sourceHost": "10.233.50.79", "externalIp": "10.233.29.205", "replicaIORunning": "No", "replicaSQLRunning": "No", "serverId": 3003, "role": "STANDBY", "lastErrno": 0, "lastError": "", "lastIOErrno": 0, "lastIOError": "", "lastSQLErrno": 0, "lastSQLError": "", "sqlRemainingDelay": null, </pre>

Table 4-9 (Cont.) cnDBTier Status API

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
					<pre> "sourceSqlStatusDetails": { "sqlNode": "ndbmysqld-3", "sourceHost": "10.233.29.205", "externalIp": "10.233.50.79", "replicaIORunning": "No", "replicaSQLRunning": "No", "serverId": 2003, "role": "STANDBY", "lastErrno": 0, "lastError": "", "lastIOErrno": 0, "lastIOError": "", "lastSQLErrno": 0, "lastSQLError": "", "sqlRemainingDelay": null } }] }, { "siteNameAndRemoteSiteName": "ses1,ses2", "siteName": "ses1", </pre>

Table 4-9 (Cont.) cnDBTier Status API

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
					<pre> "remoteSiteName": "ses2", "status": "UP", "replicationDetailsPer GroupId": [{ "replicationGroupId": 1, "status": "UP", "allSqlStatusDetails": [{ "sqlNode": "ndbmysql-0", "sourceHost": "10.233.19.91", "externalIp": "10.233.31.67", "replicaIORunning": "Yes", "replicaSQLRunning": "Yes", "serverId": 2000, "role": "ACTIVE", "lastErrno": 0, "lastError": "", "lastIOErrno": 0, "lastIOError": "", "lastSQLErrno": 0, "lastSQLError": "", </pre>

Table 4-9 (Cont.) cnDBTier Status API

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
					<pre> "sqlRemainingDelay": null, "sourceSqlStatusDetail s": { "sqlNode": "ndbmysqld-0", "sourceHost": "10.233.31.67", "externalIp": "10.233.19.91", "replicaIORunning": "Yes", "replicaSQLRunning": "Yes", "serverId": 1000, "role": "ACTIVE", "lastErrno": 0, "lastError": "", "lastIOErrno": 0, "lastIOError": "", "lastSQLErrno": 0, "lastSQLError": "", "sqlRemainingDelay": null } }, { "sqlNode": "ndbmysqld-1", "sourceHost": "10.233.6.238", </pre>

Table 4-9 (Cont.) cnDBTier Status API

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
					<pre> "externalIp": "10.233.35.109", "replicaIORunning": "No", "replicaSQLRunning": "No", "serverId": 2001, "role": "STANDBY", "lastErrno": 0, "lastError": "", "lastIOErrno": 0, "lastIOError": "", "lastSQLErrno": 0, "lastSQLError": "", "sqlRemainingDelay": null, "sourceSqlStatusDetails": { "sqlNode": "ndbmysqld-1", "sourceHost": "10.233.35.109", "externalIp": "10.233.6.238", "replicaIORunning": "No", "replicaSQLRunning": "No", "serverId": 1001, </pre>

Table 4-9 (Cont.) cnDBTier Status API

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
					<pre> "role": "STANDBY", "lastErrno": 0, "lastError": "", "lastIOErrno": 0, "lastIOError": "", "lastSQLErrno": 0, "lastSQLError": "", "sqlRemainingDelay": null }] } }, { "siteNameAndRemoteSite Name": "ses1,ses3", "siteName": "ses1", "remoteSiteName": "ses3", "status": "UP", "replicationDetailsPer GroupId": [{ "replicationGroupId": 1, "status": "UP", "allSqlStatusDetails": [{ "sqlNode": "ndbmysqld-2", </pre>

Table 4-9 (Cont.) cnDBTier Status API

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
					<pre> "sourceHost": "10.233.35.195", "externalIp": "10.233.29.41", "replicaIORunning": "Yes", "replicaSQLRunning": "Yes", "serverId": 3000, "role": "ACTIVE", "lastErrno": 0, "lastError": "", "lastIOErrno": 0, "lastIOError": "", "lastSQLErrno": 0, "lastSQLError": "", "sqlRemainingDelay": null, "sourceSqlStatusDetails": { "sqlNode": "ndbmysqld-0", "sourceHost": "10.233.29.41", "externalIp": "10.233.35.195", "replicaIORunning": "Yes", "replicaSQLRunning": </pre>

Table 4-9 (Cont.) cnDBTier Status API

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
					<pre> "Yes", "serverId": 1002, "role": "ACTIVE", "lastErrno": 0, "lastError": "", "lastIOErrno": 0, "lastIOError": "", "lastSQLErrno": 0, "lastSQLError": "", "sqlRemainingDelay": null } }, { "sqlNode": "ndbmysqld-3", "sourceHost": "10.233.35.129", "externalIp": "10.233.50.242", "replicaIORunning": "No", "replicaSQLRunning": "No", "serverId": 3001, "role": "STANDBY", "lastErrno": 0, "lastError": "", "lastIOErrno": 0, </pre>

Table 4-9 (Cont.) cnDBTier Status API

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
					<pre> "lastIOError": "", "lastSQLErrno": 0, "lastSQLError": "", "sqlRemainingDelay": null, "sourceSqlStatusDetail s": { "sqlNode": "ndbmysql-1", "sourceHost": "10.233.50.242", "externalIp": "10.233.35.129", "replicaIORunning": "No", "replicaSQLRunning": "No", "serverId": 1003, "role": "STANDBY", "lastErrno": 0, "lastError": "", "lastIOErrno": 0, "lastIOError": "", "lastSQLErrno": 0, "lastSQLError": "", "sqlRemainingDelay": null } } </pre>

Table 4-9 (Cont.) cnDBTier Status API

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
					<pre>] }] }, { "siteNameAndRemoteSiteName": "ses2,ses1", "siteName": "ses2", "remoteSiteName": "ses1", "status": "UP", "replicationDetailsPerGroupId": [{ "replicationGroupId": 1, "status": "UP", "allSqlStatusDetails": [{ "sqlNode": "ndbmysqld-0", "sourceHost": "10.233.31.67", "externalIp": "10.233.19.91", "replicaIORunning": "Yes", "replicaSQLRunning": "Yes", "serverId": 1000, "role": "ACTIVE", "lastErrno": 0, </pre>

Table 4-9 (Cont.) cnDBTier Status API

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
					<pre> "lastError": "", "lastIOErrno": 0, "lastIOError": "", "lastSQLErrno": 0, "lastSQLError": "", "sqlRemainingDelay": null, "sourceSqlStatusDetail s": { "sqlNode": "ndbmysqld-0", "sourceHost": "10.233.19.91", "externalIp": "10.233.31.67", "replicaIORunning": "Yes", "replicaSQLRunning": "Yes", "serverId": 2000, "role": "ACTIVE", "lastErrno": 0, "lastError": "", "lastIOErrno": 0, "lastIOError": "", "lastSQLErrno": 0, "lastSQLError": "", "sqlRemainingDelay": </pre>

Table 4-9 (Cont.) cnDBTier Status API

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
					<pre> null } }, { "sqlNode": "ndbmysqld-1", "sourceHost": "10.233.35.109", "externalIp": "10.233.6.238", "replicaIORunning": "No", "replicaSQLRunning": "No", "serverId": 1001, "role": "STANDBY", "lastErrno": 0, "lastError": "", "lastIOErrno": 0, "lastIOError": "", "lastSQLErrno": 0, "lastSQLError": "", "sqlRemainingDelay": null, "sourceSqlStatusDetails": { "sqlNode": "ndbmysqld-1", "sourceHost": "10.233.6.238", </pre>

Table 4-9 (Cont.) cnDBTier Status API

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
					<pre> "externalIp": "10.233.35.109", "replicaIORunning": "No", "replicaSQLRunning": "No", "serverId": 2001, "role": "STANDBY", "lastErrno": 0, "lastError": "", "lastIOErrno": 0, "lastIOError": "", "lastSQLErrno": 0, "lastSQLError": "", "sqlRemainingDelay": null } }] }] </pre> 404 Not Found: NA 503 Service Unavailable: NA

Table 4-9 (Cont.) cnDBTier Status API

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
<code>http://base-uri/db-tier/replication/status/realtime/siteName/{siteName}</code>	This API is used to get current replication status of a given site and all mates sites in the cluster. Service Type: Realtime. This API provides real time data.	GET	NA	200 OK - Cross-site replication is enabled. 400 Bad Request. 404 Not Found - The indicated mate site has not been configured. 503 Service Unavailable - Cross-site replication is not enabled and/or DOWN.	200 OK: <pre>[{ "siteNameAndRemoteSiteName": "ses1,ses2", "siteName": "ses1", "remoteSiteName": "ses2", "status": "UP", "replicationDetailsPerGroupId": [{ "replicationGroupId": 1, "status": "UP", "allSqlStatusDetails": [{ "sqlNode": "ndbmysqld-0", "sourceHost": "10.233.19.91", "externalIp": "10.233.31.67", "replicaIORunning": "Yes", "replicaSQLRunning": "Yes", "serverId": 2000, "role": "ACTIVE", "lastErrno": 0, "lastError": "",</pre>

Table 4-9 (Cont.) cnDBTier Status API

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
					<pre> "lastIOErrno": 0, "lastIOError": "", "lastSQLErrno": 0, "lastSQLError": "", "sqlRemainingDelay": null, "sourceSqlStatusDetails": { "sqlNode": "ndbmysqld-0", "sourceHost": "10.233.31.67", "externalIp": "10.233.19.91", "replicaIORunning": "Yes", "replicaSQLRunning": "Yes", "serverId": 1000, "role": "ACTIVE", "lastErrno": 0, "lastError": "", "lastIOErrno": 0, "lastIOError": "", "lastSQLErrno": 0, "lastSQLError": "", "sqlRemainingDelay": null </pre>

Table 4-9 (Cont.) cnDBTier Status API

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
					<pre> } }, { "sqlNode": "ndbmysqld-1", "sourceHost": "10.233.6.238", "externalIp": "10.233.35.109", "replicaIORunning": "No", "replicaSQLRunning": "No", "serverId": 2001, "role": "STANDBY", "lastErrno": 0, "lastError": "", "lastIOErrno": 0, "lastIOError": "", "lastSQLErrno": 0, "lastSQLError": "", "sqlRemainingDelay": null, "sourceSqlStatusDetails": { "sqlNode": "ndbmysqld-1", "sourceHost": "10.233.35.109", "externalIp": </pre>

Table 4-9 (Cont.) cnDBTier Status API

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
					<pre> "10.233.6.238", "replicaIORunning": "No", "replicaSQLRunning": "No", "serverId": 1001, "role": "STANDBY", "lastErrno": 0, "lastError": "", "lastIOErrno": 0, "lastIOError": "", "lastSQLErrno": 0, "lastSQLError": "", "sqlRemainingDelay": null } }] }, { "siteNameAndRemoteSite Name": "ses1,ses3", "siteName": "ses1", "remoteSiteName": "ses3", "status": "UP", "replicationDetailsPer GroupId": [{ "replicationGroupId": 1, </pre>

Table 4-9 (Cont.) cnDBTier Status API

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
					<pre> "status": "UP", "allSqlStatusDetails": [{ "sqlNode": "ndbmysqld-2", "sourceHost": "10.233.35.195", "externalIp": "10.233.29.41", "replicaIORunning": "Yes", "replicaSQLRunning": "Yes", "serverId": 3000, "role": "ACTIVE", "lastErrno": 0, "lastError": "", "lastIOErrno": 0, "lastIOError": "", "lastSQLErrno": 0, "lastSQLError": "", "sqlRemainingDelay": null, "sourceSqlStatusDetails": { "sqlNode": "ndbmysqld-0", "sourceHost": </pre>

Table 4-9 (Cont.) cnDBTier Status API

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
					<pre> "10.233.29.41", "externalIp": "10.233.35.195", "replicaIORunning": "Yes", "replicaSQLRunning": "Yes", "serverId": 1002, "role": "ACTIVE", "lastErrno": 0, "lastError": "", "lastIOErrno": 0, "lastIOError": "", "lastSQLErrno": 0, "lastSQLError": "", "sqlRemainingDelay": null } }, { "sqlNode": "ndbmysqld-3", "sourceHost": "10.233.35.129", "externalIp": "10.233.50.242", "replicaIORunning": "No", "replicaSQLRunning": "No", </pre>

Table 4-9 (Cont.) cnDBTier Status API

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
					<pre> "serverId": 3001, "role": "STANDBY", "lastErrno": 0, "lastError": "", "lastIOErrno": 0, "lastIOError": "", "lastSQLErrno": 0, "lastSQLError": "", "sqlRemainingDelay": null, "sourceSqlStatusDetail s": { "sqlNode": "ndbmysqld-1", "sourceHost": "10.233.50.242", "externalIp": "10.233.35.129", "replicaIORunning": "No", "replicaSQLRunning": "No", "serverId": 1003, "role": "STANDBY", "lastErrno": 0, "lastError": "", "lastIOErrno": 0, "lastIOError": "", </pre>

Table 4-9 (Cont.) cnDBTier Status API

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
					<pre> { "lastSQLErrno": 0, "lastSQLError": "", "sqlRemainingDelay": null } </pre> 404 Not Found: NA 503 Service Unavailable: NA

Table 4-9 (Cont.) cnDBTier Status API

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
http://base-uri/db-tier/replication/status/realtime/siteName/{siteName}/remotesitenam/{remoteSiteName}	This API is used to get the current replication status of a given site and specific remote site. Service Type: Realtime. This API provides real time data.	GET	NA	200 OK - Cross-site replication is enabled. 400 Bad Request. 404 Not Found - The indicated mate site has not been configured. 503 Service Unavailable - Cross-site replication is not enabled and/or DOWN.	200 OK: <pre>[{ "siteNameAndRemoteSiteName": "ses1,ses2", "siteName": "ses1", "remoteSiteName": "ses2", "status": "UP", "replicationDetailsPerGroupId": [{ "replicationGroupId": 1, "status": "UP", "allSqlStatusDetails": [{ "sqlNode": "ndbmysqld-0", "sourceHost": "10.233.19.91", "externalIp": "10.233.31.67", "replicaIORunning": "Yes", "replicaSQLRunning": "Yes", "serverId": 2000, "role": "ACTIVE", "lastErrno": 0, "lastError": "",</pre>

Table 4-9 (Cont.) cnDBTier Status API

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
					<pre> "lastIOErrno": 0, "lastIOError": "", "lastSQLErrno": 0, "lastSQLError": "", "sqlRemainingDelay": null, "sourceSqlStatusDetail s": { "sqlNode": "ndbmysqld-0", "sourceHost": "10.233.31.67", "externalIp": "10.233.19.91", "replicaIORunning": "Yes", "replicaSQLRunning": "Yes", "serverId": 1000, "role": "ACTIVE", "lastErrno": 0, "lastError": "", "lastIOErrno": 0, "lastIOError": "", "lastSQLErrno": 0, "lastSQLError": "", "sqlRemainingDelay": null </pre>

Table 4-9 (Cont.) cnDBTier Status API

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
					<pre> } }, { "sqlNode": "ndbmysqld-1", "sourceHost": "10.233.6.238", "externalIp": "10.233.35.109", "replicaIORunning": "No", "replicaSQLRunning": "No", "serverId": 2001, "role": "STANDBY", "lastErrno": 0, "lastError": "", "lastIOErrno": 0, "lastIOError": "", "lastSQLErrno": 0, "lastSQLError": "", "sqlRemainingDelay": null, "sourceSqlStatusDetails": { "sqlNode": "ndbmysqld-1", "sourceHost": "10.233.35.109", "externalIp": </pre>

Table 4-9 (Cont.) cnDBTier Status API

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
					<pre> "10.233.6.238", "replicaIORunning": "No", "replicaSQLRunning": "No", "serverId": 1001, "role": "STANDBY", "lastErrno": 0, "lastError": "", "lastIOErrno": 0, "lastIOError": "", "lastSQLErrno": 0, "lastSQLError": "", "sqlRemainingDelay": null } }] </pre> • 404 Not Found: NA • 503 Service Unavailable: NA

Table 4-9 (Cont.) cnDBTier Status API

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
http://base-uri/ db-tier/ replication/ status/realtime/ sitename/ {sitename}/ remotesitename/ {remoteSiteName} e}/replgrouppid/ {replGrouppid}	This API is used to get the current replication status of a given site and specific remote site and group ID in the cluster. Service Type: Realtime. This API provides real time data.	GET	NA	200 OK - Cross-site replication is enabled. 400 Bad Request. 404 Not Found - The indicated mate site has not been configured. 503 Service Unavailable - Cross-site replication is not enabled and/or DOWN.	200 OK: <pre>[{ "siteNameAndRemoteSiteName": "ses1,ses2", "siteName": "ses1", "remoteSiteName": "ses2", "status": "UP", "replicationDetailsPerGroupId": [{ "replicationGroupId": 1, "status": "UP", "allSqlStatusDetails": [{ "sqlNode": "ndbmysqld-0", "sourceHost": "10.233.19.91", "externalIp": "10.233.31.67", "replicaIORunning": "Yes", "replicaSQLRunning": "Yes", "serverId": 2000, "role": "ACTIVE", "lastErrno": 0, "lastError": "",</pre>

Table 4-9 (Cont.) cnDBTier Status API

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
					<pre> "lastIOErrno": 0, "lastIOError": "", "lastSQLErrno": 0, "lastSQLError": "", "sqlRemainingDelay": null, "sourceSqlStatusDetail s": { "sqlNode": "ndbmysqld-0", "sourceHost": "10.233.31.67", "externalIp": "10.233.19.91", "replicaIORunning": "Yes", "replicaSQLRunning": "Yes", "serverId": 1000, "role": "ACTIVE", "lastErrno": 0, "lastError": "", "lastIOErrno": 0, "lastIOError": "", "lastSQLErrno": 0, "lastSQLError": "", "sqlRemainingDelay": null </pre>

Table 4-9 (Cont.) cnDBTier Status API

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
					<pre> } }, { "sqlNode": "ndbmysqld-1", "sourceHost": "10.233.6.238", "externalIp": "10.233.35.109", "replicaIORunning": "No", "replicaSQLRunning": "No", "serverId": 2001, "role": "STANDBY", "lastErrno": 0, "lastError": "", "lastIOErrno": 0, "lastIOError": "", "lastSQLErrno": 0, "lastSQLError": "", "sqlRemainingDelay": null, "sourceSqlStatusDetails": { "sqlNode": "ndbmysqld-1", "sourceHost": "10.233.35.109", "externalIp": </pre>

Table 4-9 (Cont.) cnDBTier Status API

Rest URL	Description	HTTP Method	Request Payload	Response Code	Response Payload
					<pre> "10.233.6.238", "replicaIORunning": "No", "replicaSQLRunning": "No", "serverId": 1001, "role": "STANDBY", "lastErrno": 0, "lastError": "", "lastIOErrno": 0, "lastIOError": "", "lastSQLErrno": 0, "lastSQLError": "", "sqlRemainingDelay": null } }] </pre> • 404 Not Found: NA • 503 Service Unavailable: NA

Note

The value of base-uri in the table is <db-monitor-svc service name>.<namespace>:8080.
For example: curl http://mysql-cluster-db-monitor-svc.occne-cndbtier:8080/db-tier/status/replication/realtime

5

cnDBTier Metrics

cnDBTier generates metrics to record or measure the specified values in cnDBTier. You can access the metrics using the Prometheus dashboard and take necessary actions. Prometheus gets installed as part of common services during the vCNE installation. This section provides details about the available cnDBTier metrics.

Dimensions Legend for Metrics

The following table provides information about metrics dimensions:

Table 5-1 Dimensions Legend

Dimension	Description
namespace	The name of the namespace in which the cnDBTier cluster is deployed.
site_name	The name of the site in which the cnDBTier cluster is deployed.
mate_site_name	The name of the remote site where replication is established.
node_id	The node ID of the database node.
remote_node_id	The node ID of the database node in the remote site.
node_type	The type of the database node. Sample values: Data node, Management node, SQL node
node_version	The version of MySQL NDB cluster software.
block_name	The name of the associated NDB kernel block. A kernel block is responsible for managing distinct operations. For more information about NDB kernel block, see SQL documentation . Sample Values: DBLQH, DBTC
block_instance	The instance number of the NDB kernel block.
counter_name	The name of the counter. Each counter is associated with a particular NDB kernel block. For more information about counters, see SQL documentation .
thr_no	The thread ID that is specific to a node.
thr_nm	The name of the thread. Sample Values: Idm, main, recv, rep
memory_type	Type of memory. Sample Values: Data memory, Index memory, Long message buffer
error_number	The error number for which the replication error is skipped. Sample Values: 13119, 1296, 1007, 1008, 1050, 1051.
service_name	Name of the cnDBTier microservice.

Table 5-1 (Cont.) Dimensions Legend

Dimension	Description
mount_path	The path within a container's file system where a volume is mounted. This path allows the container to access the contents of that volume.
hostname	The Fully Qualified Domain Name (FQDN). This value is provided in the following format: host-name.service-name.namespace.svc.domain_name. Sample value: - ndbmtid-0.ndbmtidsvc.occn-cnodbtier.svc.cluster.local - ndbmysqld-1.ndbmysqldsdc.occn-cnodbtier.svc.cluster.local
role	The role of the mysqld replication channel. Sample Values: active, standby
replchannel_group_id	The ID of the replication channel group.
channel_id	The ID of the mysqld replication channel.
source_node_ip	The IP address of the active replication SQL service in the remote site.
replica_node_ip	The IP address or hostname of the active replication SQL service in the local site.
backup_id	The ID of the backup.
status	The status of the backup operation. Sample values: STARTED, FAILED, COMPLETED, PURGED, PURGED_EARLY, BACKUP_TRANSFER_FAILED, BACKUP_TRANSFER_IN_PROGRESS, BACKUP_TRANSFER_COMPLETED
table_id	The unique ID of a table that is generated internally by NDB.
fq_name	The fully qualified name of the fragment. Fragments are the logical partitions of the data within a table. A fragment refers to a portion of a table's data that is distributed across multiple data nodes in a cluster. For more information, see SQL documentation .
parent_fq_name	The fully qualified fragment name for the parent or any fragment of the object.
remote_server_ip	The IP address of the remote server.

5.1 cnDBTier API Node Read and Write Metrics

This section provides details about the cnDBTier API node read and write metrics.

Table 5-2 db_tier_api_node_bytes_writes

Field	Details
Description	This metric provides information about the number of byte writes for ndbappmysqld pods and ndbmysqld pods.
Type	Gauge
Dimensions	namespace node_id hostname site_name

Table 5-3 db_tier_api_node_bytes_reads

Field	Details
Description	This metric provides information about the number of byte reads for ndbappmysqld pods and ndbmysqld pods.
Type	Gauge
Dimensions	namespace node_id hostname site_name

5.2 cnDBTier Remote Server Backup Transfer Status Metrics

This section provides details about the cnDBTier remote server backup transfer status metrics.

Table 5-4 db_tier_remote_server_backup_transfer_status

Field	Details
Description	Provides the status of backup transfer to a remote server. The possible values are: "1": indicates that the transfer of backup to remote server failed. "0": indicates that the transfer of backup to remote server completed. "2": indicates that the transfer of backup to remote server is in progress.
Type	Gauge
Dimensions	namespace site_name backup_id remote_server_ip status

5.3 cnDBTier Backup Transfer Status Metrics

This section provides details about the cnDBTier backup transfer status metrics.

Table 5-5 db_tier_backup_transfer_status

Field	Details
Description	Provides the status of the backup transfer process. The possible status values are: 0: indicates success 1: indicates in progress 2 and 3: indicates failed
Type	Gauge
Dimensions	site_name namespace backup_id status

5.4 cnDBTier Parallel Backup Transfer Progress Metrics

This section provides details about the cnDBTier parallel backup transfer progress metrics.

Table 5-6 db_tier_local_backup_transfer_progress

Field	Details
Description	Provides the percentage of the backup transfer that has completed from data node to replication service on a healthy backup site. This metric can be seen when a georeplication recovery is in progress on a failed site.
Type	Gauge
Dimensions	namespace node_id site_name

Table 5-7 db_tier_remote_backup_transfer_progress

Field	Details
Description	Provides the percentage of the backup transfer that has completed from the replication service of a backup site to the replication service of an unhealthy site. This metric can be seen when a georeplication recovery is in progress on a failed site.
Type	Gauge
Dimensions	namespace node_id site_name

5.5 cnDBTier Heartbeat Metrics

This section provides details about the cnDBTier heartbeat metrics.

Table 5-8 db_tier_heartbeat_failure

Field	Details
Description	Indicates the success or failure of heartbeat when trying to connect to a remote site. 0: indicates that Heartbeat is successful and cnDBTier is able to connect to the remote site. 1: indicates that Heartbeat failed and cnDBTier is unable to connect to the remote site.
Type	Gauge
Dimensions	site_name mate_site_name replchannel_group_id namespace

5.6 cnDBTier Node Status Metrics

The section provides details about the cnDBTier node status metrics.

Table 5-9 db_tier_node_status

Field	Details
Description	The status of the cnDBTier node. The possible values are: "0": indicates that the node is DOWN "1": indicates that the node is UP
Type	Gauge
Dimensions	node_id node_type node_version site_name namespace

Table 5-10 db_tier_cluster_status

Field	Details
Description	The status of the cnDBTier cluster. The possible values are: "0": indicates that the cluster is DOWN "1": indicates that the cluster is UP
Type	Gauge

Table 5-10 (Cont.) db_tier_cluster_status

Field	Details
Dimensions	site_name namespace

Table 5-11 db_tier_cluster_disconnect

Field	Details
Description	The disconnect status of the cnDBTier cluster. the value indicates the number of cluster disconnect incidents happened in the cluster.
Type	Gauge
Dimensions	site_name namespace

5.7 cnDBTier Table Read Write Metrics

The section provides details about the cnDBTier table read write metrics.

Table 5-12 db_tier_local_operations

Field	Details
Description	The total number of local operations in cnDBTier for a node.
Type	Gauge
Dimensions	node_id block_name block_instance counter_name site_name namespace

Table 5-13 db_tier_transactions

Field	Details
Description	The total number of transactions in cnDBTier for a node.
Type	Gauge
Dimensions	node_id block_name block_instance counter_name site_name namespace

Table 5-14 db_tier_commits

Field	Details
Description	The total number of commits in cnDBTier for a node.
Type	Gauge
Dimensions	node_id block_name block_instance counter_name site_name namespace

Table 5-15 db_tier_reads

Field	Details
Description	The total number of reads in cnDBTier for a node.
Type	Gauge
Dimensions	node_id block_name block_instance counter_name site_name namespace

Table 5-16 db_tier_local_reads

Field	Details
Description	The total number of local reads in cnDBTier for a node.
Type	Gauge
Dimensions	node_id block_name block_instance counter_name site_name namespace

Table 5-17 db_tier_writes

Field	Details
Description	The total number of writes in cnDBTier cluster for a node.
Type	Gauge

Table 5-17 (Cont.) db_tier_writes

Field	Details
Dimensions	node_id block_name block_instance counter_name site_name namespace

Table 5-18 db_tier_local_writes

Field	Details
Description	The total number of local writes in cnDBTier for a node.
Type	Gauge
Dimensions	node_id block_name block_instance counter_name site_name namespace

Table 5-19 db_tier_aborts

Field	Details
Description	The total number of aborted transactions in cnDBTier for a node.
Type	Gauge
Dimensions	node_id block_name block_instance counter_name site_name namespace

Table 5-20 db_tier_table_scans

Field	Details
Description	The total number of table scans in cnDBTier for a node.
Type	Gauge

Table 5-20 (Cont.) db_tier_table_scans

Field	Details
Dimensions	node_id block_name block_instance counter_name site_name namespace

Table 5-21 db_tier_range_scans

Field	Details
Description	The total number of range scans in cnDBTier for a node.
Type	Gauge
Dimensions	node_id block_name block_instance counter_name site_name namespace

Table 5-22 db_tier_transporter_overload

Field	Details
Description	The total number of transporter overload in cnDBTier for a node.
Type	Gauge
Dimensions	node_id block_name block_instance counter_name site_name namespace

Table 5-23 db_tier_scan_slowdown

Field	Details
Description	The total number of scan slowdowns in cnDBTier for a node.
Type	Gauge

Table 5-23 (Cont.) db_tier_scan_slowdown

Field	Details
Dimensions	node_id block_name block_instance counter_name site_name namespace

5.8 cnDBTier CPU Usage Metrics

This section provides details about the cnDBTier CPU usage metrics.

Table 5-24 db_tier_cpu_os_user

Field	Details
Description	Provides the CPU user statistics per thread for the specific node.
Type	Gauge
Dimensions	node_id thr_no site_name namespace

Table 5-25 db_tier_cpu_os_system

Field	Details
Description	Provides the CPU system statistics per thread for the specific node.
Type	Gauge
Dimensions	node_id thr_no site_name namespace

Table 5-26 db_tier_cpu_os_idle

Field	Details
Description	Provides the idle CPU statistics per thread for the specific node.
Type	Gauge
Dimensions	node_id thr_no site_name namespace

5.9 cnDBTier Memory Usage Metrics

This section provides details about the cnDBTier memory usage metrics.

Table 5-27 db_tier_memory_used_bytes

Field	Details
Description	Indicates the amount of memory used by the node in bytes.
Type	Gauge
Dimensions	node_id memory_type site_name namespace

Table 5-28 db_tier_memory_total_bytes

Field	Details
Description	Indicates the total memory assigned for the node in bytes.
Type	Gauge
Dimensions	node_id memory_type site_name namespace

5.10 cnDBTier Bin Log Usage Metrics

This section provides details about the cnDBTier binlog usage metrics.

Table 5-29 db_tier_binlog_used_bytes_percentage

Field	Details
Description	Indicates the percentage of total memory used by bin log in the SQL node.
Type	Gauge
Dimensions	node_id site_name namespace

5.11 cnDBTier Replication Metrics

This section provides details about the cnDBTier replication metrics.

Table 5-30 db_tier_replication_status

Field	Details
Description	Indicates the status of replication. The possible values are: "0": indicates that the replication channel status of the local site is OFF. "1": indicates that the replication channel status of the local site is ON. "2": indicates that the replication channel status of the local site is CONNECTING.
Type	Gauge
Dimensions	node_id role replchannel_group_id mate_site_name channel_id site_name namespace

Table 5-31 db_tier_replication_replica_delay

Field	Details
Description	Indicates the time (in seconds) by which the last record read by the replica is behind the latest record written by the source.
Type	Gauge
Dimensions	channel_id source_node_ip replica_node_ip mate_site_name site_name namespace

5.12 cnDBTier Automated Backup Metrics

This section provides details about the cnDBTier automated backup metrics.

Table 5-32 db_tier_backup_used_disk_percentage

Field	Details
Description	This metric provides the percentage of disk space used by the backup. This metric is pegged after the old backups are purged and a new one is created.
Type	Gauge
Dimensions	node_id site_name namespace

Table 5-33 db_tier_backup_size

Field	Details
Description	This metric provides the size of cnDBTier backup created. This metric is pegged only when a backup completes successfully.
Type	Gauge
Dimensions	backup_id node_id site_name namespace

Table 5-34 db_tier_backup

Field	Details
Description	This metric fetches the status of cnDBTier backup. This metric is pegged at each stage of a backup life cycle: on creation, when it fails or completes, and when it is deleted.
Type	Gauge
Dimensions	site_name namespace status

Table 5-35 db_tier_ndb_backup_in_progress

Field	Details
Description	This metric states if a data node backup is in progress in the current site. The possible values are: "0": Indicates that no database backup is in progress. "1": Indicates that cnDBTier backup is in progress.
Type	Gauge
Dimensions	site_name namespace

5.13 cnDBTier Fault Recovery State Metrics

This section provides details about the cnDBTier georeplication recovery state metrics.

Table 5-36 db_tier_georeplication_recovery_state

Field	Details
Description	Indicates if the current site is undergoing georeplication recovery.
Type	Gauge

Table 5-36 (Cont.) db_tier_georeplication_recovery_state

Field	Details
Dimensions	site_name (Name of the site that is undergoing georeplication recovery) namespace

Table 5-37 db_tier_ndb_restore_meta_progress

Field	Details
Description	Provides the completion percentage of the NDB metadata restore progress.
Type	Gauge
Dimensions	site_name (Name of the site that is undergoing georeplication restore) namespace

Table 5-38 db_tier_ndb_restore_data_progress

Field	Details
Description	Provides the completion percentage of the NDB data restore progress.
Type	Gauge
Dimensions	site_name (Name of the site that is undergoing georeplication restore) namespace node_id

5.14 cnDBTier Replica Status Metrics

This section provides details about the cnDBTier replica status metrics.

Table 5-39 db_tier_api_trans_commit_count

Field	Details
Description	The number of transactions committed by this replica.
Type	Gauge
Dimensions	node_id (of the DB node) site_name namespace

Table 5-40 db_tier_api_wait_exec_complete_count

Field	Details
Description	The number of times a thread has been blocked by this replica while waiting for execution of an operation to complete.
Type	Gauge
Dimensions	node_id (of the DB node) site_name namespace

Table 5-41 db_tier_api_bytes_sent_count

Field	Details
Description	The amount of data (in bytes) sent to the data nodes by this replica.
Type	Gauge
Dimensions	node_id (of the DB node) site_name namespace

Table 5-42 db_tier_api_pk_op_count

Field	Details
Description	The number of operations performed by this replica based on or using primary keys.
Type	Gauge
Dimensions	node_id (of the DB node) site_name namespace

5.15 cnDBTier ndbinfo Transporters Metrics

This section provides details about the cnDBTier ndbinfo transporters metrics.

Table 5-43 db_tier_node_transporter_bytes_sent

Field	Details
Description	The bytes sent from data node to other nodes.
Type	Gauge
Dimensions	node_id node_type remote_node_id site_name namespace

Table 5-44 db_tier_node_transporter_bytes_received

Field	Details
Description	The bytes received from data node to other nodes.
Type	Gauge
Dimensions	node_id node_type remote_node_id site_name namespace

Table 5-45 db_tier_node_transporter_overload_count

Field	Details
Description	Indicates the number of times the current connection has entered overload state since the start of the connection.
Type	Gauge
Dimensions	node_id node_type remote_node_id site_name namespace

Table 5-46 db_tier_node_transporter_slowdown_count

Field	Details
Description	Indicates the number of times the current connection has entered slowdown state since the start of the connecting.
Type	Gauge
Dimensions	node_id node_type remote_node_id site_name namespace

5.16 cnDBTier ndbinfo Threadstat Metrics

This section provides details about the cnDBTier ndbinfo thread stats metrics.

Table 5-47 db_tier_threadstat_os_time

Field	Details
Description	Indicates the OS time of the thread (ms).
Type	Gauge

Table 5-47 (Cont.) db_tier_threadstat_os_time

Field	Details
Dimensions	node_id, thr_no thr_nm site_name namespace

Table 5-48 db_tier_threadstat_os_user_cpu_time

Field	Details
Description	Indicates the OS CPU time taken by the user (µs).
Type	Gauge
Dimensions	node_id, thr_no thr_nm site_name namespace

Table 5-49 db_tier_threadstat_os_system_cpu_time

Field	Details
Description	Indicates the OS CPU time taken by the system (µs).
Type	Gauge
Dimensions	node_id, thr_no thr_nm site_name namespace

Table 5-50 db_tier_threadstat_os_voluntary_context_switches

Field	Details
Description	The number of OS voluntary context switches that happened.
Type	Gauge
Dimensions	node_id, thr_no thr_nm site_name namespace

Table 5-51 db_tier_threadstat_os_involuntary_context_switches

Field	Details
Description	The number of OS involuntary context switches that happened.
Type	Gauge
Dimensions	node_id, thr_no thr_nm site_name namespace

5.17 cnDBTier ndbinfo Operations Per Fragment Metrics

This section provides details about the cnDBTier ndbinfo operations per fragment metrics.

Note

cnDBTier ndbinfo operations per fragment metrics provide insights on the operations performed on individual fragments and their replicas within a database. For more information, see [MySQL documentation](#).

Table 5-52 db_tier_operations_per_fragment_tot_key_reads

Field	Details
Description	The total number of key reads.
Type	Gauge
Dimensions	table_id fq_name parent_fq_name site_name namespace

Table 5-53 db_tier_operations_per_fragment_tot_key_inserts

Field	Details
Description	The total number of key inserts.
Type	Gauge
Dimensions	table_id fq_name parent_fq_name site_name namespace

Table 5-54 db_tier_operations_per_fragment_tot_key_updates

Field	Details
Description	The total number of key updates.
Type	Gauge
Dimensions	table_id fq_name parent_fq_name site_name namespace

Table 5-55 db_tier_operations_per_fragment_tot_key_writes

Field	Details
Description	The total number of key writes.
Type	Gauge
Dimensions	table_id fq_name parent_fq_name site_name namespace

Table 5-56 db_tier_operations_per_fragment_tot_key_deletes

Field	Details
Description	The total number of key deletes.
Type	Gauge
Dimensions	table_id fq_name parent_fq_name site_name namespace

Table 5-57 db_tier_operations_per_fragment_tot_key_bytes_returned

Field	Details
Description	The total size of data and metadata returned from key read operations.
Type	Gauge
Dimensions	table_id fq_name parent_fq_name site_name namespace

Table 5-58 db_tier_operations_per_fragment_tot_frag_scans

Field	Details
Description	The total number of scans.
Type	Gauge
Dimensions	table_id fq_name parent_fq_name site_name namespace

Table 5-59 db_tier_operations_per_fragment_tot_scan_rows_returned

Field	Details
Description	The total number of rows returned to the client.
Type	Gauge
Dimensions	table_id fq_name parent_fq_name site_name namespace

Table 5-60 db_tier_operations_per_fragment_tot_scan_bytes_returned

Field	Details
Description	The total size of data and metadata returned to the client.
Type	Gauge
Dimensions	table_id fq_name parent_fq_name site_name namespace

Table 5-61 db_tier_operations_per_fragment_tot_qd_frag_scans

Field	Details
Description	The total number of times the scans were queued.
Type	Gauge
Dimensions	table_id fq_name parent_fq_name site_name namespace

Table 5-62 db_tier_operations_per_fragment_tot_commits

Field	Details
Description	The total number of row changes committed.
Type	Gauge
Dimensions	table_id fq_name parent_fq_name site_name namespace

Table 5-63 db_tier_operations_per_fragment_tot_scan_rows_examined

Field	Details
Description	The total number of rows examined.
Type	Gauge
Dimensions	table_id fq_name parent_fq_name site_name namespace

5.18 cnDBTier ndbinfo Locks Per Fragment Metrics

This section provides details about the cnDBTier ndbinfo locks per fragment metrics.

Note

cnDBTier ndbinfo locks per fragment metrics provide details about the number of lock claim requests and their outcomes for each fragment within a database. For more information, see [MySQL documentation](#).

Table 5-64 db_tier_locks_per_fragment_ex_req

Field	Details
Description	The number of exclusive lock requests started.
Type	Gauge
Dimensions	table_id fq_name parent_fq_name site_name namespace

Table 5-65 db_tier_locks_per_fragment_ex_imm_ok

Field	Details
Description	The number of exclusive lock requests granted immediately.
Type	Gauge
Dimensions	table_id fq_name parent_fq_name site_name namespace

Table 5-66 db_tier_locks_per_fragment_ex_wait_ok

Field	Details
Description	The number of exclusive lock requests granted following a wait.
Type	Gauge
Dimensions	table_id fq_name parent_fq_name site_name namespace

Table 5-67 db_tier_locks_per_fragment_ex_wait_fail

Field	Details
Description	The number non-granted exclusive lock requests.
Type	Gauge
Dimensions	table_id fq_name parent_fq_name site_name namespace

Table 5-68 db_tier_locks_per_fragment_sh_req

Field	Details
Description	The number of shared lock requests started.
Type	Gauge
Dimensions	table_id fq_name parent_fq_name site_name namespace

Table 5-69 db_tier_locks_per_fragment_sh_imm_ok

Field	Details
Description	The number of shared lock requests granted immediately.
Type	Gauge
Dimensions	table_id fq_name parent_fq_name site_name namespace

Table 5-70 db_tier_locks_per_fragment_sh_wait_ok

Field	Details
Description	The number of shared lock requests granted following a wait.
Type	Gauge
Dimensions	table_id fq_name parent_fq_name site_name namespace

Table 5-71 db_tier_locks_per_fragment_sh_wait_fail

Field	Details
Description	The number non-granted shared lock requests.
Type	Gauge
Dimensions	table_id fq_name parent_fq_name site_name namespace

Table 5-72 db_tier_locks_per_fragment_wait_ok_millis

Field	Details
Description	The waiting time of the granted lock requests (in milliseconds).
Type	Gauge

Table 5-72 (Cont.) db_tier_locks_per_fragment_wait_ok_millis

Field	Details
Dimensions	table_id fq_name parent_fq_name site_name namespace

Table 5-73 db_tier_locks_per_fragment_wait_fail_millis

Field	Details
Description	The waiting time of the failed lock requests (in milliseconds).
Type	Gauge
Dimensions	table_id fq_name parent_fq_name site_name namespace

5.19 cnDBTier ndbinfo Disk Write Speed Aggregate Metrics

This section provides details about the cnDBTier ndbinfo disk write speed aggregate metrics.

Table 5-74 db_tier_disk_write_speed_aggregate_backup_lcp_speed_last_60sec

Field	Details
Description	The number of bytes written to the disk by the backup and LCP processes per second (averaged over the last 60 seconds).
Type	Gauge
Dimensions	node_id thr_no site_name namespace

Table 5-75 db_tier_disk_write_speed_aggregate_redo_speed_last_60sec

Field	Details
Description	The number of bytes written to the REDO log per second (averaged over the last 60 seconds).
Type	Gauge

Table 5-75 (Cont.) db_tier_disk_write_speed_aggregate_redo_speed_last_60sec

Field	Details
Dimensions	node_id thr_no site_name namespace

Table 5-76 db_tier_disk_write_speed_aggregate_slowdowns_due_to_io_lag

Field	Details
Description	The number of seconds since the last node start the disk writes were slowed due to the REDO log I/O lag.
Type	Gauge
Dimensions	node_id thr_no site_name namespace

Table 5-77 db_tier_disk_write_speed_aggregate_slowdowns_due_to_high_cpu

Field	Details
Description	The number of seconds since the last node start the disk writes were slowed due to high CPU usage.
Type	Gauge
Dimensions	node_id thr_no site_name namespace

Table 5-78 db_tier_disk_write_speed_aggregate_disk_write_speed_set_to_min

Field	Details
Description	The number of seconds since the last node start the disk write speed was set to minimum.
Type	Gauge
Dimensions	node_id thr_no site_name namespace

Table 5-79 db_tier_disk_write_speed_aggregate_current_target_disk_write_speed

Field	Details
Description	The actual speed of the disk writes per LDM thread (aggregated).
Type	Gauge
Dimensions	node_id thr_no site_name namespace

5.20 cnDBTier Replication Error Skip Info Metrics

This section provides details about the cnDBTier Replication Error Skip Info Metrics.

Table 5-80 db_tier_replication_halted_due_to_skiperror

Field	Details
Description	The number of times an error is skipped.
Type	Gauge
Dimensions	site_name mate_site_name replchannel_group_id error_number namespace

Table 5-81 db_tier_epochs_lost_due_to_skiperror

Field	Details
Description	The number of epochs lost due to a skipped replication error.
Type	Gauge
Dimensions	site_name mate_site_name replchannel_group_id error_number namespace

Table 5-82 db_tier_replication_switchover_due_to_clusterdisconnect

Field	Details
Description	The number of switchover that happens due to cluster disconnect error.
Type	Gauge

Table 5-82 (Cont.) db_tier_replication_switchover_due_to_clusterdisconnect

Field	Details
Dimensions	node_id site_name mate_site_name replchannel_group_id error_number namespace

5.21 cnDBTier BinLog Injector Thread Info Metrics

This section provides details about the cnDBTier replication metrics.

Table 5-83 db_tier_binlog_injector_thread

Field	Details
Description	Indicates if the Bin Log Injector thread is stalled for every replication SQL node.
Type	Gauge
Dimensions	node_id site_name namespace

Table 5-84 db_tier_binlog_injector_thread_latest_epoch

Field	Details
Description	Provides the latest epoch of the Bin Log Injector thread for every SQL node.
Type	Gauge
Dimensions	node_id site_name namespace

5.22 cnDBTier PVC Health Monitoring Metrics

This section provides details about the cnDBTier PVC Health Monitoring metrics.

Table 5-85 db_tier_pvc_read_write_speed

Field	Details
Description	Provides information about the read and write speed of the PVC.
Type	Gauge

Table 5-85 (Cont.) db_tier_pvc_read_write_speed

Field	Details
Dimensions	site_name namespace mount_path hostname

Table 5-86 db_tier_pvc_is_accessible

Field	Details
Description	Provides information about whether the PVC is accessible or not at that point of time.
Type	Gauge
Dimensions	site_name namespace mount_path hostname

Table 5-87 db_tier_pvc_failure_count

Field	Details
Description	Provides information about the number of times PVC is not accessible during the last scrape.
Type	Gauge
Dimensions	site_name namespace mount_path hostname

5.23 cnDBTier PVC Capacity Monitoring Metrics

This section provides details about the cnDBTier PVC Capacity Monitoring metrics.

Table 5-88 db_tier_volume_stats_used_bytes

Field	Details
Description	This metric gives information about how much PVC is used in bytes for ndbappmysqld, ndbmysqld, ndbmtd and replication svc pods.
Type	Gauge
Dimensions	site_name namespace mount_path hostname

Table 5-89 db_tier_volume_stats_capacity_bytes

Field	Details
Description	Provides information about the total capacity of the PVC for ndbappmysqld, ndbmysqld, ndbmtd and replication svcpods.
Type	Gauge
Dimensions	site_name namespace mount_path hostname

5.24 cnDBTier Cert Expiration Time Metrics

This section provides details about the cnDBTier Certificate Expiration Time metrics.

Table 5-90 db_tier_cert_expiry

Field	Details
Description	This metric gives information about the expiration time of cnDBTier certificates for HTTPS and Tls.
Type	Gauge
Dimensions	site_name namespace cert_type hostname

5.25 DBTier Tc Time Track Metrics

TC (Transaction Coordinator) Time track metrics provide visibility into the average duration of events. For each event type, two metrics are tracked; the total accumulated time spent on the event and the count of occurrences. The average time for an event is calculated by dividing the total time by the count of events. This approach helps monitor and understand the typical time spent on specific event types.

Table 5-91 db_tier_tc_time_track_stats_total_index_count

Field	Details
Description	Provides the total number of index-based operations coordinated by the TC. The number of operations that require indexes to be created or updated or deleted.
Type	Gauge
Dimensions	site_name namespace node_id node_type

Table 5-92 db_tier_tc_time_track_stats_total_index_time

Field	Details
Description	Provides the total cumulative time the TC spent processing index operations. The time spent creating or updating or deleting indexes.
Type	Gauge
Dimensions	site_name namespace node_id node_type

Table 5-93 db_tier_tc_time_track_stats_total_read_count

Field	Details
Description	Provides the total number of read operations coordinated by the TC. The number of stateful primary key read operations.
Type	Gauge
Dimensions	site_name namespace node_id node_type

Table 5-94 db_tier_tc_time_track_stats_total_read_time

Field	Details
Description	Provides the total cumulative time the TC spent processing read operations. The time spent on stateful primary key read operations.
Type	Gauge
Dimensions	site_name namespace node_id node_type

Table 5-95 db_tier_tc_time_track_stats_total_scan_count

Field	Details
Description	Provides the total number of table or index scan operations coordinated by the TC. The total number of scans between REPL API or APP API and Data pods.
Type	Gauge

Table 5-95 (Cont.) db_tier_tc_time_track_stats_total_scan_count

Field	Details
Dimensions	site_name namespace node_id node_type

Table 5-96 db_tier_tc_time_track_stats_total_scan_time

Field	Details
Description	Provides the total cumulative time the TC spent performing scan operations. The time spent performing scan between REPL API or APP API and Data pods.
Type	Gauge
Dimensions	site_name namespace node_id node_type

Table 5-97 db_tier_tc_time_track_stats_total_scan_fragments_count

Field	Details
Description	Provides the total number of fragment-level scans performed across all scans. The number of scan fragment operations between TC and LDM threads. It is internal between TC and LDM
Type	Gauge
Dimensions	site_name namespace node_id node_type

Table 5-98 db_tier_tc_time_track_stats_total_scan_fragments_time

Field	Details
Description	Provides the total cumulative time spent scanning fragments across all scans. The time spent performing fragment scan operation between TC and LDM threads.
Type	Gauge
Dimensions	site_name namespace node_id node_type

Table 5-99 db_tier_tc_time_track_stats_total_transaction_count

Field	Details
Description	Provides the total number of transactions coordinated by the TC. The number of transactions between REPL API or APP API pod and data pod.
Type	Gauge
Dimensions	site_name namespace node_id node_type

Table 5-100 db_tier_tc_time_track_stats_total_transaction_time

Field	Details
Description	Provides the total cumulative time the TC spent in transactions (from start to commit or abort). The time spent on transaction between REPL API or APP API pod and data pod.
Type	Gauge
Dimensions	site_name namespace node_id node_type

Table 5-101 db_tier_tc_time_track_stats_total_write_count

Field	Details
Description	Provides the total number of write operations coordinated by the TC. The number of stateful primary key write operations.
Type	Gauge
Dimensions	site_name namespace node_id node_type

Table 5-102 db_tier_tc_time_track_stats_total_write_time

Field	Details
Description	Provides the total cumulative time the TC spent processing write operations. The time spent on stateful primary key write operations.
Type	Gauge

Table 5-102 (Cont.) db_tier_tc_time_track_stats_total_write_time

Field	Details
Dimensions	site_name namespace node_id node_type

5.26 cnDBTier Replication Switchover Status Metrics

This section provides details about the cnDBTier Replication Switchover Status metrics.

Table 5-103 db_tier_stop_repl_switchover

Field	Details
Description	Provides information about whether switchover is forcibly disabled or not.
Type	Gauge
Dimensions	site_name mate_site_name namespace replchannel_group_id

5.27 JVM Memory Metrics

Table 5-104 jvm_memory_used_bytes

Field	Details
Description	Provides information about the amount of memory currently used in the JVM.
Type	Gauge
Dimensions	area id

Table 5-105 jvm_memory_committed_bytes

Field	Details
Description	Provides information about the amount of used memory committed to be available for the JVM to use.
Type	Gauge
Dimensions	area id

Table 5-106 `jvm_memory_max_bytes`

Field	Details
Description	Provides information about the maximum amount of memory that can be used for the memory management.
Type	Gauge
Dimensions	area id

6

cnDBTier Alerts

cnDBTier generates alerts when cnDBTier meets a specified condition. You can access the alerts using the Prometheus dashboard and take necessary actions. Prometheus gets installed as part of common services during the vCNE installation. This section provides details about the available cnDBTier alerts.

6.1 cnDBTier Remote Server Backup Transfer Status Alerts

This section provides details about the cnDBTier remote server backup transfer status alerts.

Table 6-1 REMOTE_SERVER_BACKUP_TRANSFER_FAILED

Field	Details
Description	This alert is triggered with major severity when the transfer of backup to a remote server fails.
Summary	Secure transfer of backup to remote server failed on cnDBTier site {{ \$labels.site_name }}
Severity	major
Condition	db_tier_remote_server_backup_transfer_status == 1
Expression Validity	NA
SNMP Trap ID	2031
Affects Service (Y/N)	N

Table 6-1 (Cont.) REMOTE_SERVER_BACKUP_TRANSFER_FAILED

Field	Details
Recommended Action	Cause: The transfer of backup to remote server failed. Diagnostic Information: Check the status of the <code>db_tier_remote_server_backup_transfer_status</code> metric (db_tier_remote_server_backup_transfer_status). Recommended Actions: This alert is cleared automatically when the backup transfer status is updated from the good state to remote server as success. 1. Check if the data nodes are able to copy the backups to db replication service pod. Log in to one of the data nodes: <pre>kubectl -n <namespace> exec -it ndbmt-d-0 -c db-backup-executor-svc -- bash ssh -i /home/mysql/.ssh/id_rsa mysql@<db-replication-service-svc> -p 2022 sftp -i /home/mysql/.ssh/id_rsa -P 2022 -o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null mysql@<leader_db_Replication_service_svc> :/var/occnedb/ <<< '\$put -r <file/dir to be copied>',</pre> 2. Check if the remote server IP address is reachable or not from the Kubernetes cluster worker nodes. Log in to the worker node: <pre>ping <remoteserveripaddress></pre> 3. Verify whether the DB replication service pod can successfully connect to the remote server. Log in to the db replication service pod where PVC is attached. <pre>kubectl -n <namespace> exec -it <leader_db_replication_service_pod> -- bash ssh -i /home/mysql/.ssh/ remoteserver_id_rsa remoteuser@remoteServerIp</pre> 4. Verify that there is adequate disk space on the remote server for storing backups. Log in to the db replication service pod where PVC is attached. <pre>kubectl -n <namespace> exec -it <leader_db_replication_service_pod> -- bash</pre>

Table 6-1 (Cont.) REMOTE_SERVER_BACKUP_TRANSFER_FAILED

Field	Details
	<pre>ssh -i /home/mysql/.ssh/ remoteserver_id_rsa remoteuser@remoteServerIp df -kh</pre> 5. Manually copy large backup files to the remote server using SFTP commands from the db_replication_service pod to ensure backups can be transferred successfully. Log in to the db replication service pod where PVC is attached. <pre>kubectl -n <namespace> exec -it <leader_db_replication_service_pod> -- bash sftp -i /home/mysql/.ssh/ remoteserver_id_rsa -o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null <remoteserveruser>@<remote_server_ip_addr ess>:<remote_server_path> <<< '\$put -r <file/dir to be copied>',</pre> 6. In case the issue persists, contact My Oracle Support. Note Use CNC NF Data Collector tool for capturing logs. For more information on how to collect logs using Data Collector tool, see Oracle Communications Cloud Native Core, Network Function Data Collector User Guide.

6.2 cnDBTier Backup Transfer Status Alerts

This section provides details about the cnDBTier backup transfer status alerts.

Table 6-2 BACKUP_TRANSFER_LOCAL_FAILED

Field	Details
Description	This alert is triggered with major severity when the system fails to transfer the backup from the data node to the replication service pod on the cnDBTier site (db_tier_backup_transfer_status metric value is 2).

Table 6-2 (Cont.) BACKUP_TRANSFER_LOCAL_FAILED

Field	Details
Summary	Failed to transfer backup from data node to replication service pod on cnDBTier site {{ \$labels.site_name }}
Severity	major
Condition	db_tier_backup_transfer_status == 2
Expression Validity	NA
SNMP Trap ID	2026
Affects Service (Y/N)	Y

Table 6-2 (Cont.) BACKUP_TRANSFER_LOCAL_FAILED

Field	Details
Recommended Action	Cause: The system failed to transfer a backup from the data node to the replication service pod on a cnDBTier site. Diagnostic Information: The <code>db_tier_backup_transfer_status</code> metric (db_tier_backup_transfer_status) provides information about the backup transfer status. Recommended Actions: 1. Ensure all data pods and db-replication-svc pods are up. <pre>kubectl get pods -n <namespace> -l dbtierapp=ndbmttd --no-headers kubectl get pods -n <namespace> -l dbtierapp=dbreplicationsvc --no-headers</pre> 2. Check if it can be connected to the local db-replication-svc pod from the data nodes. <pre>kubectl -n <namespace> exec -it ndbmttd-0 -c db-backup-executor-svc -- bash ssh -i /home/mysql/.ssh/id_rsa mysql@<db- replication-service-svc> -p 2023</pre> 3. Verify the number of data nodes and the DataMemory value configured in the ndbmttd configuration and ensure that the PVC configured in the data nodes and db-replication-svc aligns with the cnDBTier dimensions. a. Verify the data memory configured in the <code>customer_value.yaml</code> file. <pre>ndb: ndbdisksize: 60Gi ndbbackupdisksize: 100Gi datamemory: 12G db-replication-svc: dbreplsvcdeployments: pvc: name: pvc-cluster1-cluster2- replication-svc disksize: 100Gi</pre> b. Log in to one of the Bastion host and get the pvc for data nodes and db replication service. <pre>kubectl get pvc -n <namespace> grep ndbmttd kubectl get pvc -n <namespace> grep repl</pre>

Table 6-2 (Cont.) BACKUP_TRANSFER_LOCAL_FAILED

Field	Details
	4. If db replication service pvc is not configured as per the cnDBTier dimension, then increase the PVC by following the scaling procedure provided in Scaling ndbmttd Pods, and get the PVC configurations reviewed by the NF team before performing the scaling. 5. In case the issue persists, contact My Oracle Support. Note Use CNC NF Data Collector tool for capturing logs. For more information on how to collect logs using Data Collector tool, see Oracle Communications Cloud Native Core, Network Function Data Collector User Guide.

Table 6-3 BACKUP_TRANSFER_FAILED

Field	Details
Description	This alert is triggered with major severity when the backup transfer failed as the system failed to transfer the backup to the remote site from the cnDBTier site (db_tier_backup_transfer_status metric value is 3).
Summary	Failed to transfer backup to remote site from cnDBTier site {{ \$labels.site_name }}
Severity	major
Condition	db_tier_backup_transfer_status == 3
Expression Validity	NA
SNMP Trap ID	2027
Affects Service (Y/N)	Y

Table 6-3 (Cont.) BACKUP_TRANSFER_FAILED

Field	Details
Recommended Action	Cause: The system failed to transfer a backup from the cnDBTier site to a remote site. Diagnostic Information: The <code>db_tier_backup_transfer_status</code> metric (db_tier_backup_transfer_status) provides information about the backup transfer status. Recommended Actions: 1. Check if all db-replication-svc pods of current site and mate site are up or not. <pre>kubectl get pods -n <namespace> -l dbtierapp=dbreplicationsvc --no-headers</pre> 2. Check if connectcurrent site leader db-replication-svc can be connected to mate site leader db-replication-svc. Log in to the db replication service pod where PVC is attached. <pre>kubectl -n <namespace> exec -it <leader_db_replication_service_pod> -- bash ssh -i /home/mysql/.ssh/id_rsa mysql@<leader_db_replication_service_site 2_pod> -p 2022</pre> 3. Verify the number of data nodes and the DataMemory value configured in the ndbmtcd configuration and ensure that the PVC configured in the data nodes and db-replication-svc aligns with the cnDBTier dimensions. a. Verify the data memory configured in the <code>customer_value.yaml</code> file. <pre>ndb: ndbdisksize: 60Gi ndbbackupdisksize: 100Gi datamemory: 12G db-replication-svc: dbreplsvcd deployments: pvc: name: pvc-cluster1-cluster2- replication-svc disksize: 100Gi</pre>

Table 6-3 (Cont.) BACKUP_TRANSFER_FAILED

Field	Details
	b. Log in to one of the Bastion hosts and get the PVC for data nodes and db replication service. <pre>kubectl get pvc -n <namespace> grep ndbmt kubectl get pvc -n <namespace> grep repl</pre> 4. If db replication service pvc is not configured as per the cnDBTier dimension, then increase the PVC by following the scaling procedure provided in Scaling ndbmt Pods, and get the PVC configurations reviewed by the NF team before performing the scaling. 5. In case the issue persists, contact My Oracle Support. Note Use CNC NF Data Collector tool for capturing logs. For more information on how to collect logs using Data Collector tool, see Oracle Communications Cloud Native Core, Network Function Data Collector User Guide.

Table 6-4 BACKUP_TRANSFER_IN_PROGRESS

Field	Details
Description	This alert is triggered with info severity when the backup transfer is in progress on the cnDBTier site (db_tier_backup_transfer_status metric value is 1).
Summary	Backup Transfer is In Progress on cnDBTier site {{ \$labels.site_name }}
Severity	info
Condition	db_tier_backup_transfer_status == 1
Expression Validity	NA
SNMP Trap ID	2028
Affects Service (Y/N)	N

Table 6-4 (Cont.) BACKUP_TRANSFER_IN_PROGRESS

Field	Details
Recommended Action	Cause: Backup transfer is in progress on the cnDBTier site. Diagnostic Information: The <code>db_tier_backup_transfer_status</code> metric (db_tier_backup_transfer_status) provides information about the backup transfer status. Recommended Actions: 1. Wait for the backup transfer to complete. The alert should automatically clear once the transfer is finished. 2. If the backup transfer takes longer than expected, allow it to continue for an additional 15 minutes, as temporary delays may occur. 3. If the transfer still does not complete within this time frame: • Check the CPU and memory resources allocated to the <code>db-backup-executor-svc</code> container within the <code>ndbmt</code> pods. • Verify the configuration of the leader <code>db-replication-service</code> in both current and remote sites. • Ensure that all these components are configured in accordance with the cnDBTier dimension sheet. 4. In case the issue persists, contact My Oracle Support. Note Use CNC NF Data Collector tool for capturing logs. For more information on how to collect logs using Data Collector tool, see Oracle Communications Cloud Native Core, Network Function Data Collector User Guide.

6.3 cnDBTier Heartbeat Alerts

This section provides details about cnDBTier heartbeat alerts.

Table 6-5 HEARTBEAT_FAILED

Field	Details
Description	This alert is triggered with critical severity when HeartBeat fails on a remote site.
Summary	HeartBeat failed on cnDBTier site <code>{{ \$labels.site_name }}</code> connected to mate site <code>{{ \$labels.mate_site_name }}</code> on replication channel group id <code>{{ \$labels.repchannel_group_id }}</code> and kubernetes namespace <code>{{ \$labels.namespace }}</code>

Table 6-5 (Cont.) HEARTBEAT_FAILED

Field	Details
Severity	critical
Condition	db_tier_heartbeat_failure == 1
Expression Validity	NA
SNMP Trap ID	2025
Affects Service (Y/N)	Y

Table 6-5 (Cont.) HEARTBEAT_FAILED

Field	Details
Recommended Action	Cause: The system is unable to connect to remote site and Heartbeat failed. Diagnostic Information: The <code>db_tier_heartbeat_failure</code> metric (db_tier_heartbeat_failure) provides information about the heartbeat status and indicates whether the remote site is reachable or not. Recommended Actions: 1. Check the logs of the db-replication-svc service. <pre>\$ kubectl logs -n <cnDBTier Namespace> <replication service pod name> -f</pre> 2. Check if Geo Replication Recovery (GRR) is in progress. During GRR, a HEARTBEAT_FAILED alert may be triggered. <pre>\$ export IP=\$(kubectl get svc -n <namespace of failed site> -l dbtierapp=dbreplicationsvc,servicetype=ex ternal --no-headers awk '{print \$4}' head -n 1) a. Get the replication service LoadBalancer Port for the site on which restore is being done. \$ export PORT=\$(kubectl get svc -n <namespace of failed site> -l dbtierapp=dbreplicationsvc,servicetype =external --no-headers awk '{print \$5}' cut -d '/' -f 1 cut -d ':' -f 1 head -n 1) b. Run the following command to get geo replication restore status. \$ curl -X GET http://\$IP:\$PORT/db- tier/gr-recovery/site/{sitename}/ status c. If HTTPS is enabled, run the following commands: \$ kubectl exec it n <cnDBTier namespace> <current_site_db_replication_svc_pod></pre>

Table 6-5 (Cont.) HEARTBEAT_FAILED

Field	Details
	<pre> - bash \$ curl --cert client-cert.pem --cert-type PEM --key client-key.pem --key-type PEM --cacert combine-ca.pem -X GET http://\$IP:\$PORT/db-tier/gr-recovery/site/{sitename}/status </pre> 3. Check if the mate site IP and port are configured correctly: <pre> replication: # Local site replication service LoadBalancer ip can be configured. localsiteip: "" localsiteport: "80" channelgroupid: "1" matesitename: "<\$ {OCCNE_MATE_SITE_NAME}>" preferredIpFamily: "IPv4" remotesiteip: "<\$ {OCCNE_MATE_REPLICATION_SVC}>" remotesiteport: "80" service: type: LoadBalancer loadBalancerIP: "" httpport: 80 httpsport: 443 </pre> 4. If HTTPS is enabled, then check if the certificates are configured rightly. Check the db-replication-svc pod logs for ssl handshake related errors. <pre> \$ kubectl logs -n <cnDBTier Namespace> <replication service pod name> -f </pre> 5. Check if the connectivity is established between the current and remote db-replication-svc service in both directions. a. To check the Curl connection use the following command and get the replication service LoadBalancer IP for cluster1: <pre> \$ IP=\$(kubectl get svc -n <cnDBTier namespace> -l </pre> dbtierapp=dbreplicationsvc,servicetype

Table 6-5 (Cont.) HEARTBEAT_FAILED

Field	Details
	<pre>=external --no-headers awk '{print \$4}' head -n 1)</pre> b. Get the replication service LoadBalancer Port for cluster1. <pre>\$ PORT=\$(kubectl get svc -n <cnDBTier namespace> -l dbtierapp=dbreplicationsvc,servicetype =external --no-headers awk '{print \$5}' cut -d '/' -f 1 cut - d ':' -f 1 head -n 1)</pre> <pre>\$ kubectl exec it n <cnDBTier namespace> <current_site_db_replication_svc_pod> - bash \$ curl -X GET http://\$IP:\$PORT//db- tier/health/db-replication-svc/status</pre> c. If HTTPS is enabled: <pre>\$ kubectl exec it n <cnDBTier namespace> <current_site_db_replication_svc_pod> - bash \$ curl --cert client-cert.pem --cert- type PEM --key client-key.pem --key- type PEM --cacert combine-ca.pem -X GET https://\$IP:\$PORT//db-tier/ health/db-replication-svc/status</pre> 6. In case the issue persists, contact My Oracle Support. Note Use CNC NF Data Collector tool for capturing logs. For more information on how to collect logs using Data Collector tool, see Oracle Communications Cloud Native Core, Network Function Data Collector User Guide.

6.4 cnDBTier BinLog Injector Thread Alerts

This section provides details about cnDBTier BinLog injector alerts.

Table 6-6 BINLOG_INJECTOR_STOPPED

Field	Details
Description	This alert is triggered with critical severity when Bin Log Injector stops working. The value of <code>db_tier_binlog_injector_thread</code> or <code>db_tier_binlog_injector_thread_latest_epoch</code> indicates the status of Bin Log Injector: • 0: indicates that the Bin Log Injector thread is not stopped for the specified node ID • 1: indicates that the Bin Log Injector thread is stopped for the specified node ID
Summary	BinLog Injector Thread is stopped for MySQL node having node id {{ \$labels.node_id }} on cnDBTier site {{ \$labels.site_name }} and kubernetes namespace {{ \$labels.namespace }}
Severity	critical
Condition	<code>db_tier_binlog_injector_thread_latest_epoch == 1</code> or <code>db_tier_binlog_injector_thread == 1</code>
Expression Validity	NA
SNMP Trap ID	2024
Affects Service (Y/N)	Y

Table 6-6 (Cont.) BINLOG_INJECTOR_STOPPED

Field	Details
Recommended Action	Cause: Bin Log Injector thread stalled for the replication SQL node. Diagnostic Information: The <code>db_tier_binlog_injector_thread_latest_epoch</code> or <code>db_tier_binlog_injector_thread</code> metrics (db_tier_binlog_injector_thread_latest_epoch or db_tier_binlog_injector_thread) provide information whether the Bin Log Injector thread is stalled or not. Recommended Actions: 1. Check the engine ndb status by connecting to MySQL server running inside the ndbmysqld pod and run <code>SHOW ENGINE NDB STATUS</code>. Check if the <code>latest_epoch</code> in ndbappmysqld and ndbmysqld pod is changing. Check the value of the <code>latest_applied_binlog_epoch</code> in ndbmysqld pod using the following command. <pre>\$ kubectl -n <namespace> exec -it ndbmysqld-0 -- mysql -h127.0.0.1 -uroot - pNextGenCne \$ SHOW ENGINE NDB STATUS;</pre> Run this command a few times and monitor the values of <code>latest_epoch</code> and <code>latest_applied_binlog_epoch</code> in the ndbmysqld/ndbappmysqld pod. If these values are not getting changed, then it means that binlog injector thread is stalled. 2. If the alert is not getting cleared, then restart the ndbmysqld/ndbappmysqld pod where the binlog injector thread is stalled. <pre>\$ kubectl -n <namespace> delete pod <ndbmysqld/ndbappmysqld POD NAME></pre> 3. In case the issue persists, contact My Oracle Support. Note Use CNC NF Data Collector tool for capturing logs. For more information on how to collect logs using Data Collector tool, see Oracle Communications Cloud Native Core, Network Function Data Collector User Guide.

6.5 cnDBTier Replication Error Skip Alerts

This section provides details about the cnDBTier replication error skip alerts.

Table 6-7 REPLICATION_SWITCHOVER_DUE_CLUSTERDISCONNECT

Field	Details
Description	This alert is triggered when switch over happens on an API node due to configured cluster disconnect error, if skip replication error is enabled.
Summary	Replication channel on SQL node with node ID {{ \$labels.node_id }} had switchover due to cluster disconnecterror number {{ \$labels.error_number }}
Severity	info
Condition	db_tier_replication_switchover_due_to_clusterdisconnect == 1
Expression Validity	NA
SNMP Trap ID	2019
Affects Service (Y/N)	N

Table 6-7 (Cont.) REPLICATION_SWITCHOVER_DUE_CLUSTERDISCONNECT

Field	Details
Recommended Action	Cause: Skip replication error is enabled on an API node and a switchover occurred on the node as the configured cluster disconnected. Diagnostic Information: The <code>db_tier_replication_switchover_due_to_clusterdisconnect</code> metric (Table 5-82) provides information whether a switchover occurred on an API node. Recommended Actions: 1. Check cluster status and pod status in the remote site. If multiple pods restarted in the remote site then capture all the outputs for the above steps and contact My Oracle Support. a. Check the cluster status by running the following command: <pre>kubectl -n <cnDBTier remote site Namespace> exec -it ndbmcmd-0 -- ndb_mgm -e show</pre> b. Check the pod status by running the following command: <pre>kubectl get pod -n <cnDBTier remote site Namespace></pre> 2. Verify if the cnDBTier pods (ndbmtd, ndbmysqld) are configured with resources (CPUs and memory) as per cnDBTier dimensions in the remote site. 3. If CPU and memory for the pods are not configured as per the cnDBTier dimensions, then increase the CPU and memory by following the Scaling ndbmtd Pods procedure and ensure that CPU and memory configurations are reviewed by the NF team before upgrading. 4. Check for resource pressure on the pods of the remote site for which the replication is switched over. <pre>kubectl top pod -n <cnDBTier remote site Namespace></pre> 5. Check <code>db_tier_replication_switchover_due_to_clusterdisconnect</code> metric to get the error number resulting the switchover. If the replication switch over occurs because of the replication error 13119 and 1296, then their may be issue that cluster nodes got disconnected and restarted or both the replicas in the replication group got disconnected from cluster. In this case, collect the logs and contact My Oracle Support.

Table 6-7 (Cont.) REPLICATION_SWITCHOVER_DUE_CLUSTERDISCONNECT

Field	Details
	6. Check the ndbmysqld and ndbmcmd logs of the remote site and try to find out the cause of the particular error number obtained above (for example, network issue, heartbeat failures, cluster disconnection, infrastructure events like network issue or network delays occurred in the infrastructure and so on). a. Check the current and previous logs of ndbmtd pods: <pre>kubectl logs -n <cnDBTier remote site Namespace> <ndbmtd pod> -f kubectl logs -n <cnDBTier remote site Namespace> <ndbmtd pod> --previous</pre> b. Check the logs of the ndbmysql pods: i. Log in to ndbmysqld pod. <pre>kubectl -n <cnDBTier remote site Namespace> exec -it <ndbmysqld pod> -- bash</pre> ii. Change to /var/occnedb/mysql/ directory. <pre>cd /var/occnedb/mysql/</pre> iii. Open the mysqld.log file to check the logs. <pre>vim mysqld.log</pre> 7. In case if the geo replication is down with the remote site and the issue persists and alert did not automatically clear within configured threshold time(1 hour), capture all the outputs for the above steps and contact My Oracle Support. Note Use CNC NF Data Collector tool for capturing logs. For more information on how to collect logs using Data Collector tool, see Oracle Communications Cloud Native Core, Network Function Data Collector User Guide.

Table 6-8 REPLICATION_TOO_MANY_EPOCHS_LOST

Field	Details
Description	This alert is triggered when the epochs lost due to skip error is greater than 10000 and less than or equal to 80000. This alert is cleared one hour after the event.
Summary	Too many epochs are lost for skipping replication errors
Severity	major
Condition	(db_tier_epochs_lost_due_to_skiperror > 10000) and (db_tier_epochs_lost_due_to_skiperror <= 80000)
Expression Validity	NA
SNMP Trap ID	2020
Affects Service (Y/N)	N

Table 6-8 (Cont.) REPLICATION_TOO_MANY_EPOCHS_LOST

Field	Details
Recommended Action	Cause: Between 10000 and 80000 epochs are lost due to skip errors. Diagnostic Information: The <code>db_tier_epochs_lost_due_to_skiperror</code> metric (Table 5-81) provides information about the number of epochs lost due to skip errors. Recommended Actions: 1. Check cluster status and pod status in the remote site. If multiple pods restarted in the remote site then capture all the outputs for the above steps and contact My Oracle Support. a. Check the cluster status by running the following command: <pre>kubectl -n <cnDBTier remote site Namespace> exec -it ndbmcmd-0 -- ndb_mgm -e show</pre> b. Check the pod status by running the following command: <pre>kubectl get pod -n <cnDBTier remote site Namespace></pre> 2. Verify if the cnDBTier pods (ndbmtd, ndbmysqld) are configured with resources (CPUs and memory) as per cnDBTier dimensions in the remote site. 3. If CPU and memory for the pods are not configured as per the cnDBTier dimensions, then increase the CPU and memory by following the Scaling ndbmtd Pods procedure and ensure that CPU and memory configurations are reviewed by the NF team before upgrading. 4. Check for resource pressure on the pods of the remote site for which the replication is switched over. <pre>kubectl top pod -n <cnDBTier remote site Namespace></pre> 5. Check <code>db_tier_replication_switchover_due_to_clusterdisconnect</code> metric to get the error number resulting the switchover. If the replication switch over occurs because of the replication error 13119 and 1296, then their may be issue that cluster nodes got disconnected and restarted or both the replicas in the replication group got disconnected from cluster. In this case, collect the logs and contact My Oracle Support. 6. Check the ndbmysqld and ndbmcmd logs of the remote site and try to find out the cause of the particular error

Table 6-8 (Cont.) REPLICATION_TOO_MANY_EPOCHS_LOST

Field	Details
	number obtained above (for example, network issue, heartbeat failures, cluster disconnection, infrastructure events like network issue or network delays occurred in the infrastructure and so on). a. Check the current and previous logs of ndbmttd pods: <pre>kubectl logs -n <cnDBTier remote site Namespace> <ndbmttd pod> -f kubectl logs -n <cnDBTier remote site Namespace> <ndbmttd pod> --previous</pre> b. Check the logs of the ndbmysqld pods: i. Log in to ndbmysqld pod. <pre>kubectl -n <cnDBTier remote site Namespace> exec -it <ndbmysqld pod> -- bash</pre> ii. Change to /var/occnedb/mysql/ directory. <pre>cd /var/occnedb/mysql/</pre> iii. Open the mysqld.log file to check the logs. <pre>vim mysqld.log</pre> 7. In case if the geo replication is down with the remote site and the issue persists and alert did not automatically clear within configured threshold time (1 hour), capture all the outputs for the above steps and contact My Oracle Support. Note Use CNC NF Data Collector tool for capturing logs. For more information on how to collect logs using Data Collector tool, see Oracle Communications Cloud Native Core, Network Function Data Collector User Guide.

Table 6-9 REPLICATION_SKIP_ERRORS_LOW

Field	Details
Description	This alert is triggered when the replication is halted due to skip error count less than or equal to 5. This alert is cleared one hour after the event.
Summary	Cross-site replication errors are skipped
Severity	minor
Condition	(db_tier_replication_halted_due_to_skiperror > 0) and (db_tier_replication_halted_due_to_skiperror <= 5)
Expression Validity	NA
SNMP Trap ID	2021
Affects Service (Y/N)	N

Table 6-9 (Cont.) REPLICATION_SKIP_ERRORS_LOW

Field	Details
Recommended Action	Cause: Replication halted due to less than five skip errors. Diagnostic Information: The <code>db_tier_replication_halted_due_to_skiperror</code> metric (Table 5-80) provides information about the number of skip errors due to which the replication halted. Recommended Actions: 1. Monitor the alert. If new skip errors accumulate or replication halts again, it may escalate to <code>REPLICATION_SKIP_ERRORS_HIGH</code>. 2. Check cluster status and pod status in the remote site. If multiple pods restarted in the remote site then capture all the outputs for the above steps and contact My Oracle Support. a. Check the cluster status by running the following command: <pre>kubectl -n <cnDBTier remote site Namespace> exec -it ndbmcmd-0 -- ndb_mgm -e show</pre> b. Check the pod status by running the following command: <pre>kubectl get pod -n <cnDBTier remote site Namespace></pre> 3. Check for resource pressure on the pods on the remotesite. <pre>kubectl top pod -n <cnDBTier remote site Namespace></pre> 4. Check the <code>db_tier_replication_switchover_due_to_skiperror</code> metric to get the error number which is raising the alert. If the replication switch over occurs because of the replication error 13119 and 1296, then their may be issue that cluster nodes got disconnected and restarted or both the replicas in the replication group got disconnected from cluster. In this case, collect the logs and contact My Oracle Support. 5. Check the <code>ndbmysqld</code> and <code>ndbmcmd</code> logs of the remote site and try to find out the cause of the particular error number obtained above (for example, network issue, heartbeat failures, cluster disconnection, infrastructure events like network issue or network delays occurred in the infrastructure and so on).

Table 6-9 (Cont.) REPLICATION_SKIP_ERRORS_LOW

Field	Details
	a. Check the current and previous logs of ndbmtl pods: <pre>kubectl logs -n <cnDBTier remote site Namespace> < ndbmtl pod> -f kubectl logs -n <cnDBTier remote site Namespace> < ndbmtl pod> --previous</pre> b. Check the logs of the ndbmysql pods: i. Log in to ndbmysqld pod. <pre>kubectl -n <cnDBTier remote site Namespace> exec -it <ndbmysqld pod> -- bash</pre> ii. Change to /var/occnedb/mysql/ directory. <pre>cd /var/occnedb/mysql/</pre> iii. Open the mysqld.log file to check the logs. <pre>vim mysqld.log</pre> 6. In case if the geo replication is down with the remote site and the issue persists and alert did not automatically clear within configured threshold time (1 hour), capture all the outputs for the above steps and contact My Oracle Support. Note Use CNC NF Data Collector tool for capturing logs. For more information on how to collect logs using Data Collector tool, see Oracle Communications Cloud Native Core, Network Function Data Collector User Guide.

Table 6-10 REPLICATION_SKIP_ERRORS_HIGH

Field	Details
Description	This alert is triggered when the replication is halted due to skip error counts greater than 5. This alert is cleared one hour after the event.

Table 6-10 (Cont.) REPLICATION_SKIP_ERRORS_HIGH

Field	Details
Summary	Cross-site replication errors skipped are high
Severity	major
Condition	db_tier_replication_halted_due_to_skiperror > 5
Expression Validity	NA
SNMP Trap ID	2022
Affects Service (Y/N)	N

Table 6-10 (Cont.) REPLICATION_SKIP_ERRORS_HIGH

Field	Details
Recommended Action	Cause: Replication halted due to more than five skip errors. Diagnostic Information: The <code>db_tier_replication_halted_due_to_skiperror</code> metric (Table 5-80) provides information about the number of skip errors due to which the replication halted. Recommended Actions: 1. Check cluster status and pod status in the remote site. If multiple pods restarted in the remote site then capture all the outputs for the above steps and contact My Oracle Support. a. Check the cluster status by running the following command: <pre>kubectl -n <cnDBTier remote site Namespace> exec -it ndbmcmd-0 -- ndb_mgm -e show</pre> b. Check the pod status by running the following command: <pre>kubectl get pod -n <cnDBTier remote site Namespace></pre> 2. Check for resource pressure on the pods on the remotesite. <pre>kubectl top pod -n <cnDBTier remote site Namespace></pre> 3. Check the <code>db_tier_replication_switchover_due_to_skiperror</code> metric to get the error number which is raising the alert. If the replication switch over occurs because of the replication error 13119 and 1296, then their may be issue that cluster nodes got disconnected and restarted or both the replicas in the replication group got disconnected from cluster. In this case, collect the logs and contact My Oracle Support. 4. Check the <code>ndbmysqld</code> and <code>ndbmcmd</code> logs of the remote site and try to find out the cause of the particular error number obtained above (for example, network issue, heartbeat failures, cluster disconnection, infrastructure events like network issue or network delays occurred in the infrastructure and so on).

Table 6-10 (Cont.) REPLICATION_SKIP_ERRORS_HIGH

Field	Details
	a. Check the current and previous logs of ndbmttd pods: <pre>kubectl logs -n <cnDBTier remote site Namespace> <ndbmttd pod> -f kubectl logs -n <cnDBTier remote site Namespace> <ndbmttd pod> --previous</pre> b. Check the logs of the ndbmysql pods: i. Log in to ndbmysqld pod. <pre>kubectl -n <cnDBTier remote site Namespace> exec -it <ndbmysqld pod> -- bash</pre> ii. Change to /var/occnedb/mysql/ directory. <pre>cd /var/occnedb/mysql/</pre> iii. Open the mysqld.log file to check the logs. <pre>vim mysqld.log</pre> 5. In case if the geo replication is down with the remote site and the issue persists and alert did not automatically clear within configured threshold time (1 hour), capture all the outputs for the above steps and contact My Oracle Support. Note Use CNC NF Data Collector tool for capturing logs. For more information on how to collect logs using Data Collector tool, see Oracle Communications Cloud Native Core, Network Function Data Collector User Guide.

Table 6-11 REPLICATION_EPOCHS_LOST

Field	Details
Description	This alert is triggered when the epochs lost due to skip error is greater than 0 and less than 2000. This alert is cleared one hour after the event.

Table 6-11 (Cont.) REPLICATION_EPOCHS_LOST

Field	Details
Summary	Epochs are lost for skipping replication errors
Severity	info
Condition	db_tier_epochs_lost_due_to_skiperror > 0 and db_tier_epochs_lost_due_to_skiperror <= <Configured epoch interval lower threshold>
Expression Validity	NA
SNMP Trap ID	2023
Affects Service (Y/N)	N

Table 6-11 (Cont.) REPLICATION_EPOCHS_LOST

Field	Details
Recommended Action	Cause: Less than 2000 epochs are lost due to skip errors. Diagnostic Information: The <code>db_tier_epochs_lost_due_to_skiperror</code> metric (Table 5-81) provides information about the number of epochs lost due to skip errors. Recommended Actions: 1. Monitor the alert, If new skip errors accumulate or replication halts again, it may escalate to <code>REPLICATION_TOO_MANY_EPOCHSLOST</code>. 2. Check cluster status and pod status in the remote site. If multiple pods restarted in the remote site then capture all the outputs for the above steps and contact My Oracle Support. a. Check the cluster status by running the following command: <pre>kubectl -n <cnDBTier remote site Namespace> exec -it ndbmcmd-0 -- ndb_mgm -e show</pre> b. Check the pod status by running the following command: <pre>kubectl get pod -n <cnDBTier remote site Namespace></pre> 3. Check for resource pressure on the pods on the remotesite. <pre>kubectl top pod -n <cnDBTier remote site Namespace></pre> 4. Check the <code>db_tier_epochs_lost_due_to_skiperror</code> metric to get the error number which is raising the alert. If the replication switch over occurs because of the replication error 13119 and 1296, then their may be issue that cluster nodes got disconnected and restarted or both the replicas in the replication group got disconnected from cluster. In this case, collect the logs and contact My Oracle Support. 5. Check the <code>ndbmysqld</code> and <code>ndbmcmd</code> logs of the remote site and try to find out the cause of the particular error number obtained above (for example, network issue, heartbeat failures, cluster disconnection, infrastructure events like network issue or network delays occurred in the infrastructure and so on).

Table 6-11 (Cont.) REPLICATION_EPOCHS_LOST

Field	Details
	a. Check the current and previous logs of ndbmtid pods: <pre>kubectl logs -n <cnDBTier remote site Namespace> < ndbmtid pod> -f kubectl logs -n <cnDBTier remote site Namespace> < ndbmtid pod> --previous</pre> b. Check the logs of the ndbmysql pods: i. Log in to ndbmysqld pod. <pre>kubectl -n <cnDBTier remote site Namespace> exec -it <ndbmysqld pod> -- bash</pre> ii. Change to /var/occnedb/mysql/ directory. <pre>cd /var/occnedb/mysql/</pre> iii. Open the mysqld.log file to check the logs. <pre>vim mysqld.log</pre> 6. In case if the geo replication is down with the remote site and the issue persists and alert did not automatically clear within configured threshold time (1 hour), capture all the outputs for the above steps and contact My Oracle Support. Note Use CNC NF Data Collector tool for capturing logs. For more information on how to collect logs using Data Collector tool, see Oracle Communications Cloud Native Core, Network Function Data Collector User Guide.

6.6 cnDBTier Georeplication Recovery Status Alerts

This section provides details about the cnDBTier georeplication recovery status alerts.

Table 6-12 GEOREPLICATION_RECOVERY_IN_PROGRESS

Field	Details
Description	This alert is triggered with critical severity when the georeplication recovery is in progress and the alert is cleared when georeplication recovery is complete.
Summary	Identified cnDBTier Site {{ \$labels.site_name }} georeplication recovery is in progress for kubernetes namespace {{ \$labels.namespace }}
Severity	critical
Condition	db_tier_georeplication_recovery_state == 1
Expression Validity	1m
SNMP Trap ID	2018
Affects Service (Y/N)	Y

Table 6-12 (Cont.) GEOREPLICATION_RECOVERY_IN_PROGRESS

Field	Details
Recommended Action	Cause: When you perform georeplication recovery to recover failed site from a healthy site, that is when georeplication recovery is in progress. Diagnostic Information: The <code>db_tier_georeplication_recovery_state</code> metric (db_tier_georeplication_recovery_state) provides information whether georeplication recovery is in progress. Recommended Actions: 1. Since the time required to complete the GRR process depends on the database size, wait for the process to complete. The alert will be automatically cleared once the GRR process is successfully completed. 2. Monitor the status of GRR using the GRR Status REST API. a. If <code>gr_state</code> is "COMPLETED", then database restoration and re-establishing the replication channels are completed. Get the replication service LoadBalancer IP for the site on which restore is being done: <pre>\$ export IP=\$(kubectl get svc -n <namespace of failed site> -l dbtierapp=dbreplicationsvc,servicetype=external --no-headers awk '{print \$4}' head -n 1)</pre> b. Get the replication service LoadBalancer Port for the site on which restore is being done: <pre>\$ export PORT=\$(kubectl get svc -n <namespace of failed site> -l dbtierapp=dbreplicationsvc,servicetype=external --no-headers awk '{print \$5}' cut -d '/' -f 1 cut -d ':' -f 1 head -n 1)</pre> c. Run the following command to get the geo replication restore status: <pre>\$ curl -X GET http://\$IP:\$PORT/db-tier/gr-recovery/site/{sitename}/status</pre> 3. Monitor the GRR states and check if any other alerts related to backup transfer has failed are raised: 4. In case the issue persists, capture all the outputs for the above steps and contact My Oracle Support.

Table 6-12 (Cont.) GEOREPLICATION_RECOVERY_IN_PROGRESS

Field	Details
	 Note Use CNC NF Data Collector tool for capturing logs. For more information on how to collect logs using Data Collector tool, see Oracle Communications Cloud Native Core, Network Function Data Collector User Guide.

6.7 cnDBTier Cluster Status Alerts

This section provides details about cnDBTier cluster status alerts.

Table 6-13 CLUSTER_DOWN

Field	Details
Description	This alert is triggered with critical severity when cnDBTier NDB cluster is not UP.
Summary	MySQL Cluster is down for cnDBTier site {{ \$labels.site_name }} and kubernetes namespace {{ \$labels.namespace }}
Severity	critical
Condition	db_tier_cluster_status == 0
Expression Validity	1m
SNMP Trap ID	2017
Affects Service (Y/N)	Y

Table 6-13 (Cont.) CLUSTER_DOWN

Field	Details
Recommended Action	Cause: - When pod restarts due to Kubernetes liveness or readiness probe failures. - When cnDBTier application restarts or fails to start. Diagnostic Information: - Run the following command to check the status of cnDBTier namespace: <pre>kubectl -n <namespace> exec -it ndbmgmd-0 -- ndb_mgm -e show</pre> The cluster is down if: - the ndbappmysqld pods are down, not running, and not connected - the remaining pods are not running and not connected - Check Kubernetes events for probe failures in the platform logs. - Check if any exception is reported in the cnDBTier application logs. Recommended Actions: - If the <code>GEOREPLICATION_RECOVERY_IN_PROGRESS</code> alert is raised along with the <code>CLUSTER_DOWN</code> alert, wait until the GRR process is completed. - Get the replication service LoadBalancer IP for the site on which restore is being done. <pre>\$ export IP=\$(kubectl get svc -n <namespace of failed site> -l dbtierapp=dbreplicationsvc,servicetype =external --no-headers awk '{print \$4}' head -n 1)</pre> - Get the replication service LoadBalancer Port for the site on which restore is being done: <pre>\$ export PORT=\$(kubectl get svc -n <namespace of failed site> -l dbtierapp=dbreplicationsvc,servicetype =external --no-headers awk '{print \$5}' cut -d '/' -f 1 cut -d ':' -f 1 head -n 1)</pre>

Table 6-13 (Cont.) CLUSTER_DOWN

Field	Details
	c. Run the following command to get Georeplication restore status: <pre>\$ curl -X GET http://\$IP:\$PORT/db-tier/gr-recovery/site/{sitename}/status</pre> 2. Run the following command to check the status of cnDBTier namespace: The following command checks the node status: <pre>\$ kubectl -n \${OCCNE_NAMESPACE} exec -it ndbmgmd-0 -- ndb_mgm -e show</pre> 3. Check the management pod logs also for checking if there are any frequent heartbeat missed warnings or other ALERT are logged for the pod. <pre>\$ kubectl -n <namespace> exec -it ndbmgmd-0 -- bash \$ tail -f /var/occnedb/mysqlndbcluster/ndbmgmd_cluster.log</pre> 4. Check the pod status, if the pod is not coming up then analyze the previous container logs of the pod to see the error information. <pre>\$ kubectl -n <namespace> logs <ndbmgmd/ndbmysqld/ndbappmysqld podname> -- previous</pre> 5. Verify if any network or infrastructure-related events occurred that might have disrupted communication between pods. 6. In case the issue persists, capture all the outputs for the above steps and contact My Oracle Support Note Use CNC NF Data Collector tool for capturing logs. For more information on how to collect logs using Data Collector tool, see Oracle Communications Cloud Native Core, Network Function Data Collector User Guide.

Table 6-14 MYSQL_NDB_CLUSTER_DISCONNECT

Field	Details
Description	This alert is triggered with critical severity when cnDBTier NDB cluster is not UP.
Summary	MySQL NDB Cluster Disconnected {{ \$value }} times for cnDBTier site {{ \$labels.site_name }} and kubernetes namespace {{ \$labels.namespace }}
Severity	critical
Condition	db_tier_cluster_disconnect > 0
Expression Validity	1m
SNMP Trap ID	2034
Affects Service (Y/N)	Y

Table 6-14 (Cont.) MYSQL_NDB_CLUSTER_DISCONNECT

Field	Details
Recommended Action	Cause: - When all ndbmttd pods or all ndbmttd pods of the same node group restart due to Kubernetes probe failures or infrastructure related issues. Diagnostic Information: - Run the following command to check the status of cnDBTier namespace: <pre>kubectl -n <namespace> exec -it ndbmgmd-0 -- ndb_mgm -e show</pre> The cluster is down if: all data nodes or all data nodes of the same node group are not connected. - Check Kubernetes events for probe failures in the platform logs. - Check if there is any network fluctuation or platform related issue which can cause the ndbmttd pods to restart. - Check if any exception is reported in the cnDBTier application logs. Recommended Actions: - Run the following command to check the status of cnDBTier namespace: The following command checks the node status: <pre>\$ kubectl -n \${OCCNE_NAMESPACE} exec -it ndbmgmd-0 -- ndb_mgm -e show</pre> - Check the management pod logs also for checking if there are any frequent heartbeat missed warnings or other ALERT are logged for the pod. <pre>\$ kubectl -n <namespace> exec -it ndbmgmd-0 -- bash \$ tail -f /var/occnedb/mysqlndbcluster/ ndbmgmd_cluster.log</pre> - Check the pod status, if the pod is not coming up then analyze the previous container logs of the pod to see the error information. <pre>\$ kubectl -n <namespace> logs <ndbmttd/ ndbmysqld/ndbappmysqld podname> -- previous</pre> - Verify if any network or infrastructure-related events occurred that might have disrupted communication between pods.

Table 6-14 (Cont.) MYSQL_NDB_CLUSTER_DISCONNECT

Field	Details
	5. Check if any exception is reported in the cnDBTier application logs: <pre>\$ kubectl logs -n <cnDBTier Namespace> <cnDBTier Pod name> -f</pre> 6. If the pod alone is down and all the other pods are up and running then delete the pod and its pvc, so that pod can restart by re-initializing: <pre>\$ kubectl -n <namespace> exec -it <ndbmt/ndbmysqld/ndbappmysqld podname> - c db-infra-monitor-svc -- bash \$ ndb_mgm -c ndbmcmd-0.ndbmcmdsvc:1186 - e show</pre> 7. In case the issue persists, capture all the outputs for the above steps and contact My Oracle Support Note Use CNC NF Data Collector tool for capturing logs. For more information on how to collect logs using Data Collector tool, see Oracle Communications Cloud Native Core, Network Function Data Collector User Guide.

6.8 cnDBTier Automated Backup Alerts

This section provides details about the cnDBTier automated backup alerts.

Table 6-15 BACKUP_FAILED

Field	Details
Description	This alert is triggered with minor severity when the backup service fails to complete the backup successfully.
Summary	Could not backup database for cnDBTier site {{ \$labels.site_name }} and kubernetes namespace {{ \$labels.namespace }}
Severity	minor
Condition	db_tier_backup{status='FAILED'}
Expression Validity	N/A
SNMP Trap ID	2011

Table 6-15 (Cont.) BACKUP_FAILED

Field	Details
Affects Service (Y/N)	N

Table 6-15 (Cont.) BACKUP_FAILED

Field	Details
Recommended Action	Cause: - When backup service fails to complete the backup successfully. - When PVC size is not enough and as a result the backup fails. Diagnostic Information: The <code>db_tier_backup</code> metric (db_tier_backup) provides information if the backup failed or not. Recommended Actions: - Run the following command to check if all data pods are up and running: <pre>kubectl get pods -n <namespace> -l dbtierapp=ndbmttd --no-headers</pre> - Check the data pod status, if the pod is not coming up then analyze the previous container logs of the pod to see the error information. <pre>kubectl -n <namespace> logs <ndbmttd pod name> --previous</pre> - Verify the number of data nodes and the DataMemory value configured in the ndbmttd configuration and ensure that the PVC configured in data nodes aligns with the cnDBTier dimensions. - Verify the data memory configured in <code>custom_values.yaml</code>. <pre>ndb: ndbdisksize: 60Gi ndbbackupdisksize: 100Gi datamemory: 12G</pre> - Log in to one of the Bastion hosts and get the pvc for data nodes and db replication service: <pre>kubectl get pvc -n <namespace> grep ndbmttd</pre> - If the PVC is not configured as per cnDBTier dimensions, then increase the database capacity by following the scaling procedures below, and get the PVC resources configuration reviewed by the NF team before performing the scaling: - Horizontal scaling by adding more data nodes. - Vertical scaling by increasing the PVC of data nodes.

Table 6-15 (Cont.) BACKUP_FAILED

Field	Details
	5. Validate the network connectivity between data nodes and management nodes and make sure that ports and nodes are reachable. 6. In case the issue persists, capture all the outputs for the above steps and contact My Oracle Support. Note Use CNC NF Data Collector tool for capturing logs. For more information on how to collect logs using Data Collector tool, see Oracle Communications Cloud Native Core, Network Function Data Collector User Guide.

Table 6-16 BACKUP_PURGED_EARLY

Field	Details
Description	This alert is triggered with minor severity when the backup service purges old backups earlier than expected to create space for new backup.
Summary	A backup was deleted prematurely for cnDBTier site {{ \$labels.site_name }} and kubernetes namespace {{ \$labels.namespace }}
Severity	minor
Condition	db_tier_backup{status='PURGED_EARLY'}
Expression Validity	N/A
SNMP Trap ID	2012
Affects Service (Y/N)	N

Table 6-16 (Cont.) BACKUP_PURGED_EARLY

Field	Details
Recommended Action	Cause: When the backup service purges the old backups earlier than the expected time, to create space for a new backup. Diagnostic Information: The <code>db_tier_backup</code> metric (db_tier_backup) provides information if the backup is purged earlier than expected. Recommended Actions: 1. Verify if there is sufficient disk space available on data pods: <pre>kubectl exec -it ndbmt-d-0 -c mysqlndbcluster -n <namespace> -- df -kh</pre> 2. Validate that DBTIER_BACKUP_INFO retains the correct number of backups <code>retainbackupno</code> and purges older one. a. Check the <code>retainbackupno</code> of the cnDBTier cluster: <pre>\$ kubectl exec -it <db-backup-manager- svc pod> -n <namespace> -- printenv RETAIN_BACKUP_NO</pre> b. Identify the exception tables using the following query: <pre>\$ kubectl -n <namespace> exec -it ndbappmysqld-0 -- mysql -h 127.0.0.1 - uroot -p<PASSWORD> mysql> SELECT * FROM backup_info.DBTIER_BACKUP_INFO ORDER BY creation_ts DESC;</pre> 3. Verify the number of data nodes and the DataMemory value configured in the <code>ndbmt-d</code> configuration and ensure that the PVC configured in data nodes aligns with the cnDBTier dimensions. a. Verify the data memory configured in <code>custom_values.yaml</code>. <pre>ndb: ndbdisksize: 60Gi ndbbackupdisksize: 100Gi datamemory: 12G</pre> b. Log in to one of the Bastion hosts and get the pvc for data nodes and db replication service: <pre>kubectl get pvc -n <namespace> grep ndbmt-d</pre>

Table 6-16 (Cont.) BACKUP_PURGED_EARLY

Field	Details
	4. If the PVC is not configured as per cnDBTier dimensions, then increase the database capacity by following the scaling procedures below, and get the PVC resources configuration reviewed by the NF team before performing the scaling: a. Horizontal scaling by adding more data nodes. b. Vertical scaling by increasing the PVC of data nodes. 5. Validate the network connectivity between data nodes and management nodes and make sure that ports and nodes are reachable. 6. In case the issue persists, capture all the outputs for the above steps and contact My Oracle Support. Note Use CNC NF Data Collector tool for capturing logs. For more information on how to collect logs using Data Collector tool, see Oracle Communications Cloud Native Core, Network Function Data Collector User Guide.

Table 6-17 BACKUP_SIZE_GROWTH

Field	Details
Description	This alert is triggered with minor severity whenever the current backup size exceeds 20% of the average of the previous backups.
Summary	Backup size exceeded expected size for cnDBTier site {{ \$labels.site_name }} and kubernetes namespace {{ \$labels.namespace }}
Severity	minor
Condition	$(db\_tier\_backup\_used\_disk\_percentage / (avg\_over\_time(db\_tier\_backup\_used\_disk\_percentage[5d])) > 1.05$
Expression Validity	N/A
SNMP Trap ID	2013
Affects Service (Y/N)	N

Table 6-17 (Cont.) BACKUP_SIZE_GROWTH

Field	Details
Recommended Action	Cause: When the current backup size exceeds 20% of the average of the previous backups. Diagnostic Information: The <code>db_tier_backup_used_disk_percentage</code> metric (db_tier_backup_used_disk_percentage) provides information if the current backup size exceeds 20% of the average. Recommended Actions: 1. This alert detects a spike in the disk usage by comparing the current value of <code>db_tier_backup_used_disk_percentage</code> to the 5 days average. If the spike exceeds a defined <code>backupSizeGrowthAlertThreshold</code> this alert will be triggered. 2. The alert will automatically clear when the size of the current backup does not exceed 20% above the average size of the backups taken over the last five days. 3. In case the issue persists, capture all the outputs for the above steps and contact My Oracle Support. Note Use CNC NF Data Collector tool for capturing logs. For more information on how to collect logs using Data Collector tool, see Oracle Communications Cloud Native Core, Network Function Data Collector User Guide.

Table 6-18 BACKUP_STORAGE_LOW

Field	Details
Description	This alert is triggered with minor severity when the total backup size of the data node is $\geq 70\%$ and $< 80\%$ of the total data node disk size.
Summary	Disk storage on DATA node with node ID <code>{{ \$labels.node_id }}</code> at <code>{{ \$value }}</code> percent for cnDBTier site <code>{{ \$labels.site_name }}</code> and kubernetes namespace <code>{{ \$labels.namespace }}</code>
Severity	minor
Condition	$(\text{avg\_over\_time}(\text{db\_tier\_backup\_used\_disk\_percentage}[5m]) \geq 70)$ and $(\text{avg\_over\_time}(\text{db\_tier\_backup\_used\_disk\_percentage}[5m]) < 80)$
Expression Validity	N/A

Table 6-18 (Cont.) BACKUP_STORAGE_LOW

Field	Details
SNMP Trap ID	2014
Affects Service (Y/N)	N

Table 6-18 (Cont.) BACKUP_STORAGE_LOW

Field	Details
Recommended Action	Cause: When the total backup size of the data node is $\geq 70\%$ and $< 80\%$ of the total data node disk size. Diagnostic Information: The <code>db_tier_backup_used_disk_percentage</code> metric (db_tier_backup_used_disk_percentage) provides information if the current backup size is $\geq 70\%$ and $< 80\%$ of the total data node disk size. Recommended Actions: 1. Verify the number of data nodes and the DataMemory value configured in the <code>ndbmt</code> configuration and ensure that the PVC configured in data nodes aligns with the cnDBTier dimensions. a. Verify the data memory configured in <code>custom_values.yaml</code>. <pre> ndb: ndbdisksize: 60Gi ndbbackupdisksize: 100Gi datamemory: 12G </pre> b. Log in to one of the Bastion hosts and get the pvc for data nodes and db replication service: <pre> kubectl get pvc -n <namespace> grep ndbmt </pre> 2. If the PVC is not configured as per cnDBTier dimensions, then increase the database capacity by following the scaling procedures below, and get the PVC resources configuration reviewed by the NF team before performing the scaling: a. Horizontal scaling by adding more data nodes. b. Vertical scaling by increasing the PVC of data nodes. 3. The alert will be cleared once the <code>db_tier_backup_used_disk_percentage</code> is less than 70% or more than or equal to 80%. 4. In case the issue persists, capture all the outputs for the above steps and contact My Oracle Support. Note Use CNC NF Data Collector tool for capturing logs. For more information on how to collect logs using Data Collector tool, see Oracle Communications Cloud Native Core, Network

Table 6-18 (Cont.) BACKUP_STORAGE_LOW

Field	Details
	 Function Data Collector User Guide.

Table 6-19 BACKUP_STORAGE_LOW

Field	Details
Description	This alert is triggered with major severity when the total backup size of the data node is $\geq 80\%$ and $< 95\%$ of the total data node disk size.
Summary	Disk storage on DATA node with node ID {{ \$labels.node_id }} at {{ \$value }} percent for cnDBTier site {{ \$labels.site_name }} and kubernetes namespace {{ \$labels.namespace }}
Severity	major
Condition	(avg_over_time(db_tier_backup_used_disk_percentage[5m]) ≥ 80) and (avg_over_time(db_tier_backup_used_disk_percentage[5m]) < 95)
Expression Validity	N/A
SNMP Trap ID	2015
Affects Service (Y/N)	N

Table 6-19 (Cont.) BACKUP_STORAGE_LOW

Field	Details
Recommended Action	Cause: When the total backup size of the data node is $\geq 80\%$ and $< 95\%$ of the total data node disk size. Diagnostic Information: The <code>db_tier_backup_used_disk_percentage</code> metric (db_tier_backup_used_disk_percentage) provides information if the current backup size is $\geq 80\%$ and $< 95\%$ of the total data node disk size. Recommended Actions: 1. Verify the number of data nodes and the DataMemory value configured in the <code>ndbmt</code> configuration and ensure that the PVC configured in data nodes aligns with the cnDBTier dimensions. a. Verify the data memory configured in <code>custom_values.yaml</code>. <pre>ndb: ndbdisksize: 60Gi ndbbackupdisksize: 100Gi datamemory: 12G</pre> b. Log in to one of the Bastion hosts and get the pvc for data nodes and db replication service: <pre>kubectl get pvc -n <namespace> grep ndbmt</pre> 2. If the PVC is not configured as per cnDBTier dimensions, then increase the database capacity by following the scaling procedures below, and get the PVC resources configuration reviewed by the NF team before performing the scaling: a. Horizontal scaling by adding more data nodes. b. Vertical scaling by increasing the PVC of data nodes. 3. The alert will be cleared once the <code>db_tier_backup_used_disk_percentage</code> is less than 80% or more than or equal to 95%. 4. In case the issue persists, capture all the outputs for the above steps and contact My Oracle Support. Note Use CNC NF Data Collector tool for capturing logs. For more information on how to collect logs using Data Collector tool, see Oracle Communications Cloud Native Core, Network

Table 6-19 (Cont.) BACKUP_STORAGE_LOW

Field	Details
	 Function Data Collector User Guide.

Table 6-20 BACKUP_STORAGE_FULL

Field	Details
Description	This alert is triggered with critical severity when the total backup size of the data node is $\geq 95\%$ of the total data node disk size.
Summary	Disk storage on DATA node with node ID {{ \$labels.node_id }} is full for cnDBTier site {{ \$labels.site_name }} and kubernetes namespace {{ \$labels.namespace }}
Severity	critical
Condition	(avg_over_time(db_tier_backup_used_disk_percentage[5m]) ≥ 95)
Expression Validity	N/A
SNMP Trap ID	2016
Affects Service (Y/N)	N

Table 6-20 (Cont.) BACKUP_STORAGE_FULL

Field	Details
Recommended Action	Cause: When the total backup size of the data node is $\geq 95\%$ of the total data node disk size. Diagnostic Information: The <code>db_tier_backup_used_disk_percentage</code> metric (db_tier_backup_used_disk_percentage) provides information if the current backup size is $\geq 95\%$ of the total data node disk size. Recommended Actions: 1. Verify the number of data nodes and the DataMemory value configured in the <code>ndbmt</code> configuration and ensure that the PVC configured in data nodes aligns with the cnDBTier dimensions. a. Verify the data memory configured in <code>custom_values.yaml</code>. <pre> ndb: ndbdisksize: 60Gi ndbbackupdisksize: 100Gi datamemory: 12G </pre> b. Log in to one of the Bastion hosts and get the pvc for data nodes and db replication service: <pre> kubectl get pvc -n <namespace> grep ndbmt </pre> 2. If the PVC is not configured as per cnDBTier dimensions, then increase the database capacity by following the scaling procedures below, and get the PVC resources configuration reviewed by the NF team before performing the scaling: a. Horizontal scaling by adding more data nodes. b. Vertical scaling by increasing the PVC of data nodes. 3. The alert will be cleared once the <code>db_tier_backup_used_disk_percentage</code> is less than 95%. 4. In case the issue persists, capture all the outputs for the above steps and contact My Oracle Support. Note Use CNC NF Data Collector tool for capturing logs. For more information on how to collect logs using Data Collector tool, see Oracle Communications Cloud Native Core, Network

Table 6-20 (Cont.) BACKUP_STORAGE_FULL

Field	Details
	 Function Data Collector User Guide.

Table 6-21 DB_TIER_NDB_BACKUP_IN_PROGRESS

Field	Details
Description	This alert is triggered with minor severity when a data node backup is in progress in the current site.
Summary	Indicates that a data node backup process is in progress in the current site.
Severity	minor
Condition	db_tier_ndb_backup_in_progress == 1
Expression Validity	N/A
SNMP Trap ID	2037
Affects Service (Y/N)	N

Table 6-21 (Cont.) DB_TIER_NDB_BACKUP_IN_PROGRESS

Field	Details
Recommended Action	Cause: When a data node backup is in progress in the current site. Diagnostic Information: The <code>db_tier_ndb_backup_in_progress</code> metric (Table 5-35) provides information if a data node backup is in progress or not. Ensure that you don't make any schema changes until the backup completes. Recommended Actions: 1. Wait for the backup to complete. Once completed, the alert should automatically clear. 2. Check if all data pods are up and running by running the following command: <pre>kubectl get pods -n <namespace> -l dbtierapp=ndbmttd --no-headers</pre> 3. Check the logs of the <code>db-backup-manager-svc</code> service: <pre>\$ kubectl logs -n <cnDBTier Namespace> <backup manager service pod name> -f</pre> 4. Verify and check if any backup is in progress state: <pre>\$ kubectl -n cluster1 exec -it mysql- cluster-db-backup-manager-svc-b49488f8f- lbpb -- bash \$ curl -X GET http://mysql-cluster-db- backup-manager-svc:8080/db-tier/backup/ status</pre> 5. In case the issue persists, capture all the outputs for the above steps and contact My Oracle Support. Note Use CNC NF Data Collector tool for capturing logs. For more information on how to collect logs using Data Collector tool, see Oracle Communications Cloud Native Core, Network Function Data Collector User Guide.

6.9 cnDBTier Bin Log Usage Alerts

This section provides details about the cnDBTier binlog usage alerts.

Table 6-22 BINLOG_STORAGE_LOW

Field	Details
Description	This alert is triggered with a minor severity when the total BinLog size of the SQL node is $\geq 70\%$ and $< 80\%$ of the total SQL node disk size.
Summary	Disk storage on SQL node with node ID {{ \$labels.node_id }} at {{ \$value }} percent for cnDBTier site {{ \$labels.site_name }} and kubernetes namespace {{ \$labels.namespace }}
Severity	minor
Condition	(avg_over_time(db_tier_binlog_used_bytes_percentage[5m]) ≥ 70) and (avg_over_time(db_tier_binlog_used_bytes_percentage[5m]) < 80)
Expression Validity	5m
SNMP Trap ID	2007
Affects Service (Y/N)	N

Table 6-22 (Cont.) BINLOG_STORAGE_LOW

Field	Details
Recommended Action	Cause: When the total BinLog size of the SQL node is $\geq 70\%$ and $< 80\%$ of the total SQL node disk size. Diagnostic Information: The <code>db_tier_binlog_used_bytes_percentage</code> metric (db_tier_binlog_used_bytes_percentage) provides information if the total BinLog size of the SQL node is $\geq 70\%$ and $< 80\%$ of total SQL node disk size. Recommended Actions: 1. Verify if the PVC configured for the <code>ndbmysqld</code> pods is as per the cnDBTier dimensions. if PVC is not configured as per the cnDBTier dimensions, then increase the PVC by following the scaling procedures below, and get the PVC configurations reviewed by the NF team before performing the scaling: For more information, see the Scaling cnDBTier Pods section to scale the cnDBTier pod for increasing the CPU allocation. 2. Monitor BinLog growth closely over the next few hours and check if binlogs are getting purged. a. Run the following command inside the <code>ndbmysqld</code> pod: <pre>\$ kubectl exec -it -n <cnDBTier Namespace> ndbmysqld-0 -- mysql - h127.0.0.1 -uroot -pNextGenCne;</pre> b. Run the command to monitor Binlog growth: <pre>\$ show binary logs;</pre> 3. Check if the relay logs are increasing continuously in the <code>ndbmysqld</code> pod. Run the following commands inside <code>ndbmysqld</code> pod and check all the relay logs. It should not be more than 3 or 4 <pre>\$ kubectl exec -it -n <cnDBTier Namespace> ndbmysqld-0 -- bash \$ cd /var/ occnedb/mysld/\$ ls -lrth grep 'relay'</pre> 4. In case the issue persists, capture all the outputs for the above steps and contact My Oracle Support. Note Use CNC NF Data Collector tool for capturing logs. For more information on how to collect logs using Data Collector tool,

Table 6-22 (Cont.) BINLOG_STORAGE_LOW

Field	Details
	see Oracle Communications Cloud Native Core, Network Function Data Collector User Guide.

Table 6-23 BINLOG_STORAGE_LOW

Field	Details
Description	This alert is triggered with major severity when the total BinLog size of the SQL node is $\geq 80\%$ and $< 95\%$ of the total SQL node disk size.
Summary	Disk storage on SQL node with node ID {{ \$labels.node_id }} at {{ \$value }} percent for cnDBTier site {{ \$labels.site_name }} and kubernetes namespace {{ \$labels.namespace }}
Severity	major
Condition	(avg_over_time(db_tier_binlog_used_bytes_percentage[5m]) ≥ 80) and (avg_over_time(db_tier_binlog_used_bytes_percentage[5m]) < 95)
Expression Validity	5m
SNMP Trap ID	2036
Affects Service (Y/N)	N

Table 6-23 (Cont.) BINLOG_STORAGE_LOW

Field	Details
Recommended Action	Cause: When the total BinLog size of the SQL node is $\geq 80\%$ and $< 95\%$ of the total SQL node disk size. Diagnostic Information: The <code>db_tier_binlog_used_bytes_percentage</code> metric (db_tier_binlog_used_bytes_percentage) provides information if the total BinLog size of the SQL node is $\geq 80\%$ and $< 95\%$ of total SQL node disk size. Recommended Actions: 1. Verify if the PVC configured for the <code>ndbmysqld</code> pods is as per the cnDBTier dimensions. If PVC is not configured as per the cnDBTier dimensions, then increase the PVC by following the scaling procedures below, and get the PVC configurations reviewed by the NF team before performing the scaling: For more information, see the Scaling cnDBTier Pods section to scale the cnDBTier pod for increasing the CPU allocation. 2. Monitor BinLog growth closely over the next few hours and check if binlogs are getting purged. a. Run the following command inside the <code>ndbmysqld</code> pod: <pre>\$ kubectl exec -it -n <cnDBTier Namespace> ndbmysqld-0 -- mysql - h127.0.0.1 -uroot -pNextGenCne;</pre> b. Run the command to monitor Binlog growth: <pre>\$ show binary logs;</pre> 3. Check if the relay logs are increasing continuously in the <code>ndbmysqld</code> pod. Run the following commands inside <code>ndbmysqld</code> pod and check all the relay logs. It should not be more than 3 or 4: <pre>\$ kubectl exec -it -n <cnDBTier Namespace> ndbmysqld-0 -- bash \$ cd /var/ occnedb/mysld/\$ ls -lrth grep 'relay'</pre> 4. In case the issue persists, capture all the outputs for the above steps and contact My Oracle Support. Note Use CNC NF Data Collector tool for capturing logs. For more information on how to collect logs using Data Collector tool,

Table 6-23 (Cont.) BINLOG_STORAGE_LOW

Field	Details
	see Oracle Communications Cloud Native Core, Network Function Data Collector User Guide.

Table 6-24 BINLOG_STORAGE_FULL

Field	Details
Description	This alert is triggered with critical severity when the total BinLog size of the SQL node is >= 95% of the total SQL node disk size.
Summary	Disk storage on SQL node with node ID {{ \$labels.node_id }} is full for cnDBTier site {{ \$labels.site_name }} and kubernetes namespace {{ \$labels.namespace }}
Severity	critical
Condition	avg_over_time(db_tier_binlog_used_bytes_percentage[5m]) >= 95
Expression Validity	N/A
SNMP Trap ID	2008
Affects Service (Y/N)	Y

Table 6-24 (Cont.) BINLOG_STORAGE_FULL

Field	Details
Recommended Action	Cause: When the total BinLog size of the SQL node is $\geq$ 95% of the total SQL node disk size. Diagnostic Information: The <code>db_tier_binlog_used_bytes_percentage</code> metric (db_tier_binlog_used_bytes_percentage) provides information if the total BinLog size of the SQL node is $\geq$ 95% of total SQL node disk size. Recommended Actions: 1. Verify if the PVC configured for the <code>ndbmysqld</code> pods is as per the cnDBTier dimensions. if PVC is not configured as per the <code>cndbtier</code> dimensions, then increase the PVC by following the scaling procedures below, and get the PVC configurations reviewed by the NF team before performing the scaling: For more information, see the Scaling cnDBTier Pods section to scale the cnDBTier pod for increasing the CPU allocation. 2. Monitor BinLog growth closely over the next few hours and check if binlogs are getting purged. a. Run the following command inside the <code>ndbmysqld</code> pod: <pre>\$ kubectl exec -it -n <cnDBTier Namespace> ndbmysqld-0 -- mysql - h127.0.0.1 -uroot -pNextGenCne;</pre> b. Run the command to monitor Binlog growth: <pre>\$ show binary logs;</pre> 3. Check if the relay logs are increasing continuously in the <code>ndbmysqld</code> pod. Run the following commands inside <code>ndbmysqld</code> pod and check all the relay logs. It should not be more than 3 or 4 <pre>\$ kubectl exec -it -n <cnDBTier Namespace> ndbmysqld-0 -- bash \$ cd /var/ occnedb/mysld/\$ ls -lrth grep 'relay'</pre> 4. In case the issue persists, capture all the outputs for the above steps and contact My Oracle Support. Note Use CNC NF Data Collector tool for capturing logs. For more information on how to collect logs using Data Collector tool,

Table 6-24 (Cont.) BINLOG_STORAGE_FULL

Field	Details
	see Oracle Communications Cloud Native Core, Network Function Data Collector User Guide.

6.10 cnDBTier Replication Alerts

This section provides details about cnDBTier replication alerts.

Table 6-25 REPLICATION_CHANNEL_DOWN

Field	Details
Description	This alert is triggered with major severity when an ACTIVE channel goes to the FAILED state.
Summary	Cross-site replication is down on node {{ \$labels.node_id }} for cnDBTier site {{ \$labels.site_name }} and kubernetes namespace {{ \$labels.namespace }}
Severity	major
Condition	(db_tier_replication_status{role="failed"} == 0) or (db_tier_replication_status{role="active"} == 0)
Expression Validity	N/A
SNMP Trap ID	2005
Affects Service (Y/N)	N

Table 6-25 (Cont.) REPLICATION_CHANNEL_DOWN

Field	Details
Recommended Action	Cause: When any ACTIVE channel goes to the FAILED state when the crosssite replication is down on a node. Diagnostic Information: The following metrics provide information if the replication channel is down: • <code>db_tier_replication_status{role="failed"} == 0</code> (db_tier_replication_status{role="failed"} == 0) • <code>db_tier_replication_status{role="active"} == 0</code> (db_tier_replication_status{role="active"} == 0) Recommended Actions: 1. Verify the replication status of all the ndbmysqld pods using the 'SHOW REPLICA STATUS' command: <pre>kubectl exec -it ndbmysqld-0 -- namespace=<namesapce>-- mysql - h127.0.0.1 -uroot -p<PASSWORD>-e "SHOW REPLICA STATUS\G"</pre> 2. Check the pod status, if the pod is not coming up then analyze the previous container logs of the pod to see the error information. <pre>\$ kubectl -n <namespace> logs <ndbmttd/ ndbmysqld/ndbappmysqld podname> -- previous</pre> 3. Check the channel details for all cnDBTier sites. Identify the exception tables using the following query: <pre>\$ kubectl -n <namespace> exec -it ndbappmysqld-0 -- mysql -h 127.0.0.1 - uroot -p<PASSWORD> mysql> SELECT remote_site_name,channel_id,role, replchan nel_group_id FROM replication_info.DBTIER_REPLICATION_CHANN EL_INFO;</pre> 4. Check the management pod logs also for checking if there are any frequent heartbeat missed warnings or other alerts are logged for the pod. <pre>\$ kubectl -n <namespace> exec -it ndbmgmd-0 -- bash \$ tail -f /var/occnedb/mysqlndbcluster/ ndbmgmd_cluster.log</pre> 5. Validate the network connectivity between the cluster nodes and make sure that pods and nodes are reachable.

Table 6-25 (Cont.) REPLICATION_CHANNEL_DOWN

Field	Details
	6. In case the issue persists, capture all the outputs for the above steps and contact My Oracle Support. Note Use CNC NF Data Collector tool for capturing logs. For more information on how to collect logs using Data Collector tool, see Oracle Communications Cloud Native Core, Network Function Data Collector User Guide.

Table 6-26 REPLICATION_FAILED

Field	Details
Description	This alert is triggered with critical severity when all the channels are in the STANDBY or FAILED state.
Summary	Cross-site replication is down for cnDBTier site {{ \$labels.site_name }} and kubernetes namespace {{ \$labels.namespace }}
Severity	critical
Condition	(count by (site_name, namespace, replchannel_group_id) (db_tier_replication_status) == count by (site_name, namespace, replchannel_group_id) (db_tier_replication_status{role="standby"})) or (count by (namespace, replchannel_group_id, site_name) (db_tier_replication_status) == count by (namespace, replchannel_group_id, site_name) (db_tier_replication_status{role="failed"}))
Expression Validity	N/A
SNMP Trap ID	2006
Affects Service (Y/N)	Y

Table 6-26 (Cont.) REPLICATION_FAILED

Field	Details
Recommended Action	Cause: When all the channels are in the STANDBY or FAILED state as the cross-site replication is down for the cnDBTier site. Diagnostic Information: The <code>db_tier_replication_status</code> metric (db_tier_replication_status) provides information if the replication failed. Recommended Actions: 1. Verify the replication status of all the ndbmysqld pods using the 'SHOW REPLICA STATUS' command: <pre>kubectl exec -it ndbmysqld-0 -- namespace=<namesapce>-- mysql - h127.0.0.1 -uroot -p<PASSWORD>-e "SHOW REPLICA STATUS\G"</pre> 2. Check the pod status, if the pod is not coming up then analyze the previous container logs of the pod to see the error information. <pre>\$ kubectl -n <namespace> logs <ndbmttd/ ndbmysqld/ndbappmysqld podname> -- previous</pre> 3. Check the channel details for all cnDBTier sites. Retrieve the replication channels configurations <pre>\$ kubectl -n <namespace> exec -it ndbappmysqld-0 -- mysql -h 127.0.0.1 - uroot -p<PASSWORD> mysql> SELECT remote_site_name,channel_id,role,repchan nel_group_id FROM replication_info.DBTIER_REPLICATION_CHANN EL_INFO;</pre> 4. Check the management pod logs also for checking if there are any frequent heartbeat missed warnings or other alerts are logged for the pod. <pre>\$ kubectl -n <namespace> exec -it ndbmgmd-0 -- bash \$ tail -f /var/occnedb/mysqlndbcluster/ ndbmgmd_cluster.log</pre> 5. Validate the network connectivity between the cluster nodes and make sure that pods and nodes are reachable. 6. In case the issue persists, capture all the outputs for the above steps and contact My Oracle Support.

Table 6-26 (Cont.) REPLICATION_FAILED

Field	Details
	 Note Use CNC NF Data Collector tool for capturing logs. For more information on how to collect logs using Data Collector tool, see Oracle Communications Cloud Native Core, Network Function Data Collector User Guide.

Table 6-27 REPLICA_REPLICATION_DELAY_HIGH

Field	Details
Description	This alert is triggered when the last record read by the replica is more than five minutes behind the latest record written by the source.
Summary	Replica replication on SQL node at {{ \$labels.replica_node_ip }} is {{ \$value }} seconds behind the source for cnDBTier site {{ \$labels.site_name }} and kubernetes namespace {{ \$labels.namespace }}
Severity	major
Condition	avg(avg_over_time(db_tier_replication_replica_delay[5m])) by (source_node_ip,replica_node_ip) >= 300 and avg(avg_over_time(db_tier_replication_replica_delay[5m])) by (source_node_ip,replica_node_ip) < 48*3600
Expression Validity	1m
SNMP Trap ID	2009
Affects Service (Y/N)	N

Table 6-27 (Cont.) REPLICA_REPLICATION_DELAY_HIGH

Field	Details
Recommended Action	Cause: When the last record read by the worker node is more than 5 minutes and less than 48 hours behind the latest record written by the controller. Diagnostic Information: The <code>db_tier_replication_replica_delay</code> metric (db_tier_replication_slave_delay) provides information if there is a delay in the worker node replication. Recommended Actions: 1. Verify the replication status of all the <code>ndbmysqld</code> pods using the 'SHOW REPLICA STATUS' command: <pre>kubectl exec -it ndbmysqld-0 -- namespace=<namesapce>-- mysql - h127.0.0.1 -uroot -p<PASSWORD>-e "SHOW REPLICA STATUS\G"</pre> 2. Check the management pod logs also for checking if there are any frequent heartbeat missed warnings or other alerts are logged for the pod. <pre>\$ kubectl -n <namespace> exec -it ndbmgmd-0 -- bash \$ tail -f /var/occnedb/mysqlndbcluster/ ndbmgmd_cluster.log</pre> 3. Validate the network connectivity between the cluster nodes and make sure that pods and nodes are reachable. 4. In case the issue persists, capture all the outputs for the above steps and contact My Oracle Support. Note Use CNC NF Data Collector tool for capturing logs. For more information on how to collect logs using Data Collector tool, see Oracle Communications Cloud Native Core, Network Function Data Collector User Guide.

Table 6-28 REPLICATION_REPLICATION_FAILED

Field	Details
Description	This alert is triggered when the last record read by the replica is more than 48 hours behind the latest record written by the source.
Summary	Replica replication has fallen more than 48 hours behind the source. Manual restore from backup may be required for cnDBTier site {{ \$labels.site_name }} and kubernetes namespace {{ \$labels.namespace }}
Severity	critical
Condition	avg(avg_over_time(db_tier_replication_replica_delay[5m])) by (source_node_ip,replica_node_ip) >= 48*3600
Expression Validity	1m
SNMP Trap ID	2010
Affects Service (Y/N)	Y

Table 6-28 (Cont.) REPLICA_REPLICATION_FAILED

Field	Details
Recommended Action	Cause: When the last record read by the worker node is more than 48 hours behind the latest record written by the controller. Diagnostic Information: The <code>db_tier_replication_replica_delay</code> metric (db_tier_replication_replica_delay) provides information if the worker node replication failed. Recommended Actions: 1. Verify the replication status of all the <code>ndbmysqld</code> pods using the 'SHOW REPLICA STATUS' command: <pre>kubectl exec -it ndbmysqld-0 -- namespace=<namesapce>-- mysql - h127.0.0.1 -uroot -p<PASSWORD>-e "SHOW REPLICA STATUS\G"</pre> 2. Check the channel details for all cnDBTier sites. Retrieve the replication channels configurations <pre>\$ kubectl -n <namespace> exec -it ndbappmysqld-0 -- mysql -h 127.0.0.1 - uroot -p<PASSWORD> mysql> SELECT remote_site_name,channel_id,role,repchan nel_group_id FROM replication_info.DBTIER_REPLICATION_CHANN EL_INFO;</pre> 3. Check the management pod logs also for checking if there are any frequent heartbeat missed warnings or other alerts are logged for the pod. <pre>\$ kubectl -n <namespace> exec -it ndbmgmd-0 -- bash \$ tail -f /var/occnedb/mysqlndbcluster/ ndbmgmd_cluster.log</pre> 4. Check the pod status, if the pod is not coming up then analyze the previous container logs of the pod to see the error information. <pre>\$ kubectl -n <namespace> logs <ndbmttd/ ndbmysqld/ndbappmysqld podname> -- previous</pre> 5. Validate the network connectivity between the cluster nodes and make sure that pods and nodes are reachable. 6. In case the issue persists, capture all the outputs for the above steps and contact My Oracle Support.

Table 6-28 (Cont.) REPLICA_REPLICATION_FAILED

Field	Details
	 Note Use CNC NF Data Collector tool for capturing logs. For more information on how to collect logs using Data Collector tool, see Oracle Communications Cloud Native Core, Network Function Data Collector User Guide.

Table 6-29 GEOREPLICATION_RECOVERY_FAILED

Field	Details
Description	This alert is triggered with critical severity when georeplication recovery fails on a unhealthy site where georeplication recovery was started.
Summary	Georeplication recovery has failed on cnDBTier Site {{ \$labels.site_name }} from kubernetes namespace {{ \$labels.namespace }}
Severity	critical
Condition	db_tier_georeplication_recovery_state == 2
Expression Validity	NA
SNMP Trap ID	2033
Affects Service (Y/N)	Y

Table 6-29 (Cont.) GEOREPLICATION_RECOVERY_FAILED

Field	Details
Recommended Action	Cause: Incorrect disk size, incorrect SSH key configurations, or other similar reasons. Diagnostic Information: This alert indicates that georeplication recovery failed on a unhealthy site and replication couldn't be reestablished using the georeplication recovery procedure. This alert requires immediate attention. Recommended Actions: 1. Verify the replication status of all the ndbmysqld pods using the 'SHOW REPLICA STATUS' command: <pre>kubectl exec -it ndbmysqld-0 -- namespace=<namesapce>-- mysql - h127.0.0.1 -uroot -p<PASSWORD>-e "SHOW REPLICA STATUS\G"</pre> 2. Check the pod status, if the pod is not coming up then analyze the previous container logs of the pod to see the error information. <pre>\$ kubectl -n <namespace> logs <ndbmtld/ ndbmysqld/ndbappmysqld podname> -- previous</pre> 3. Check the channel details for all cnDBTier sites. Retrieve the replication channels configurations: <pre>\$ kubectl -n <namespace> exec -it ndbappmysqld-0 -- mysql -h 127.0.0.1 - uroot -p<PASSWORD> mysql> SELECT remote_site_name,channel_id,role,repchan nel_group_id FROM replication_info.DBTIER_REPLICATION_CHANN EL_INFO;</pre> 4. Check the management pod logs also for checking if there are any frequent heartbeat missed warnings or other alerts are logged for the pod. <pre>\$ kubectl -n <namespace> exec -it ndbmgmd-0 -- bash \$ tail -f /var/occnedb/mysqlndbcluster/ ndbmgmd_cluster.log</pre> 5. Monitor the GRR state during the execution of GEOREPLICATION_RECOVERY and identify the specific state after which GRR is failing. 6. Check if backups can be successfully transferred from a healthy site to the recovering site.

Table 6-29 (Cont.) GEOREPLICATION_RECOVERY_FAILED

Field	Details
	Log in to one of the leader replication svc pod of healthy site: <pre>\$ kubectl -n <namespace> exec -it <leader-replication-pod> -- bash ssh -i /home/mysql/.ssh/id_rsa mysql@<db- replication-service-svc> -p 2022 \$ sftp -i /home/mysql/.ssh/id_rsa -P 2022 -o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null mysql@<leader_db_Replication_service_svc> :/var/occnedb/ <<< '\$put -r <file/dir to be copied>'</pre> 7. Validate the network connectivity between the cluster nodes and make sure that pods and nodes are reachable. 8. In case the issue persists, capture all the outputs for the above steps and contact My Oracle Support. Note Use CNC NF Data Collector tool for capturing logs. For more information on how to collect logs using Data Collector tool, see Oracle Communications Cloud Native Core, Network Function Data Collector User Guide.

6.11 cnDBTier Memory Usage Alerts

This section provides details about the cnDBTier memory usage alerts.

Table 6-30 LOW_MEMORY

Field	Details
Description	This alert is triggered when the RAM usage of any node is greater than or equal to 80%.
Summary	Node ID {{ \$labels.node_id }}, memory utilization at {{ \$value }} percent for memory type {{ \$labels.memory_type }} for cnDBTier site {{ \$labels.site_name }} and kubernetes namespace {{ \$labels.namespace }}
Severity	major

Table 6-30 (Cont.) LOW_MEMORY

Field	Details
Condition	((avg_over_time(db_tier_memory_used_bytes{memory_type="Data memory"}[1m]) / avg_over_time(db_tier_memory_total_bytes{memory_type="Data memory"}[1m])) * 100) >= 80
Expression Validity	1m
SNMP Trap ID	2003
Affects Service (Y/N)	N

Table 6-30 (Cont.) LOW_MEMORY

Field	Details
Recommended Action	Cause: When the RAM or memory usage of any node reaches the major level of threshold value. Diagnostic Information: Check if the memory usage of the following metrics are too high: - db_tier_memory_used_bytes (db_tier_memory_used_bytes) - db_tier_memory_total_bytes (db_tier_memory_total_bytes) Recommended Actions: - Check the memory (Data and Index) usage of the data nodes. <pre>\$ kubectl -n <NAMESPACE> exec -it ndbmgmd-0 -- ndb_mgm -e "ALL REPORT memoryusage"</pre> - Check if the exceptions in the exception tables are continuously increasing, and if the record count is high, involve the NF team to clean up the entries and investigate the root cause. - Identifying the exception tables using the following query: <pre>\$ kubectl -n <namespace> exec -it ndbappmysqld-0 -- mysql -h 127.0.0.1 - uroot -p<PASSWORD>mysql> select TABLE_SCHEMA, TABLE_NAME from information_schema.TABLES where TABLE_NAME = '%\$EX';</pre> - Get the count in the exception table from the exception table lists: <pre>mysql> select count(*) AS COUNT from <ABOVE_TABLE_SCHEMA>.<ABOVE_TABLE_NAME >;</pre> - If the number of subscribers continues to increase, scale the database capacity by following the procedures below. Before proceeding with the scaling, ensure that the memory resources and DataMemory configuration are reviewed and approved by the NF team: - Horizontal scaling by adding more data nodes. - Vertical scaling by increasing the memory allocated to ndbmttd pods and updating the DataMemory setting accordingly. - In case the issue persists, capture all the outputs for the above steps and contact My Oracle Support.

Table 6-30 (Cont.) LOW_MEMORY

Field	Details
	 Note Use CNC NF Data Collector tool for capturing logs. For more information on how to collect logs using Data Collector tool, see Oracle Communications Cloud Native Core, Network Function Data Collector User Guide.

Table 6-31 OUT_OF_MEMORY

Field	Details
Description	This alert is triggered with critical severity when the RAM usage of any node is greater than or equal to 90%.
Summary	Node ID {{ \$labels.node_id }} out of memory for memory type {{ \$labels.memory_type }} for cnDBTier site {{ \$labels.site_name }} and kubernetes namespace {{ \$labels.namespace }}
Severity	critical
Condition	((avg_over_time(db_tier_memory_used_bytes{memory_type="Data memory"}[1m]) / avg_over_time(db_tier_memory_total_bytes{memory_type="Data memory"}[1m])) * 100) >= 90
Expression Validity	1m
SNMP Trap ID	2004
Affects Service (Y/N)	Y

Table 6-31 (Cont.) OUT_OF_MEMORY

Field	Details
Recommended Action	Cause: When the RAM or memory usage of any node reaches the critical level of threshold value. Diagnostic Information: Check if the memory usage of the following metrics are too high: - db_tier_memory_used_bytes (db_tier_memory_used_bytes) - db_tier_memory_total_bytes (db_tier_memory_total_bytes) Recommended Actions: - Check the memory (Data and Index) usage of the data nodes. \$ kubectl -n <NAMESPACE> exec -it ndbmcmd-0 -- ndb_mgm -e "ALL REPORT memoryusage" - Check if the exceptions in the exception tables are continuously increasing, and if the record count is high, involve the NF team to clean up the entries and investigate the root cause. - Identifying the exception tables using the following query: \$ kubectl -n <namespace> exec -it ndbappmysqld-0 -- mysql -h 127.0.0.1 -u root -p<PASSWORD>mysql> select TABLE_SCHEMA, TABLE_NAME from information_schema.TABLES where TABLE_NAME = '%\$EX'; - Get the count in the exception table from the exception table lists: mysql> select count(*) AS COUNT from <ABOVE_TABLE_SCHEMA>.<ABOVE_TABLE_NAME>; - If the number of subscribers continues to increase, scale the database capacity by following the procedures below. Before proceeding with the scaling, ensure that the memory resources and DataMemory configuration are reviewed and approved by the NF team: - Horizontal scaling by adding more data nodes. - Vertical scaling by increasing the memory allocated to ndbmtd pods and updating the DataMemory setting accordingly. - In case the issue persists, capture all the outputs for the above steps and contact My Oracle Support.

Table 6-31 (Cont.) OUT_OF_MEMORY

Field	Details
	 Note Use CNC NF Data Collector tool for capturing logs. For more information on how to collect logs using Data Collector tool, see Oracle Communications Cloud Native Core, Network Function Data Collector User Guide.

6.12 cnDBTier CPU Usage Alerts

This section provides details about cnDBTier CPU usage alerts.

Table 6-32 HIGH_CPU

Field	Details
Description	This alert is triggered with major severity when the CPU usage of any data node is greater than or equal to 80%, and less than 90%.
Summary	Node ID {{ \$labels.node_id }} CPU utilization at {{ \$value }} for cnDBTier site {{ \$labels.site_name }} and kubernetes namespace {{ \$labels.namespace }}
Severity	major
Condition	((100 - (avg(avg_over_time(db_tier_cpu_os_idle[10m])) by (node_id))) >= 80) and ((100 - (avg(avg_over_time(db_tier_cpu_os_idle[10m])) by (node_id))) < 90)
Expression Validity	1m
SNMP Trap ID	2002
Affects Service (Y/N)	N

Table 6-32 (Cont.) HIGH_CPU

Field	Details
Recommended Action	Cause: When the CPU utilization of any data node is greater than or equal to 80%, and less than 90%. Diagnostic Information: Check the CPU threshold level status from the cnDBTier worker pod logs. Recommended Actions: 1. Monitor the CPU utilization of the pods. 2. If CPU is not configured as per the cnDBTier dimensions, increase the CPU allocation by following the scaling procedures below. Ensure that the updated CPU configuration is reviewed and approved by the NF team before proceeding with the scaling operation. For more information, see the Scaling ndbmttd Pods section to scale cndbtier pod for increasing the CPU allocation. 3. Check if the traffic is exceeding the configured cnDBTier capacity as per the cnDBTier dimensions. 4. Check the logs of the specific node where CPU utilization is high. kubectl -n <namespace> logs <ndbmttd/ ndbmysqld/ndbappmysqld podname> 5. Verify if any worker nodes, network or infrastructure-related events occurred that could be causing the CPU starvation for the pods. 6. In case the issue persists, capture all the outputs for the above steps and contact My Oracle Support. Note Use CNC NF Data Collector tool for capturing logs. For more information on how to collect logs using Data Collector tool, see Oracle Communications Cloud Native Core, Network Function Data Collector User Guide.

Table 6-33 HIGH_CPU

Field	Details
Description	This alert is triggered with critical severity when the CPU usage of any data node is greater than or equal to 90%.
Summary	Node ID {{ \$labels.node_id }} CPU utilization at {{ \$value }} for cnDBTier site {{ \$labels.site_name }} and kubernetes namespace {{ \$labels.namespace }}
Severity	critical

Table 6-33 (Cont.) HIGH_CPU

Field	Details
Condition	$(100 - (\text{avg}(\text{avg\_over\_time}(\text{db\_tier\_cpu\_os\_idle}[10\text{m}]))\text{BY}(\text{node\_id}))) \geq 90$
Expression Validity	1m
SNMP Trap ID	2035
Affects Service (Y/N)	N
Recommended Action	Cause: When the CPU utilization of any data node is greater than or equal to 90%. Diagnostic Information: Check the CPU threshold level status from the cnDBTier worker pod logs. Recommended Actions: 1. Monitor the CPU utilization of the pods. 2. If CPU is not configured as per the cnDBTier dimensions, increase the CPU allocation by following the scaling procedures below. Ensure that the updated CPU configuration is reviewed and approved by the NF team before proceeding with the scaling operation. For more information, see the Scaling ndbmttd Pods section to scale cnDBTier pod for increasing the CPU allocation. 3. Check if the traffic is exceeding the configured cnDBTier capacity as per the cnDBTier dimensions. 4. Check the logs of the specific node where CPU utilization is high. <code>kubectl -n <namespace> logs <ndbmttd/ndbmysqld/ndbappmysqld podname></code> 5. Verify if any worker nodes, network or infrastructure-related events occurred that could be causing the CPU starvation for the pods. 6. In case the issue persists, capture all the outputs for the above steps and contact My Oracle Support. Note Use CNC NF Data Collector tool for capturing logs. For more information on how to collect logs using Data Collector tool, see Oracle Communications Cloud Native Core, Network Function Data Collector User Guide.

6.13 cnDBTier Node Status Alerts

The section provides details about cnDBTier node status alerts.

Table 6-34 NODE_DOWN

Field	Details
Description	This alert is raised with critical severity when the data node is down. db_tier_node_status value: • 0: indicates that a node is DOWN • 1: indicates that the node is UP
Summary	MySQL {{ \$labels.node_type }} node having node id {{ \$labels.node_id }} is down for cnDBTier site {{ \$labels.site_name }} and kubernetes namespace {{ \$labels.namespace }}
Severity	critical
Condition	db_tier_node_status == 0
Expression Validity	N/A
SNMP Trap ID	2001
Affects Service (Y/N)	Y

Table 6-34 (Cont.) NODE_DOWN

Field	Details
Recommended Action	Cause: - When pod restarts due to Kubernetes liveness or readiness probe failures - When cnDBTier application restarts or fails to start Diagnostic Information: - Run the following command to check the status of cnDBTier namespace: <pre>kubectl -n <namespace> exec -it ndbmgmd-0 -- ndb_mgm -e show</pre> - Check the Kubernetes events for probe failures in the platform logs. - Check if any exception is reported in the cnDBTier application logs. Recommended Actions: - Refer to the application logs on Kibana and filter based on pod name. - Check the pod status, if the pod is not coming up then analyze the previous container logs of the pod to see the error information. <pre>\$ kubectl -n <namespace> logs <ndbmtld/ ndbmysqld/ndbappmysqld podname> -- previous</pre> - Check the management pod logs also for checking if there are any frequent heartbeat missed warnings are logged for the pod. <pre>\$ kubectl -n <namespace> exec -it ndbmgmd-0 -- bash \$ tail -f /var/occnedb/mysqlndbcluster/ ndbmgmd_cluster.log</pre> - Check if the pod is able to connect to the management pods from pod to other pods. - If the pod alone is down and all the other pods are up and running then delete the pod and its pvc, so that pod can restart by re-initializing. <pre>\$ kubectl -n <namespace> exec -it <ndbmtld/ndbmysqld/ndbappmysqld podname> - c db-infra-monitor-svc -- bash \$ ndb_mgm -c ndbmgmd-0.ndbmgmdsvc:1186 - e show</pre> Follow the Disaster Recovery guide for restoring the single node.

Table 6-34 (Cont.) NODE_DOWN

Field	Details
	6. In case the issue persists, capture all the outputs for the above steps and contact My Oracle Support. Note Use CNC NF Data Collector tool for capturing logs. For more information on how to collect logs using Data Collector tool, see Oracle Communications Cloud Native Core, Network Function Data Collector User Guide.

6.14 cnDBTier Node Data Volume Alerts

This section provides details about cnDBTier node data volume alerts.

Table 6-35 DB_TIER_API_SEND_NODE_DATA_VOLUME_LOW

Field	Details
Description	This alert is triggered when any NDB application node sends less data to NDB when compared to the other NDB application nodes.
Summary	Send Node Data Volume Low for API Node ID {{ \$labels.remote_node_id }} at kubernetes namespace {{ \$labels.namespace }}
Severity	critical
Condition	(((sum by (remote_node_id,namespace) (avg_over_time(rate(db_tier_node_transporter_bytes_received{node_type=~"ndbapp_node",namespace=~"\$CNDBTIER_NAMESPACE"}[5m])[15m:5m])))/scalar(sum (avg_over_time(rate(db_tier_node_transporter_bytes_received{node_type=~"ndbapp_node",namespace=~"\$CNDBTIER_NAMESPACE"}[5m])[15m:5m]))))*100) < (100/(scalar(count(count by (remote_node_id) (db_tier_node_transporter_bytes_received{node_type=~"ndbapp_node",namespace=~"\$CNDBTIER_NAMESPACE"}[5m])[15m:5m]))))*1.6))
Expression Validity	NA
SNMP Trap ID	3001
Affects Service (Y/N)	Y

Table 6-35 (Cont.) DB_TIER_API_SEND_NODE_DATA_VOLUME_LOW

Field	Details
Recommended Action	Cause: When NDB application node sends less data to NDB when compared to other NDB application nodes. Diagnostic Information: The alert indicates that the NDB application node is slow, therefore check the underlying infrastructure. Recommended Actions: 1. Identify the worker node. \$ kubectl get nodes -o wide 2. Check if an SSH connection can be successfully established to the worker node. 3. Capture a PCAP file and analyze it to identify any network disconnects or transmission activity. 4. Check the file system and underlying hardware for any issues that might be causing slowness. 5. In case the issue persists, capture all the outputs for the above steps and contact My Oracle Support. Note Use CNC NF Data Collector tool for capturing logs. For more information on how to collect logs using Data Collector tool, see Oracle Communications Cloud Native Core, Network Function Data Collector User Guide.

Table 6-36 DB_TIER_API_RECEIVE_NODE_DATA_VOLUME_LOW

Field	Details
Description	This alert is triggered when any NDB sends less data to any specific NDB application node when compared to the other NDB application nodes.
Summary	Receive Node Data Volume Low for API Node ID {{ \$labels.remote_node_id }} at kubernetes namespace {{ \$labels.namespace }}
Severity	critical

Table 6-36 (Cont.) DB_TIER_API_RECEIVE_NODE_DATA_VOLUME_LOW

Field	Details
Condition	$(((\text{sum by (remote\_node\_id, namespace)} \\ (\text{avg\_over\_time}(\text{rate}(\text{db\_tier\_node\_transporter\_bytes\_sent}\{\text{node\_type}=\sim\text{"ndbapp\_node", namespace}=\text{"<\$"} \\ \{\text{CNDBTIER\_NAMESPACE}\}>\text{"}\}[5\text{m}]\}[15\text{m}:5\text{m}]\]))) / \text{scalar}(\text{sum} \\ (\text{avg\_over\_time}(\text{rate}(\text{db\_tier\_node\_transporter\_bytes\_sent}\{\text{node\_type}=\sim\text{"ndbapp\_node", namespace}=\text{"<\$"} \\ \{\text{CNDBTIER\_NAMESPACE}\}>\text{"}\}[5\text{m}]\}[15\text{m}:5\text{m}]\]))) * 100) < \\ (100 / (\text{scalar}(\text{count}(\text{count by (remote\_node\_id)} \\ (\text{db\_tier\_node\_transporter\_bytes\_sent}\{\text{node\_type}=\sim\text{"ndbapp\_node", namespace}=\text{"<\$"} \\ \{\text{CNDBTIER\_NAMESPACE}\}>\text{"}\})))) * 1.6)))$
Expression Validity	NA
SNMP Trap ID	3002
Affects Service (Y/N)	Y
Recommended Action	Cause: When NDB application node sends less data to any specific NDB application node when compared to other NDB application nodes due to the communication not happening intermittently or fully or the underlying platform is slow. Diagnostic Information: The alert indicates that the NDB application node is slow, or underlying infrastructure is slow or communication did not happen intermittently or fully, therefore check the underlying infrastructure. Recommended Actions: 1. Identify the worker node. \$ kubectl get nodes -o wide 2. Check if an SSH connection can be successfully established to the worker node. 3. Capture a PCAP file and analyze it to identify any network disconnects or transmission activity. 4. Check the file system and underlying hardware for any issues that might be causing the slowness. 5. In case the issue persists, capture all the outputs for the above steps and contact My Oracle Support. Note Use CNC NF Data Collector tool for capturing logs. For more information on how to collect logs using Data Collector tool, see Oracle Communications Cloud Native Core, Network Function Data Collector User Guide.

Table 6-37 DB_TIER_SEND_DATA_NODE_DATA_VOLUME_LOW

Field	Details
Description	This alert is triggered when any NDB doesn't send the traffic data in the required speed or when the speed is slower when compared to another data node.
Summary	Send Data Node Data Volume Low for DATA Node ID {{ \$labels.node_id }} at kubernetes namespace {{ \$labels.namespace }}
Severity	critical
Condition	((sum by (node_id,namespace) (avg_over_time(rate(db_tier_node_transporter_bytes_sent{namespace="<\${CNDBTIER_NAMESPACE}>"}[5m])[15m:5m]))/scalar(sum (avg_over_time(rate(db_tier_node_transporter_bytes_sent{namespace="<\${CNDBTIER_NAMESPACE}>"}[5m])[15m:5m]))) * 100) < (100/(scalar(count(count by (node_id) (db_tier_node_transporter_bytes_sent{namespace="<\${CNDBTIER_NAMESPACE}>"}))))*1.6))
Expression Validity	NA
SNMP Trap ID	3003
Affects Service (Y/N)	Y

Table 6-37 (Cont.) DB_TIER_SEND_DATA_NODE_DATA_VOLUME_LOW

Field	Details
Recommended Action	Cause: When any NDB doesn't send the traffic data in the required speed or when the speed is slower when compared to another data node. Diagnostic Information: The alert indicates that the NDB application node is slow, therefore check the underlying infrastructure. Recommended Actions: 1. Identify the worker node. \$ kubectl get nodes -o wide 2. Check if an SSH connection can be successfully established to the worker node. 3. Capture a PCAP file and analyze it to identify any network disconnects or transmission activity. 4. Check the file system and underlying hardware for any issues that might be causing the slowness. 5. In case the issue persists, capture all the outputs for the above steps and contact My Oracle Support. Note Use CNC NF Data Collector tool for capturing logs. For more information on how to collect logs using Data Collector tool, see Oracle Communications Cloud Native Core, Network Function Data Collector User Guide.

Table 6-38 DB_TIER_RECEIVE_DATA_NODE_DATA_VOLUME_LOW

Field	Details
Description	This alert is triggered when any NDB doesn't receive the traffic data in the required speed or when the speed is slower when compared to another data node.
Summary	Receive Data Node Data Volume Low for DATA Node ID {{ \$labels.node_id }} at kubernetes namespace {{ \$labels.namespace }}
Severity	critical

Table 6-38 (Cont.) DB_TIER_RECEIVE_DATA_NODE_DATA_VOLUME_LOW

Field	Details
Condition	$((\text{sum by (node\_id, namespace)} \\ (\text{avg\_over\_time}(\text{rate}(\text{db\_tier\_node\_transporter\_bytes\_received}\{\text{namespace}="<\{\text{CNDBTIER\_NAMESPACE}\}>"\}[5\text{m}]) \\ [15\text{m}:5\text{m}]))/\text{scalar}(\text{sum} \\ (\text{avg\_over\_time}(\text{rate}(\text{db\_tier\_node\_transporter\_bytes\_received}\{\text{namespace}="<\{\text{CNDBTIER\_NAMESPACE}\}>"\}[5\text{m}]) \\ [15\text{m}:5\text{m}])))) * 100) < (100/(\text{scalar}(\text{count}(\text{count by (node\_id)} \\ (\text{db\_tier\_node\_transporter\_bytes\_received}\{\text{namespace}="<\{\text{CNDBTIER\_NAMESPACE}\}>"\}))) * 1.6))$
Expression Validity	NA
SNMP Trap ID	3004
Affects Service (Y/N)	Y
Recommended Action	Cause: When any NDB doesn't receive the traffic data in the required speed or when the speed is slower when compared to another data node. Diagnostic Information: The alert indicates that the NDB application node is slow, therefore check the underlying infrastructure. Recommended Actions: 1. Identify the worker node. \$ kubectl get nodes -o wide 2. Check if an SSH connection can be successfully established to the worker node. 3. Capture a PCAP file and analyze it to identify any network disconnects or transmission activity. 4. Check the file system and underlying hardware for any issues that might be causing slow response. 5. In case the issue persists, capture all the outputs for the above steps and contact My Oracle Support. For any assistance, contact My Oracle Support. Note Use CNC NF Data Collector tool for capturing logs. For more information on how to collect logs using Data Collector tool, see Oracle Communications Cloud Native Core, Network Function Data Collector User Guide.

Table 6-39 DB_TIER_DATA_NODE_SCAN_FRAGMENT_SLOW

Field	Details
Description	This alert is triggered when any data node scan fragment is slow when compared with other data nodes.
Summary	Scan Fragment is Slow for DATA Node ID {{ \$labels.node_id }} at kubernetes namespace {{ \$labels.namespace }}
Severity	critical
Condition	$(((\text{sum by (node\_id,comm\_node\_id,namespace)} \\ (\text{rate}(\text{db\_tier\_tc\_time\_track\_stats\_total\_scan\_fragments\_time}\{\text{path\_type}=\text{"INTERNAL"},\text{namespace}=\text{"<"}\{\text{CNDBTIER\_NAMESPACE}\}>\text{"}[15m]}\}))/\text{sum by} \\ (\text{node\_id,comm\_node\_id,namespace)} \\ (\text{rate}(\text{db\_tier\_tc\_time\_track\_stats\_total\_scan\_fragments\_count}\{\text{path\_type}=\text{"INTERNAL"},\text{namespace}=\text{"<"}\{\text{CNDBTIER\_NAMESPACE}\}>\text{"}[15m]}\}))/(\text{scalar}(\text{sum}(\text{sum by} \\ (\text{node\_id,comm\_node\_id,namespace)} \\ (\text{rate}(\text{db\_tier\_tc\_time\_track\_stats\_total\_scan\_fragments\_time}\{\text{path\_type}=\text{"INTERNAL"},\text{namespace}=\text{"<"}\{\text{CNDBTIER\_NAMESPACE}\}>\text{"}[15m]}\}))/\text{sum by} \\ (\text{node\_id,comm\_node\_id,namespace)} \\ (\text{rate}(\text{db\_tier\_tc\_time\_track\_stats\_total\_scan\_fragments\_count}\{\text{path\_type}=\text{"INTERNAL"},\text{namespace}=\text{"<"}\{\text{CNDBTIER\_NAMESPACE}\}>\text{"}[15m]}\}))))))\ast 100) > ((100/ \\ \text{scalar}(\text{count}(\text{sum by (node\_id,comm\_node\_id,namespace)} \\ (\text{db\_tier\_tc\_time\_track\_stats\_total\_scan\_fragments\_time}\{\text{path\_type}=\text{"INTERNAL"},\text{namespace}=\text{"<"}\{\text{CNDBTIER\_NAMESPACE}\}>\text{"}[15m]}\}))))))\ast 1.6)$
Expression Validity	NA
SNMP Trap ID	3005
Affects Service (Y/N)	Y

Table 6-39 (Cont.) DB_TIER_DATA_NODE_SCAN_FRAGMENT_SLOW

Field	Details
Recommended Action	Cause: When the scan fragment for any particular data node is slow. Diagnostic Information: The alert indicates that the NDB application node is slow, therefore check the underlying infrastructure. Recommended Actions: 1. Identify the worker node. \$ kubectl get nodes -o wide 2. Check if an SSH connection can be successfully established to the worker node. 3. Capture a PCAP file and analyze it to identify any network disconnects or transmission activity. 4. Check the file system and underlying hardware for any issues that might be causing slow response. 5. In case the issue persists, capture all the outputs for the above steps and contact My Oracle Support. Note Use CNC NF Data Collector tool for capturing logs. For more information on how to collect logs using Data Collector tool, see Oracle Communications Cloud Native Core, Network Function Data Collector User Guide.

6.15 cnDBTier Certificate Expiry Alerts

This section provides details about cnDBTier certificate expiry alerts.

Table 6-40 DBTIER_CERTIFICATE_EXPIRY_INFO

Field	Details
Description	This alert is triggered with <code>info</code> severity whenever the certificate for a cnDBTier is set to expire within the next 90 days.
Summary	dbtier Certificate {{ \$labels.certType }}for {{ \$labels.hostname }} is expiring with in 90 days for cnDBTier site {{ \$labels.site_name }}and kubernetes namespace {{ \$labels.namespace }}
Severity	info

Table 6-40 (Cont.) DBTIER_CERTIFICATE_EXPIRY_INFO

Field	Details
Condition	(db_tier_cert_expiry / 1000 - time()) > 2592000 and (db_tier_cert_expiry / 1000 - time()) <= 7776000
Expression Validity	NA
OID	1.3.6.1.4.1.323.5.3.50.1.2.2045
Metric Used	db_tier_cert_expiry
Affects Service (Y/N)	N
Recommended Action	Cause: This alert is triggered when any cnDBTier certificate is going to expire in next 90 days. Diagnostic Information: This alert is triggered with info severity whenever the certificate for a cnDBTier is set to expire within the next 90 days. Recommended Actions: 1. Update the cnDBTier certificates by following the steps provided in the "Update Certificate" section in <i>Oracle Communications Cloud Native Core, cnDBTier User Guide</i>. 2. In case the issue persists, capture all the outputs for the above steps and contact My Oracle Support. Note Use CNC NF Data Collector tool for capturing logs. For more information on how to collect logs using Data Collector tool, see Oracle Communications Cloud Native Core, Network Function Data Collector User Guide.
Available in OCI	Yes

Table 6-41 DBTIER_CERTIFICATE_EXPIRY_MAJOR

Field	Details
Description	This alert is triggered with major severity whenever the certificate for a cnDBTier is set to expire within the next 30 days.
Summary	dbtier Certificate {{ \$labels.certType }}for {{ \$labels.hostname }} is expiring with in 30 days for cnDBTier site {{ \$labels.site_name }}and kubernetes namespace {{ \$labels.namespace }}
Severity	major
Condition	(db_tier_cert_expiry / 1000 - time()) > 604800 and (db_tier_cert_expiry / 1000 - time()) <= 2592000
OID	1.3.6.1.4.1.323.5.3.50.1.2.2040
Metric Used	db_tier_cert_expiry
Expression Validity	NA
Affects Service (Y/N)	N

Table 6-41 (Cont.) DBTIER_CERTIFICATE_EXPIRY_MAJOR

Field	Details
Recommended Action	Cause: This alert is triggered when any cnDBTier certificate is going to expire in next 30 days. Diagnostic Information: This alert is triggered with info severity whenever the certificate for a cnDBTier is set to expire within the next 30 days. Recommended actions: 1. Update the cnDBTier certificates by following the steps provided in the "Update Certificate" section in <i>Oracle Communications Cloud Native Core, cnDBTier User Guide</i>. Depending on the certificate type in alerts, follow the below procedures appropriately: • Modifying cnDBTier Certificates to Establish TLS for Communication with NFs • Modifying HTTPS Certificates 2. In case the issue persists, capture all the outputs for the above steps and contact My Oracle Support. Note Use CNC NF Data Collector tool for capturing logs. For more information on how to collect logs using Data Collector tool, see Oracle Communications Cloud Native Core, Network Function Data Collector User Guide.
Available in OCI	Yes

Table 6-42 DBTIER_CERTIFICATE_EXPIRY_CRITICAL

Field	Details
Description	This alert is triggered with critical severity whenever the certificate for a cnDBTier is set to expire within the next 7 days.
Summary	dbtier Certificate {{ \$labels.certType }}for {{ \$labels.hostname }} is expiring with in 7 days for cnDBTier site {{ \$labels.site_name }}and kubernetes namespace {{ \$labels.namespace }}
Severity	critical
Condition	(db_tier_cert_expiry / 1000 - time()) > 0 and (db_tier_cert_expiry / 1000 - time()) <= 604800
OID	1.3.6.1.4.1.323.5.3.50.1.2.2041
Metric Used	db_tier_cert_expiry
Expression Validity	NA
Affects Service (Y/N)	Y

Table 6-42 (Cont.) DBTIER_CERTIFICATE_EXPIRY_CRITICAL

Field	Details
Recommended Action	Cause: This alert is triggered when any cnDBTier certificate is going to expire in next 7 days. Diagnostic Information: This alert is triggered with critical severity whenever the certificate for a cnDBTier is set to expire within the next 7 days. Recommended actions: 1. Update the cnDBTier certificates by following the steps provided in the "Update Certificate" section in <i>Oracle Communications Cloud Native Core, cnDBTier User Guide</i>. Depending on the certificate type in alerts, follow the below procedures appropriately: • Modifying cnDBTier Certificates to Establish TLS for Communication with NFs • Modifying HTTPS Certificates 2. In case the issue persists, capture all the outputs for the above steps and contact My Oracle Support. Note Use CNC NF Data Collector tool for capturing logs. For more information on how to collect logs using Data Collector tool, see Oracle Communications Cloud Native Core, Network Function Data Collector User Guide.
Available in OCI	Yes

Table 6-43 DBTIER_CERTIFICATE_EXPIRED

Field	Details
Description	This alert is triggered with critical severity when any cnDBTier certificate has expired.
Summary	dbtier Certificate {{ \$labels.certType }}for {{ \$labels.hostname }} is expiredfor cnDBTier site {{ \$labels.site_name }}and kubernetes namespace {{ \$labels.namespace }}
Severity	critical
Condition	(db_tier_cert_expiry / 1000 - time()) <= 0
OID	1.3.6.1.4.1.323.5.3.50.1.2.2041
Metric Used	db_tier_cert_expiry
Expression Validity	NA
Affects Service (Y/N)	Y

Table 6-43 (Cont.) DBTIER_CERTIFICATE_EXPIRED

Field	Details
Recommended Action	Cause: This alert is triggered when any cnDBTier certificate is going to expire in next 7 days. Diagnostic Information: This alert is triggered with critical severity whenever the certificate for a cnDBTier is set to expire within the next 7 days. Recommended actions: 1. Update the cnDBTier certificates by following the steps provided in the "Update Certificate" section in <i>Oracle Communications Cloud Native Core, cnDBTier User Guide</i>. Depending on the certificate type in alerts, follow the below procedures appropriately: • Modifying cnDBTier Certificates to Establish TLS for Communication with NFs • Modifying HTTPS Certificates 2. In case the issue persists, capture all the outputs for the above steps and contact My Oracle Support. Note Use CNC NF Data Collector tool for capturing logs. For more information on how to collect logs using Data Collector tool, see Oracle Communications Cloud Native Core, Network Function Data Collector User Guide.
Available in OCI	Yes

6.16 cnDBTier PVC Health Alerts

This section provides details about cnDBTier PVC health related alerts.

Table 6-44 PVC_NOT_ACCESSIBLE

Field	Details
Description	This alert is triggered with critical severity when db_tier_pvc_is_accesible condition is zero. • If the value of db_tier_pvc_is_accesible is 0, indicates that PVC is not accessible. • If the value of db_tier_pvc_is_accesible is 1, indicates that PVC is accessible.
Summary	PVC is not accessible on cnDBTier site { { \$labels.site_name } }
Severity	critical
Condition	db_tier_pvc_is_accesible == 0
OID	1.3.6.1.4.1.323.5.3.50.1.2.2029
Metric Used	db_tier_pvc_is_accesible

Table 6-44 (Cont.) PVC_NOT_ACCESSIBLE

Field	Details
Expression Validity	1m
Affects Service (Y/N)	Y

Table 6-44 (Cont.) PVC_NOT_ACCESSIBLE

Field	Details
Recommended Action	Cause: When PVC is not accessible for read or write operation. Diagnostic Information: The <code>db_tier_pvc_is_accessible</code> metric provides information about the PVC is accessible or not. Recommended Actions: 1. Verify the Cluster and Pod status. Run the following command to check the cluster status: <pre>kubectl -n <cnDBTier Namespace> exec -it ndbmcmd-0 -- ndb_mgm -e show</pre> Run the following command to check the pod status: <pre>kubectl get pod -n <cnDBTier Namespace></pre> 2. Retrieve the pod name whose PVC is not accessible using <code>db_tier_pvc_is_accessible</code> metric. It is the <code>hostname</code> attribute. 3. After retrieving the the name of the pod, get the PVC associated with it and describe the pod as well as PVC. Look for PVC bound status, any mount errors, events indicating mount failure or volume timeout. To describe the pod, run the following command: <pre>kubectl -n <cnDBTier Namespace> describe pod <pod name></pre> To retrieve the PVC associated with the Pod, run the following command: <pre>kubectl -n <cnDBTier Namespace> get pvc</pre> To describe the PVC, run the following command: <pre>kubectl -n <cnDBTier Namespace> describe pvc <pvc name></pre> 4. Check PVC Mounting Inside the pod. Get <code>mount_path</code> from the <code>db_tier_pvc_is_accessible</code>. If <code>mount_path</code> is missing or empty. It confirms the PVC isn't mounted properly. Run the following commands to check the <code>mount_path</code> by logging in to the pod: <pre>kubectl -n <cnDBTier Namespace> exec -it <pod name> -- bash</pre>

Table 6-44 (Cont.) PVC_NOT_ACCESSIBLE

Field	Details
	<pre>ls -l /var/occnedb df -h grep occnedb</pre> 5. Restart the pod. Sometimes a simple restart remounts the PVC correctly. Restart the podkubectl -n <cnDBTier Namespace> delete pod <pod name> 6. Check the logs of the pod. Run the following command to check the logs of the main container: <pre>kubectl -n <cnDBTier Namespace> logs <pod name> -c <main container name></pre> Run the following command to check logs of the infra monitor svc container: <pre>kubectl -n <cnDBTier Namespace> logs <pod name> -c <db-infra-monitor-svc container name></pre> 7. This alert will be cleared automatically when the PVC metric db_tier_pvc_failure_count become zero. In case if the issue persists, capture all the outputs for the above steps and contact My Oracle Support. Note Use CNC NF Data Collector tool for capturing logs. For more information on how to collect logs using Data Collector tool, see <i>Oracle Communications Cloud Native Core, Network Function Data Collector User Guide</i>.

Table 6-45 PVC_STORAGE_FULL

Field	Details
Description	The PVC_STORAGE_FULL alert is triggered with critical severity when a pod's PVC reaches full capacity.
Summary	PVC is not accessible on cnDBTier site {{ \$labels.site_name }}

Table 6-45 (Cont.) PVC_STORAGE_FULL

Field	Details
Severity	critical
Condition	db_tier_pvc_is_accessible == 0
OID	1.3.6.1.4.1.323.5.3.50.1.2.2029
Metric Used	db_tier_pvc_is_accessible
Expression Validity	1m
Affects Service (Y/N)	Y

Table 6-45 (Cont.) PVC_STORAGE_FULL

Field	Details
Recommended Action	Cause: This alert is triggered when the PVC reaches full capacity, preventing further write operations.. Diagnostic Information: The system detects that the PVC has no available space, leading to storage-related failures. Recommended steps: 1. Verify the Cluster and Pod status. Run the following command to check the cluster status: <pre>kubectl -n <cnDBTier Namespace> exec -it ndbmgmd-0 -- ndb_mgm -e show</pre> Run the following command to check the pod status: <pre>kubectl get pod -n <cnDBTier Namespace></pre> 2. From <code>db_tier_volume_stats_used_bytes</code> get the name of the pod whose pvc storage is full. It is there as the <code>hostname</code> attribute. 3. Verify that the cnDBTier pods are configured with the resources (CPUs, Memory and PVC size) as per cnDBTier dimensions. 4. If the PVC is not configured as per cnDBTier dimensions, then increase the database capacity by following the scaling procedures below before performing the scaling: • Vertical ScalingHorizontal Scaling 5. Ensure that the application managing the PVC properly handles storage utilization. This alert will be cleared automatically when sufficient space becomes available. In case if the issue persists, capture all the outputs for the above steps and and contact My Oracle Support. Note Use CNC NF Data Collector tool for capturing logs. For more information on how to collect logs using Data Collector tool, see <i>Oracle Communications Cloud Native Core, Network Function Data Collector User Guide</i>.

6.17 cnDBTier Backup Manager Svc Down Alerts

This section provides details about cnDBTier backup manager Svc down alerts.

Table 6-46 DB_BACKUP_MANAGER_SVC_DOWN

Field	Details
Description	This alert is triggered with <code>critical</code> severity when <code>db_backup_manager_svc</code> pod is down.
Summary	PVC is not accessible on cnDBTier site { { \$labels.site_name } }
Severity	<code>critical</code>
Condition	<code>kube_deployment_status_replicas_available{deployment=~".*db-backup-manager-svc.*"} == 0</code>
OID	1.3.6.1.4.1.323.5.3.50.1.2.2039
Metric Used	<code>kube_deployment_status_replicas_available{deployment=~".*db-backup-manager-svc.*"}</code>
Expression Validity	1m
Affects Service (Y/N)	Y

Table 6-46 (Cont.) DB_BACKUP_MANAGER_SVC_DOWN

Field	Details
Recommended Action	Cause: When db_backup_manager_svc pod is down. Diagnostic Information: The system detects that the db_backup_manager_svc pod is not up and unable to connect to database. Recommended Actions: 1. Check the db-backup-manager service pod status. Look for CrashLoopBackOff, ImagePullBackOff, OOMKilled, Init container failures. Run the following command to check the pod status: <pre>kubectl -n <cnDBTier namespace> get pods grep "db-backup-manager-svc"</pre> Run the following command to describe the pod: <pre>kubectl -n <cnDBTier namespace> describe pod <backup-manager-pod name></pre> 2. Check the deployment status. Look for Available replicas: 0, Events section for scheduling or image pull errors. Run the following command to get the db-backup-manager-svc deployment: <pre>kubectl -n <cnDBTier namespace> get deployment grep "db-backup-manager-svc"</pre> Run the following command to describe the deployment: <pre>kubectl -n <cnDBTier namespace> describe deployment <backup-manger-svc deployment></pre> 3. Check the DB Backup Manager service pod logs to see if there are a lot of database connectivity issues. Run the following command to get the backup-manager-svc pod: <pre>kubectl -n <cnDBTier namespace> get pods grep "db-backup-manager-svc"</pre> Run the following command to check the logs of the backup-manger svc pod: <pre>kubectl -n <cnDBTier namespace> logs <db-backup-manager-svc pod name></pre> 4. In case if the issue persists, capture all the outputs for the above steps and contact My Oracle Support.

Table 6-46 (Cont.) DB_BACKUP_MANAGER_SVC_DOWN

Field	Details
	 Note Use CNC NF Data Collector tool for capturing logs. For more information on how to collect logs using Data Collector tool, see <i>Oracle Communications Cloud Native Core, Network Function Data Collector User Guide</i>.

6.18 cnDBTier Forced Switchover Disabled Alerts

This section provides details about cnDBTier forced switchover disabled alerts.

Table 6-47 DB_TIER_FORCED_SWITCHOVER_DISABLED

Field	Details
Description	This alert is triggered with <code>critical</code> severity when switchover is disabled forcefully.
Summary	dbtier switchover is disabled forcefully for cnDBTier {{ \$labels.site_name }} and kubernetes namespace {{ \$labels.namespace }}
Severity	critical
Condition	<code>kube_deployment_status_replicas_available{deployment=~".*db-backup-manager-svc.*"} == 0</code>
OID	1.3.6.1.4.1.323.5.3.50.1.2.2039
Metric Used	<code>kube_deployment_status_replicas_available{deployment=~".*db-backup-manager-svc.*"}</code>
Expression Validity	1m
Affects Service (Y/N)	Y

Table 6-47 (Cont.) DB_TIER_FORCED_SWITCHOVER_DISABLED

Field	Details
Recommended Action	Cause: When switchover is disabled forcefully. Diagnostic Information: The alert informs the operator that switchover is currently disabled and needs to be updated. Recommended Actions: 1. Check the DBTIER_REPL_SITE_INFO table for the stop_repl_switchover value. If the value of stop_repl_switchover for the current site is 1, it means switchover is disabled forcefully. Get the site_name, mate_site_name and replchannel_group_id from the db_tier_stop_repl_switchover metric's attribute. Run the following command to check the replication_info.DBTIER_REPL_SITE_INFO table <pre>kubect1 -n <cnDBTier namespace> ndbmysqld-0 -- mysql -h127.0.0.1 -uroot - p<root user password>;</pre> Run the following query to get the value of the column stop_repl_switchover: <pre>select stop_repl_switchover from DBTIER_REPL_SITE_INFO where site_name='<site name>' and mate_site_name='<mate site name>' and replchannel_group_id=<replication channel grp id>;</pre> 2. The alert will automatically clear once the switchover is enabled. If the stop_repl_switchover value retrieved from the above step is 1 and if you wish to re-enable the switchover, then call the API mentioned in the cnDBTier Switchover APIs section to re-enable the switchover. 3. In case if the issue persists, capture all the outputs for the above steps and contact My Oracle Support. Note Use CNC NF Data Collector tool for capturing logs. For more information on how to collect logs using Data Collector tool, see <i>Oracle Communications Cloud Native Core, Network Function Data Collector User Guide</i>.

7

Maintenance Procedures

This section provides the procedures for managing and maintaining cnDBTier.

Note

- The "occne-cndbtier" namespace name used in the procedures is only an example. Ensure that you configure the namespace name according to your environment.
- The OCCNE_NAMESPACE variable in the maintenance procedures is only an example. Before running any command that contains the OCCNE_NAMESPACE variable, ensure that you have set this variable to the cnDBTier namespace as stated in the following code block:

```
export OCCNE_NAMESPACE=<namespace>
```

where, <namespace> is the cnDBTier namespace.

- cnDBTier doesn't support monitoring or alarming about any certificate expiry. Ensure that you track the certificates and renew them before it expires. Refer to the following procedures depending on the certificates you want to renew:
 - [Modifying cnDBTier Certificates to Establish TLS for Communication with NFs](#)
 - [Modifying cnDBTier Certificates to Establish TLS Between Georeplication Sites](#)
 - [Modifying HTTPS Certificates](#)

7.1 Starting or Stopping cnDBTier

This section provides the procedures to start or stop cnDBTier.

7.1.1 Starting cnDBTier

To start cnDBTier, perform the following steps:

1. Run the following command to scale up the management nodes:

```
$ kubectl -n <CNDBTIER_NAMESPACE_NAME> scale sts ndbmcmd --replicas=<replica count for MGM pods>
```

Example:

```
$ kubectl -n occne-cndbtier scale sts ndbmcmd --replicas=2
```

2. Run the following command to scale up the data nodes:

```
$ kubectl -n <CNDBTIER_NAMESPACE_NAME> scale sts ndbmtdd --replicas=<replica count for DATA pods>
```

Example:

```
$ kubectl -n occne-cndbtier scale sts ndbmtdd --replicas=4
```

3. Run the following command to scale up the non-georeplication SQL nodes:

```
$ kubectl -n <CNDBTIER_NAMESPACE_NAME> scale sts ndbappmysqld --replicas=<replica count for non geo SQL pods>
```

Example:

```
$ kubectl -n cluster1 scale sts ndbappmysqld --replicas=2
```

4. Run the following command to scale up the SQL nodes:

```
$ kubectl -n <CNDBTIER_NAMESPACE_NAME> scale sts ndbmysqld --replicas=<replica count for SQL pods>
```

Example:

```
$ kubectl -n cluster1 scale sts ndbmysqld --replicas=2
```

5. Run the following commands to scale up db-monitor-svc, db-replication-svc, and db-backup-manager-svc:

- a. To scale up the db-monitor-svc:

```
$ kubectl -n <CNDBTIER_NAMESPACE_NAME> scale deploy <cnDBTier monitor svc deployment name> --replicas=1
```

Example:

```
$ kubectl -n occne-cndbtier scale deploy mysql-cluster-db-monitor-svc --replicas=1
```

- b. To scale up the db-replication-svc:

```
$ kubectl -n <CNDBTIER_NAMESPACE_NAME> scale deploy <cnDBTier replication svc deployment name> --replicas=1
```

Example:

```
$ kubectl -n occne-cndbtier scale deploy mysql-cluster-Cluster1-Cluster2-replication-svc --replicas=1
```

- c. To scale up the db-backup-manager-svc:

```
$ kubectl -n <CNDBTIER_NAMESPACE_NAME> scale deploy <cnDBTier backup
manager svc deployment name> --replicas=1
```

Example:

```
$ kubectl -n occne-cndbtier scale deploy mysql-cluster-db-backup-
manager-svc --replicas=1
```

6. Run the following command on each mate site to check the replication status:

```
$ kubectl -n <namespace of cnDBTier cluster> exec -it ndbmysqld-0 -- curl
http://mysql-cluster-db-monitor-svc.<namespace of cnDBTier
cluster>:8080/db-tier/status/replication/realtime
```

The value of replicationStatus in the output indicates if the local site is able to replicate data from that remote site:

- "UP": Indicates that the local site is able to replicate data from that remote site.
- "DOWN": Indicates that the local site is not able to replicate data from the respective remote site. In this case, perform a georeplication recovery. For georeplication recovery procedure, see the *"Restoring Georeplication Failure"* section in *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide*.

For example, run the following command to check the georeplication status of cnDBTier cluster2 configured with other remote sites:

```
$ kubectl -n cluster2 exec -it ndbmysqld-0 -- curl http://mysql-cluster-db-
monitor-svc.cluster2:8080/db-tier/status/replication/realtime
```

Sample output:

```
[
  {
    "localSiteName": "cluster2",
    "remoteSiteName": "cluster1",
    "replicationStatus": "UP",
    "secondsBehindRemote": 0,
    "replicationGroupDelay": [
      {
        "replchannel_group_id": "1",
        "secondsBehindRemote": 0
      }
    ]
  },
  {
    "localSiteName": "cluster2",
    "remoteSiteName": "cluster3",
    "replicationStatus": "UP",
    "secondsBehindRemote": 0,
    "replicationGroupDelay": [
      {
```

```

        "replchannel_group_id": "1",
        "secondsBehindRemote": 0
    }
]
},
{
    "localSiteName": "cluster2",
    "remoteSiteName": "cluster4",
    "replicationStatus": "UP",
    "secondsBehindRemote": 0,
    "replicationGroupDelay": [
        {
            "replchannel_group_id": "1",
            "secondsBehindRemote": 0
        }
    ]
}
]

```

In the sample output, the replicationStatus is "UP" for the localSiteName cluster2 for remoteSiteName cluster1, cluster3, and cluster4. This indicates that the localSiteName cluster2 is able to replicate data from remoteSiteName cluster1, cluster3, and cluster4.

① Note

If georeplication is enabled, then run this procedure on all sites.

7.1.2 Stopping cnDBTier

To stop cnDBTier, perform the following steps:

1. Run the following command to scale down the non-georeplication SQL nodes:

```
$ kubectl -n <CNDBTIER_NAMESPACE_NAME> scale sts ndbappmysql - --replicas=0
```

For example:

```
$ kubectl -n cluster1 scale sts ndbappmysql - --replicas=0
```

2. Run the following commands to scale down db-monitor-svc, db-replication-svc, and db-backup-manager-svc:

- a. Run the following commands to scale down db-monitor-svc:

```
$ kubectl -n <CNDBTIER_NAMESPACE_NAME> scale deploy <cnDBTier monitor  
svc deployment name> --replicas=0
```

For example:

```
$ kubectl -n cluster1 scale deploy mysql-cluster-db-monitor-svc --  
replicas=0
```

- b. Run the following commands to scale down db-replication-svc:

```
$ kubectl -n <CNDBTIER_NAMESPACE_NAME> scale deploy <cnDBTier  
replication svc deployment name> --replicas=0
```

For example:

```
$ kubectl -n cluster1 scale deploy mysql-cluster-Cluster1-Cluster2-  
replication-svc --replicas=0
```

- c. Run the following commands to scale down db-backup-manager-svc:

```
$ kubectl -n <CNDBTIER_NAMESPACE_NAME> scale deploy <cnDBTier backup  
manager svc deployment name> --replicas=0
```

For example:

```
$ kubectl -n cluster1 scale deploy mysql-cluster-db-backup-manager-svc  
--replicas=0
```

3. Run the following command to scale down the SQL nodes:

```
$ kubectl -n <CNDBTIER_NAMESPACE_NAME> scale sts ndbmysqld --replicas=0
```

For example:

```
$ kubectl -n cluster1 scale sts ndbmysqld --replicas=0
```

4. Run the following command to scale down the data nodes:

```
$ kubectl -n <CNDBTIER_NAMESPACE_NAME> scale sts ndbmtdd --replicas=0
```

For example:

```
$ kubectl -n cluster1 scale sts ndbmtdd --replicas=0
```

5. Run the following command to scale down the management nodes:

```
$ kubectl -n <CNDBTIER_NAMESPACE_NAME> scale sts ndbmgrmd --replicas=0
```

For example:

```
$ kubectl -n cluster1 scale sts ndbmgrmd --replicas=0
```

7.2 Starting or Stopping cnDBTier Georeplication Service

This section provides the procedures to start or stop cnDBTier georeplication service.

It can be done in either of the following ways:

- Starting or stopping cnDBTier georeplication using dbtreplmgr script

- Starting or stopping cnDBTier georeplication between sites manually

7.2.1 Starting or Stopping georeplication using `dbtrep_lmgr` script

To increase the probability of gracefully and successfully stopping or starting georeplication between sites while simplifying the procedure, reducing human errors, and running limited system and connectivity tests, an automation script `dbtrep_lmgr` is used.

`dbtrep_lmgr` is a bash script used to gracefully stop or start georeplication between sites in a cnDBTier deployment. The script can be run from any of the sites as long as communication between `db-replication-svc` exists.

The `dbtrep_lmgr` script goes through the following phases:

1. **Collecting Site Information:**

In this phase, `dbtrep_lmgr` script gathers data about the current site and other sites that are part of the georeplication setup. It identifies the current site (the site where the script is running) and gather information about the mate sites (peer sites) the "good" site knows about.

2. **Run Sanity Checks:**

In this phase, the script runs sanity checks to try to ensure successful completion.

3. **Get IPs:**

In this phase, the script collects the required IPs from the already obtained site information.

4. **Start or Stop Georeplication:**

In this phase, the script runs the necessary commands to stop or start georeplication between the given sites.

Prerequisites

Before running the `dbtrep_lmgr` script, ensure the following prerequisites are met:

- `dbtrep_lmgr` requires Bash version 4.3.
- The `source_me` file is packaged along with the `dbtrep_lmgr` script. This file must be sourced from the script as it sets the file path and variables required to run the `dbtrep_lmgr` script. It must be source'd from the directory where it is located. In the following command syntax, `./` represents the relative path.

```
source ./source_me
```

- `DBTIER_NAMESPACE` must be set to the `cndbtier` namespace of the recovered site.

Note

`dbtrep_lmgr` script does not support stopping or starting georeplication down to individual replication channel group IDs. It stops all group IDs between the involved sites.

Installing `dbtrep_lmgr` script

Source the `source_me` file which then performs the following actions:

1. Adds the `/bin` along with the script to the local file path.
2. Sets the cnDBTier namespace as `DBTIER_NAMESPACE`.

3. Sets the environment variable DBTIER_LIB with the path for the directory containing the libraries required by the dbtreplmgr script.

```
# Source to set up the current bash shell before running dbtreplmgr
cd Artifacts/Scripts/tools
source ./source_me
```

Using the dbtreplmgr script

The dbtreplmgr script provides the following options. Run the following commands to view the script options:

```
dbtreplmgr [-h | --help]
dbtreplmgr [-u | --usage]

source ./source_me
dbtreplmgr [OPTIONS] --stop-repl-to-from=<site_id>

. ./source_me
dbtreplmgr [OPTIONS] --start-repl-to-from=<site_id>

. ./source_me
dbtreplmgr [OPTIONS] --stop-repl-between-two-sites=<site_a_id>,<site_b_id>

. ./source_me
dbtreplmgr [OPTIONS] --start-repl-between-two-sites=<site_a_id>,<site_b_id>
```

Table 7-1 dbtreplmgr Script Options

Options	Descriptions
-h --help	Displays the help message and exits.
-u --usage	Displays the usage message and exits.
-v --version	Displays the version of the dbtreplmgr script.
--debug	Displays the DEBUG log message to standard error, stderr.
--no-colors	Displays the output on default terminal font color instead of using dbtreplmgr colors.
--use--regex	Forces the use of regular expressions instead of labels to find Kubernetes components (it requires Kubernetes version earlier than 23.3.x). such as, pod names, deployments, sts, svc, and so on.
--use--ipv4	Ensures that dbtreplmgr operates using IPv4, which is applicable in IPv4-only environments or in DualStack Kubernetes clusters where IPv6 might cause issues. It forces the tool to use IPv4 addresses (like loopback IP address 127.0.0.1).
--use--ipv6	Ensures that dbtreplmgr operates using IPv6. It forces the tool to use IPv6 addresses (like loopback IP address : : 1).
--connect-timeout=<connect_timeout_in_seconds>	Displays the wait time (in seconds) before it tries to reconnect. This option is used by the dbtreplmgr when running curl, mysql, mysqladmin and ssh. Default is 15 seconds.

Table 7-1 (Cont.) dbtreplmgr Script Options

Options	Descriptions
--skip-namespace-test	Skips testing of the namespace if DBTIER_NAMESPACE exists in the current cluster.
--skip-tests	Skips sanity tests.
--tests-only	Runs only sanity tests.
--start-repl-to-from=<site_id>	Gracefully start the replication process from <site_id> to and from all other sites in the setup. Note: Replication must be gracefully stopped.
--start-repl-between-two-sites=<site_a_id>,<site_b_id>	Gracefully start the replication process between <site_a_id> and <site_b_id>. Note: Replication must be gracefully stopped.
--stop-repl-to-from=<site_id>	Gracefully stop the replication process from <site_id> to and from all other sites in the setup.
--stop-repl-between-two-sites=<site_a_id>,<site_b_id>	Gracefully stop the replication process between <site_a_id> and <site_b_id>.

Examples to use dbtreplmgr script

```
$ dbtreplmgr
```

Sample output:

To stop the georeplication to and from site3, run the following commands:

```
dbtreplmgr --stop-repl-to-from=3
```

To start the georeplication to and from site3, run the following commands:

```
dbtreplmgr --start-repl-to-from=3
```

To stop the georeplication between site 2 and site 3, run the following commands:

```
dbtreplmgr --stop-repl-between-two-sites=2,3
```

To start the georeplication between site 2 and site 3, run the following commands:

```
dbtreplmgr --start-repl-between-two-sites=2,3
```

7.2.2 Starting cnDBTier Georeplication Between Sites Manually

This section provides the procedure to start cnDBTier georeplication service using the cnDBTier switchover APIs manually.

Starting Georeplication between two sites manually

To start georeplication between the two sites, perform the switchover on each site with respect to the other site.

Following are some example scenarios:

1. In a 2 site scenario, to start the georeplication between site 1 and site 2, perform the following steps:
 - a. Start switchover on site 1 with respect to site 2.
 - b. Start switchover on site 2 with respect to site 1.
2. In a 3 site scenario, to start the georeplication between site 1, site 2, and site 3, perform the following steps:
 - a. Start switchover on site 1 with respect to site 2.
 - b. Start switchover on site 1 with respect to site 3.
 - c. Start switchover on site 2 with respect to site 1.
 - d. Start switchover on site 2 with respect to site 3.
 - e. Start switchover on site 3 with respect to site 1.
 - f. Start switchover on site 3 with respect to site 2.
3. In a 4 site scenario, to start georeplication between site 1, site 2, site3, and site4, perform the following steps:
 - a. Start switchover on site 1 with respect to site 2.
 - b. Start switchover on site 1 with respect to site 3.
 - c. Start switchover on site 1 with respect to site 4.
 - d. Start switchover on site 2 with respect to site 1.
 - e. Start switchover on site 2 with respect to site 3.
 - f. Start switchover on site 2 with respect to site 4.
 - g. Start switchover on site 3 with respect to site 1.
 - h. Start switchover on site 3 with respect to site 2.
 - i. Start switchover on site 3 with respect to site 4.
 - j. Start switchover on site 4 with respect to site 1.
 - k. Start switchover on site 4 with respect to site 2.
 - l. Start switchover on site 4 with respect to site 3.

cnDBTier provides a REST API to start the switchover which can be used to make different API calls depending on the requirement:

Note

Ensure that you perform the switchover on all the sites to start the replication. For example, if you want to start the georeplication between site 1 and site 2, switchover API call on both site 1 and site 2.

Use the following APIs based on your requirement:

1. PUT `/ocdbtier/georeplication/switchover/start/sitename/{siteName}`: This call allows you to start the replication switchover on a site (siteName) with respect to all the other sites.

2. PUT `/ocdbtier/georeplication/switchover/start/siteName/{siteName}/remotesitename/{remoteSiteName}`: This call allows you to start the replication switchover on a site (siteName) with respect to a remote site (remoteSiteName).
3. PUT `/ocdbtier/georeplication/switchover/start/siteName/{siteName}/remotesitename/{remoteSiteName}/replchannelgroupid/{replChannelGroupId}`: This call allows you to start the replication switchover on a site (siteName) with respect to a remote site (remoteSiteName) for a channel group (replChannelGroupId).

To start cnDBTier georeplication service, perform the following:

1. Run the following command to get the LoadBalancer IP of the replication service on the site:

```
$ IP=$(kubectl get svc -n <namespace> | grep repl | awk '{print $4}' | head -n 1 )
```

where, <namespace> is the namespace of the failed site.

For example, run the following command to get the LoadBalancer IP of the replication service on cluster1:

```
$ IP=$(kubectl get svc -n cluster1 | grep repl | awk '{print $4}' | head -n 1 )
```

2. Run the following command to get the LoadBalancer Port of the replication service on the site:

```
$ PORT=$(kubectl get svc -n <namespace> | grep repl | awk '{print $5}' | cut -d '/' -f 1 | cut -d ':' -f 1 | head -n 1)
```

where, <namespace> is the namespace of the failed site.

For example, run the following command to get the LoadBalancer port of the replication service on cluster1:

```
$ PORT=$(kubectl get svc -n cluster1 | grep repl | awk '{print $5}' | cut -d '/' -f 1 | cut -d ':' -f 1 | head -n 1)
```

3. Run the following command to start the replication service in cnDBTier with respect to siteName:

Note

Replace \$IP and \$PORT in the command with the LoadBalancer IP and port numbers obtained from steps 1 and 2.

```
$ curl -X PUT http://$IP:$PORT/ocdbtier/georeplication/switchover/start/siteName/{siteName}
```

For example, run the following command to start the replication service in cnDBTier with respect to cluster1:

```
$ curl -X PUT http://$IP:$PORT/ocdbtier/georeplication/switchover/start/  
sitename/ cluster1
```

Sample output:

```
{  
  "replicationSwitchOver":"start"  
}
```

4. Run the following command to start the replication service in cnDBTier with respect to siteName and remoteSiteName:

Note

Replace \$IP and \$PORT in the command with the LoadBalancer IP and port numbers obtained from steps 1 and 2.

```
$ curl -X PUT http://$IP:$PORT/ocdbtier/georeplication/switchover/start/  
sitename/{siteName}/remotesitename/{remoteSiteName}
```

For example, run the following command to start the replication service in cnDBTier with respect to cluster1 and cluster2:

```
$ curl -X PUT http://$IP:$PORT/ocdbtier/georeplication/switchover/start/  
sitename/cluster1/remotesitename/cluster2
```

Sample output:

```
{  
  "replicationSwitchOver":"start"  
}
```

Alternatively, run the following command to start the replication service in cnDBTier with respect to siteName, remoteSiteName, and replChannelGroupId only:

Note

Replace \$IP and \$PORT in the command with the LoadBalancer IP and port numbers obtained from steps 1 and 2.

```
$ curl -X PUT http://$IP:$PORT/ocdbtier/georeplication/switchover/start/  
sitename/{siteName}/remotesitename/{remoteSiteName}/replchannelgroupid/  
{replChannelGroupId}
```

For example, run the following command to start the replication service in cnDBTier with respect to cluster1, cluster2, and replication channel group ID "1":

```
$ curl -X PUT http://$IP:$PORT/ocdbtier/georeplication/switchover/start/  
sitename/cluster1/remotesitename/cluster2/replchannelgroupid/1
```

Sample output:

```
{  
  "replicationSwitchOver":"start"  
}
```

Note

For more information about the API response payloads, error codes, and curl commands for starting georeplication service, see the [cnDBTier Switchover APIs](#) section.

Example scenario: 3 site setup (site 1, site 2, and site 3)

The following example provides the sample steps to restart replication in a three-site setup (site 1, site 2, site 3) considering the following scenarios:

Following are assumptions for this scenario:

- Georeplication must be restarted to and from site 3.
- Georeplication between site1 and site2 is up and running successfully.

Procedure to restart the replication in a 3 site setup:

1. Perform step 1 and step 2 to get the site IP and PORT details. This example considers the following site details.

- Site 1:
 - site name: cluster 1
 - site1-site3-replication-svc Loadbalancer service IP: 10.233.51.13
 - site1-site3-replication-svc Loadbalancer service PORT: 80
- Site 2:
 - Site name: cluster2
 - site2-site3-replication-svc Loadbalancer service IP: 10.233.52.23
 - site2-site3-replication-svc Loadbalancer service PORT: 80
- Site 3:
 - Site name: cluster3
 - site3-site1-replication-svc Loadbalancer service IP: 10.233.53.31
 - site3-site1-replication-svc Loadbalancer service PORT: 80

Note

Perform the steps 2 to 4 from a bash shell in any db-replication-pod.

Start the replication switchover on site 3 with respect to other sites, that is, site 1 and site 2:

```
curl -X PUT http://10.233.53.31:80/ocdbtier/georeplication/switchover/  
start/sitename/cluster3
```

3. Start the replication switchover on site 1 with respect to site 3:

```
curl -X PUT http://10.233.51.13:80/ocdbtier/georeplication/switchover/  
start/sitename/cluster1/remotesitename/cluster3
```

4. Start the replication switchover on site 2 with respect to site 3:

```
curl -X PUT http://10.233.52.23:80/ocdbtier/georeplication/switchover/  
start/sitename/cluster2/remotesitename/cluster3
```

5. Verify if the replication to and from site 3 has started.

Note

This procedure can be extended to a 4 site setup scenario.

7.2.3 Stopping cnDBTier Georeplication Between Sites Manually

This section provides the procedure to stop the cnDBTier georeplication service using cnDBTier switchover and stop replica APIs.

To stop the georeplication between the two sites, perform the stop switchover and replica on each site with respect to the other site.

Following are some example scenarios:

1. In a 2 site setup, to stop the georeplication between site 1 and site 2, perform the following steps:

Site 1:

- Stop the switchover on site 1 with respect to site 2.
- Stop the replica on site 1 with respect to site 2.

Site 2:

- Stop the switchover on site 2 with respect to site 1.
- Stop the replica on site 2 with respect to site 1.

2. In a 3 site setup, to stop the georeplication between site 1, site 2, and site 3, perform the following steps:

Site 1:

- Stop the switchover on site 1 with respect to site 2.
- Stop the replica on site 1 with respect to site 2.
- Stop the switchover on site 1 with respect to site 3.
- Stop the replica on site 1 with respect to site 3.

Site 2:

- Stop the switchover on site 2 with respect to site 1.
- Stop the replica on site 2 with respect to site 1.
- Stop the switchover on site 2 with respect to site 3.
- Stop the replica on site 2 with respect to site 3.

Site 3:

- Stop the switchover on site 3 with respect to site 1.
- Stop the replica on site 3 with respect to site 1.
- Stop the switchover on site 3 with respect to site 2.
- Stop the replica on site 3 with respect to site 2.

3. In a 4 site setup, to stop the georeplication between site 1 and site 2 and site3 and site4, you must perform the following steps:

Site 1:

- Stop the switchover on site 1 with respect to site 2.
- Stop the replica on site 1 with respect to site 2.
- Stop the switchover on site 1 with respect to site 3.
- Stop the replica on site 1 with respect to site 3.
- Stop the switchover on site 1 with respect to site 4.
- Stop the replica on site 1 with respect to site 4.

Site 2:

- Stop the switchover on site 2 with respect to site 1.
- Stop the replica on site 2 with respect to site 1.
- Stop the switchover on site 2 with respect to site 3.
- Stop the replica on site 2 with respect to site 3.
- Stop the switchover on site 2 with respect to site 4.
- Stop the replica on site 2 with respect to site 4.

Site 3:

- Stop the switchover on site 3 with respect to site 1.
- Stop the replica on site 3 with respect to site 1.
- Stop the switchover on site 3 with respect to site 2.
- Stop the replica on site 3 with respect to site 2.
- Stop the switchover on site 3 with respect to site 4.
- Stop the replica on site 3 with respect to site 4.

Site 4:

- Stop the switchover on site 4 with respect to site 1.
- Stop the replica on site 4 with respect to site 1.
- Stop the switchover on site 4 with respect to site 2.
- Stop the replica on site 4 with respect to site 2.
- Stop the switchover on site 4 with respect to site 3.

- Stop the replica on site 4 with respect to site 3.

cnDBTier provides two REST APIs to stop the switchover and to stop the replica. These APIs can be used to make different API calls depending on your requirement:

① Note

Ensure that you perform the switchover on all the sites to stop the replication. For example, if you want to stop the georeplication between site 1 and site 2, stop the switchover and stop the replica using the API on both site 1 and site 2.

Use one of the following APIs to stop the switchover and stop the replica on both site 1 and site 2 based on the requirement:

1. PUT `/ocdbtier/georeplication/switchover/stop/sitename/{siteName}`: This call allows you to stop replication switchover on a site (siteName) with respect to all the other mate sites.
PUT `/ocdbtier/georeplication/stopreplica/sitename/{siteName}`: This call allows you to stop replica on a site (siteName) with respect to all the other mate sites.
2. PUT `/ocdbtier/georeplication/switchover/stop/sitename/{siteName}/remotesitename/{remoteSiteName}`: This call allows you to stop replication switchover on a site (siteName) with respect to a remote site (remoteSiteName).
PUT `/ocdbtier/georeplication/stopreplica/sitename/{siteName}/remotesitename/{remoteSiteName}`: This call allows you to stop replica on a site (siteName) with respect to a remote site (remoteSiteName).
3. PUT `/ocdbtier/georeplication/switchover/stop/sitename/{siteName}/remotesitename/{remoteSiteName}/replchannelgroupid/{replChannelGroupId}`: This call allows you to stop replication switchover on a site (siteName) with respect to a remote site (remoteSiteName) for a channel group (replChannelGroupId).
PUT `/ocdbtier/georeplication/stopreplica/sitename/{siteName}/remotesitename/{remoteSiteName}/replchannelgroupid/{replChannelGroupId}`: This call allows you to stop replica on a site (siteName) with respect to a remote site (remoteSiteName) for a channel group (replChannelGroupId).

Perform the following steps to stop the cnDBTier georeplication service using the stop replication APIs:

1. Perform the following steps to stop the cnDBTier replication service switchover and stop the replication channel:
 - a. Run the following command to get the LoadBalancer IP of the replication service on the site:

```
$ IP=$(kubectl get svc -n <namespace> | grep repl | awk '{print $4}' | head -n 1 )
```

where, <namespace> is the namespace of the failed site.

For example, run the following command to get the LoadBalancer IP of the replication service on cluster1:

```
$ IP=$(kubectl get svc -n cluster1 | grep repl | awk '{print $4}' | head -n 1 )
```

- b. Run the following command to get the LoadBalancer Port of the replication service on the site:

```
$ PORT=$(kubectl get svc -n <namespace> | grep repl | awk '{print $5}'  
| cut -d '/' -f 1 | cut -d ':' -f 1 | head -n 1)
```

where, <namespace> is the namespace of the failed site.

For example, run the following command to get the LoadBalancer Port of the replication service on cluster1:

```
$ PORT=$(kubectl get svc -n cluster1 | grep repl | awk '{print $5}' |  
cut -d '/' -f 1 | cut -d ':' -f 1 | head -n 1)
```

- c. Run the following command to stop the replication service switchover and stop the replication channel in cnDBTier with respect to siteName:

Note

Replace \$IP and \$PORT in the command with the LoadBalancer IP and port numbers obtained from steps 1 and 2.

```
$ curl -X PUT http://$IP:$PORT/ocdbtier/georeplication/switchover/stop/  
sitename/{siteName}
```

```
$ curl -X PUT http://$IP:$PORT/ocdbtier/georeplication/stopreplica/  
sitename/{siteName}
```

For example, run the following command to stop the replication service switchover and stop the replication channel in cnDBTier with respect to cluster1:

```
$ curl -X PUT http://$IP:$PORT/ocdbtier/georeplication/switchover/stop/  
sitename/cluster1
```

Sample output:

```
{  
  "replicationSwitchOver": "stop"  
}
```

Run the following command to stop the replication channel in cnDBTier with respect to cluster 1:

```
$ curl -X PUT http://$IP:$PORT/ocdbtier/georeplication/stopreplica/  
sitename/cluster1
```


Sample output:

```
{
  "stopReplica":"stop"
}
```

Alternatively, run the following command to stop the replication service switchover and stop the replication channel in cnDBTier with respect to siteName and remoteSiteName:

Note

Replace \$IP and \$PORT in the command with the LoadBalancer IP and port numbers obtained from steps 1 and 2.

```
$ curl -X PUT http://$IP:$PORT/ocdbtier/georeplication/switchover/stop/
sitename/{siteName}/remotesitename/{remoteSiteName}
```

```
$ curl -X PUT http://$IP:$PORT/ocdbtier/georeplication/stopreplica/
sitename/{siteName}/remotesitename/{remoteSiteName}
```

For example, run the following command to stop the replication service switchover and stop the replication channel in cnDBTier with respect to cluster1 and cluster2:

```
$ curl -X PUT http://$IP:$PORT/ocdbtier/georeplication/switchover/stop/
sitename/cluster1/remotesitename/cluster2
```

Sample output:

```
{
  "replicationSwitchOver":"stop"
}
```

```
$ curl -X PUT http://$IP:$PORT/ocdbtier/georeplication/stopreplica/
sitename/cluster1/remotesitename/cluster2
```

Sample output:

```
{
  "stopReplica":"stop"
}
```

Run the following command to stop the replication service switchover in cnDBTier with respect to siteName, remoteSiteName, and replChannelGroupId:

Note

Replace \$IP and \$PORT in the command with the LoadBalancer IP and port numbers obtained from steps 1 and 2.

```
$ curl -X PUT http://$IP:$PORT/ocdbtier/georeplication/switchover/stop/
sitename/{siteName}/remotesitename/{remoteSiteName}/replchannelgroupid/
{replChannelGroupId}
```

```
$ curl -X PUT http://$IP:$PORT/ocdbtier/georeplication/stopreplica/
sitename/{siteName}/remotesitename/{remoteSiteName}/replchannelgroupid/
{replChannelGroupId}
```

For example, run the following command to stop the replication service switchover in cnDBTier with respect to cluster1, cluster2, and replication channel group ID "1":

```
$ curl -X PUT http://$IP:$PORT/ocdbtier/georeplication/switchover/stop/
sitename/cluster1/remotesitename/cluster2/replchannelgroupid/1
```

Sample output:

```
{
  "replicationSwitchOver":"stop"
}
```

```
$ curl -X PUT http://$IP:$PORT/ocdbtier/georeplication/stopreplica/
sitename/cluster1/remotesitename/cluster2/replchannelgroupid/1
```

Sample output:

```
{
  "stopReplica":"stop"
}
```

Note

For more information about the API response payloads, error codes, and curl commands to stop replication service switchover and stop the replication channel, see [cnDBTier Switchover APIs](#) and [cnDBTier Stop Replica APIs](#).

Example scenario: 3 site setup (site 1, site 2, site 3)

The following example provides the sample steps to stop replication in a three-site setup (site 1, site 2, site 3) considering the following scenarios:

Following are assumptions for this scenario:

- Georeplication to and from site 3 must be stopped.
- Georeplication between site1 and site2 must continue to run successfully.

Procedure to stop the replication in a 3 site setup

1. Perform steps 1a and 1b, or steps 2a and 2b to get the site IP and PORT details. This example considers the following site details.

- Site 1:
 - site name: cluster 1
 - site1-site3-replication-svc Loadbalancer service IP: 10.233.51.13
 - site1-site3-replication-svc Loadbalancer service PORT: 80
- Site 2:
 - Site name: cluster2
 - site2-site3-replication-svc Loadbalancer service IP: 10.233.52.23
 - site2-site3-replication-svc Loadbalancer service PORT: 80
- Site 3:
 - Site name: cluster3
 - site3-site1-replication-svc Loadbalancer service IP: 10.233.53.31
 - site3-site1-replication-svc Loadbalancer service PORT: 80

2. Note

Perform the steps 2 to 4 from a bash shell in any db-replication-pod.

Stop the replication switchover and stop the replica on site 3 with respect to the other sites, that is, site 1 and site 2.

2:

```
curl -X PUT http://10.233.53.31:80/ocdbtier/georeplication/switchover/stop/  
sitename/cluster3
```

```
{  
  "replicationSwitchOver":"stop"  
}
```

```
curl -X PUT http://10.233.53.31:80/ocdbtier/georeplication/stopreplica/  
sitename/cluster3
```

```
{  
  "stopReplica":"stop"  
}
```

3. Stop the replication switchover and stop the replica on site 1 with respect to site 3:

```
curl -X PUT http://10.233.51.13:80/ocdbtier/georeplication/switchover/stop/  
sitename/cluster1/remotesitename/cluster3
```

```
{  
  "replicationSwitchOver":"stop"  
}
```

```
curl -X PUT http://10.233.51.13:80/ocdbtier/georeplication/stopreplica/  
sitename/cluster1/remotesitename/cluster3
```

```
{  
  "stopReplica":"stop"  
}
```

4. Stop replication switchover and Stop replica on site 2 with respect to site 3:

```
curl -X PUT http://10.233.52.23:80/ocdbtier/georeplication/switchover/stop/  
sitename/cluster2/remotesitename/cluster3
```

```
{"replicationSwitchOver":"stop"}
```

```
curl -X PUT http://10.233.52.23:80/ocdbtier/georeplication/stopreplica/  
sitename/cluster2/remotesitename/cluster3
```

```
{  
  "stopReplica":"stop"  
}
```

5. Verify if the replication to and from site 3 has stopped.

Note

This procedure can be extended to a four site setup scenario.

7.3 Scaling cnDBTier Pods

cnDBTier 25.2.100 supports horizontal and vertical scaling of pods for improved performance. This section describes the horizontal and vertical scaling procedures for SQL and database pods.

7.3.1 Horizontal Scaling

This section describes the procedures to horizontally scale **ndbappmysqlid** and **ndbmtid** pods.

7.3.1.1 Scaling ndbappmysqld Pods

cnDBTier 25.2.100 supports autoscaling of cnDBTier ndbappmysqld pods when cnDBTier is deployed with a service account. For more information, see [Scaling ndbappmysqld Pods With Service Account](#). When cnDBTier is deployed without a service account, you must manually scale the ndbappmysqld pods. This section describes the procedures to scale the cnDBTier ndbappmysqld pods when cnDBTier is deployed without service account and configurations to consider when cnDBTier is deployed with service account.

Note

Before scaling, ensure that the worker nodes have adequate resources to support scaling.

Considerations

The examples in these procedures consider a cnDBTier setup deployed with two ndbmngmd, two ndbmtd, two ndbmysqld, and two ndbappmysqld pods. The ndbappmysqld pods of this cnDBTier setup are scaled up from replica count two to four in this procedure.

7.3.1.1.1 Scaling ndbappmysqld Pods Without Service Account

This section describes the procedure to manually scale the cnDBTier ndbappmysqld pods when cnDBTier is deployed without service account.

Note

ndbappmysqld auto scaling requires service account. If the cnDBTier is deployed without the service account, then auto scaling is disabled.

1. Increase the replica count of the ndbappmysqld pods in the `custom_values.yaml` file under `global.ndbappReplicaCount`:

```
global:
  ndbappReplicaCount: 4
```

Note

The cnDBTier is initially deployed with two ndbappmysqld pods and increased to a replica count of four in this step.

2. Upgrade cnDBTier with the new ndbappReplicaCount by following the upgrade procedure in *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide*.

Note

cnDBTier supports upgrade in a TLS enabled cluster in this scenario only.

When the helm upgrade completes, the ndbappmysql replica count is increased to the value updated in the `custom_values.yaml` file.

7.3.1.1.2 Scaling ndbappmysql Pods With Service Account

This section describes the procedure to scale the cnDBTier ndbappmysql pods when cnDBTier is deployed with service account.

Scaling cnDBTier ndbappmysql When Autoscaling is Enabled

When cnDBTier is deployed with a service account and autoscaling is enabled, the ndbappmysql pods are scaled automatically.

Ensure that autoscaling is enabled for cnDBTier ndbappmysql pods in the `custom_values.yaml` file under `autoscaling.ndbapp`:

```
autoscaling:
  ndbapp:
    enabled: true
```

Note

Apply the above change to the cnDBTier Helm chart by following the "Upgrading cnDBTier Clusters" procedure in the *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide*. cnDBTier supports upgrade in a TLS enabled cluster in this scenario only.

When autoscaling is enabled, ndbappmysql is scaled between `ndbappReplicaCount` and `ndbappReplicaMaxCount` according to the CPU and RAM usage defined in the `custom_values.yaml` file:

```
horizontalPodAutoscaler:
  memory:
    enabled: true
    averageUtilization: 80
  cpu:
    enabled: false
    averageUtilization: 80
```

Scaling cnDBTier ndbappmysql When Autoscaling is Disabled

1. Increase the replica count of the ndbappmysql pods in the `custom_values.yaml` file under `global.ndbappReplicaCount`:

```
global:
  ndbappReplicaCount: 4
```

Note

cnDBTier is initially deployed with two ndbappmysql pods and increased to a replica count of four in this step.

2. Upgrade cnDBTier with the new `ndbappReplicaCount` by following the upgrade procedure in *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide*.

Note

cnDBTier supports upgrade in a TLS enabled cluster in this scenario only.

When the Helm upgrade completes, the `ndbappmysqld` replica count is increased to the value updated in the `custom_values.yaml` file.

7.3.1.2 Scaling ndbmttd Pods

cnDBTier supports only scaling up of data pods, and doesn't support scaling down of data pods. This section describes the manual procedure to scale up the cnDBTier ndbmttd pods.

Note

- Before scaling, ensure that the worker nodes have adequate resources to support scaling.
- Divert the traffic during the horizontal scaling of the ndbmttd pods as the MySQL NDB cluster may go offline during the scaling process.

Considerations

The examples in this procedure consider a cnDBTier setup deployed with 2 ndbmgm, 2 ndbmttd, 2 ndbmysqld sql, and 2 ndbappmysqld pods. The ndbmttd pods of this cnDBTier setup are scaled up from replica count 2 to 4 in this procedure.

Procedure

1. Before scaling the data pods, perform the Helm test on the cnDBTier cluster to check the health of the cluster:

```
$ helm test mysql-cluster -n <namespace>
```

Example:

```
$ helm test mysql-cluster -n occne-cndbtier
```

Sample output:

```
NAME: mysql-cluster
LAST DEPLOYED: Mon May 20 08:10:11 2025
NAMESPACE: occne-cndbtier
STATUS: deployed
REVISION: 2
TEST SUITE: mysql-cluster-node-connection-test
Last Started: Mon May 20 10:03:51 2025
Last Completed: Mon May 20 10:04:16 2025
Phase: Succeeded
```

You can also check if all the NDB pods are connected by checking the status of cnDBTier from the management ndb_mgm console:

```
$ kubectl -n occne-cndbtier exec ndbmgmd-0 -- ndb_mgm -e show
```

Sample output:

```
Connected to Management Server at: localhost:1186
Cluster Configuration
-----
[ndbd(NDB)] 2 node(s)
id=1 @10.233.74.65 (mysql-8.4.3 ndb-8.4.3, Nodegroup: 0)
id=2 @10.233.84.68 (mysql-8.4.3 ndb-8.4.3, Nodegroup: 0, *)

[ndb_mgmd(MGM)] 2 node(s)
id=49 @10.233.73.61 (mysql-8.4.3 ndb-8.4.3)
id=50 @10.233.84.67 (mysql-8.4.3 ndb-8.4.3)

[mysqld(API)] 10 node(s)
id=56 @10.233.84.69 (mysql-8.4.3 ndb-8.4.3)
id=57 @10.233.73.62 (mysql-8.4.3 ndb-8.4.3)
id=70 @10.233.78.70 (mysql-8.4.3 ndb-8.4.3)
id=71 @10.233.72.49 (mysql-8.4.3 ndb-8.4.3)
id=72 (not connected, accepting connect from ndbappmysqld-2.ndbappmysqldsvc.occne-
cndbtier.svc.occne1-cgbu-cne-dbtier)
id=73 (not connected, accepting connect from ndbappmysqld-3.ndbappmysqldsvc.occne-
cndbtier.svc.occne1-cgbu-cne-dbtier)
id=222 (not connected, accepting connect from any host)
id=223 (not connected, accepting connect from any host)
id=224 (not connected, accepting connect from any host)
id=225 (not connected, accepting connect from any host)
```

Note

The node IDs 222 to 225 in the sample output are shown as "not connected" as these nodes are added as empty slot IDs used for fault recovery. You can ignore these nodes.

2. Enable `replicationskiperrors` in the `custom_values.yaml` file and apply the changes using the Helm upgrade procedure for all cnDBTier sites.

Note

Skip this step if `replicationskiperrors` is already enabled in the cnDBTier cluster.

```
global:
  replicationskiperrors:
    enable: true
```

3. Increase the replica count of the `ndbmtid` pod in the `custom_values.yaml` file:

Note

- The cnDBTier is deployed with 2 ndbmttd and increased to a replica count of 4 in this step.
- You can only scale up the ndbmttd pod by an even number of increment and can't scale down the pods.

```
global:
  ndbReplicaCount: 4
```

4. Run the following command to upgrade the cnDBTier to increase the ndbmttd replica count:

Note

After the upgrade, the new ndbmttd pods can crash and restart. You can this error. Verify the status of the new pod before partitioning of data in [step 14](#).

```
$ helm upgrade --no-hooks --set global.updateStrategy=OnDelete mysql-
cluster --namespace <namespace> occndbtier -f <path to custom_values.yaml>
```

5. Perform the following steps to scale the ndbmttd pods back to their original replica count (the replica count that the ndbmttd pods had before performing Step 3):

Note

In this case, the replica count was increased from 2 to 4 in Step 3. Therefore, in this step, scale back the replica count to 2.

- a. Run the following commands to patch the horizontal pod autoscaling (hpa):

```
$ kubectl -n <namespace> patch hpa ndbmttd -p '{"spec": {"minReplicas":
2}}'
$ kubectl -n <namespace> patch hpa ndbmttd -p '{"spec": {"maxReplicas":
2}}'
```

- b. Run the following command to scale down the ndbmttd sts replica count to 2:

```
$ kubectl -n <namespace> scale sts ndbmttd --replicas=2
```

- c. After scaling down the ndbmttd sts replica count, terminate the newly added ndbmttd pods. Retain only the existing ndbmttd pods in the cluster.

6. Once the upgrade is complete, delete all the management pods simultaneously:

```
$ kubectl -n <namespace> delete pods ndbmgmd-0 ndbmgmd-1
```

7. Wait for the management pods to come back to the running state and connect to the cluster. Run the following command to verify the status of the pods:

```
$ kubectl -n occne-cndbtier exec ndbmgmd-0 -- ndb_mgm -e show
```

8. Delete the data pods one at a time. Wait for a deleted pod to return to the running state and connect to cluster before deleting the next one:

Note

Delete the data pods in descending order (ndbmt-d-n, ndbmt-d-(n-1), ndbmt-d-(n-2), and so on).

For example:

- a. Run the following command to delete pod ndbmt-d-2:

```
$ kubectl -n <namespace> delete pod ndbmt-d-2
```

- b. Wait for ndbmt-d-2 to return to the running state and connect back to the NDB cluster.

- c. Run the following command to delete pod ndbmt-d-1:

```
$ kubectl -n <namespace> delete pod ndbmt-d-1
```

- d. Wait for ndbmt-d-1 to return to the running state and connect back to the NDB cluster.

- e. Run the following command to delete pod ndbmt-d-0:

```
$ kubectl -n <namespace> delete pod ndbmt-d-0
```

9. Wait until all the data pods are up and running, and get connected to the NDB cluster. To check if the data pods are connected to the NDB cluster, log in to one of the management pods and check the status of the NDB cluster from the ndb_mgm console:

```
$ kubectl -n occne-cndbtier exec ndbmgmd-0 -- ndb_mgm -e show
```

Sample output:

```
Connected to Management Server at: localhost:1186
```

```
Cluster Configuration
```

```
-----
```

```
[ndbd(NDB)] 4 node(s)
```

```
id=1 @10.233.74.65 (mysql-8.4.3 ndb-8.4.3, Nodegroup: 0)
```

```
id=2 @10.233.84.68 (mysql-8.4.3 ndb-8.4.3, Nodegroup: 0, *)
```

```
id=3 (not connected, accepting connect from ndbmt-d-2.ndbmt-dsvc.occne-cndbtier.svc.occne1-cgbu-cne-dbtier)
```

```
id=4 (not connected, accepting connect from ndbmt-d-3.ndbmt-dsvc.occne-cndbtier.svc.occne1-cgbu-cne-dbtier)
```

```
[ndb_mgmd(MGM)] 2 node(s)
```

```
id=49 @10.233.73.61 (mysql-8.4.3 ndb-8.4.3)
```

```
id=50 @10.233.84.67 (mysql-8.4.3 ndb-8.4.3)
```

```
[mysqld(API)] 10 node(s)
```

```
id=56 @10.233.84.69 (mysql-8.4.3 ndb-8.4.3)
```

```
id=57 @10.233.73.62 (mysql-8.4.3 ndb-8.4.3)
```

```

id=70 @10.233.78.70 (mysql-8.4.3 ndb-8.4.3)
id=71 @10.233.72.49 (mysql-8.4.3 ndb-8.4.3)
id=72 (not connected, accepting connect from ndbappmysqld-2.ndbappmysqldsvc.occne-
cndbtier.svc.occne1-cgbu-cne-dbtier)
id=73 (not connected, accepting connect from ndbappmysqld-3.ndbappmysqldsvc.occne-
cndbtier.svc.occne1-cgbu-cne-dbtier)
id=222 (not connected, accepting connect from any host)
id=223 (not connected, accepting connect from any host)
id=224 (not connected, accepting connect from any host)
id=225 (not connected, accepting connect from any host)

```

Note

- The newly added data pods, node IDs 3 and 4 in the sample output, are shown as "starting, no nodegroup". You can ignore these nodes.
- The node IDs 222 to 225 in the sample output are shown as "not connected" as these nodes are added as empty slot IDs used for fault recovery. You can ignore these nodes.

10. Delete the ndbmysqld pods one at a time. Wait for a deleted pod to return to the running state and connect to the cluster before deleting the next one:

Note

Delete the pods in descending order (ndbmysqld-n, ndbmysqld-(n-1), ndbmysqld-(n-2), and so on).

For example:

- a. Run the following command to delete pod ndbmysqld-2:

```
$ kubectl -n <namespace> delete pod ndbmysqld-2
```

- b. Wait for ndbmysqld-2 to return to the running state and connect back to the NDB cluster.

- c. Run the following command to delete pod ndbmysqld-1:

```
$ kubectl -n <namespace> delete pod ndbmysqld-1
```

- d. Wait for ndbmysqld-1 to return to the running state and connect back to the NDB cluster.

- e. Run the following command to delete pod ndbmysqld-0:

```
$ kubectl -n <namespace> delete pod ndbmysqld-0
```

11. Wait for the ndbmysqld pods to return to running state and connect to the NDB cluster. To check if the data pods are connected to the NDB cluster, log in to one of the management pods and check the status of the NDB cluster from the ndb_mgm console:

```
$ kubectl -n occne-cndbtier exec -it ndbmgmd-0 -- ndb_mgm -e show
```

Sample output:

```

Connected to Management Server at: localhost:1186
Cluster Configuration
-----
[ndbd(NDB)] 4 node(s)
id=1 @10.233.74.65 (mysql-8.4.3 ndb-8.4.3, Nodegroup: 0)
id=2 @10.233.84.68 (mysql-8.4.3 ndb-8.4.3, Nodegroup: 0, *)
id=3 (not connected, accepting connect from ndbmtid-2.ndbmtidsvc.occne-
cndbtier.svc.occne1-cgbu-cne-dbtier)
id=4 (not connected, accepting connect from ndbmtid-3.ndbmtidsvc.occne-
cndbtier.svc.occne1-cgbu-cne-dbtier)

[ndb_mgmd(MGM)] 2 node(s)
id=49 @10.233.73.61 (mysql-8.4.3 ndb-8.4.3)
id=50 @10.233.84.67 (mysql-8.4.3 ndb-8.4.3)

[mysqld(API)] 10 node(s)
id=56 @10.233.84.69 (mysql-8.4.3 ndb-8.4.3)
id=57 @10.233.73.62 (mysql-8.4.3 ndb-8.4.3)
id=70 @10.233.78.70 (mysql-8.4.3 ndb-8.4.3)
id=71 @10.233.72.49 (mysql-8.4.3 ndb-8.4.3)
id=72 (not connected, accepting connect from ndbappmysqld-2.ndbappmysqldsvc.occne-
cndbtier.svc.occne1-cgbu-cne-dbtier)
id=73 (not connected, accepting connect from ndbappmysqld-3.ndbappmysqldsvc.occne-
cndbtier.svc.occne1-cgbu-cne-dbtier)
id=222 (not connected, accepting connect from any host)
id=223 (not connected, accepting connect from any host)
id=224 (not connected, accepting connect from any host)
id=225 (not connected, accepting connect from any host)

```

Note

The node IDs 222 to 225 in the sample output are shown as "not connected" as these nodes are added as empty slot IDs used for fault recovery. You can ignore these nodes.

12. Delete the ndbappmysqld pods one at a time. Wait for a deleted pod to return back to the running state before deleting the next pod.

Note

Delete the pods in descending order (ndbappmysqld-n, ndbappmysqld-(n-1), ndbappmysqld-(n-2), and so on).

For example:

- a. Run the following command to delete pod ndbappmysqld-2:

```
$ kubectl -n <namespace> delete pod ndbappmysqld-2
```

- b. Wait for ndbappmysqld-2 to return to the running state and connect back to the NDB cluster.
- c. Run the following command to delete pod ndbappmysqld-1:

```
$ kubectl -n <namespace> delete pod ndbappmysqld-1
```

- d. Wait for ndbappmysqld-1 to return to the running state and connect back to the NDB cluster.
- e. Run the following command to delete pod ndbappmysqld-0:

```
$ kubectl -n <namespace> delete pod ndbappmysqld-0
```

13. Wait for the ndbappmysqld pods to return to the running state and connect to the NDB cluster. To check if the data pods are connected to the NDB cluster, log in to one of the management pods and check the status of the NDB cluster from the ndb_mgm console:

```
$ kubectl -n occne-cndbtier exec -it ndbmcmd-0 -- ndb_mgm -e show
```

Sample output:

```
Connected to Management Server at: localhost:1186
```

```
Cluster Configuration
```

```
-----
```

```
[ndbd(NDB)] 4 node(s)
```

```
id=1 @10.233.74.65 (mysql-8.4.3 ndb-8.4.3, Nodegroup: 0)
```

```
id=2 @10.233.84.68 (mysql-8.4.3 ndb-8.4.3, Nodegroup: 0, *)
```

```
id=3 (not connected, accepting connect from ndbmtid-2.ndbmtidsvc.occne-cndbtier.svc.occne1-cgbu-cne-dbtier)
```

```
id=4 (not connected, accepting connect from ndbmtid-3.ndbmtidsvc.occne-cndbtier.svc.occne1-cgbu-cne-dbtier)
```

```
[ndb_mgmd(MGM)] 2 node(s)
```

```
id=49 @10.233.73.61 (mysql-8.4.3 ndb-8.4.3)
```

```
id=50 @10.233.84.67 (mysql-8.4.3 ndb-8.4.3)
```

```
[mysqld(API)] 10 node(s)
```

```
id=56 @10.233.84.69 (mysql-8.4.3 ndb-8.4.3)
```

```
id=70 @10.233.78.70 (mysql-8.4.3 ndb-8.4.3)
```

```
id=71 @10.233.72.49 (mysql-8.4.3 ndb-8.4.3)
```

```
id=72 (not connected, accepting connect from ndbappmysqld-2.ndbappmysqldsvc.occne-cndbtier.svc.occne1-cgbu-cne-dbtier)
```

```
id=73 (not connected, accepting connect from ndbappmysqld-3.ndbappmysqldsvc.occne-cndbtier.svc.occne1-cgbu-cne-dbtier)
```

```
id=222 (not connected, accepting connect from any host)
```

```
id=223 (not connected, accepting connect from any host)
```

```
id=224 (not connected, accepting connect from any host)
```

```
id=225 (not connected, accepting connect from any host)
```

Note

The node IDs 222 to 225 in the sample output are shown as "not connected" as these nodes are added as empty slot IDs used for fault recovery. You can ignore these nodes.

14. Perform the following steps to scale up the ndbmtid pods to their increased replica count. That is, the replica count configured at Step 3:

Note

In this case, the replica count was increased to 4 in Step 3. Therefore, in this step, scale back the replica count to 4.

- a. Run the following commands to patch horizontal pod autoscaler (hpa):

```
$ kubectl -n <namespace> patch hpa ndbmtl -p '{"spec": {"maxReplicas": 4}}'
$ kubectl -n <namespace> patch hpa ndbmtl -p '{"spec": {"minReplicas": 4}}'
```

- b. Scale up the ndbmtl sts replica count to 4:

```
$ kubectl -n <namespace> scale sts ndbmtl --replicas=4
```

- c. Wait for the ndbmtl pods to return to the running state and connect to the NDB cluster. To check if the data pods are connected to the NDB cluster, log in to one of the management pods and check the status of the NDB cluster from the ndb_mgm console:

```
$ kubectl -n occne-cndbtier exec -it ndbmcmd-0 -- ndb_mgm -e show
```

Sample output:

```
Connected to Management Server at: localhost:1186
Cluster Configuration
-----
[ndbd(NDB)] 4 node(s)
id=1 @10.233.74.65 (mysql-8.4.3 ndb-8.4.3, Nodegroup: 0)
id=2 @10.233.84.68 (mysql-8.4.3 ndb-8.4.3, Nodegroup: 0, *)
id=3 @10.233.117.60 (mysql-8.4.3 ndb-8.4.3, starting, no nodegroup)
id=4 @10.233.111.64 (mysql-8.4.3 ndb-8.4.3, starting, no nodegroup)

[ndb_mgmd(MGM)] 2 node(s)
id=49 @10.233.73.61 (mysql-8.4.3 ndb-8.4.3)
id=50 @10.233.84.67 (mysql-8.4.3 ndb-8.4.3)

[mysqld(API)] 10 node(s)
id=56 @10.233.84.69 (mysql-8.4.3 ndb-8.4.3)
id=57 @10.233.73.62 (mysql-8.4.3 ndb-8.4.3)
id=70 @10.233.78.70 (mysql-8.4.3 ndb-8.4.3)
id=71 @10.233.72.49 (mysql-8.4.3 ndb-8.4.3)
id=72 (not connected, accepting connect from
ndbappmysqld-2.ndbappmysqldsvc.occne-cndbtier.svc.occne1-cgbu-cne-dbtier)
id=73 (not connected, accepting connect from
ndbappmysqld-3.ndbappmysqldsvc.occne-cndbtier.svc.occne1-cgbu-cne-dbtier)
id=222 (not connected, accepting connect from any host)
id=223 (not connected, accepting connect from any host)
id=224 (not connected, accepting connect from any host)
id=225 (not connected, accepting connect from any host)
```

Note

- When the newly added ndbmtl pods are connected to the NDB cluster, they are shown in the "no nodegroup" state.
- The node IDs 222 to 225 in the sample output are shown as "not connected" as these nodes are added as empty slot IDs used for fault recovery. You can ignore these nodes.

15. Run the following command in the NDB Cluster management client to create the new node group. Node ID for the newly added ndbmtod pod is <sequence no of the ndbmtod pod + 1>.

```
ndb_mgm> CREATE NODEGROUP
<node_id_for_pod_ndbmtod-2>,<node_id_for_pod_ndbmtod-3>
```

For example:

```
ndb_mgm> CREATE NODEGROUP <(sequence no of pod ndbmtod-2) + 1>,<(sequence no of pod
ndbmtod-3) + 1>
```

```
[mysql@ndbmgmd-0 ~]$ ndb_mgm
-- NDB Cluster -- Management Client --
ndb_mgm> CREATE NODEGROUP 3,4
```

Sample output:

```
Connected to Management Server at: localhost:1186
Nodegroup 1 created
```

Note

- This example considers that cnDBTier is initially created with two data pods (node IDs 1 and 2). In this step, the data pod count is scaled up from 2 to 4. Therefore, the node IDs 3 and 4 are assigned to the newly created data nodes.
- If you are adding more than two ndbmtod nodes to the cluster, then create the node groups for the first two nodes, then for the next pair, and so on. For example, if you are adding four ndbmtod nodes with node IDs x1, x2, x3, x4, then first create node groups for x1 and x2 and then for x3 and x4 as shown in the following code block:

```
ndb_mgm> CREATE NODEGROUP x1, x2
ndb_mgm> CREATE NODEGROUP x3, x4
```

16. Depending on the following conditions, use georeplication recovery procedure or the dbt_reorg_table_partition script to redistribute the cluster data:
- If the mate site is available, then perform the georeplication recovery procedure to redistribute the data across all the data nodes (including the new data nodes). For georeplication procedure, see the "Restoring Georeplication Failure" section in *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide*.
 - If the mate site is not available, perform the following steps to run the dbt_reorg_table_partition script to redistribute the data across all the data nodes (including the new data nodes).
 - If the mate site is available, however you don't want to perform the georeplication recovery, then perform the following steps to run the dbt_reorg_table_partition script to redistribute the data across all the data nodes (including the new data nodes).
- a. Run the following commands to source the source_me file. This adds the bin directory with the program to the user PATH and prompts you to enter the namespace. The

namespace is used to set DBTIER_NAMESPACE. Additionally, it sets the DBTIER_LIB environment variable to the path of the directory containing the libraries needed by dbt_reorg_table_partition:

```
$ cd Artifacts/Scripts/tools
$ source ./source_me
```

Sample output:

NOTE: source_me must be sourced while your current directory is the directory with the source_me file.

```
Enter cndbtier namespace: sitea
DBTIER_NAMESPACE = "sitea"
```

```
DBTIER_LIB=/home/cloud-user/user/deploy_cndbtier_sitea_25.2.100/
Artifacts/Scripts/tools/dbtier/lib
```

```
Adding /home/cloud-user/user/deploy_cndbtier_sitea_25.2.100/Artifacts/
Scripts/tools/dbtier/bin to PATH
PATH=/home/cloud-user/.local/bin:/home/cloud-user/bin:/usr/local/
bin:/usr/bin:/usr/local/sbin:/usr/sbin:/var/occne/cluster/occne3-user/
artifacts/istio-1.15.3/bin/:/var/occne/cluster/occne3-user/artifacts:/
home/cloud-user/user/deploy_cndbtier_sitea_25.2.100/Artifacts/Scripts/
tools/dbtier/bin
```

b. Run the dbt_reorg_table_partition script:

```
$ dbt_reorg_table_partition
```

Sample output:

```
dbt_reorg_table_partition 25.2.100
Copyright (c) 2024 Oracle and/or its affiliates. All rights reserved.
2024-01-17T05:11:45Z INFO - Getting sts and sts pod info...
2024-01-17T05:11:45Z INFO - Getting MGM sts and sts pod info...
2024-01-17T05:11:45Z INFO - MGM_STS="ndbmcmd"
2024-01-17T05:11:45Z INFO - MGM_REPLICAS="2"
2024-01-17T05:11:45Z INFO - MGM_PODS:
    ndbmcmd-0
    ndbmcmd-1
2024-01-17T05:11:45Z INFO - Getting NDB sts and sts pod info...
2024-01-17T05:11:45Z INFO - NDB_STS="ndbmttd"
2024-01-17T05:11:45Z INFO - NDB_REPLICAS="2"
2024-01-17T05:11:45Z INFO - NDB_PODS:
    ndbmttd-0
    ndbmttd-1
2024-01-17T05:11:45Z INFO - Getting API sts and sts pod info...
2024-01-17T05:11:45Z INFO - API_STS="ndbmysqld"
2024-01-17T05:11:45Z INFO - API_REPLICAS="2"
2024-01-17T05:11:45Z INFO - API_PODS:
    ndbmysqld-0
    ndbmysqld-1
2024-01-17T05:11:45Z INFO - Getting APP sts and sts pod info...
```



```

2024-01-17T05:11:45Z INFO - APP_STS="ndbappmysqld"
2024-01-17T05:11:45Z INFO - APP_REPLICAS="2"
2024-01-17T05:11:45Z INFO - APP_PODS:
    ndbappmysqld-0
    ndbappmysqld-1
2024-01-17T05:11:45Z INFO - Getting deployment pod info...
2024-01-17T05:11:45Z INFO - grepping for backup-man (BAK_CHART_NAME)...
2024-01-17T05:11:46Z INFO - BAK_PODS:
    mysql-cluster-db-backup-manager-svc-57fb8ff49c-49p55
2024-01-17T05:11:46Z INFO - BAK_DEPLOY:
    mysql-cluster-db-backup-manager-svc
2024-01-17T05:11:46Z INFO - grepping for db-mon (MON_CHART_NAME)...
2024-01-17T05:11:46Z INFO - MON_PODS:
    mysql-cluster-db-monitor-svc-7b7559cd45-shm8r
2024-01-17T05:11:46Z INFO - MON_DEPLOY:
    mysql-cluster-db-monitor-svc
2024-01-17T05:11:46Z INFO - grepping for repl (REP_CHART_NAME)...
2024-01-17T05:11:46Z INFO - REP_PODS:
    mysql-cluster-siteb-local-replication-svc-9c4c59f87-6zl57
2024-01-17T05:11:46Z INFO - REP_DEPLOY:
    mysql-cluster-siteb-local-replication-svc
2024-01-17T05:11:46Z INFO - Reorganizing table partitions...
mysql.ndb_apply_status optimize status OK
backup_info.DBTIER_BACKUP_INFO optimize status OK
backup_info.DBTIER_BACKUP_COMMAND_QUEUE optimize status OK
backup_info.DBTIER_BACKUP_TRANSFER_INFO optimize status OK
mysql.ndb_replication optimize status OK
hbreplica_info.DBTIER_HEARTBEAT_INFO optimize status OK
replication_info.DBTIER_SITE_INFO optimize status OK
replication_info.DBTIER_REPL_SITE_INFO optimize status OK
replication_info.DBTIER_REPLICATION_CHANNEL_INFO optimize
status OK
replication_info.DBTIER_INITIAL_BINLOG_POSTION optimize status
OK
replication_info.DBTIER_REPL_ERROR_SKIP_INFO optimize status
OK
replication_info.DBTIER_REPL_EVENT_INFO optimize status OK
replication_info.DBTIER_REPL_SVC_INFO optimize status OK
2024-01-17T05:12:10Z INFO - Reorganized Tables Partition Successfully

```

17. Run the following command in the management client to check if the data is distributed:

```
ndb_mgm> ALL REPORT MEMORY
```

For example:

```
ndb_mgm> ALL REPORT MEMORY
```

Sample output:

```

Connected to Management Server at: localhost:1186
Node 1: Data usage is 0%(58 32K pages of total 12755)
Node 1: Index usage is 0%(45 32K pages of total 12742)
Node 2: Data usage is 0%(58 32K pages of total 12755)
Node 2: Index usage is 0%(45 32K pages of total 12742)
Node 3: Data usage is 0%(15 32K pages of total 12780)

```

```
Node 3: Index usage is 0%(20 32K pages of total 12785)
Node 4: Data usage is 0%(15 32K pages of total 12780)
Node 4: Index usage is 0%(20 32K pages of total 12785)
```

18. Run the Helm test on the cnDBTier cluster to check the health of the cluster:

```
$ helm test mysql-cluster -n occne-cndbtier
```

Sample output:

```
NAME: mysql-cluster
LAST DEPLOYED: Mon May 20 08:10:11 2025
NAMESPACE: occne-cndbtier
STATUS: deployed
REVISION: 2
TEST SUITE: mysql-cluster-node-connection-test
Last Started: Mon May 20 10:27:49 2025
Last Completed: Mon May 20 10:28:11 2025
Phase: Succeeded
```

You can also verify the status of the cnDBTier from the ndb_mgm console by running the following command:

```
$ kubectl -n occne-cndbtier exec ndbmcmd-0 -- ndb_mgm -e show
```

Sample output:

```
Connected to Management Server at: localhost:1186
Cluster Configuration
-----
[ndbd(NDB)] 4 node(s)
id=1 @10.233.92.53 (mysql-8.4.3 ndb-8.4.3, Nodegroup: 0)
id=2 @10.233.72.66 (mysql-8.4.3 ndb-8.4.3, Nodegroup: 0)
id=3 @10.233.117.60 (mysql-8.4.3 ndb-8.4.3, Nodegroup: 1)
id=4 @10.233.111.64 (mysql-8.4.3 ndb-8.4.3, Nodegroup: 1, *)

[ndb_mgmd(MGM)] 2 node(s)
id=49 @10.233.111.62 (mysql-8.4.3 ndb-8.4.3)
id=50 @10.233.117.59 (mysql-8.4.3 ndb-8.4.3)

[mysqld(API)] 8 node(s)
id=56 @10.233.72.65 (mysql-8.4.3 ndb-8.4.3)
id=57 @10.233.92.51 (mysql-8.4.3 ndb-8.4.3)
id=70 @10.233.72.64 (mysql-8.4.3 ndb-8.4.3)
id=71 @10.233.92.52 (mysql-8.4.3 ndb-8.4.3)
id=222 (not connected, accepting connect from any host)
id=223 (not connected, accepting connect from any host)
id=224 (not connected, accepting connect from any host)
id=225 (not connected, accepting connect from any host)
```

Note

The node IDs 222 to 225 in the sample output are shown as "not connected" as these nodes are added as empty slot IDs used for fault recovery. You can ignore these nodes.

19. Disable `replicationskiperrors` in the `custom_values.yaml` file and apply the changes using the Helm upgrade procedure.

Note

Skip this step if you skipped Step 2.

```
global:
  replicationsskiperrors:
    enable: false
```

20. Run the following command to patch the ndbmtd, ndbmysqld, and ndbappmysqld statefulsets and update updateStrategy to RollingUpdate:

```
$ kubectl -n <namespace> patch sts <ndbmtd sts name> -p '{"spec":
{"updateStrategy":{"type":"RollingUpdate"}}}'
```

```
$ kubectl -n <namespace> patch sts <ndbmysqld sts name> -p '{"spec":
{"updateStrategy":{"type":"RollingUpdate"}}}'
```

```
$ kubectl -n <namespace> patch sts <ndbappmysqld sts name> -p '{"spec":
{"updateStrategy":{"type":"RollingUpdate"}}}'
```

7.3.2 Vertical Scaling

Currently, cnDBTier supports only manual vertical scaling of ndbmtd and SQL pods. This section describes the procedure to manually scale ndbmtd, ndbappmysqld, and ndbmysqld pods.

7.3.2.1 Scaling ndbmtd Pods

This section provides the procedures to vertically scale up ndbmtd pods.

Note

- Before scaling the pods, ensure that the worker nodes have adequate resources to support scaling.
- Perform the Helm test on the deployed cnDBTier cluster to check the health of the cluster:

```
$ helm test mysql-cluster -n occne-cndbtier
```

Sample output:

```
NAME: mysql-cluster
LAST DEPLOYED: Mon May 20 03:24:03 2025
NAMESPACE: occne-cndbtier
STATUS: deployed
REVISION: 1
TEST SUITE:      mysql-cluster-node-connection-test
Last Started:    Mon May 20 04:03:01 2025
Last Completed:  Mon May 20 04:03:26 2025
Phase:          Succeeded
```

You can also verify the status of the cnDBTier from the ndb_mgm console by running the following command:

```
$ kubectl -n occne-cndbtier exec -it ndbmcmd-0 -- ndb_mgm -e show
```

Sample output:

```
Connected to Management Server at: localhost:1186
Cluster Configuration
-----
[ndbd(NDB)]      2 node(s)
id=1   @10.233.95.87  (mysql-8.0.34 ndb-8.0.34, Nodegroup: 0, *)
id=2   @10.233.99.254 (mysql-8.0.34 ndb-8.0.34, Nodegroup: 0)

[ndb_mgmd(MGM)] 2 node(s)
id=49  @10.233.110.157 (mysql-8.0.34 ndb-8.0.34)
id=50  @10.233.101.213 (mysql-8.0.34 ndb-8.0.34)

[mysqld(API)]   10 node(s)
id=56  @10.233.99.243  (mysql-8.0.34 ndb-8.0.34)
id=57  @10.233.101.221 (mysql-8.0.34 ndb-8.0.34)
id=70  @10.233.100.228 (mysql-8.0.34 ndb-8.0.34)
id=71  @10.233.101.217 (mysql-8.0.34 ndb-8.0.34)
id=222 (not connected, accepting connect from any host)
id=223 (not connected, accepting connect from any host)
id=224 (not connected, accepting connect from any host)
id=225 (not connected, accepting connect from any host)
```

The node IDs 222 to 225 in the sample output are shown as "not connected" as these nodes are added as empty slot IDs used for fault recovery. You can ignore these nodes.

Updating CPU and RAM

Perform the following steps to update CPU and RAM using the `custom_values.yaml` file:

1. Configure the required CPU and memory values in the `global` section of the `custom_values.yaml` file.

For example:

The following code block shows the old CPU and RAM values in the `custom_values.yaml` file

```
global:
  ndb:
    datamemory: 400M
ndb:
  resources:
    limits:
      cpu: 1
      memory: 4Gi
    requests:
      cpu: 1
      memory: 4Gi
```

The following code block shows the updated CPU and RAM values in the `custom_values.yaml` file:

```
global:
  ndb:
    datamemory: 800M
ndb:
  resources:
    limits:
      cpu: 2
      memory: 8Gi
    requests:
      cpu: 2
      memory: 8Gi
```

2. Upgrade cnDBTier by performing a Helm upgrade with the modified `custom_values.yaml` file. For more information about the upgrade procedure, see *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide*.

Note

cnDBTier supports upgrade in a TLS enabled cluster in this scenario only.

Updating PVC

Updating PVC Using Helm Upgrade

Perform the following steps to update the PVC value using `custom_values.yaml`:

1. Update the `custom_values.yaml` file with the new PVC values for `ndbmttd` pods:
- For example:

The following code block shows the old PVC values in the `custom_values.yaml` file:

```
global:
  ndb:
```

```
ndbdisksize: 3Gi
ndbbakupdisksize: 3Gi
```

The following code block shows the updated PVC values in the `custom_values.yaml` file:

```
global:
  ndb:
    ndbdisksize: 6Gi
    ndbbakupdisksize: 6Gi
```

2. Run the following command to delete the `ndbmt-d` statefulset and set the dependency to *orphan*:

```
$ kubectl -n occne-cndbtier delete sts --cascade=orphan ndbmt-d
```

3. Run the following command to delete the `ndbmt-d-1` pod and patch the pod's PVC values with the values updated in Step a:

```
$ kubectl -n occne-cndbtier delete pod ndbmt-d-1

$ kubectl -n occne-cndbtier patch -p '{ "spec": { "resources":
{ "requests": { "storage": "6Gi" }}}}' pvc pvc-ndbmt-d-ndbmt-d-1
$ kubectl -n occne-cndbtier patch -p '{ "spec": { "resources":
{ "requests": { "storage": "6Gi" }}}}' pvc pvc-backup-ndbmt-d-ndbmt-d-1
```

4. After patching the PVC values, wait for the new PV to link to the PVC. Run the following command to check if the PV reflects the updated PVC values:

```
$ kubectl get pv | grep -w occne-cndbtier | grep ndbmt-d-1
```

Sample output:

```
pvc-53fe7988-4a81-40d3-a366-fd894de89535 6Gi RW0 Delete Bound occne-
cndbtier/pvc-ndbmt-d-ndbmt-d-1 occne-dbtier-sc 62m
```

5. Upgrade cnDBTier with the modified `custom_values.yaml` file:

```
$ helm upgrade mysql-cluster occndbtier -f occndbtier/custom_values.yaml -
n occne-cndbtier --no-hooks
```

6. Perform Step b through Step e for the `ndbmt-d-0` pod.

Note

If you have more than two data pods, then follow Steps b to e for each `ndbmt-d` pod in the following order: `ndbmt-d-n`, `ndbmt-d-(n-1)`, and so on up to `ndbmt-d-0`.

7. As cnDBTier Helm upgrade is performed with the `--no-hooks` option in step e, `upgradeStrategy` of the cnDBTier StatefulSets is changed to `OnDelete`. Therefore, perform the cnDBTier upgrade one more time to restore `upgradeStrategy` to `RollingRestart`:

Note

Before running the following command, ensure that you set the value of `OCNE_NAMESPACE` variable in the command with your cnDBTier namespace.

```
helm -n ${OCNE_NAMESPACE} upgrade ${OCNE_RELEASE_NAME} occndbtier -f  
occndbtier/custom_values.yaml
```

8. Run the Helm test on the cnDBTier cluster to check the health of the cluster:

```
$ helm test mysql-cluster -n occne-cndbtier
```

Sample output:

```
NAME: mysql-cluster  
LAST DEPLOYED: Mon May 20 04:43:20 2025  
NAMESPACE: occne-cndbtier  
STATUS: deployed  
REVISION: 4  
TEST SUITE: mysql-cluster-node-connection-test  
Last Started: Mon May 20 04:45:15 2025  
Last Completed: Mon May 20 04:45:35 2025  
Phase: Succeeded
```

Updating PVC Using dbtscale_vertical_pvc

`dbtscale_vertical_pvc` is an automated script to update PVC without manual intervention. Perform the following steps to update the PVC value using `custom_values.yaml`:

1. Update the `custom_values.yaml` file with the new PVC values for `ndbmt` pods. Ensure that you only modify the `global.ndb.ndbdisksize`, `global.ndb.ndbbakupdisksize`, or both the sections.
For example:

The following code block shows the old PVC values in the `custom_values.yaml` file:

```
global:  
  ndb:  
    ndbdisksize: 2Gi  
    ndbbakupdisksize: 3Gi
```

The following code block shows the updated PVC values in the `custom_values.yaml` file:

```
global:  
  ndb:  
    ndbdisksize: 3Gi  
    ndbbakupdisksize: 4Gi
```

2. Run the following commands to navigate to the `Artifacts/Scripts/tools/` directory and source the `source_me` file:

Note

- Ensure that the Artifacts/Scripts/tools/ directory contains the source_me file.
- Enter the namespace of the cnDBTier cluster when prompted.

```
$ cd Artifacts/Scripts/tools/
$ source ./source_me
```

Sample output:

```
Enter cndbtier namespace: occne-cndbtier
DBTIER_NAMESPACE = "occne-cndbtier"
```

```
DBTIER_LIB=/home/dbtuser/occne/cluster/site-1/Artifacts/Scripts/tools/lib
```

```
Adding /home/dbtuser/occne/cluster/site-1/Artifacts/Scripts/tools/bin to
PATH
```

```
Adding /home/dbtuser/occne/cluster/site-1/Artifacts/Scripts/tools/bin/
rollbackscripts to PATH
```

```
PATH=/home/dbtuser/occne/cluster/site-1/Artifacts/Scripts/tools/bin/
rollbackscripts:/home/dbtuser/occne/cluster/site-1/Artifacts/Scripts/tools/
bin:/home/dbtuser/.local/bin:/home/dbtuser/bin:/usr/local/bin:/usr/
bin:/usr/local/sbin:/usr/sbin
```

3. Run the dbtscale_vertical_pvc script with the -pods=ndbmt option and pass the helm charts and the updated custom_values.yaml file as parameters:

```
$ dbtscale_vertical_pvc --pods=ndbmt occndbtier
custom_values_scale_pvc.yaml
```

Sample output:

```
2024-12-17T14:40:15Z INFO - HELM_CHARTS = occndbtier
2024-12-17T14:40:15Z INFO - CUSTOM_VALUES_FILE =
custom_values_scale_pvc.yaml
dbtscale_vertical_pvc 25.1.100
Copyright (c) 2024 Oracle and/or its affiliates. All rights reserved.
2024-12-17T14:40:15Z INFO - Started timer dbtscale_vertical_pvc: 1734446415
2024-12-17T14:40:15Z INFO - Started timer PHASE 0: 1734446415
2024-12-17T14:40:15Z INFO -
*****
*****
2024-12-17T14:40:15Z INFO - BEGIN PHASE 0: Collect Site information
2024-12-17T14:40:15Z INFO -
*****
*****
2024-12-17T14:40:15Z INFO - Using IPv4: LOOPBACK_IP="127.0.0.1"
2024-12-17T14:40:15Z INFO - DBTIER_NAMESPACE = occne-cndbtier
...
2024-12-17T14:40:26Z INFO - PODS_TO_SCALE="ndbmt"
```



```

2024-12-17T14:40:26Z INFO - POD_TYPE="ndb"
2024-12-17T14:40:26Z INFO - STS_TO_DELETE="ndbmt-d"
2024-12-17T14:40:26Z INFO - PODS_TO_RESTART="NDB_PODS"
2024-12-17T14:40:27Z INFO - REPL_LEADER_DEPLOY = mysql-cluster-site-1-
site-2-replication-svc
2024-12-17T14:40:27Z INFO - REPL_LEADER_PVC = pvc-site-1-site-2-
replication-svc
2024-12-17T14:40:27Z INFO - REPL_LEADER_POD = mysql-cluster-site-1-site-2-
replication-svc-74df5fw98q
2024-12-17T14:40:27Z INFO -
*****
*****
2024-12-17T14:40:27Z INFO - END PHASE 0: Collect Site information
2024-12-17T14:40:27Z INFO -
*****
*****
2024-12-17T14:40:27Z INFO - Ended timer PHASE 0: 1734446427
2024-12-17T14:40:27Z INFO - PHASE 0 took: 00 hr. 00 min. 12 sec.
2024-12-17T14:40:27Z INFO - Started timer PHASE 1: 1734446427
2024-12-17T14:40:27Z INFO -
*****
*****
2024-12-17T14:40:27Z INFO - BEGIN PHASE 1: Verify disk sizes are supported
2024-12-17T14:40:27Z INFO -
*****
*****
2024-12-17T14:40:27Z INFO - Current PVC sizes:
NAME                                STATUS
VOLUME                             CAPACITY  ACCESS MODES
STORAGECLASS  VOLUMEATTRIBUTESCLASS  AGE
pvc-backup-ndbmt-d-ndbmt-d-0        Bound
pvc-55d60044-2df4-47b7-92aa-4d27d166db7f  2Gi      RWX
standard      <unset>                        30h
pvc-backup-ndbmt-d-ndbmt-d-1        Bound
pvc-1d81991c-7c9a-4d79-9265-ee6c5ddaa8f2  2Gi      RWX
standard      <unset>                        30h
pvc-ndbmt-d-ndbmt-d-0               Bound
pvc-8d55a76a-17e8-47d0-bb4b-8a6c31c76c96  3Gi      RWX
standard      <unset>                        30h
pvc-ndbmt-d-ndbmt-d-1               Bound
ab48-40f4-ad23-43d9b030f03f  3Gi      RWX
<unset>                        30h
2024-12-17T14:40:27Z INFO - Requested PVC sizes from
custom_values_scale_pvc.yaml:
ndb:
  ndbdisksize: 4Gi
  ndbbackupdisksize: 3Gi
  use_separate_backup_disk: true
2024-12-17T14:40:27Z INFO - Current and requested disk values:
2024-12-17T14:40:27Z INFO - current_pvcsz=3Gi (3221225472),
requested_pvcsz=4Gi (4294967296)
2024-12-17T14:40:27Z INFO - requested_use_separate_backup_disk=true
2024-12-17T14:40:27Z INFO - current_pvcbakupsz=2Gi (2147483648),
requested_pvcbakupsz=3Gi (3221225472)
2024-12-17T14:40:27Z INFO - Verifying current and requested disk sizes...
2024-12-17T14:40:27Z INFO - Requested PVC values should not be equal to

```

```
current - PASSED
2024-12-17T14:40:27Z INFO - No requested value should be smaller than
current - PASSED
2024-12-17T14:40:27Z INFO - No requested value should be zero or negative
- PASSED
2024-12-17T14:40:27Z INFO - At least one requested value should be larger
than its current value - PASSED
2024-12-17T14:40:27Z INFO - Should not remove backup PVC - PASSED
2024-12-17T14:40:27Z INFO - db-replication-svc requested PVC values should
equal to current - PASSED
2024-12-17T14:40:27Z INFO - ndbmysqld requested PVC values should equal to
current - PASSED
2024-12-17T14:40:27Z INFO - ndbappmysqld requested PVC values should equal
to current - PASSED
2024-12-17T14:40:27Z INFO - Verified current and requested disk sizes.
2024-12-17T14:40:27Z INFO -
*****
*****
2024-12-17T14:40:27Z INFO - END PHASE 1: Verify disk sizes are supported
2024-12-17T14:40:27Z INFO -
*****
*****
2024-12-17T14:40:27Z INFO - Ended timer PHASE 1: 1734446427
2024-12-17T14:40:27Z INFO - PHASE 1 took: 00 hr. 00 min. 00 sec.
2024-12-17T14:40:27Z INFO - Started timer PHASE 2: 1734446427
2024-12-17T14:40:27Z INFO -
*****
*****
2024-12-17T14:40:27Z INFO - BEGIN PHASE 2: Delete/scale down statefulset/
deployment
2024-12-17T14:40:27Z INFO -
*****
*****
2024-12-17T14:40:27Z INFO - Deleting STS...
2024-12-17T14:40:27Z INFO - kubectl -n occne-cndbtier delete sts --
cascade=orphan "ndbmttd"
statefulset.apps "ndbmttd" deleted
2024-12-17T14:40:27Z INFO - STS deleted
2024-12-17T14:40:27Z INFO -
*****
*****
2024-12-17T14:40:27Z INFO - END PHASE 2: Delete/scale down statefulset/
deployment
2024-12-17T14:40:27Z INFO -
*****
*****
2024-12-17T14:40:27Z INFO - Ended timer PHASE 2: 1734446427
2024-12-17T14:40:27Z INFO - PHASE 2 took: 00 hr. 00 min. 00 sec.
2024-12-17T14:40:27Z INFO - Started timer PHASE 3: 1734446427
2024-12-17T14:40:27Z INFO -
*****
*****
2024-12-17T14:40:27Z INFO - BEGIN PHASE 3: Upgrade cnDBTier --no-hooks or
patch db-replication-svc leader PVC
2024-12-17T14:40:27Z INFO -
*****
```

```

*****
2024-12-17T14:40:27Z INFO - Upgrading cnDBTier
(custom_values_scale_pvc.yaml)...
Release "mysql-cluster" has been upgraded. Happy Helming!
NAME: mysql-cluster
LAST DEPLOYED: Tue Dec 17 14:40:28 2024
NAMESPACE: occne-cndbtier
STATUS: deployed
REVISION: 10
2024-12-17T14:40:32Z INFO - Helm upgrade returned
2024-12-17T14:40:32Z INFO -
*****
*****
2024-12-17T14:40:32Z INFO - END PHASE 3: Upgrade cnDBTier --no-hooks or
patch db-replication-svc leader PVC
2024-12-17T14:40:32Z INFO -
*****
*****
2024-12-17T14:40:32Z INFO - Ended timer PHASE 3: 1734446432
2024-12-17T14:40:32Z INFO - PHASE 3 took: 00 hr. 00 min. 05 sec.
2024-12-17T14:40:32Z INFO - Started timer PHASE 4: 1734446432
2024-12-17T14:40:32Z INFO -
*****
*****
2024-12-17T14:40:32Z INFO - BEGIN PHASE 4: Delete PVCs and restart pods or
wait for repl PV with new site to bound
2024-12-17T14:40:32Z INFO -
*****
*****
2024-12-17T14:40:32Z INFO - Determine NDB Node President...
2024-12-17T14:40:33Z INFO - PRESIDENT_NODE=1
2024-12-17T14:40:33Z INFO - PRESIDENT_GROUP=0
2024-12-17T14:40:33Z INFO - NODES_IN_PRESIDENTS_GROUP=(1 2)
2024-12-17T14:40:33Z INFO - pod_name_witout_id = "ndbmt-d-"
2024-12-17T14:40:33Z INFO - pods_in_president_group = (ndbmt-d-0 ndbmt-d-1)
2024-12-17T14:40:33Z INFO - Restart pods after deleting their PVCs...
2024-12-17T14:40:33Z INFO - Deleting pods NOT in president's group (0)...
2024-12-17T14:40:33Z INFO - first_set_of_pods_to_delete = ()
2024-12-17T14:40:33Z INFO - delete_sts_pod_and_its_pvcs_one_at_a_time:
nothing to do; pods array is empty ()
2024-12-17T14:40:33Z INFO - Deleting president pod (ndbmt-d-0)...
2024-12-17T14:40:33Z INFO - president_set = (ndbmt-d-0)
2024-12-17T14:40:33Z INFO - Deleting PVCs for ndbmt-d-0 (pvc-backup-ndbmt-d-
ndbmt-d-0 pvc-ndbmt-d-ndbmt-d-0)...
2024-12-17T14:40:33Z INFO - deleting pod ndbmt-d-0 ...
persistentvolumeclaim "pvc-backup-ndbmt-d-ndbmt-d-0" deleted
persistentvolumeclaim "pvc-ndbmt-d-ndbmt-d-0" deleted
pod "ndbmt-d-0" deleted
2024-12-17T14:40:45Z INFO - Waiting for condition: is_pod ndbmt-d-0...
2024-12-17T14:40:45Z INFO - Condition occurred
2024-12-17T14:41:28Z INFO - Waiting for condition: is_pod_ready ndbmt-d-0...
2024-12-17T14:41:28Z INFO - Condition occurred
2024-12-17T14:41:28Z INFO - kubectl -n occne-cndbtier get pod ndbmt-d-0
NAME          READY   STATUS    RESTARTS   AGE
ndbmt-d-0     3/3     Running   0           43s
2024-12-17T14:41:28Z INFO - kubectl -n occne-cndbtier get pvc pvc-backup-

```

```

ndbmt-d-ndbmt-d-0 pvc-ndbmt-d-ndbmt-d-0
NAME                                STATUS
VOLUME                                CAPACITY  ACCESS MODES
STORAGECLASS  VOLUMEATTRIBUTESCLASS  AGE
pvc-backup-ndbmt-d-ndbmt-d-0  Bound  pvc-83582dba-f394-4a1e-
b1b1-3a1466f643e8  3Gi    RW0    standard
<unset>                                43s
pvc-ndbmt-d-ndbmt-d-0  Bound  pvc-0aa26c58-471d-4be3-
b148-9c192165daf6  4Gi    RW0    standard
<unset>                                43s
2024-12-17T14:41:28Z INFO - Deleting non-president pods in president's
group (0)...
2024-12-17T14:41:28Z INFO - pods_in_president_group_without_president =
(ndbmt-d-1)
2024-12-17T14:41:28Z INFO - Deleting PVCs for ndbmt-d-1 (pvc-backup-ndbmt-d-
ndbmt-d-1 pvc-ndbmt-d-ndbmt-d-1)...
2024-12-17T14:41:28Z INFO - deleting pod ndbmt-d-1 ...
persistentvolumeclaim "pvc-backup-ndbmt-d-ndbmt-d-1" deleted
persistentvolumeclaim "pvc-ndbmt-d-ndbmt-d-1" deleted
pod "ndbmt-d-1" deleted
2024-12-17T14:41:39Z INFO - Waiting for condition: is_pod ndbmt-d-1...
2024-12-17T14:41:39Z INFO - Condition occurred
2024-12-17T14:42:20Z INFO - Waiting for condition: is_pod_ready ndbmt-d-1...
2024-12-17T14:42:20Z INFO - Condition occurred
2024-12-17T14:42:20Z INFO - kubectl -n occne-cnDBTier get pod ndbmt-d-1
NAME      READY  STATUS   RESTARTS  AGE
ndbmt-d-1  3/3    Running  0          41s
2024-12-17T14:42:20Z INFO - kubectl -n occne-cnDBTier get pvc pvc-backup-
ndbmt-d-ndbmt-d-1 pvc-ndbmt-d-ndbmt-d-1
NAME                                STATUS
VOLUME                                CAPACITY  ACCESS MODES
STORAGECLASS  VOLUMEATTRIBUTESCLASS  AGE
pvc-backup-ndbmt-d-ndbmt-d-1  Bound  pvc-419ea3ef-af00-4e58-
a5fb-9a774c764a6e  3Gi    RW0    standard
<unset>                                41s
pvc-ndbmt-d-ndbmt-d-1  Bound  pvc-fa0c7826-337d-4805-
a1dd-2b6f2961270a  4Gi    RW0    standard
<unset>                                41s
2024-12-17T14:42:20Z INFO - Pods restarted and their PVCs recreated.
2024-12-17T14:42:20Z INFO -
*****
*****
2024-12-17T14:42:20Z INFO - END PHASE 4: Delete PVCs and restart pods or
wait for repl PV with new site to bound
2024-12-17T14:42:20Z INFO -
*****
*****
2024-12-17T14:42:20Z INFO - Ended timer PHASE 4: 1734446540
2024-12-17T14:42:20Z INFO - PHASE 4 took: 00 hr. 01 min. 48 sec.
2024-12-17T14:42:20Z INFO - Started timer PHASE 5: 1734446540
2024-12-17T14:42:20Z INFO -
*****
*****
2024-12-17T14:42:20Z INFO - BEGIN PHASE 5: Upgrade cnDBTier
2024-12-17T14:42:20Z INFO -
*****

```

```

*****
2024-12-17T14:42:20Z INFO - Upgrading cnDBTier
(custom_values_scale_pvc.yaml)...
Release "mysql-cluster" has been upgraded. Happy Helming!
NAME: mysql-cluster
LAST DEPLOYED: Tue Dec 17 14:42:21 2024
NAMESPACE: occne-cndbtier
STATUS: deployed
REVISION: 11
2024-12-17T14:43:51Z INFO - Helm upgrade returned
2024-12-17T14:43:51Z INFO -
*****
*****
2024-12-17T14:43:51Z INFO - END PHASE 5: Upgrade cnDBTier
2024-12-17T14:43:51Z INFO -
*****
*****
2024-12-17T14:43:51Z INFO - Ended timer PHASE 5: 1734446631
2024-12-17T14:43:51Z INFO - PHASE 5 took: 00 hr. 01 min. 31 sec.
2024-12-17T14:43:51Z INFO - Started timer PHASE 6: 1734446631
2024-12-17T14:43:51Z INFO -
*****
*****
2024-12-17T14:43:51Z INFO - BEGIN PHASE 6: Post-processing
2024-12-17T14:43:51Z INFO -
*****
*****
2024-12-17T14:43:51Z INFO - CURRENT_PVCS:
NAME                                STATUS
VOLUME                             CAPACITY  ACCESS MODES
STORAGECLASS  VOLUMEATTRIBUTESCLASS  AGE
pvc-backup-ndbmt-d-0                                Bound
pvc-55d60044-2df4-47b7-92aa-4d27d166db7f  2Gi      RW0
standard      <unset>                                30h
pvc-backup-ndbmt-d-1                                Bound
pvc-1d81991c-7c9a-4d79-9265-ee6c5ddaa8f2  2Gi      RW0
standard      <unset>                                30h
pvc-ndbmt-d-0                                Bound
pvc-8d55a76a-17e8-47d0-bb4b-8a6c31c76c96  3Gi      RW0
standard      <unset>                                30h
pvc-ndbmt-d-1                                Bound
ab48-40f4-ad23-43d9b030f03f  3Gi      RW0
<unset>                                30h
2024-12-17T14:43:51Z INFO - after_pvcs:
NAME                                STATUS
VOLUME                             CAPACITY  ACCESS MODES
STORAGECLASS  VOLUMEATTRIBUTESCLASS  AGE
pvc-backup-ndbmt-d-0                                Bound
f394-4a1e-b1b1-3a1466f643e8  3Gi      RW0
<unset>                                3m6s
pvc-backup-ndbmt-d-1                                Bound
af00-4e58-a5fb-9a774c764a6e  3Gi      RW0
<unset>                                2m12s
pvc-ndbmt-d-0                                Bound
pvc-0aa26c58-471d-4be3-b148-9c192165daf6  4Gi      RW0
standard      <unset>                                3m6s

```

```

pvc-ndbmt-d-ndbmt-d-1          Bound      pvc-
fa0c7826-337d-4805-a1dd-2b6f2961270a    4Gi        RW0
standard      <unset>          2m12s
2024-12-17T14:43:51Z INFO -
*****
*****
2024-12-17T14:43:51Z INFO - END PHASE 6: Post-processing
2024-12-17T14:43:51Z INFO -
*****
*****
2024-12-17T14:43:51Z INFO - Ended timer PHASE 6: 1734446631
2024-12-17T14:43:51Z INFO - PHASE 6 took: 00 hr. 00 min. 00 sec.
2024-12-17T14:43:51Z INFO - Ended timer dbtscale_vertical_pvc: 1734446631
2024-12-17T14:43:51Z INFO - dbtscale_vertical_pvc took: 00 hr. 03 min. 36
sec.
2024-12-17T14:43:51Z INFO - dbtscale_vertical_pvc completed successfully

```

7.3.2.2 Scaling ndbappmysqld Pods

This section provides the procedures to vertically scale up ndbappmysqld pods.

❗ Note

- Before scaling the pods, ensure that the worker nodes have adequate resources to support scaling.
- Before you proceed with the vertical scaling procedure for ndbappmysqld, perform a Helm test and ensure that all the cnDBTier services are running smoothly.

Updating CPU and RAM

Perform the following steps to update CPU and RAM using the `custom_values.yaml` file:

1. Configure the required CPU and memory values in the `global` section of the `custom_values.yaml` file.

For example:

```

ndbapp:
  resources:
    limits:
      cpu: 8
      memory: 10Gi
    requests:
      cpu: 8
      memory: 10Gi

```

2. Upgrade cnDBTier by performing a Helm upgrade with the modified `custom_values.yaml` file. For more information about the upgrade procedure, see *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide*.

❗ Note

cnDBTier supports upgrade in a TLS enabled cluster in this scenario only.

Updating PVC

Updating PVC Using Helm Upgrade

Perform the following steps to update the PVC value using `custom_values.yaml`:

1. Update the `custom_values.yaml` file with the new PVC values for `ndbappmysqld` pods:

```
global:
  ndbapp:
    ndbdisksize: 10Gi
```

2. Delete the `ndbappmysqld` statefulset and make the dependencies as orphan:

```
$ kubectl -n occne-cndbtier delete sts --cascade=orphan ndbappmysqld
```

3. Delete the `ndbappmysqld-1` pod and patch its PVCs with the new PVC value:

- a. Delete the `ndbappmysqld-1` pod:

```
$ kubectl -n occne-cndbtier delete pod ndbappmysqld-1
```

Sample output:

```
pod "ndbappmysqld-1" deleted
```

- b. Patch the PVCs of `ndbappmysqld-1` pod with the new PVC values:

```
$ kubectl -n occne-cndbtier patch -p '{ "spec": { "resources": { "requests": { "storage": "10Gi" } } } }' pvc pvc-ndbappmysqld-ndbappmysqld-1
```

- c. Wait for the new PV to attach with the PVC.

4. Upgrade `cnDBTier` with the modified `custom_values.yaml` file:

```
$ helm upgrade mysql-cluster occndbtier -f occndbtier/custom_values.yaml -n occne-cndbtier --no-hooks
```

5. Repeat steps b through d for `ndbappmysqld-0`.
6. As `cnDBTier` Helm upgrade is performed with the `--no-hooks` option in step d, `upgradeStrategy` of the `cnDBTier` StatefulSets is changed to `OnDelete`. Therefore, perform the `cnDBTier` upgrade one more time to restore `upgradeStrategy` to `RollingRestart`:

Note

Before running the following command, ensure that you set the value of `OCNE_NAMESPACE` variable in the command with your `cnDBTier` namespace.

```
helm -n ${OCNE_NAMESPACE} upgrade ${OCNE_RELEASE_NAME} occndbtier -f occndbtier/custom_values.yaml
```

Updating PVC Using `dbtscale_vertical_pvc`

`dbtscale_vertical_pvc` is an automated script to update PVC without manual intervention. Perform the following steps to update the PVC value using `custom_values.yaml`:

1. Update the `custom_values.yaml` file with the new PVC values for `ndbmt` pods. Ensure that you modify the `global.ndbapp.ndbdisksize` section only.
For example:

The following code block shows the old PVC values in the `custom_values.yaml` file:

```
global:
  ndbapp:
    ndbdisksize: 3Gi
```

The following code block shows the updated PVC values in the `custom_values.yaml` file:

```
global:
  ndbapp:
    ndbdisksize: 4Gi
```

2. Run the following commands to navigate to the `Artifacts/Scripts/tools/` directory and source the `source_me` file:

Note

- Ensure that the `Artifacts/Scripts/tools/` directory contains the `source_me` file.
- After running the script, enter the namespace of the cnDBTier cluster when prompted.

```
$ cd Artifacts/Scripts/tools/
$ source ./source_me
```

Sample output:

```
Enter cndbtier namespace: occne-cndbtier
DBTIER_NAMESPACE = "occne-cndbtier"
```

```
DBTIER_LIB=/home/dbtuser/occne/cluster/site-1/Artifacts/Scripts/tools/lib
```

```
Adding /home/dbtuser/occne/cluster/site-1/Artifacts/Scripts/tools/bin to
PATH
```

```
Adding /home/dbtuser/occne/cluster/site-1/Artifacts/Scripts/tools/bin/
rollbackscripts to PATH
```

```
PATH=/home/dbtuser/occne/cluster/site-1/Artifacts/Scripts/tools/bin/
rollbackscripts:/home/dbtuser/occne/cluster/site-1/Artifacts/Scripts/tools/
bin:/home/dbtuser/.local/bin:/home/dbtuser/bin:/usr/local/bin:/usr/
bin:/usr/local/sbin:/usr/sbin
```

3. Run the `dbtscale_vertical_pvc` script with the `--pods=ndbappmysqld` option and pass the helm charts and the updated `custom_values.yaml` file as parameters:

```
$ dbtscale_vertical_pvc --pods=ndbappmysqld occndbtier
custom_values_scale_pvc.yaml
```


Sample output:

```

2024-12-17T18:36:39Z INFO - HELM_CHARTS = occndbtier
2024-12-17T18:36:39Z INFO - CUSTOM_VALUES_FILE =
custom_values_scale_pvc.yaml
dbtscale_vertical_pvc 25.1.100
Copyright (c) 2024 Oracle and/or its affiliates. All rights reserved.
2024-12-17T18:36:39Z INFO - Started timer dbtscale_vertical_pvc: 1734460599
2024-12-17T18:36:39Z INFO - Started timer PHASE 0: 1734460599
2024-12-17T18:36:39Z INFO -
*****
*****
2024-12-17T18:36:39Z INFO - BEGIN PHASE 0: Collect Site information
2024-12-17T18:36:39Z INFO -
*****
*****
2024-12-17T18:36:39Z INFO - Using IPv4: LOOPBACK_IP="127.0.0.1"
2024-12-17T18:36:39Z INFO - DBTIER_NAMESPACE = occne-cndbtier
...
2024-12-17T18:36:51Z INFO - PODS_TO_SCALE="ndbappmysqld"
2024-12-17T18:36:51Z INFO - POD_TYPE="ndbapp"
2024-12-17T18:36:51Z INFO - STS_TO_DELETE="ndbappmysqld"
2024-12-17T18:36:51Z INFO - PODS_TO_RESTART="APP_PODS"
2024-12-17T18:36:51Z INFO - REPL_LEADER_DEPLOY = mysql-cluster-site-1-
site-2-replication-svc
2024-12-17T18:36:51Z INFO - REPL_LEADER_PVC = pvc-site-1-site-2-
replication-svc
2024-12-17T18:36:51Z INFO - REPL_LEADER_POD = mysql-cluster-site-1-site-2-
replication-svc-58856dds8ql
2024-12-17T18:36:51Z INFO -
*****
*****
2024-12-17T18:36:51Z INFO - END PHASE 0: Collect Site information
2024-12-17T18:36:51Z INFO -
*****
*****
2024-12-17T18:36:51Z INFO - Ended timer PHASE 0: 1734460611
2024-12-17T18:36:51Z INFO - PHASE 0 took: 00 hr. 00 min. 12 sec.
2024-12-17T18:36:51Z INFO - Started timer PHASE 1: 1734460611
2024-12-17T18:36:51Z INFO -
*****
*****
2024-12-17T18:36:51Z INFO - BEGIN PHASE 1: Verify disk sizes are supported
2024-12-17T18:36:51Z INFO -
*****
*****
2024-12-17T18:36:51Z INFO - Current PVC sizes:
NAME                                STATUS
VOLUME                             CAPACITY  ACCESS MODES
STORAGECLASS  VOLUMEATTRIBUTESCLASS  AGE
pvc-ndbappmysqld-ndbappmysqld-0    Bound
pvc-64e675c9-458f-48dd-a827-f43784f695ae  3Gi      RWO
standard      <unset>                33h
pvc-ndbappmysqld-ndbappmysqld-1    Bound    pvc-4e9879d8-
ce42-4115-9d7b-4d079d7d84ea  3Gi      RWO      standard
<unset>                33h

```

```

2024-12-17T18:36:51Z INFO - Requested PVC sizes from
custom_values_scale_pvc.yaml:
  ndbapp:
    ndbdisksize: 4Gi
2024-12-17T18:36:51Z INFO - Current and requested disk values:
2024-12-17T18:36:51Z INFO - current_pvcsz=3Gi (3221225472),
requested_pvcsz=4Gi (4294967296)
2024-12-17T18:36:51Z INFO - Verifying current and requested disk sizes...
2024-12-17T18:36:51Z INFO - Requested PVC values should not be equal to
current - PASSED
2024-12-17T18:36:51Z INFO - No requested value should be smaller than
current - PASSED
2024-12-17T18:36:51Z INFO - No requested value should be zero or negative
- PASSED
2024-12-17T18:36:51Z INFO - At least one requested value should be larger
than its current value - PASSED
2024-12-17T18:36:51Z INFO - db-replication-svc requested PVC values should
equal to current - PASSED
2024-12-17T18:36:51Z INFO - ndbmysqld requested PVC values should equal to
current - PASSED
2024-12-17T18:36:51Z INFO - ndbmtd requested PVC values should equal to
current - PASSED
2024-12-17T18:36:51Z INFO - Verified current and requested disk sizes.
2024-12-17T18:36:51Z INFO -
*****
*****
2024-12-17T18:36:51Z INFO - END PHASE 1: Verify disk sizes are supported
2024-12-17T18:36:51Z INFO -
*****
*****
2024-12-17T18:36:51Z INFO - Ended timer PHASE 1: 1734460611
2024-12-17T18:36:51Z INFO - PHASE 1 took: 00 hr. 00 min. 00 sec.
2024-12-17T18:36:51Z INFO - Started timer PHASE 2: 1734460611
2024-12-17T18:36:51Z INFO -
*****
*****
2024-12-17T18:36:51Z INFO - BEGIN PHASE 2: Delete/scale down statefulset/
deployment
2024-12-17T18:36:51Z INFO -
*****
*****
2024-12-17T18:36:51Z INFO - Deleting STS...
2024-12-17T18:36:51Z INFO - kubectl -n occne-cndbtier delete sts --
cascade=orphan "ndbappmysqld"
statefulset.apps "ndbappmysqld" deleted
2024-12-17T18:36:52Z INFO - STS deleted
2024-12-17T18:36:52Z INFO -
*****
*****
2024-12-17T18:36:52Z INFO - END PHASE 2: Delete/scale down statefulset/
deployment
2024-12-17T18:36:52Z INFO -
*****
*****
2024-12-17T18:36:52Z INFO - Ended timer PHASE 2: 1734460612
2024-12-17T18:36:52Z INFO - PHASE 2 took: 00 hr. 00 min. 01 sec.

```

```
2024-12-17T18:36:52Z INFO - Started timer PHASE 3: 1734460612
2024-12-17T18:36:52Z INFO -
*****
*****
2024-12-17T18:36:52Z INFO - BEGIN PHASE 3: Upgrade cnDBTier --no-hooks or
patch db-replication-svc leader PVC
2024-12-17T18:36:52Z INFO -
*****
*****
2024-12-17T18:36:52Z INFO - Upgrading cnDBTier
(custom_values_scale_pvc.yaml)...
Release "mysql-cluster" has been upgraded. Happy Helming!
NAME: mysql-cluster
LAST DEPLOYED: Tue Dec 17 18:36:52 2024
NAMESPACE: occne-cndbtier
STATUS: deployed
REVISION: 12
2024-12-17T18:36:56Z INFO - Helm upgrade returned
2024-12-17T18:36:56Z INFO -
*****
*****
2024-12-17T18:36:56Z INFO - END PHASE 3: Upgrade cnDBTier --no-hooks or
patch db-replication-svc leader PVC
2024-12-17T18:36:56Z INFO -
*****
*****
2024-12-17T18:36:56Z INFO - Ended timer PHASE 3: 1734460616
2024-12-17T18:36:56Z INFO - PHASE 3 took: 00 hr. 00 min. 04 sec.
2024-12-17T18:36:56Z INFO - Started timer PHASE 4: 1734460616
2024-12-17T18:36:56Z INFO -
*****
*****
2024-12-17T18:36:56Z INFO - BEGIN PHASE 4: Delete PVCs and restart pods or
wait for repl PV with new site to bound
2024-12-17T18:36:56Z INFO -
*****
*****
2024-12-17T18:36:56Z INFO - Restart pods after deleting their PVCs...
2024-12-17T18:36:56Z INFO - pods_to_delete = (ndbappmysql-0
ndbappmysql-1)
2024-12-17T18:36:56Z INFO - Deleting PVCs for ndbappmysql-0 (pvc-
ndbappmysql-ndbappmysql-0)...
2024-12-17T18:36:56Z INFO - deleting pod ndbappmysql-0 ...
persistentvolumeclaim "pvc-ndbappmysql-ndbappmysql-0" deleted
pod "ndbappmysql-0" deleted
2024-12-17T18:37:01Z INFO - Waiting for condition: is_pod ndbappmysql-0...
2024-12-17T18:37:01Z INFO - Condition occurred
2024-12-17T18:37:51Z INFO - Waiting for condition: is_pod_ready
ndbappmysql-0...
2024-12-17T18:37:51Z INFO - Condition occurred
2024-12-17T18:37:51Z INFO - Deleting PVCs for ndbappmysql-1 (pvc-
ndbappmysql-ndbappmysql-1)...
2024-12-17T18:37:51Z INFO - deleting pod ndbappmysql-1 ...
persistentvolumeclaim "pvc-ndbappmysql-ndbappmysql-1" deleted
pod "ndbappmysql-1" deleted
2024-12-17T18:37:55Z INFO - Waiting for condition: is_pod ndbappmysql-1...
```

```

2024-12-17T18:37:55Z INFO - Condition occurred
2024-12-17T18:38:36Z INFO - Waiting for condition: is_pod_ready
ndbappmysqld-1...
2024-12-17T18:38:36Z INFO - Condition occurred
2024-12-17T18:38:36Z INFO - kubectl -n occne-cndbtier get pod
ndbappmysqld-0 ndbappmysqld-1
NAME                READY   STATUS    RESTARTS   AGE
ndbappmysqld-0      2/2     Running   0           95s
ndbappmysqld-1      2/2     Running   0           41s
2024-12-17T18:38:36Z INFO - kubectl -n occne-cndbtier get pvc pvc-
ndbappmysqld-ndbappmysqld-0 pvc-ndbappmysqld-ndbappmysqld-1
NAME                STATUS
VOLUME              CAPACITY   ACCESS MODES
STORAGECLASS        VOLUMEATTRIBUTESCLASS  AGE
pvc-ndbappmysqld-ndbappmysqld-0  Bound      pvc-ccd68518-
ea7e-46b7-95f3-3db730a0f914  4Gi        RWX          standard
<unset>              95s
pvc-ndbappmysqld-ndbappmysqld-1  Bound      pvc-0f5ba7b1-bb0a-4125-
bd5e-738b8b486bc6  4Gi        RWX          standard
<unset>              41s
2024-12-17T18:38:36Z INFO - Pods restarted and their PVCs recreated.
2024-12-17T18:38:36Z INFO -
*****
*****
2024-12-17T18:38:36Z INFO - END PHASE 4: Delete PVCs and restart pods or
wait for repl PV with new site to bound
2024-12-17T18:38:36Z INFO -
*****
*****
2024-12-17T18:38:36Z INFO - Ended timer PHASE 4: 1734460716
2024-12-17T18:38:36Z INFO - PHASE 4 took: 00 hr. 01 min. 40 sec.
2024-12-17T18:38:36Z INFO - Started timer PHASE 5: 1734460716
2024-12-17T18:38:36Z INFO -
*****
*****
2024-12-17T18:38:36Z INFO - BEGIN PHASE 5: Upgrade cnDBTier
2024-12-17T18:38:36Z INFO -
*****
*****
2024-12-17T18:38:36Z INFO - Upgrading cnDBTier
(custom_values_scale_pvc.yaml)...
Release "mysql-cluster" has been upgraded. Happy Helming!
NAME: mysql-cluster
LAST DEPLOYED: Tue Dec 17 18:38:36 2024
NAMESPACE: occne-cndbtier
STATUS: deployed
REVISION: 13
2024-12-17T18:40:17Z INFO - Helm upgrade returned
2024-12-17T18:40:17Z INFO -
*****
*****
2024-12-17T18:40:17Z INFO - END PHASE 5: Upgrade cnDBTier
2024-12-17T18:40:17Z INFO -
*****
*****
2024-12-17T18:40:17Z INFO - Ended timer PHASE 5: 1734460817

```

```

2024-12-17T18:40:17Z INFO - PHASE 5 took: 00 hr. 01 min. 41 sec.
2024-12-17T18:40:17Z INFO - Started timer PHASE 6: 1734460817
2024-12-17T18:40:17Z INFO -
*****
*****

2024-12-17T18:40:17Z INFO - BEGIN PHASE 6: Post-processing
2024-12-17T18:40:17Z INFO -
*****
*****

2024-12-17T18:40:17Z INFO - CURRENT_PVCS:
NAME                                     STATUS
VOLUME                                CAPACITY  ACCESS MODES
STORAGECLASS  VOLUMEATTRIBUTESCLASS  AGE
pvc-ndbappmysql-ndbappmysql-0          Bound
pvc-64e675c9-458f-48dd-a827-f43784f695ae 3Gi       RW0
standard      <unset>                          33h
pvc-ndbappmysql-ndbappmysql-1          Bound      pvc-4e9879d8-
ce42-4115-9d7b-4d079d7d84ea 3Gi       RW0      standard
<unset>      33h
2024-12-17T18:40:17Z INFO - after_pvcs:
NAME                                     STATUS
VOLUME                                CAPACITY  ACCESS MODES
STORAGECLASS  VOLUMEATTRIBUTESCLASS  AGE
pvc-ndbappmysql-ndbappmysql-0          Bound      pvc-ccd68518-
ea7e-46b7-95f3-3db730a0f914 4Gi       RW0      standard
<unset>      3m16s
pvc-ndbappmysql-ndbappmysql-1          Bound      pvc-0f5ba7b1-
bb0a-4125-bd5e-738b8b486bc6 4Gi       RW0      standard
<unset>      2m22s
2024-12-17T18:40:17Z INFO -
*****
*****

2024-12-17T18:40:17Z INFO - END PHASE 6: Post-processing
2024-12-17T18:40:17Z INFO -
*****
*****

2024-12-17T18:40:17Z INFO - Ended timer PHASE 6: 1734460817
2024-12-17T18:40:17Z INFO - PHASE 6 took: 00 hr. 00 min. 00 sec.
2024-12-17T18:40:17Z INFO - Ended timer dbtscale_vertical_pvc: 1734460817
2024-12-17T18:40:17Z INFO - dbtscale_vertical_pvc took: 00 hr. 03 min. 38
sec.
2024-12-17T18:40:17Z INFO - dbtscale_vertical_pvc completed successfully

```

7.3.2.3 Scaling ndbmysql Pods

This section provides the procedures to vertically scale up ndbmysql pods.

Note

- Before scaling the pods, ensure that the worker nodes have adequate resources to support scaling.
- Before you proceed with the vertical scaling procedure for ndbmysql, perform a Helm test and ensure that all the cnDBTier services are running smoothly.

Updating CPU and RAM

Perform the following steps to update CPU and RAM using the `custom_values.yaml` file:

1. Configure the required CPU and memory values in the global section of the `custom_values.yaml` file.

For example:

```
api:
  resources:
    limits:
      cpu: 8
      memory: 10Gi
    requests:
      cpu: 8
      memory: 10Gi
```

2. Upgrade cnDBTier by performing a Helm upgrade with the modified `custom_values.yaml` file. For more information about the upgrade procedure, see *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide*.

Updating PVC

Updating PVC Using Helm Upgrade

Perform the following steps to update the PVC value using `custom_values.yaml`:

1. Update the `custom_values.yaml` file with the new PVC values for `ndbmysqld` pods:

```
global:
  api:
    ndbdisksize: 10Gi
```

2. Delete the `ndbmysqld` statefulset and make the dependencies as orphan:

```
$ kubectl -n occne-cndbtier delete sts --cascade=orphan ndbmysqld
```

3. Delete the `ndbmysqld-1` pod and patch its PVCs with the new PVC value:

- a. Delete the `ndbmysqld-1` pod:

```
$ kubectl -n occne-cndbtier delete pod ndbmysqld-1
```

Sample output:

```
pod "ndbmysqld-1" deleted
```

- b. Patch the PVCs of `ndbmysqld-1` pod with the new PVC values:

```
$ kubectl -n occne-cndbtier patch -p '{ "spec": { "resources": { "requests": { "storage": "10Gi" }}}}' pvc pvc-ndbmysqld-ndbmysqld-1
```

4. Wait for the new PV to attach with the PVC. Run the following command to check if the PV reflects the updated PVC values:

```
$ kubectl get pv | grep -w occne-cndbtier | grep ndbmysqld-1
```

Sample output:

```
pvc-ee960e06-6fae-48de-bd8d-1212fb26c24b 10Gi RW0
Delete Bound occne-cndbtier/pvc-ndbmysqld-ndbmysqld-1
```

5. Upgrade cnDBTier with the modified custom_values.yaml file:

```
$ helm upgrade mysql-cluster occndbtier -f occndbtier/custom_values.yaml -
n occne-cndbtier --no-hooks
```

6. Repeat steps b through e for ndbmysqld-0.
7. As cnDBTier Helm upgrade is performed with the "--no-hooks" option in step d, upgradeStrategy of the cnDBTier StatefulSets is changed to OnDelete. Therefore, perform the cnDBTier upgrade one more time to restore upgradeStrategy to RollingRestart:

Note

Before running the following command, ensure that you set the value of OCCNE_NAMESPACE variable in the command with your cnDBTier namespace.

```
helm -n ${OCCNE_NAMESPACE} upgrade ${OCCNE_RELEASE_NAME} occndbtier -f
occndbtier/custom_values.yaml
```

Updating PVC Using dbtscale_vertical_pvc

dbtscale_vertical_pvc is an automated script to update PVC without manual intervention. Perform the following steps to update the PVC value using custom_values.yaml:

1. Update the custom_values.yaml file with the new PVC values for ndbmysqld pods. Ensure that you modify the global.api.ndbdisksize section only.
For example:

The following code block shows the old PVC values in the custom_values.yaml file:

```
global:
  api:
    ndbdisksize: 3Gi
```

The following code block shows the updated PVC values in the custom_values.yaml file:

```
global:
  api:
    ndbdisksize: 4Gi
```

2. Run the following commands to navigate to the Artifacts/Scripts/tools/ directory and source the source_me file:

Note

- Ensure that the `Artifacts/Scripts/tools/` directory contains the `source_me` file.
- After running the script, enter the namespace of the cnDBTier cluster when prompted.

```
$ cd Artifacts/Scripts/tools/
$ source ./source_me
```

Sample output:

```
Enter cndbtier namespace: occne-cndbtier
DBTIER_NAMESPACE = "occne-cndbtier"
```

```
DBTIER_LIB=/home/dbtuser/occne/cluster/site-1/Artifacts/Scripts/tools/lib
```

```
Adding /home/dbtuser/occne/cluster/site-1/Artifacts/Scripts/tools/bin to
PATH
```

```
Adding /home/dbtuser/occne/cluster/site-1/Artifacts/Scripts/tools/bin/
rollbackscripts to PATH
```

```
PATH=/home/dbtuser/occne/cluster/site-1/Artifacts/Scripts/tools/bin/
rollbackscripts:/home/dbtuser/occne/cluster/site-1/Artifacts/Scripts/tools/
bin:/home/dbtuser/.local/bin:/home/dbtuser/bin:/usr/local/bin:/usr/
bin:/usr/local/sbin:/usr/sbin
```

3. Run the `dbtscale_vertical_pvc` script with the `--pods=ndbmysqld` option and pass the helm charts and the updated `custom_values.yaml` file as parameters:

```
$ dbtscale_vertical_pvc --pods=ndbmysqld occndbtier
custom_values_scale_pvc.yaml
```

Sample output:

```
2024-12-17T18:54:40Z INFO - HELM_CHARTS = occndbtier
2024-12-17T18:54:40Z INFO - CUSTOM_VALUES_FILE =
custom_values_scale_pvc.yaml
dbtscale_vertical_pvc 25.1.100
Copyright (c) 2024 Oracle and/or its affiliates. All rights reserved.
2024-12-17T18:54:40Z INFO - Started timer dbtscale_vertical_pvc: 1734461680
2024-12-17T18:54:40Z INFO - Started timer PHASE 0: 1734461680
2024-12-17T18:54:40Z INFO -
*****
*****
2024-12-17T18:54:40Z INFO - BEGIN PHASE 0: Collect Site information
2024-12-17T18:54:40Z INFO -
*****
*****
2024-12-17T18:54:40Z INFO - Using IPv4: LOOPBACK_IP="127.0.0.1"
2024-12-17T18:54:40Z INFO - DBTIER_NAMESPACE = occne-cndbtier
...
```



```

2024-12-17T18:54:52Z INFO - PODS_TO_SCALE="ndbmysqld"
2024-12-17T18:54:52Z INFO - POD_TYPE="api"
2024-12-17T18:54:52Z INFO - STS_TO_DELETE="ndbmysqld"
2024-12-17T18:54:52Z INFO - PODS_TO_RESTART="API_PODS"
2024-12-17T18:54:52Z INFO - REPL_LEADER_DEPLOY = mysql-cluster-site-1-
site-2-replication-svc
2024-12-17T18:54:52Z INFO - REPL_LEADER_PVC = pvc-site-1-site-2-
replication-svc
2024-12-17T18:54:52Z INFO - REPL_LEADER_POD = mysql-cluster-site-1-site-2-
replication-svc-58856dds8ql
2024-12-17T18:54:52Z INFO -
*****
*****
2024-12-17T18:54:52Z INFO - END PHASE 0: Collect Site information
2024-12-17T18:54:52Z INFO -
*****
*****
2024-12-17T18:54:52Z INFO - Ended timer PHASE 0: 1734461692
2024-12-17T18:54:52Z INFO - PHASE 0 took: 00 hr. 00 min. 12 sec.
2024-12-17T18:54:52Z INFO - Started timer PHASE 1: 1734461692
2024-12-17T18:54:52Z INFO -
*****
*****
2024-12-17T18:54:52Z INFO - BEGIN PHASE 1: Verify disk sizes are supported
2024-12-17T18:54:52Z INFO -
*****
*****
2024-12-17T18:54:52Z INFO - Current PVC sizes:
NAME                                STATUS
VOLUME                             CAPACITY  ACCESS MODES
STORAGECLASS  VOLUMEATTRIBUTESCLASS  AGE
pvc-ndbmysqld-ndbmysqld-0          Bound
pvc-09a977c0-8fea-47cc-b010-8ac02f870e53  3Gi      RWX
standard      <unset>                34h
pvc-ndbmysqld-ndbmysqld-1          Bound    pvc-d136cd51-
f4f0-4142-b92d-45f40d4fe65c  3Gi      RWX    standard
<unset>                34h
2024-12-17T18:54:52Z INFO - Requested PVC sizes from
custom_values_scale_pvc.yaml:
  api:
    ndbdisksize: 4Gi
2024-12-17T18:54:52Z INFO - Current and requested disk values:
2024-12-17T18:54:52Z INFO - current_pvcsz=3Gi (3221225472),
requested_pvcsz=4Gi (4294967296)
2024-12-17T18:54:52Z INFO - Verifying current and requested disk sizes...
2024-12-17T18:54:52Z INFO - Requested PVC values should not be equal to
current - PASSED
2024-12-17T18:54:52Z INFO - No requested value should be smaller than
current - PASSED
2024-12-17T18:54:52Z INFO - No requested value should be zero or negative
- PASSED
2024-12-17T18:54:52Z INFO - At least one requested value should be larger
than its current value - PASSED
2024-12-17T18:54:52Z INFO - db-replication-svc requested PVC values should
equal to current - PASSED
2024-12-17T18:54:52Z INFO - ndbappmysqld requested PVC values should equal

```

```

to current - PASSED
2024-12-17T18:54:52Z INFO - ndbmtld requested PVC values should equal to
current - PASSED
2024-12-17T18:54:52Z INFO - Verified current and requested disk sizes.
2024-12-17T18:54:52Z INFO -
*****
*****
2024-12-17T18:54:52Z INFO - END PHASE 1: Verify disk sizes are supported
2024-12-17T18:54:52Z INFO -
*****
*****
2024-12-17T18:54:52Z INFO - Ended timer PHASE 1: 1734461692
2024-12-17T18:54:52Z INFO - PHASE 1 took: 00 hr. 00 min. 00 sec.
2024-12-17T18:54:52Z INFO - Started timer PHASE 2: 1734461692
2024-12-17T18:54:52Z INFO -
*****
*****
2024-12-17T18:54:52Z INFO - BEGIN PHASE 2: Delete/scale down statefulset/
deployment
2024-12-17T18:54:52Z INFO -
*****
*****
2024-12-17T18:54:52Z INFO - Deleting STS...
2024-12-17T18:54:52Z INFO - kubectl -n occne-cndbtier delete sts --
cascade=orphan "ndbmysqld"
statefulset.apps "ndbmysqld" deleted
2024-12-17T18:54:52Z INFO - STS deleted
2024-12-17T18:54:52Z INFO -
*****
*****
2024-12-17T18:54:52Z INFO - END PHASE 2: Delete/scale down statefulset/
deployment
2024-12-17T18:54:52Z INFO -
*****
*****
2024-12-17T18:54:52Z INFO - Ended timer PHASE 2: 1734461692
2024-12-17T18:54:52Z INFO - PHASE 2 took: 00 hr. 00 min. 00 sec.
2024-12-17T18:54:52Z INFO - Started timer PHASE 3: 1734461692
2024-12-17T18:54:52Z INFO -
*****
*****
2024-12-17T18:54:52Z INFO - BEGIN PHASE 3: Upgrade cnDBTier --no-hooks or
patch db-replication-svc leader PVC
2024-12-17T18:54:52Z INFO -
*****
*****
2024-12-17T18:54:52Z INFO - Upgrading cnDBTier
(custom_values_scale_pvc.yaml)...
Release "mysql-cluster" has been upgraded. Happy Helming!
NAME: mysql-cluster
LAST DEPLOYED: Tue Dec 17 18:54:53 2024
NAMESPACE: occne-cndbtier
STATUS: deployed
REVISION: 14
2024-12-17T18:54:57Z INFO - Helm upgrade returned
2024-12-17T18:54:57Z INFO -

```

```

*****
*****
2024-12-17T18:54:57Z INFO - END PHASE 3: Upgrade cnDBTier --no-hooks or
patch db-replication-svc leader PVC
2024-12-17T18:54:57Z INFO -
*****
*****
2024-12-17T18:54:57Z INFO - Ended timer PHASE 3: 1734461697
2024-12-17T18:54:57Z INFO - PHASE 3 took: 00 hr. 00 min. 05 sec.
2024-12-17T18:54:57Z INFO - Started timer PHASE 4: 1734461697
2024-12-17T18:54:57Z INFO -
*****
*****
2024-12-17T18:54:57Z INFO - BEGIN PHASE 4: Delete PVCs and restart pods or
wait for repl PV with new site to bound
2024-12-17T18:54:57Z INFO -
*****
*****
2024-12-17T18:54:57Z INFO - Restart pods after deleting their PVCs...
2024-12-17T18:54:57Z INFO - pods_to_delete = (ndbmysqld-0 ndbmysqld-1)
2024-12-17T18:54:57Z INFO - Deleting PVCs for ndbmysqld-0 (pvc-ndbmysqld-
ndbmysqld-0)...
2024-12-17T18:54:57Z INFO - deleting pod ndbmysqld-0 ...
persistentvolumeclaim "pvc-ndbmysqld-ndbmysqld-0" deleted
pod "ndbmysqld-0" deleted
2024-12-17T18:55:08Z INFO - Waiting for condition: is_pod ndbmysqld-0...
2024-12-17T18:55:08Z INFO - Condition occurred
2024-12-17T18:55:49Z INFO - Waiting for condition: is_pod_ready
ndbmysqld-0...
2024-12-17T18:55:49Z INFO - Condition occurred
2024-12-17T18:55:49Z INFO - Deleting PVCs for ndbmysqld-1 (pvc-ndbmysqld-
ndbmysqld-1)...
2024-12-17T18:55:49Z INFO - deleting pod ndbmysqld-1 ...
persistentvolumeclaim "pvc-ndbmysqld-ndbmysqld-1" deleted
pod "ndbmysqld-1" deleted
2024-12-17T18:56:00Z INFO - Waiting for condition: is_pod ndbmysqld-1...
2024-12-17T18:56:00Z INFO - Condition occurred
2024-12-17T18:56:40Z INFO - Waiting for condition: is_pod_ready
ndbmysqld-1...
2024-12-17T18:56:40Z INFO - Condition occurred
2024-12-17T18:56:40Z INFO - kubectl -n occne-cndbtier get pod ndbmysqld-0
ndbmysqld-1
NAME          READY   STATUS    RESTARTS   AGE
ndbmysqld-0   3/3     Running   0           92s
ndbmysqld-1   3/3     Running   0           41s
2024-12-17T18:56:40Z INFO - kubectl -n occne-cndbtier get pvc pvc-
ndbmysqld-ndbmysqld-0 pvc-ndbmysqld-ndbmysqld-1
NAME          STATUS
VOLUME
STORAGECLASS  VOLUMEATTRIBUTESCLASS  AGE
pvc-ndbmysqld-ndbmysqld-0   Bound
pvc-84025c03-7f43-4c9a-8042-1c241ba365b5   4Gi          RW0
standard        <unset>          92s
pvc-ndbmysqld-ndbmysqld-1   Bound   pvc-7f626556-8516-46fb-ac1c-
b0196b02fecf   4Gi          RW0          standard
<unset>          41s

```

```

2024-12-17T18:56:40Z INFO - Pods restarted and their PVCs recreated.
2024-12-17T18:56:40Z INFO -
*****
*****
2024-12-17T18:56:40Z INFO - END PHASE 4: Delete PVCs and restart pods or
wait for repl PV with new site to bound
2024-12-17T18:56:40Z INFO -
*****
*****
2024-12-17T18:56:40Z INFO - Ended timer PHASE 4: 1734461800
2024-12-17T18:56:40Z INFO - PHASE 4 took: 00 hr. 01 min. 43 sec.
2024-12-17T18:56:40Z INFO - Started timer PHASE 5: 1734461800
2024-12-17T18:56:40Z INFO -
*****
*****
2024-12-17T18:56:40Z INFO - BEGIN PHASE 5: Upgrade cnDBTier
2024-12-17T18:56:40Z INFO -
*****
*****
2024-12-17T18:56:40Z INFO - Upgrading cnDBTier
(custom_values_scale_pvc.yaml)...
Release "mysql-cluster" has been upgraded. Happy Helming!
NAME: mysql-cluster
LAST DEPLOYED: Tue Dec 17 18:56:41 2024
NAMESPACE: occne-cndbtier
STATUS: deployed
REVISION: 15
2024-12-17T18:59:02Z INFO - Helm upgrade returned
2024-12-17T18:59:02Z INFO -
*****
*****
2024-12-17T18:59:02Z INFO - END PHASE 5: Upgrade cnDBTier
2024-12-17T18:59:02Z INFO -
*****
*****
2024-12-17T18:59:02Z INFO - Ended timer PHASE 5: 1734461942
2024-12-17T18:59:02Z INFO - PHASE 5 took: 00 hr. 02 min. 22 sec.
2024-12-17T18:59:02Z INFO - Started timer PHASE 6: 1734461942
2024-12-17T18:59:02Z INFO -
*****
*****
2024-12-17T18:59:02Z INFO - BEGIN PHASE 6: Post-processing
2024-12-17T18:59:02Z INFO -
*****
*****
2024-12-17T18:59:02Z INFO - CURRENT_PVCS:
NAME                                STATUS
VOLUME                             CAPACITY  ACCESS MODES
STORAGECLASS  VOLUMEATTRIBUTESCLASS  AGE
pvc-ndbmysql-ndbmysql-0            Bound
pvc-09a977c0-8fea-47cc-b010-8ac02f870e53  3Gi      RW0
standard      <unset>                34h
pvc-ndbmysql-ndbmysql-1            Bound
f4f0-4142-b92d-45f40d4fe65c  3Gi      RW0
<unset>                34h
2024-12-17T18:59:02Z INFO - after_pvcs:

```

```

NAME                                     STATUS
VOLUME                                CAPACITY  ACCESS MODES
STORAGECLASS  VOLUMEATTRIBUTESCLASS  AGE
pvc-ndbmysqld-ndbmysqld-0              Bound
pvc-84025c03-7f43-4c9a-8042-1c241ba365b5  4Gi        RW0
standard      <unset>                3m54s
pvc-ndbmysqld-ndbmysqld-1              Bound
pvc-7f626556-8516-46fb-ac1c-b0196b02fecf  4Gi        RW0
standard      <unset>                3m3s
2024-12-17T18:59:02Z INFO -
*****
*****
2024-12-17T18:59:02Z INFO - END PHASE 6: Post-processing
2024-12-17T18:59:02Z INFO -
*****
*****
2024-12-17T18:59:02Z INFO - Ended timer PHASE 6: 1734461942
2024-12-17T18:59:02Z INFO - PHASE 6 took: 00 hr. 00 min. 00 sec.
2024-12-17T18:59:02Z INFO - Ended timer dbtscale_vertical_pvc: 1734461942
2024-12-17T18:59:02Z INFO - dbtscale_vertical_pvc took: 00 hr. 04 min. 22
sec.
2024-12-17T18:59:02Z INFO - dbtscale_vertical_pvc completed successfully

```

7.3.2.4 Vertical Scaling of db-replication-svc Pods

This section provides the procedures to vertically scale up db-replication-svc pods.

Note

- Before scaling the pods, ensure that the worker nodes have adequate resources to support scaling.
- Before you proceed with the vertical scaling of db-replication-svc, perform a Helm test and ensure that all the cnDBTier services are running smoothly.

Updating CPU and RAM

Perform the following steps to update CPU and RAM using the `custom_values.yaml` file:

1. Configure the required CPU, and memory values in the `custom_values.yaml` file. Following is an example configuration for the leader db-replication-svc (that is, the one with PVC attached):

```

db-replication-svc:
  grrecoveryresources:
    limits:
      cpu: 2
      memory: 12Gi
      ephemeral-storage: 1Gi
    requests:
      cpu: 2
      memory: 12Gi
      ephemeral-storage: 90Mi

```

Following is an example configuration for the non-leader db-replication-svc (that is, the one without PVC attached):

```
db-replication-svc:
  resources:
    limits:
      cpu: 1
      memory: 2048Mi
      ephemeral-storage: 1Gi
    requests:
      cpu: 0.6
      memory: 1024Mi
      ephemeral-storage: 90Mi
```

2. Upgrade cnDBTier by performing a Helm upgrade with the modified `custom_values.yaml` file. For more information about the upgrade procedure, see *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide*.

Note

cnDBTier supports upgrade in a TLS enabled cluster in this scenario only.

Updating PVC

Updating PVC Using Helm Upgrade

Perform the following steps to update the PVC value using `custom_values.yaml`:

1. Perform the following steps to scale down the replication service pod:
 - a. Identify the replication service of cnDBTier cluster1 with respect to cnDBTier cluster2:

```
$ kubectl get deployment --namespace=cluster1
```

Sample output:

NAME	READY	UP-TO-DATE
AVAILABLE AGE		
mysql-cluster-cluster1-cluster2-replication-svc	1/1	1
1 18m		
mysql-cluster-db-backup-manager-svc	1/1	1
1 18m		
mysql-cluster-db-monitor-svc	1/1	1
1 18m		

- b. Scale down the replication service of cnDBTier cluster1 with respect to cnDBTier cluster2:

```
$ kubectl scale deployment mysql-cluster-cluster1-cluster2-replication-svc --namespace=cluster1 --replicas=0
```

Sample output:

```
deployment.apps/mysql-cluster-cluster1-cluster2-replication-svc scaled
```

2. Update the `custom_values.yaml` file with the new PVC values for `ndbmysqld` pods:
The following example shows the PVC value in the `custom_values.yaml` file updated from 8Gi to 10Gi:

Old PVC value:

```
db-replication-svc:
  dbreplsvcdeployments:
    - name: cluster1-cluster2-replication-svc
      enabled: true
      pvc:
        name: pvc-cluster1-cluster2-replication-svc
        disksize: 8Gi
```

New PVC value:

```
db-replication-svc:
  dbreplsvcdeployments:
    - name: cluster1-cluster2-replication-svc
      enabled: true
      pvc:
        name: pvc-cluster1-cluster2-replication-svc
        disksize: 10Gi
```

Note

If you are providing a pod prefix for the DB replication service name, then use a name that is unique and small.

3. Patch PVCs of the DB replication service with the new PVC value:

```
$ kubectl -n cluster1 patch -p '{ "spec": { "resources": { "requests": { "storage": "10Gi" }}}}' pvc pvc-cluster1-cluster2-replication-svc
```

Sample output:

```
persistentvolumeclaim/pvc-cluster1-cluster2-replication-svc patched
```

4. Wait until new PV returns and gets attached to the existing PVC. Run the following commands to check the events:

```
$ kubectl get events -n cluster1
$ kubectl get pv | grep cluster1 | grep repl
```

Sample output:

```
pvc-5b9917ff-53d4-44cb-a5ef-3e37b201c376    8Gi          RW0
Delete                                     Bound        cluster2/pvc-cluster2-cluster1-replication-svc
occne-dbtier-sc                           12m
pvc-e6526259-d007-4868-a4e2-d53a8569edc8    10Gi         RW0
Delete                                     Bound        cluster1/pvc-cluster1-cluster2-replication-svc
occne-dbtier-sc                           15m
```

5. Upgrade cnDBTier with the modified `custom_values.yaml` file:

```
$ helm upgrade mysql-cluster occndbtier -f occndbtier/custom_values.yaml -n cluster1
```

When the helm upgrade is complete, the `db-replication-svc` pod comes up with the extended PVC size.

Updating PVC Using `dbtscale_vertical_pvc`

`dbtscale_vertical_pvc` is an automated script to update PVC without manual intervention. Perform the following steps to update the PVC value using `custom_values.yaml`:

1. Update the `custom_values.yaml` file with the new PVC values for the replication service pods. Ensure that you modify the `.db-replication-svc.dbreplsvcdeployments[0].pvc.disksize` section only. For example:

The following code block shows the old PVC values in the `custom_values.yaml` file:

```
db-replication-svc:
  dbreplsvcdeployments:
    # if pod prefix is given then use the unique smaller name for this db
    replication service.
    - name: site-1-site-2-replication-svc
      enabled: true
      pvc:
        name: pvc-site-1-site-2-replication-svc
        disksize: 3Gi
```

The following code block shows the updated PVC values in the `custom_values.yaml` file:

```
db-replication-svc:
  dbreplsvcdeployments:
    # if pod prefix is given then use the unique smaller name for this db
    replication service.
    - name: site-1-site-2-replication-svc
      enabled: true
      pvc:
        name: pvc-site-1-site-2-replication-svc
        disksize: 4Gi
```

2. Run the following commands to navigate to the `Artifacts/Scripts/tools/` directory and source the `source_me` file:

Note

- Ensure that the `Artifacts/Scripts/tools/` directory contains the `source_me` file.
- After running the script, enter the namespace of the cnDBTier cluster when prompted.

```
$ cd Artifacts/Scripts/tools/
$ source ./source_me
```

Sample output:

```
Enter cndbtier namespace: occne-cndbtier
DBTIER_NAMESPACE = "occne-cndbtier"
```

```
DBTIER_LIB=/home/dbtuser/occne/cluster/site-1/Artifacts/Scripts/tools/lib
```

```
Adding /home/dbtuser/occne/cluster/site-1/Artifacts/Scripts/tools/bin to
PATH
```

```
Adding /home/dbtuser/occne/cluster/site-1/Artifacts/Scripts/tools/bin/
rollbackscripts to PATH
```

```
PATH=/home/dbtuser/occne/cluster/site-1/Artifacts/Scripts/tools/bin/
rollbackscripts:/home/dbtuser/occne/cluster/site-1/Artifacts/Scripts/tools/
bin:/home/dbtuser/.local/bin:/home/dbtuser/bin:/usr/local/bin:/usr/
bin:/usr/local/sbin:/usr/sbin
```

3. Run the `dbtscale_vertical_pvc` script with the `-pods=db-replication-svc` option and pass the helm charts and the updated `custom_values.yaml` file as parameters:

```
$ dbtscale_vertical_pvc --pods=db-replication-svc occndbtier
custom_values_scale_pvc.yaml
```

Sample output:

```
2024-12-17T19:07:02Z INFO - HELM_CHARTS = occndbtier
2024-12-17T19:07:02Z INFO - CUSTOM_VALUES_FILE =
custom_values_scale_pvc.yaml
dbtscale_vertical_pvc <24.3.0>
Copyright (c) 2024 Oracle and/or its affiliates. All rights reserved.
2024-12-17T19:07:02Z INFO - Started timer dbtscale_vertical_pvc: 1734462422
2024-12-17T19:07:02Z INFO - Started timer PHASE 0: 1734462422
2024-12-17T19:07:02Z INFO -
*****
*****
2024-12-17T19:07:02Z INFO - BEGIN PHASE 0: Collect Site information
2024-12-17T19:07:02Z INFO -
*****
*****
2024-12-17T19:07:03Z INFO - Using IPv4: LOOPBACK_IP="127.0.0.1"
2024-12-17T19:07:03Z INFO - DBTIER_NAMESPACE = occne-cndbtier
...
```

```

2024-12-17T19:07:15Z INFO - PODS_TO_SCALE="db-replication-svc"
2024-12-17T19:07:15Z INFO - POD_TYPE="db-replication-svc"
2024-12-17T19:07:15Z INFO - REPL_LEADER_DEPLOY = mysql-cluster-site-1-
site-2-replication-svc
2024-12-17T19:07:15Z INFO - REPL_LEADER_PVC = pvc-site-1-site-2-
replication-svc
2024-12-17T19:07:15Z INFO - REPL_LEADER_POD = mysql-cluster-site-1-site-2-
replication-svc-98476bwhhlp
2024-12-17T19:07:15Z INFO -
*****
*****
2024-12-17T19:07:15Z INFO - END PHASE 0: Collect Site information
2024-12-17T19:07:15Z INFO -
*****
*****
2024-12-17T19:07:15Z INFO - Ended timer PHASE 0: 1734462435
2024-12-17T19:07:15Z INFO - PHASE 0 took: 00 hr. 00 min. 13 sec.
2024-12-17T19:07:15Z INFO - Started timer PHASE 1: 1734462435
2024-12-17T19:07:15Z INFO -
*****
*****
2024-12-17T19:07:15Z INFO - BEGIN PHASE 1: Verify disk sizes are supported
2024-12-17T19:07:15Z INFO -
*****
*****
2024-12-17T19:07:15Z INFO - Current PVC sizes:
NAME                                STATUS
VOLUME                             CAPACITY  ACCESS MODES
STORAGECLASS  VOLUMEATTRIBUTESCLASS  AGE
pvc-site-1-site-2-replication-svc  Bound    pvc-1212dd3c-5100-467d-
ac54-0c569116a507  3Gi      RWX          standard
<unset>                                37h
2024-12-17T19:07:15Z INFO - Requested PVC sizes from
custom_values_scale_pvc.yaml:
  - name: lfg-site-1-site-2-replication-svc
    enabled: true
    pvc:
      name: pvc-site-1-site-2-replication-svc
      disksize: 4Gi
2024-12-17T19:07:15Z INFO - Current and requested disk values:
2024-12-17T19:07:15Z INFO - current_pvcsz=3Gi (3221225472),
requested_pvcsz=4Gi (4294967296)
2024-12-17T19:07:15Z INFO - Verifying current and requested disk sizes...
2024-12-17T19:07:15Z INFO - Requested PVC values should not be equal to
current - PASSED
2024-12-17T19:07:15Z INFO - No requested value should be smaller than
current - PASSED
2024-12-17T19:07:15Z INFO - No requested value should be zero or negative
- PASSED
2024-12-17T19:07:15Z INFO - At least one requested value should be larger
than its current value - PASSED
2024-12-17T19:07:15Z INFO - ndbmysqld requested PVC values should equal to
current - PASSED
2024-12-17T19:07:15Z INFO - ndbappmysqld requested PVC values should equal
to current - PASSED
2024-12-17T19:07:15Z INFO - ndbmtd requested PVC values should equal to

```

```
current - PASSED
2024-12-17T19:07:15Z INFO - Verified current and requested disk sizes.
2024-12-17T19:07:15Z INFO -
*****
*****
2024-12-17T19:07:15Z INFO - END PHASE 1: Verify disk sizes are supported
2024-12-17T19:07:15Z INFO -
*****
*****
2024-12-17T19:07:15Z INFO - Ended timer PHASE 1: 1734462435
2024-12-17T19:07:15Z INFO - PHASE 1 took: 00 hr. 00 min. 00 sec.
2024-12-17T19:07:15Z INFO - Started timer PHASE 2: 1734462435
2024-12-17T19:07:15Z INFO -
*****
*****
2024-12-17T19:07:15Z INFO - BEGIN PHASE 2: Delete/scale down statefulset/
deployment
2024-12-17T19:07:15Z INFO -
*****
*****
2024-12-17T19:07:15Z INFO - Scaling down deployment...
2024-12-17T19:07:15Z INFO - REPL_LEADER_DEPLOY = mysql-cluster-site-1-
site-2-replication-svc
2024-12-17T19:07:15Z INFO - kubectl -n ocne-cndbtier scale deployment
mysql-cluster-site-1-site-2-replication-svc --replicas=0
deployment.apps/mysql-cluster-site-1-site-2-replication-svc scaled
2024-12-17T19:07:16Z INFO - Deployment scaled down
2024-12-17T19:07:16Z INFO -
*****
*****
2024-12-17T19:07:16Z INFO - END PHASE 2: Delete/scale down statefulset/
deployment
2024-12-17T19:07:16Z INFO -
*****
*****
2024-12-17T19:07:16Z INFO - Ended timer PHASE 2: 1734462436
2024-12-17T19:07:16Z INFO - PHASE 2 took: 00 hr. 00 min. 01 sec.
2024-12-17T19:07:16Z INFO - Started timer PHASE 3: 1734462436
2024-12-17T19:07:16Z INFO -
*****
*****
2024-12-17T19:07:16Z INFO - BEGIN PHASE 3: Upgrade cnDBTier --no-hooks or
patch db-replication-svc leader PVC
2024-12-17T19:07:16Z INFO -
*****
*****
2024-12-17T19:07:16Z INFO - Patch pvc-site-1-site-2-replication-svc with
the new value (4294967296).
persistentvolumeclaim/pvc-site-1-site-2-replication-svc patched
2024-12-17T19:07:16Z INFO - Replication service PVC patched.
2024-12-17T19:07:16Z INFO -
*****
*****
2024-12-17T19:07:16Z INFO - END PHASE 3: Upgrade cnDBTier --no-hooks or
patch db-replication-svc leader PVC
2024-12-17T19:07:16Z INFO -
```

```

*****
*****
2024-12-17T19:07:16Z INFO - Ended timer PHASE 3: 1734462436
2024-12-17T19:07:16Z INFO - PHASE 3 took: 00 hr. 00 min. 00 sec.
2024-12-17T19:07:16Z INFO - Started timer PHASE 4: 1734462436
2024-12-17T19:07:16Z INFO -
*****
*****
2024-12-17T19:07:16Z INFO - BEGIN PHASE 4: Delete PVCs and restart pods or
wait for repl PV with new site to bound
2024-12-17T19:07:16Z INFO -
*****
*****
2024-12-17T19:07:16Z INFO - Wait for the new PV to bound to existing pvc.
2024-12-17T19:07:19Z INFO - Waiting for condition:
is_requested_size_and_bound...
2024-12-17T19:07:19Z INFO - Condition occurred
2024-12-17T19:07:19Z INFO - PV attached to pvc-site-1-site-2-replication-
svc with requested size: 4294967296
2024-12-17T19:07:19Z INFO - PV:
NAME                                CAPACITY  ACCESS MODES
RECLAIM POLICY  STATUS
CLAIM
                                STORAGECLASS
VOLUMEATTRIBUTESCLASS  REASON  AGE
pvc-1212dd3c-5100-467d-ac54-0c569116a507  4Gi      RW0
Delete          Bound    occne-cndbtier/pvc-site-1-site-2-replication-
svc
standard        <unset>      37h
2024-12-17T19:07:19Z INFO -
*****
*****
2024-12-17T19:07:19Z INFO - END PHASE 4: Delete PVCs and restart pods or
wait for repl PV with new site to bound
2024-12-17T19:07:19Z INFO -
*****
*****
2024-12-17T19:07:19Z INFO - Ended timer PHASE 4: 1734462439
2024-12-17T19:07:19Z INFO - PHASE 4 took: 00 hr. 00 min. 03 sec.
2024-12-17T19:07:19Z INFO - Started timer PHASE 5: 1734462439
2024-12-17T19:07:19Z INFO -
*****
*****
2024-12-17T19:07:19Z INFO - BEGIN PHASE 5: Upgrade cnDBTier
2024-12-17T19:07:19Z INFO -
*****
*****
2024-12-17T19:07:19Z INFO - Upgrading cnDBTier
(custom_values_scale_pvc.yaml)...
Release "mysql-cluster" has been upgraded. Happy Helming!
NAME: mysql-cluster
LAST DEPLOYED: Tue Dec 17 19:07:20 2024
NAMESPACE: occne-cndbtier
STATUS: deployed
REVISION: 16
2024-12-17T19:09:19Z INFO - Helm upgrade returned

```

```

2024-12-17T19:09:19Z INFO -
*****
*****
2024-12-17T19:09:19Z INFO - END PHASE 5: Upgrade cnDBTier
2024-12-17T19:09:19Z INFO -
*****
*****
2024-12-17T19:09:19Z INFO - Ended timer PHASE 5: 1734462559
2024-12-17T19:09:19Z INFO - PHASE 5 took: 00 hr. 02 min. 00 sec.
2024-12-17T19:09:19Z INFO - Started timer PHASE 6: 1734462559
2024-12-17T19:09:19Z INFO -
*****
*****
2024-12-17T19:09:19Z INFO - BEGIN PHASE 6: Post-processing
2024-12-17T19:09:19Z INFO -
*****
*****
2024-12-17T19:09:19Z INFO - CURRENT_PVCS:
NAME                                STATUS
VOLUME                             CAPACITY  ACCESS MODES
STORAGECLASS  VOLUMEATTRIBUTESCLASS  AGE
pvc-site-1-site-2-replication-svc  Bound    pvc-1212dd3c-5100-467d-
ac54-0c569116a507  3Gi      RW0          standard
<unset>                                37h
2024-12-17T19:09:19Z INFO - after_pvcs:
NAME                                STATUS
VOLUME                             CAPACITY  ACCESS MODES
STORAGECLASS  VOLUMEATTRIBUTESCLASS  AGE
pvc-site-1-site-2-replication-svc  Bound    pvc-1212dd3c-5100-467d-
ac54-0c569116a507  4Gi      RW0          standard
<unset>                                37h
2024-12-17T19:09:19Z INFO -
*****
*****
2024-12-17T19:09:19Z INFO - END PHASE 6: Post-processing
2024-12-17T19:09:19Z INFO -
*****
*****
2024-12-17T19:09:19Z INFO - Ended timer PHASE 6: 1734462559
2024-12-17T19:09:19Z INFO - PHASE 6 took: 00 hr. 00 min. 00 sec.
2024-12-17T19:09:19Z INFO - Ended timer dbtscale_vertical_pvc: 1734462559
2024-12-17T19:09:19Z INFO - dbtscale_vertical_pvc took: 00 hr. 02 min. 17
sec.
2024-12-17T19:09:19Z INFO - dbtscale_vertical_pvc completed successfully

```

7.3.2.5 Scaling of ndbmgmd Pods

This section provides the procedures to vertically scale up ndbmgmd pods.

Note

- Before scaling the pods, ensure that the worker nodes have adequate resources to support scaling.
- Before you proceed with the vertical scaling procedure for ndbmcmd, perform a Helm Test and ensure that all the cnDBTier services are running smoothly.

Updating CPU and RAM

Perform the following steps to update CPU and RAM using the `custom_values.yaml` file:

1. Configure the required CPU and memory values in the `global` section of the `custom_values.yaml` file.

For example:

```
mgm:
  ...
  resources:
    limits:
      cpu: 6
      memory: 12Gi
      ephemeral-storage: 1Gi
    requests:
      cpu: 6
      memory: 12Gi
      ephemeral-storage: 90Mi
```

2. Upgrade cnDBTier by performing a Helm upgrade with the modified `custom_values.yaml` file. For more information about the upgrade procedure, see *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide*.

Updating PVC**Updating PVC Using Helm Upgrade**

Perform the following steps to update the PVC value using `custom_values.yaml`:

1. Update the `custom_values.yaml` file with the new PVC values for ndbmcmd pods:

```
global:
  mgm:
    ndbdisksize: 10Gi
```

2. Delete the ndbmcmd statefulset and make the dependencies as orphan:

```
$ kubectl -n occne-cndbtier delete sts --cascade=orphan ndbmcmd
```

3. Delete the ndbmcmd-1 pod and patch its PVCs with the new PVC value:

- a. Delete the ndbmcmd-1 pod:

```
$ kubectl -n occne-cndbtier delete pod ndbmcmd-1
```

Sample output:

```
pod "ndbmcmd-1" deleted
```

- b. Patch the PVCs of ndbmcmd-1 pod with the new PVC values:

```
$ kubectl -n occne-cndbtier patch -p '{ "spec": { "resources":  
  { "requests": { "storage": "10Gi" }}}}' pvc pvc-ndbmcmd-ndbmcmd-1
```

4. Wait for the new PV to attach with the PVC. Run the following command to check if the PV reflects the updated PVC values:

```
$ kubectl get pv | grep -w occne-cndbtier | grep ndbmcmd-1
```

Sample output:

```
pvc-8a58a7b4-25bb-4b32-a07a-728h8d792835    10Gi          RWX  
Delete              Bound          occne-cndbtier/pvc-ndbmcmd-ndbmcmd-1
```

5. Upgrade cnDBTier with the modified custom_values.yaml file:

```
$ helm upgrade mysql-cluster occndbtier -f occndbtier/custom_values.yaml -  
n occne-cndbtier --no-hooks
```

6. Repeat steps b through e for ndbmcmd-0.
7. As cnDBTier Helm upgrade is performed with the "--no-hooks" option in step e, upgradeStrategy of the cnDBTier StatefulSets is changed to OnDelete. Therefore, perform the cnDBTier upgrade one more time to restore upgradeStrategy to RollingRestart:

```
helm -n ${OCCNE_NAMESPACE} upgrade ${OCCNE_RELEASE_NAME} occndbtier -f  
occndbtier/custom_values.yaml
```

Updating PVC Using dbtscale_vertical_pvc

dbtscale_vertical_pvc is an automated script to update PVC without manual intervention. Perform the following steps to update the PVC value using custom_values.yaml:

1. Update the custom_values.yaml file with the new PVC values for the ndbmcmd pods. Ensure that you modify the global.mgm.ndbdisksize section only. For example:

The following code block shows the old PVC values in the custom_values.yaml file:

```
global:  
  mgm:  
    ndbdisksize: 2Gi
```

The following code block shows the updated PVC values in the custom_values.yaml file:

```
global:  
  mgm:  
    ndbdisksize: 3Gi
```

2. Run the following commands to navigate to the Artifacts/Scripts/tools/ directory and source the source_me file:

Note

- Ensure that the `Artifacts/Scripts/tools/` directory contains the `source_me` file.
- After running the script, enter the namespace of the cnDBTier cluster when prompted.

```
$ cd Artifacts/Scripts/tools/
$ source ./source_me
```

Sample output:

```
Enter cndbtier namespace: occne-cndbtier
DBTIER_NAMESPACE = "occne-cndbtier"
```

```
DBTIER_LIB=/home/dbtuser/occne/cluster/site-1/Artifacts/Scripts/tools/lib
```

```
Adding /home/dbtuser/occne/cluster/site-1/Artifacts/Scripts/tools/bin to
PATH
```

```
Adding /home/dbtuser/occne/cluster/site-1/Artifacts/Scripts/tools/bin/
rollbackscripts to PATH
```

```
PATH=/home/dbtuser/occne/cluster/site-1/Artifacts/Scripts/tools/bin/
rollbackscripts:/home/dbtuser/occne/cluster/site-1/Artifacts/Scripts/tools/
bin:/home/dbtuser/.local/bin:/home/dbtuser/bin:/usr/local/bin:/usr/
bin:/usr/local/sbin:/usr/sbin
```

3. Run the `dbtscale_vertical_pvc` script with the `--pods=ndbmcmd` option and pass the helm charts and the updated `custom_values.yaml` file as parameters:

```
$ dbtscale_vertical_pvc --pods=ndbmcmd occndbtier
custom_values_scale_pvc.yaml
```

Sample output:

```
2024-12-17T22:07:40Z INFO - HELM_CHARTS = occndbtier
2024-12-17T22:07:40Z INFO - CUSTOM_VALUES_FILE =
custom_values_scale_pvc.yaml
dbtscale_vertical_pvc <24.3.0>
Copyright (c) 2024 Oracle and/or its affiliates. All rights reserved.
2024-12-17T22:07:40Z INFO - Started timer dbtscale_vertical_pvc: 1734473260
2024-12-17T22:07:40Z INFO - Started timer PHASE 0: 1734473260
2024-12-17T22:07:40Z INFO -
*****
*****
2024-12-17T22:07:40Z INFO - BEGIN PHASE 0: Collect Site information
2024-12-17T22:07:40Z INFO -
*****
*****
2024-12-17T22:07:40Z INFO - Using IPv4: LOOPBACK_IP="127.0.0.1"
2024-12-17T22:07:40Z INFO - DBTIER_NAMESPACE = occne-cndbtier
...
```



```

2024-12-17T22:07:51Z INFO - PODS_TO_SCALE="ndbmgmd"
2024-12-17T22:07:51Z INFO - POD_TYPE="mgm"
2024-12-17T22:07:51Z INFO - STS_TO_DELETE="ndbmgmd"
2024-12-17T22:07:51Z INFO - PODS_TO_RESTART="MGM_PODS"
2024-12-17T22:07:51Z INFO - REPL_LEADER_DEPLOY = mysql-cluster-site-1-
site-2-replication-svc
2024-12-17T22:07:51Z INFO - REPL_LEADER_PVC = pvc-site-1-site-2-
replication-svc
2024-12-17T22:07:51Z INFO - REPL_LEADER_POD = mysql-cluster-site-1-site-2-
replication-svc-6f549c46qt9
2024-12-17T22:07:51Z INFO -
*****
*****
2024-12-17T22:07:51Z INFO - END PHASE 0: Collect Site information
2024-12-17T22:07:51Z INFO -
*****
*****
2024-12-17T22:07:51Z INFO - Ended timer PHASE 0: 1734473271
2024-12-17T22:07:51Z INFO - PHASE 0 took: 00 hr. 00 min. 11 sec.
2024-12-17T22:07:51Z INFO - Started timer PHASE 1: 1734473271
2024-12-17T22:07:51Z INFO -
*****
*****
2024-12-17T22:07:51Z INFO - BEGIN PHASE 1: Verify disk sizes are supported
2024-12-17T22:07:51Z INFO -
*****
*****
2024-12-17T22:07:51Z INFO - Current PVC sizes:
NAME                                STATUS
VOLUME                             CAPACITY  ACCESS MODES
STORAGECLASS  VOLUMEATTRIBUTESCLASS  AGE
pvc-ndbmgmd-ndbmgmd-0              Bound    pvc-b718e19d-
c90a-4b5d-8d92-5c6f528d3643      2Gi      RWO          standard
<unset>                             40m
pvc-ndbmgmd-ndbmgmd-1              Bound    pvc-
f5de0e40-170e-4e34-9ee0-7166466554ff  2Gi      RWO          standard
<unset>                             40m
2024-12-17T22:07:51Z INFO - Requested PVC sizes from
custom_values_scale_pvc.yaml:
  mgm:
    ndbdisksize: 3Gi
2024-12-17T22:07:51Z INFO - Current and requested disk values:
2024-12-17T22:07:51Z INFO - current_pvcsz=2Gi (2147483648),
requested_pvcsz=3Gi (3221225472)
2024-12-17T22:07:51Z INFO - Verifying current and requested disk sizes...
2024-12-17T22:07:51Z INFO - Requested PVC values should not be equal to
current - PASSED
2024-12-17T22:07:51Z INFO - No requested value should be smaller than
current - PASSED
2024-12-17T22:07:51Z INFO - No requested value should be zero or negative
- PASSED
2024-12-17T22:07:51Z INFO - At least one requested value should be larger
than its current value - PASSED
2024-12-17T22:07:51Z INFO - db-replication-svc requested PVC values should
equal to current - PASSED
2024-12-17T22:07:52Z INFO - ndbmysqld requested PVC values should equal to

```

```
current - PASSED
2024-12-17T22:07:52Z INFO - ndbappmysqld requested PVC values should equal
to current - PASSED
2024-12-17T22:07:52Z INFO - ndbmttd requested PVC values should equal to
current - PASSED
2024-12-17T22:07:52Z INFO - Verified current and requested disk sizes.
2024-12-17T22:07:52Z INFO -
*****
*****
2024-12-17T22:07:52Z INFO - END PHASE 1: Verify disk sizes are supported
2024-12-17T22:07:52Z INFO -
*****
*****
2024-12-17T22:07:52Z INFO - Ended timer PHASE 1: 1734473272
2024-12-17T22:07:52Z INFO - PHASE 1 took: 00 hr. 00 min. 01 sec.
2024-12-17T22:07:52Z INFO - Started timer PHASE 2: 1734473272
2024-12-17T22:07:52Z INFO -
*****
*****
2024-12-17T22:07:52Z INFO - BEGIN PHASE 2: Delete/scale down statefulset/
deployment
2024-12-17T22:07:52Z INFO -
*****
*****
2024-12-17T22:07:52Z INFO - Deleting STS...
2024-12-17T22:07:52Z INFO - kubectl -n occne-cndbtier delete sts --
cascade=orphan "ndbmgmd"
statefulset.apps "ndbmgmd" deleted
2024-12-17T22:07:52Z INFO - STS deleted
2024-12-17T22:07:52Z INFO -
*****
*****
2024-12-17T22:07:52Z INFO - END PHASE 2: Delete/scale down statefulset/
deployment
2024-12-17T22:07:52Z INFO -
*****
*****
2024-12-17T22:07:52Z INFO - Ended timer PHASE 2: 1734473272
2024-12-17T22:07:52Z INFO - PHASE 2 took: 00 hr. 00 min. 00 sec.
2024-12-17T22:07:52Z INFO - Started timer PHASE 3: 1734473272
2024-12-17T22:07:52Z INFO -
*****
*****
2024-12-17T22:07:52Z INFO - BEGIN PHASE 3: Upgrade cnDBTier --no-hooks or
patch db-replication-svc leader PVC
2024-12-17T22:07:52Z INFO -
*****
*****
2024-12-17T22:07:52Z INFO - Upgrading cnDBTier
(custom_values_scale_pvc.yaml)...
Release "mysql-cluster" has been upgraded. Happy Helming!
NAME: mysql-cluster
LAST DEPLOYED: Tue Dec 17 22:07:52 2024
NAMESPACE: occne-cndbtier
STATUS: deployed
REVISION: 15
```

```

2024-12-17T22:07:55Z INFO - Helm upgrade returned
2024-12-17T22:07:55Z INFO -
*****
*****
2024-12-17T22:07:56Z INFO - END PHASE 3: Upgrade cnDBTier --no-hooks or
patch db-replication-svc leader PVC
2024-12-17T22:07:56Z INFO -
*****
*****
2024-12-17T22:07:56Z INFO - Ended timer PHASE 3: 1734473276
2024-12-17T22:07:56Z INFO - PHASE 3 took: 00 hr. 00 min. 04 sec.
2024-12-17T22:07:56Z INFO - Started timer PHASE 4: 1734473276
2024-12-17T22:07:56Z INFO -
*****
*****
2024-12-17T22:07:56Z INFO - BEGIN PHASE 4: Delete PVCs and restart pods or
wait for repl PV with new site to bound
2024-12-17T22:07:56Z INFO -
*****
*****
2024-12-17T22:07:56Z INFO - Restart pods after deleting their PVCs...
2024-12-17T22:07:56Z INFO - pods_to_delete = (ndbmcmd-0 ndbmcmd-1)
2024-12-17T22:07:56Z INFO - Deleting PVCs for ndbmcmd-0 (pvc-ndbmcmd-
ndbmcmd-0)...
2024-12-17T22:07:56Z INFO - deleting pod ndbmcmd-0 ...
persistentvolumeclaim "pvc-ndbmcmd-ndbmcmd-0" deleted
pod "ndbmcmd-0" deleted
2024-12-17T22:08:07Z INFO - Waiting for condition: is_pod ndbmcmd-0...
2024-12-17T22:08:07Z INFO - Condition occurred
2024-12-17T22:08:29Z INFO - Waiting for condition: is_pod_ready
ndbmcmd-0...
2024-12-17T22:08:29Z INFO - Condition occurred
2024-12-17T22:08:29Z INFO - Deleting PVCs for ndbmcmd-1 (pvc-ndbmcmd-
ndbmcmd-1)...
2024-12-17T22:08:29Z INFO - deleting pod ndbmcmd-1 ...
persistentvolumeclaim "pvc-ndbmcmd-ndbmcmd-1" deleted
pod "ndbmcmd-1" deleted
2024-12-17T22:08:37Z INFO - Waiting for condition: is_pod ndbmcmd-1...
2024-12-17T22:08:37Z INFO - Condition occurred
2024-12-17T22:08:59Z INFO - Waiting for condition: is_pod_ready
ndbmcmd-1...
2024-12-17T22:08:59Z INFO - Condition occurred
2024-12-17T22:08:59Z INFO - kubectl -n occne-cndbtier get pod ndbmcmd-0
ndbmcmd-1
NAME          READY   STATUS    RESTARTS   AGE
ndbmcmd-0     2/2     Running   0           52s
ndbmcmd-1     2/2     Running   0           22s
2024-12-17T22:09:00Z INFO - kubectl -n occne-cndbtier get pvc pvc-ndbmcmd-
ndbmcmd-0 pvc-ndbmcmd-ndbmcmd-1
NAME          STATUS
VOLUME          CAPACITY   ACCESS MODES
STORAGECLASS  VOLUMEATTRIBUTESCLASS  AGE
pvc-ndbmcmd-ndbmcmd-0  Bound      pvc-9bb687b0-3bae-42d5-
a3f0-5cf0009cf246  3Gi        RW0          standard
<unset>         53s
pvc-ndbmcmd-ndbmcmd-1  Bound      pvc-bc40c80c-39b1-4818-a1db-

```

```

ef204ff55c1f 3Gi RW0 standard
<unset> 23s
2024-12-17T22:09:00Z INFO - Pods restarted and their PVCs recreated.
2024-12-17T22:09:00Z INFO -
*****
*****
2024-12-17T22:09:00Z INFO - END PHASE 4: Delete PVCs and restart pods or
wait for repl PV with new site to bound
2024-12-17T22:09:00Z INFO -
*****
*****
2024-12-17T22:09:00Z INFO - Ended timer PHASE 4: 1734473340
2024-12-17T22:09:00Z INFO - PHASE 4 took: 00 hr. 01 min. 04 sec.
2024-12-17T22:09:00Z INFO - Started timer PHASE 5: 1734473340
2024-12-17T22:09:00Z INFO -
*****
*****
2024-12-17T22:09:00Z INFO - BEGIN PHASE 5: Upgrade cnDBTier
2024-12-17T22:09:00Z INFO -
*****
*****
2024-12-17T22:09:00Z INFO - Upgrading cnDBTier
(custom_values_scale_pvc.yaml)...
Release "mysql-cluster" has been upgraded. Happy Helming!
NAME: mysql-cluster
LAST DEPLOYED: Tue Dec 17 22:09:00 2024
NAMESPACE: occne-cndbtier
STATUS: deployed
REVISION: 16
2024-12-17T22:11:04Z INFO - Helm upgrade returned
2024-12-17T22:11:04Z INFO -
*****
*****
2024-12-17T22:11:04Z INFO - END PHASE 5: Upgrade cnDBTier
2024-12-17T22:11:04Z INFO -
*****
*****
2024-12-17T22:11:04Z INFO - Ended timer PHASE 5: 1734473464
2024-12-17T22:11:04Z INFO - PHASE 5 took: 00 hr. 02 min. 04 sec.
2024-12-17T22:11:04Z INFO - Started timer PHASE 6: 1734473464
2024-12-17T22:11:04Z INFO -
*****
*****
2024-12-17T22:11:04Z INFO - BEGIN PHASE 6: Post-processing
2024-12-17T22:11:04Z INFO -
*****
*****
2024-12-17T22:11:04Z INFO - CURRENT_PVCS:
NAME STATUS
VOLUME CAPACITY ACCESS MODES
STORAGECLASS VOLUMEATTRIBUTESCLASS AGE
pvc-ndbmcmd-ndbmcmd-0 Bound pvc-b718e19d-
c90a-4b5d-8d92-5c6f528d3643 2Gi RW0 standard
<unset> 40m
pvc-ndbmcmd-ndbmcmd-1 Bound pvc-
f5de0e40-170e-4e34-9ee0-7166466554ff 2Gi RW0

```

```

standard          <unset>          40m
2024-12-17T22:11:04Z INFO - after_pvc:
NAME                                STATUS
VOLUME                             CAPACITY  ACCESS MODES
STORAGECLASS  VOLUMEATTRIBUTESCLASS  AGE
pvc-ndbmgmd-ndbmgmd-0              Bound
pvc-9bb687b0-3bae-42d5-a3f0-5cf0009cf246  3Gi      RW0
standard          <unset>          2m57s
pvc-ndbmgmd-ndbmgmd-1              Bound    pvc-
bc40c80c-39b1-4818-a1db-ef204ff55c1f  3Gi      RW0
standard          <unset>          2m27s
2024-12-17T22:11:04Z INFO -
*****
*****
2024-12-17T22:11:04Z INFO - END PHASE 6: Post-processing
2024-12-17T22:11:04Z INFO -
*****
*****
2024-12-17T22:11:04Z INFO - Ended timer PHASE 6: 1734473464
2024-12-17T22:11:04Z INFO - PHASE 6 took: 00 hr. 00 min. 00 sec.
2024-12-17T22:11:04Z INFO - Ended timer dbtscale_vertical_pvc: 1734473464
2024-12-17T22:11:04Z INFO - dbtscale_vertical_pvc took: 00 hr. 03 min. 24
sec.
2024-12-17T22:11:04Z INFO - dbtscale_vertical_pvc completed successfully

```

7.4 Managing TLS

This section provides the procedures to enable TLS and modify cnDBTier certificates that are used to establish an encrypted connection for georeplication and for communication with NFs.

7.4.1 Managing TLS for Georeplication

This section provides the procedures to enable TLS for georeplication and modify cnDBTier certificates that are used to establish an encrypted connection between georeplication sites.

7.4.1.1 Enabling TLS for Georeplication

Perform the following steps to enable TLS for georeplication between cnDBTier sites.

1. Create all the required secrets using the "Creating Secret for TLS Certificates" procedure in *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide*.
2. Enable TLS feature in the `custom_values.yaml` file:

```

global:
  tls:
    enable: true

```

3. Provide all the required certificates (such as, ca certificate, client certificate, and server certificate) for the respective ndbmysqld pods in the `custom_values.yaml` file:

```

tls:
  enable: true
  caCertificate: "<ca certificate file name>"

```

```

tlsversion: "TLSv1.3"
tlsMode: "<TLS mode>"
ciphers:
  - TLS_AES_128_GCM_SHA256
  - TLS_AES_256_GCM_SHA384
  - TLS_CHACHA20_POLY1305_SHA256
  - TLS_AES_128_CCM_SHA256
certificates:
  - name: ndbmysqld-0
    serverCertificate: "<server certificate name>"
    serverCertificateKey: "<server key name>"
    clientCertificate: "<client certificate name>"
    clientCertificateKey: "<client key name>"
  - name: ndbmysqld-1
    serverCertificate: "<server certificate name>"
    serverCertificateKey: "<server key name>"
    clientCertificate: "<client certificate name>"
    clientCertificateKey: "<client key name>"

```

For example:

```

tls:
  enable: true
  caCertificate: "combine-ca.pem"
  tlsversion: "TLSv1.3"
  tlsMode: "VERIFY_CA"
  ciphers:
    - TLS_AES_128_GCM_SHA256
    - TLS_AES_256_GCM_SHA384
    - TLS_CHACHA20_POLY1305_SHA256
    - TLS_AES_128_CCM_SHA256
  certificates:
    - name: ndbmysqld-0
      serverCertificate: "server1-cert.pem"
      serverCertificateKey: "server1-key.pem"
      clientCertificate: "client1-cert.pem"
      clientCertificateKey: "client1-key.pem"
    - name: ndbmysqld-1
      serverCertificate: "server1-cert.pem"
      serverCertificateKey: "server1-key.pem"
      clientCertificate: "client1-cert.pem"
      clientCertificateKey: "client1-key.pem"

```

4. Follow the installation procedure in *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide* to perform a Helm install with the updated `custom_values.yaml` file.

7.4.1.2 Modifying cnDBTier Certificates to Establish TLS Between Georeplication Sites

This section provides the procedure to modify cnDBTier certificates to establish an encrypted connection between georeplication sites using TLS.

Note

- If you update cnDBTier certificates in one site, ensure that you update the relevant Certificate Authority (CA) certificates in all the mate sites. This ensures that the new or switchover replication doesn't break due to incorrect certificates.
- While recreating the secrets, ensure that the file names of the updated certificates are same as the old certificates.

1. Get the list of existing CA certificates of the cnDBTier cluster.

Note

If the file name has a dot either in the extension or in its name, use an escape character \ before the dot.

```
$ kubectl get secret cndbtier-trust-store-secret -n
<cndbtier_namespace> -o jsonpath="{.data.<ca_certificate_file_name>}" |
base64 --decode
```

For example:

```
$ kubectl get secret cndbtier-trust-store-secret -n occne-cndbtier -o
jsonpath="{.data.ca\\.pem}" | base64 --decode
```

2. If you want to modify the secret of the existing CA certificates in the cnDBTier cluster, update the secret using the following command.

```
$ TLS_CA_CERT=$(base64 < "<updated-certificate-name>" | tr -d '\n')
$ kubectl -n <namespace> patch secret cndbtier-trust-store-secret -p
"{\"data\":{\"<ca_certificate>\":\"${TLS_CA_CERT}\"}}"
```

where, <namespace> is the name of the cnDBTier namespace.

For example:

```
$ TLS_CA_CERT=$(base64 < "combine-ca.pem" | tr -d '\n')
$ kubectl -n occne-cndbtier patch secret cndbtier-trust-store-secret -p
"{\"data\":{\"combine-ca.pem\":\"${TLS_CA_CERT}\"}}"
```

3. Perform the following steps to get the existing server certificate and the server key of the cnDBTier cluster.

Note

If the file names have dots either in the extension or in its name, use an escape character \ before the dots.

- a. Run the following command to get the server certificate:

```
$ kubectl get secret cndbtier-server-secret -n <cndbtier_namespace> -
o jsonpath="{.data.<server_certificate_file_name>}" | base64 --decode
```

For example:

```
$ kubectl get secret cndbtier-server-secret -n occne-cndbtier -o
jsonpath="{.data.server-cert.pem}" | base64 --decode
```

- b. Run the following command to get the server certificate key:

```
$ kubectl get secret cndbtier-server-secret -n <cndbtier_namespace> -
o jsonpath="{.data.<server_certificate_key_file_name>}" | base64 --
decode
```

For example:

```
$ kubectl get secret cndbtier-server-secret -n occne-cndbtier -o
jsonpath="{.data.server-key.pem}" | base64 --decode
```

4. If you want to modify the secret of the existing server certificate and server key secret, update the secret using the following command. Otherwise, skip this step and move to Step 7.

```
$ kubectl -n <cndbtier_namespace> delete secret cndbtier-server-secret
```

For example:

```
$ kubectl -n occne-cndbtier delete secret cndbtier-server-secret
```

5. If you deleted the secret of the existing server certificate and server key in the previous step, then create the cndbtier-server-secret secret with the new server certificate and server key. For more information, see the "*Creating Secrets for TLS Certificates*" section in *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide*.

```
$ kubectl -n ${OCCNE_NAMESPACE} create secret generic cndbtier-server-
secret --from-file=<server_certificate> --from-
file=<server_certificate_key>
```

For example:

```
$ kubectl -n ${OCCNE_NAMESPACE} create secret generic cndbtier-server-
secret --from-file=server-cert.pem --from-file=server-key.pem
```

6. [Optional] Perform the following steps to get the existing client certificate and client key of the cnDBTier cluster. You can skip this step if you know the list of certificates.

Note

If the file names have dots either in the extension or in its name, use an escape character \ before the dots.

- a. Run the following command to get the client certificate:

```
$ kubectl get secret cndbtier-client-secret -n <cndbtier_namespace> -o jsonpath="{.data.<client_certificate_file_name>}" | base64 --decode
```

For example:

```
$ kubectl get secret cndbtier-client-secret -n occne-cndbtier -o jsonpath="{.data.client-cert\.pem}" | base64 --decode
```

- b. Run the following command to get the client certificate key:

```
$ kubectl get secret cndbtier-client-secret -n <cndbtier_namespace> -o jsonpath="{.data.<client_certificate_key_file_name>}" | base64 --decode
```

For example:

```
$ kubectl get secret cndbtier-client-secret -n occne-cndbtier -o jsonpath="{.data.client-key\.pem}" | base64 --decode
```

7. If you want to modify the secret of the existing client certificate and client key secret, delete the secret using the following command. Otherwise, skip this step and move to Step 10.

```
$ kubectl -n <cndbtier_namespace> delete secret cndbtier-client-secret
```

For example:

```
$ kubectl -n occne-cndbtier delete secret cndbtier-client-secret
```

8. If you deleted the secret of the existing client certificate and client key in the previous step, then create the cndbtier-client-secret secret with the new client certificate and client key. For more information, see the "*Creating Secrets for TLS Certificates*" section in *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide*.

```
$ kubectl -n ${OCCNE_NAMESPACE} create secret generic cndbtier-client-secret --from-file=<client_certificate> --from-file=<client_certificate_key>
```

For example:

```
$ kubectl -n ${OCCNE_NAMESPACE} create secret generic cndbtier-client-secret --from-file=client-cert.pem --from-file=client-key.pem
```

9. If you modified the CA certificates in the current site, then you must update the certificates in other sites too. Perform Steps 1 through 9 to modify the relevant certificates on all the mate sites.
10. Wait for the new certificates of the recreated secret to be automatically mounted to the replication SQL pods by the Kubernetes API server.

Note

This could take around 30 seconds. However, the wait time depends on the Kubernetes configuration and varies from environment to environment.

11. Perform the following steps to reload the TLS context on each replication SQL pod in the cnDBTier cluster:

- a. Check the number of replication SQL pods present in the cnDBTier cluster:

```
$ kubectl get pods --namespace=<namespace of cnDBTier cluster> | grep ndbmysql
```

For example:

```
$ kubectl get pods --namespace=ocne-cndbtier | grep ndbmysql
```

Sample output:

ndbmysqld-0		3/3
Running	0	103m
ndbmysqld-1		3/3
Running	0	4h39m
ndbmysqld-2		3/3
Running	0	4h37m
ndbmysqld-3		3/3
Running	0	4h36m
ndbmysqld-4		3/3
Running	0	4h34m
ndbmysqld-5		3/3
Running	0	4h32m

- b. Reload the TLS context on each replication SQL pod:

```
$ kubectl exec -it <replication SQL pod> --namespace=<namespace of cnDBTier cluster> -- mysql -h127.0.0.1 -uroot -p<root password> -e "ALTER INSTANCE RELOAD TLS;"
```

For example:

- Run the following command to reload the TLS context on ndbmysqld-0:

```
$ kubectl exec -it ndbmysqld-0 --namespace=ocne-cndbtier -- mysql -h127.0.0.1 -uroot -pNextGenCne -e "ALTER INSTANCE RELOAD TLS;"
```

Sample output:

```
Defaulted container "mysqlndbcluster" out of: mysqlndbcluster, init-sidecar, db-infra-monitor-svc
mysql: [Warning] Using a password on the command line interface can be insecure.
```

- Run the following command to reload the TLS context on ndbmysqld-1:

```
$ kubectl exec -it ndbmysqld-1 --namespace=occne-cndbtier -- mysql -h127.0.0.1 -uroot -pNextGenCne -e "ALTER INSTANCE RELOAD TLS;"
```

Sample output:

```
Defaulted container "mysqlndbcluster" out of: mysqlndbcluster, init-sidecar, db-infra-monitor-svc
mysql: [Warning] Using a password on the command line interface can be insecure.
```

- Run the following command to reload the TLS context on ndbmysqld-2:

```
$ kubectl exec -it ndbmysqld-2 --namespace=occne-cndbtier -- mysql -h127.0.0.1 -uroot -pNextGenCne -e "ALTER INSTANCE RELOAD TLS;"
```

Sample output:

```
Defaulted container "mysqlndbcluster" out of: mysqlndbcluster, init-sidecar, db-infra-monitor-svc
mysql: [Warning] Using a password on the command line interface can be insecure.
```

- Run the following command to reload the TLS context on ndbmysqld-3:

```
$ kubectl exec -it ndbmysqld-3 --namespace=occne-cndbtier -- mysql -h127.0.0.1 -uroot -pNextGenCne -e "ALTER INSTANCE RELOAD TLS;"
```

Sample output:

```
Defaulted container "mysqlndbcluster" out of: mysqlndbcluster, init-sidecar, db-infra-monitor-svc
mysql: [Warning] Using a password on the command line interface can be insecure.
```

- Run the following command to reload the TLS context on ndbmysqld-4:

```
$ kubectl exec -it ndbmysqld-4 --namespace=occne-cndbtier -- mysql -h127.0.0.1 -uroot -pNextGenCne -e "ALTER INSTANCE RELOAD TLS;"
```

Sample output:

```
Defaulted container "mysqlndbcluster" out of: mysqlndbcluster, init-sidecar, db-infra-monitor-svc
```

mysql: [Warning] Using a password on the command line interface can be insecure.

- Run the following command to reload the TLS context on ndbmysqld-5:

```
$ kubectl exec -it ndbmysqld-5 --namespace=occne-cndbtier -- mysql -h127.0.0.1 -uroot -pNextGenCne -e "ALTER INSTANCE RELOAD TLS;"
```

Sample output:

```
Defaulted container "mysqlndbcluster" out of: mysqlndbcluster, init-sidecar, db-infra-monitor-svc
mysql: [Warning] Using a password on the command line interface can be insecure.
```

12. Repeat Step 12 on all the mate sites where you have updated the certificates.
13. Perform the following steps to run a rollout restart of the ndbmysqld pods in the cnDBTier cluster:
 - a. Run the following command to get the statefulset name of the ndbmysqld pod:

```
$ kubectl get statefulset --namespace=<namespace of cnDBTier cluster>
```

For example:

```
$ kubectl get statefulset --namespace=occne-cndbtier
```

Sample output:

NAME	READY	AGE
ndbappmysqld	2/2	27h
ndbmgmd	2/2	27h
ndbmttd	2/2	27h
ndbmysqld	2/2	27h

- b. Perform a rollout restart of the ndbmsqld pod:

```
$ kubectl rollout restart statefulset <statefulset name of ndbmysqld> --namespace=<namespace of cnDBTier cluster>
```

For example:

```
$ kubectl rollout restart statefulset ndbmysqld --namespace=occne-cndbtier
```

Sample output:

```
statefulset.apps/ndbmysqld restarted
```

- c. Wait for the rollout restart of the ndbmysqld pod to complete and check the status:

```
$ kubectl rollout status statefulset <statefulset name of ndbmysqld> --
namespace=<namespace of cnDBTier cluster>
```

For example:

```
$ kubectl rollout status statefulset ndbmysqld --namespace=occne-
cndbtier
```

Sample output:

```
Waiting for 1 pods to be ready...
waiting for statefulset rolling update to complete 1 pods at revision
ndbmysqld-7c6b9c9f84...
Waiting for 1 pods to be ready...
Waiting for 1 pods to be ready...
statefulset rolling update complete 2 pods at revision
ndbmysqld-7c6b9c9f84...
```

14. Repeat Step 14 on all the mate sites where you have updated the certificates.

7.4.2 Managing TLS for Communication with NFs

This section provides the procedures to enable TLS for communication with NFs and modify cnDBTier certificates that are used to establish a TLS connection between cnDBTier and NFs.

7.4.2.1 Enabling TLS for Communication with NFs

Perform the following steps to enable TLS for communication between cnDBTier and NFs.

1. Create all the required secrets using the "Creating Secret for TLS Certificates" procedure in *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide*.
2. Set the global/ndbappTLS/enable parameter to *true* in the custom_values.yaml file to enable the TLS feature for NF communication:

```
global:
  ndbappTLS:
    enable: true
```

3. Provide all the required certificates (such as, ca certificate, client certificate, and server certificate) for the application SQL pods in the custom_values.yaml file for the cnDBTier site:

```
ndbappTLS:
  enable: true
  caSecret: cndbtier-ndbapp-trust-store-secret
  serverSecret: cndbtier-ndbapp-server-secret
  tlsversion: "TLSv1.3"
  caCertificate: "<ca certificate file name>"
  serverCertificate: "<server certificate name>"
  serverCertificateKey: "<server key name>"
```

For example:

```
ndbappTLS:
  enable: true
  caSecret: cndbtier-ndbapp-trust-store-secret
  serverSecret: cndbtier-ndbapp-server-secret
  tlsversion: "TLSv1.3"
  caCertificate: "combine-ca.pem"
  serverCertificate: "server1-cert.pem"
  serverCertificateKey: "server1-key.pem"
```

4. Follow the installation procedure in *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide* to perform a Helm install with the updated `custom_values.yaml` file.

7.4.2.2 Modifying cnDBTier Certificates to Establish TLS for Communication with NFs

This section provides the procedure to modify cnDBTier certificates to establish an encrypted connection between cnDBTier and NFs using TLS.

① Note

- If you update the cnDBTier certificates, ensure that you update the relevant Certificate Authority (CA) certificates on all relevant NFs. This ensures that NF traffic to cnDBTier is not disrupted due to incorrect certificates.
- While recreating the secrets, ensure that the file names of the updated certificates are same as the old certificates.

1. Get the list of the existing CA certificates that are configured for the application SQL pod of the cnDBTier cluster:

① Note

If the file name has a dot either in the extension or in its name, use an escape character `\` before the dot.

```
$ kubectl get secret cndbtier-ndbapp-trust-store-secret -n
<cndbtier_namespace> -o jsonpath="{.data.<ca_certificate_file_name>}" |
base64 --decode
```

For example:

```
$ kubectl get secret cndbtier-ndbapp-trust-store-secret -n occne-cndbtier
-o jsonpath="{.data.ca\\.pem}" | base64 --decode
```

2. If you want to modify the secret of the existing CA certificates in the cnDBTier cluster, delete the existing CA certificate(s) secret configured for the application SQL pod. Otherwise, skip this step and move to Step 4.

```
$ kubectl -n <namespace> delete secret cndbtier-ndbapp-trust-store-secret
```

where, <namespace> is the name of the cnDBTier namespace.

For example:

```
$ kubectl -n occne-cndbtier delete secret cndbtier-ndbapp-trust-store-secret
```

3. If you deleted the secret of the existing CA certificates in the previous step, then create the cndbtier-ndbapp-trust-store-secret secret with the new CA certificates. For more information, see the "Create Secret for TLS Certificates" section in *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide*.

```
$ kubectl -n ${OCCNE_NAMESPACE} create secret generic cndbtier-ndbapp-trust-store-secret --from-file=<ca_certificate>
```

For example:

```
$ kubectl -n ${OCCNE_NAMESPACE} create secret generic cndbtier-ndbapp-trust-store-secret --from-file=combine-ca.pem
```

4. Perform the following steps to get the existing server certificate and the server key of the cnDBTier cluster.

Note

If the file names have dots either in the extension or in its name, use an escape character \ before the dots.

- a. Run the following command to get the server certificate:

```
$ kubectl get secret cndbtier-ndbapp-server-secret -n  
<cndbtier_namespace> -o  
jsonpath="{.data.<server_certificate_file_name>}" | base64 --decode
```

For example:

```
$ kubectl get secret cndbtier-ndbapp-server-secret -n occne-cndbtier -  
o jsonpath="{.data.server-cert\.pem}" | base64 --decode
```

- b. Run the following command to get the server certificate key:

```
$ kubectl get secret cndbtier-ndbapp-server-secret -n  
<cndbtier_namespace> -o  
jsonpath="{.data.<server_certificate_key_file_name>}" | base64 --decode
```

For example:

```
$ kubectl get secret cndbtier-ndbapp-server-secret -n occne-cndbtier -
o jsonpath="{.data.server-key\.pem}" | base64 --decode
```

5. If you want to modify the secret of the existing server certificate and server key secret, delete the secret using the following command. Otherwise, skip this step and move to Step 7.

```
$ kubectl -n <cndbtier_namespace> delete secret cndbtier-ndbapp-server-
secret
```

For example:

```
$ kubectl -n occne-cndbtier delete secret cndbtier-ndbapp-server-secret
```

6. If you deleted the secret of the existing server certificate and server key in the previous step, then create the cndbtier-ndbapp-server-secret secret with the new server certificate and server key. For more information, see the "Creating Secrets for TLS Certificates" section in *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide*.

```
$ kubectl -n ${OCCNE_NAMESPACE} create secret generic cndbtier-ndbapp-
server-secret --from-file=<server_certificate> --from-
file=<server_certificate_key>
```

For example:

```
$ kubectl -n ${OCCNE_NAMESPACE} create secret generic cndbtier-ndbapp-
server-secret --from-file=server-cert.pem --from-file=server-key.pem
```

7. If you modified the CA certificates in the current site, then you must update the certificates in other NFs too. Perform Steps 1 through 6 to modify the relevant certificates on all relevant NFs.
8. Wait for the new certificates of the recreated secret to be automatically mounted to the application SQL pods by the Kubernetes API server.

Note

This could take around 30 seconds. However, the wait time depends on the Kubernetes configuration and varies from environment to environment.

9. Perform the following steps to reload the TLS context on each application SQL pod in the cnDBTier cluster:
 - a. Check the number of application SQL pods present in the cnDBTier cluster:

```
$ kubectl get pods --namespace=<namespace of cnDBTier cluster> | grep
ndbappmysql
```


For example:

```
$ kubectl get pods --namespace=occne-cndbtier | grep ndbmysql
```

Sample output:

```
ndbappmysqld-0
3/3      Running    0          103m
ndbappmysqld-1
3/3      Running    0          4h39m
```

- b. Reload the TLS context on each application SQL pod:

```
$ kubectl exec -it <APP SQL pod> --namespace=<namespace of cnDBTier
cluster> -- mysql -h127.0.0.1 -uroot -p<root password> -e "ALTER
INSTANCE RELOAD TLS;"
```

For example:

- Run the following command to reload the TLS context on ndbappmysqld-0:

```
$ kubectl exec -it ndbappmysqld-0 --namespace=occne-cndbtier --
mysql -h127.0.0.1 -uroot -pNextGenCne -e "ALTER INSTANCE RELOAD
TLS;"
```

Sample output:

```
Defaulted container "mysqlndbcluster" out of: mysqlndbcluster, init-
sidecar, db-infra-monitor-svc
mysql: [Warning] Using a password on the command line interface can
be insecure.
```

- Run the following command to reload the TLS context on ndbappmysqld-1:

```
$ kubectl exec -it ndbappmysqld-1 --namespace=occne-cndbtier --
mysql -h127.0.0.1 -uroot -pNextGenCne -e "ALTER INSTANCE RELOAD
TLS;"
```

Sample output:

```
Defaulted container "mysqlndbcluster" out of: mysqlndbcluster, init-
sidecar, db-infra-monitor-svc
mysql: [Warning] Using a password on the command line interface can
be insecure.
```

10. Perform the following steps to run a rollout restart of the ndbappmysqld pods in the cnDBTier cluster:

- a. Run the following command to get the statefulset name of the ndbappmysqld pod:

```
$ kubectl get statefulset --namespace=<namespace of cnDBTier cluster>
```

For example:

```
$ kubectl get statefulset --namespace=occne-cndbtier
```

Sample output:

NAME	READY	AGE
ndbappmysqld	2/2	27h
ndbmgmd	2/2	27h
ndbmttd	2/2	27h
ndbmysqld	2/2	27h

- b. Perform a rollout restart of the ndbappmysqld pod:

```
$ kubectl rollout restart statefulset <statefulset name of  
ndbappmysqld> --namespace=<namespace of cnDBTier cluster>
```

For example:

```
$ kubectl rollout restart statefulset ndbappmysqld --namespace=occne-  
cndbtier
```

Sample output:

```
statefulset.apps/ndbappmysqld restarted
```

- c. Wait for the rollout restart of the ndbappmysqld pod to complete and check the status:

```
$ kubectl rollout status statefulset <statefulset name of ndbappmysqld>  
--namespace=<namespace of cnDBTier cluster>
```

For example:

```
$ kubectl rollout status statefulset ndbappmysqld --namespace=occne-  
cndbtier
```

Sample output:

```
Waiting for 1 pods to be ready...  
waiting for statefulset rolling update to complete 1 pods at revision  
ndbappmysqld-7c6b9c9f84...  
Waiting for 1 pods to be ready...  
Waiting for 1 pods to be ready...  
statefulset rolling update complete 2 pods at revision  
ndbappmysqld-7c6b9c9f84...
```

7.5 Adding a Georedundant cnDBTier Cluster

This section describes the procedures to add a cnDBTier georedundant cluster to an existing single site, two site, and three site georedundant cnDBTier clusters.

Note

- If the existing Georedundant cnDBTier cluster is configured with single replication channel, then the new cnDBTier cluster being added by following this procedure must also be configured with a single replication channel.
- If the existing Georedundant cnDBTier cluster is configured with multiple replication channels, then the new cnDBTier cluster being added by following this procedure must also be configured with multiple replication channel groups.

The procedures in this section use the following terms to identify the clusters:

1. cnDBTier Cluster1 (a.k.a. cluster1): The first cloud native based DBTier cluster in a two site, three site, or four site georeplication setup.
2. cnDBTier Cluster2 (a.k.a. cluster2): The second cloud native based DBTier cluster in a two site, three site, or four site georeplication setup.
3. cnDBTier Cluster3 (a.k.a. cluster3): The third cloud native based DBTier cluster in a two site, three site, or four site georeplication setup.
4. cnDBTier Cluster4 (a.k.a. cluster4): The fourth cloud native based DBTier cluster in a two site, three site, or four site georeplication setup.

Prerequisites

1. All the cnDBTier data nodes and SQL nodes that participate in georeplication, and at least one management node in the existing clusters must be up and running.
2. Georeplication must be established between the existing cnDBTier clusters.
3. All the cnDBTier clusters must be installed using cnDBTier 22.1.x or above.
4. All the cnDBTier clusters must have the same number of data nodes and node groups.

7.5.1 Adding cnDBTier Georedundant Cluster to Single Site cnDBTier Cluster

This section describes the procedure to add a cnDBTier georedundant cluster (cnDBTier cluster2) to an existing single site cnDBTier cluster (cnDBTier cluster1).

1. If TLS was configured on cnDBTier cluster1, then follow the [Modifying cnDBTier Certificates to Establish TLS Between Georeplication Sites](#) procedure to update the certificates in cnDBTier cluster1.

Note

Perform this step on cnDBTier cluster1 only if you want to reconfigure the CA certificates for the new cnDBTier cluster.

2. Check if georeplication is enabled in cnDBTier cluster1 by performing the following steps:

Note

The following commands and examples are applicable for a single replication channel group only. If multiple replication channel groups are enabled, then check if georeplication is enabled in cnDBTier cluster1 for every replication group.

- a. Ensure that the DB replication service in cnDBTier cluster1 is enabled and running successfully with respect to cnDBTier cluster2.

```
$ kubectl -n cluster1 get pods | grep repl
```

Sample output:

```
mysql-cluster-cluster1-cluster2-replication-svc-5bdc46crzzhb 1/1 Running
0 3m11s
```

- b. Ensure that at least two georeplication SQL nodes are configured in cluster1.

```
$ kubectl -n cluster1 get pods | grep ndbmysqld
```

Sample output:

```
ndbmysqld-0 3/3 Running 0 4m14s
ndbmysqld-1 3/3 Running 0 3m53s
```

- c. Check the status of cnDBTier cluster1.

```
$ kubectl -n cluster1 exec -it ndbmcmd-0 -- ndb_mgm -e show
```

Sample output:

```
Connected to Management Server at: localhost:1186 Cluster Configuration
```

```
-----
[ndbd(NDB)] 2 node(s)
id=1 @10.233.85.92 (mysql-8.4.3 ndb-8.4.3, Nodegroup: 0)
id=2 @10.233.114.33 (mysql-8.4.3 ndb-8.4.3, Nodegroup: 0, *)
```

```
[ndb_mgmd(MGM)] 2 node(s)
id=49 @10.233.65.167 (mysql-8.4.3 ndb-8.4.3)
id=50 @10.233.127.115 (mysql-8.4.3 ndb-8.4.3)
```

```
[mysqld(API)] 8 node(s)
id=56 @10.233.114.34 (mysql-8.4.3 ndb-8.4.3)
id=57 @10.233.85.91 (mysql-8.4.3 ndb-8.4.3)
id=70 @10.233.127.117 (mysql-8.4.3 ndb-8.4.3)
id=71 @10.233.85.93 (mysql-8.4.3 ndb-8.4.3)
id=222 (not connected, accepting connect from any host)
id=223 (not connected, accepting connect from any host)
id=224 (not connected, accepting connect from any host)
id=225 (not connected, accepting connect from any host)
```

Note

- If cluster1 satisfies all the conditions in Step 1, then georeplication is enabled in cnDBTier cluster1. Otherwise, georeplication is not enabled in cluster1.
- The node IDs 222 to 225 in the sample output are shown as "not connected" as these nodes are added as empty slot IDs used for georeplication recovery. You can ignore these nodes. This note is applicable to all similar outputs in this procedure.

3. If georeplication is not enabled in cluster1 as per [Step 1](#), then perform the following steps to enable georeplication in cluster1. Else, skip to [Step 3](#).

Note

- This step is not used to enable Multi Replication Channel Group on the existing cnDBTier cluster1. Multi replication channel groups with the required number of channel groups must be already configured on cnDBTier cluster1.
- Do not run a Helm test when performing a conversion procedure.

- a. Configure the remote mate site name (cluster2) in cnDBTier cluster1 by enabling and configuring the replication service using the `custom_values.yaml` file.

Example:

```
$ vi ocndbtier/custom_values.yaml
```

Sample output:

```
dbreplsvcdeployments:
  # if pod prefix is given then use the unique smaller name for this db
  replication service.
  - name: cluster1-cluster2-replication-svc
    enabled: true
    ....
    ....
    ....
  replication:
    # Local site replication service LoadBalancer ip can be configured.
    localsiteip: ""
    matesitename: "cluster2"
    remotesiteip: ""
```

- b. Configure the number of SQL pods in cnDBTier cluster1 to at least two in the `custom_values.yaml` file.

Example with output:

```
$ vi ocndbtier/custom_values.yaml
apiReplicaCount: 2
```

- c. Upgrade cnDBTier cluster1 by using the CSAR package.

```
$ helm upgrade mysql-cluster occndbtier --namespace cluster1 -f
occndbtier/custom_values.yaml
```

Sample output:

```
Release "mysql-cluster" has been upgraded. Happy Helming!
NAME: mysql-cluster
LAST DEPLOYED: Mon May 20 11:10:32 2025
NAMESPACE: cluster1
STATUS: deployed
REVISION: 2
```

- d. Wait until the upgrade is complete and all the pods of cnDBTier cluster 1 are restarted. Verify the status of the cluster by performing the following steps:
 - i. Check the status of pods running cluster1:

```
$ kubectl get pods --namespace=cluster1
```

Sample output:

NAME	READY	STATUS	RESTARTS	AGE
mysql-cluster-cluster1-cluster2-replication-svc-7676cc7bd62zssl	1/1	Running	0	153m
mysql-cluster-db-backup-manager-svc-c77df67b7-tqnd2	1/1	Running	0	153m
mysql-cluster-db-monitor-svc-69ff969477-646tz	1/1	Running	0	147m
ndbappmysqld-0	2/2	Running	0	148m
ndbappmysqld-1	2/2	Running	0	147m
ndbmgmd-0	1/1	Running	0	152m
ndbmgmd-1	1/1	Running	0	152m
ndbmt-d-0	2/2	Running	0	150m
ndbmt-d-1	2/2	Running	0	151m
ndbmysqld-0	2/2	Running	0	147m
ndbmysqld-1	2/2	Running	0	147m

- ii. Check the status of cnDBTier cluster1:

```
$ kubectl -n cluster1 exec -it ndbmgmd-0 -- ndb_mgm -e show
```

Sample output:

```
Connected to Management Server at: localhost:1186 Cluster
Configuration
```

```

-----
[ndbd(NDB)]      2 node(s)
id=1   @10.233.85.92  (mysql-8.4.3 ndb-8.4.3, Nodegroup: 0)
id=2   @10.233.114.33 (mysql-8.4.3 ndb-8.4.3, Nodegroup: 0, *)

[ndb_mgmd(MGM)]  2 node(s)
id=49   @10.233.65.167 (mysql-8.4.3 ndb-8.4.3)
id=50   @10.233.127.115 (mysql-8.4.3 ndb-8.4.3)

[mysqld(API)]    8 node(s)
id=56   @10.233.114.34 (mysql-8.4.3 ndb-8.4.3)
id=57   @10.233.85.91  (mysql-8.4.3 ndb-8.4.3)
id=70   @10.233.127.117 (mysql-8.4.3 ndb-8.4.3)
id=71   @10.233.85.93  (mysql-8.4.3 ndb-8.4.3)
id=222  (not connected, accepting connect from any host)
id=223  (not connected, accepting connect from any host)
id=224  (not connected, accepting connect from any host)
id=225  (not connected, accepting connect from any host)

```

Note

Node IDs 222 to 225 in the sample output are shown as "not connected" as these are added as empty slot IDs that are used for georeplication recovery. You can ignore these node IDs.

4. Install cnDBTier cluster2 by configuring cnDBTier cluster1 as mate site with at least two SQL pods. For information on installing a cnDBTier cluster, see *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide*.

Note

- If multiple replication channel groups are enabled, then install cnDBTier cluster2 by configuring cnDBTier cluster1 as mate site with at least four SQL pods.
- Do not run a Helm test when performing the conversion procedure.
- If you are going to enable encryption, ensure that the new site uses the same encryption key.

5. Check if the georeplication channels are established between cnDBTier cluster1 and cnDBTier cluster2 by running the following commands:

Note

The following commands and examples are applicable for a single replication channel group only. If multi replication channel groups are enabled, then check if the georeplication channels are established between cnDBTier cluster1 and cnDBTier cluster2 for every replication channel group.

```

$ kubectl -n cluster1 exec -it ndbmysqld-0 -- bash
$ mysql -h 127.0.0.1 -uroot -p

```

Password:

```
mysql> select * from replication_info.DBTIER_REPLICATION_CHANNEL_INFO;
```

Sample output:

```
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
| remote_site_name | remote_server_id | channel_id | remote_signaling_ip | role |
| start_epoch | site_name | server_id | start_ts | replchannel_group_id |
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
| cluster1 | | 1000 | 2005601 | 10.233.51.94 | ACTIVE |
| NULL | cluster2 | 2000 | 2024-02-19 19:10:45 | 1 |
| cluster1 | | 1001 | 2005702 | 10.233.10.190 | STANDBY |
| NULL | cluster2 | 2001 | 2024-02-19 19:10:44 | 1 |
| cluster2 | | 2000 | 2005601 | 10.233.8.24 | ACTIVE |
| NULL | cluster1 | 1000 | 2024-02-19 19:10:44 | 1 |
| cluster2 | | 2001 | 2005702 | 10.233.61.239 | STANDBY |
| NULL | cluster1 | 1001 | 2024-02-19 19:10:45 | 1 |
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
```

4 rows in set (0.01 sec)

6. Restore the cnDBTier cluster1 data in cnDBTier cluster2 and reestablish the georeplication channels by performing the fault recovery procedure. You can perform the georeplication recovery using cnDBTier or CNC Console. For more information, see the *Georeplication Failure Between cnDBTier Clusters in Two Site Replication* section of *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide*.
7. Verify the replication status in cnDBTier cluster1 and cnDBTier cluster2 by performing the following steps:

Note

The following commands and examples are applicable for a single replication channel group only. If multiple replication channel groups are enabled, then verify the replication status in cnDBTier cluster1 and cnDBTier cluster2 for every replication group.

- a. Log in to the cnDBTier cluster1 Bastion Host and check the replication status in the active replication channel:

```
$ kubectl -n cluster1 exec -it ndbmysqld-0 -- bash
```

```
$ mysql -h 127.0.0.1 -uroot -p
```

Password:

```
mysql> select * from replication_info.DBTIER_REPLICATION_CHANNEL_INFO;
```

Sample output:

```
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
+-----+
| remote_site_name | remote_server_id | channel_id | remote_signaling_ip |
| role | start_epoch | site_name | server_id | start_ts |
| replchannel_group_id |
+-----+-----+-----+-----+-----+
```



```

+-----+-----+-----+-----+-----+-----+
+-----+
| cluster1 | | 1000 | | 2005601 | | 10.233.51.94 | |
ACTIVE | | NULL | cluster2 | | 2000 | | 2024-02-19 19:10:45 |
| | 1 |
| cluster1 | | 1001 | | 2005702 | | 10.233.10.190 | |
STANDBY | | NULL | cluster2 | | 2001 | | 2024-02-19 19:10:44 |
| | 1 |
| cluster2 | | 2000 | | 2005601 | | 10.233.8.24 | |
ACTIVE | | NULL | cluster1 | | 1000 | | 2024-02-19 19:10:44 |
| | 1 |
| cluster2 | | 2001 | | 2005702 | | 10.233.61.239 | |
STANDBY | | NULL | cluster1 | | 1001 | | 2024-02-19 19:10:45 |
| | 1 |
+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
+-----+
4 rows in set (0.01 sec)

```

- b. Check if the replication is turned on (that is, Replica_IO_Running and Replica_SQL_Running are set to Yes) in the ACTIVE replication channel.
Example to check cluster 1:

```

$ kubectl -n cluster1 exec -it ndbmysqld-0 -- bash
$ mysql -h 127.0.0.1 -uroot -p

```

Sample output:

```

Password:
mysql> SHOW REPLICA STATUS\G;
***** 1. row *****
      Replica_IO_State: Waiting for master to send event
      Source_Host: 10.233.0.147
      Source_User: occnerepluser
      Source_Port: 3306
      Connect_Retry: 60
      Source_Log_File: mysql-bin.000007
      Read_Source_Log_Pos: 16442
      Relay_Log_File: mysql-relay-bin.000002
      Relay_Log_Pos: 11203
      Relay_Source_Log_File: mysql-bin.000007
      Replica_IO_Running: Yes
      Replica_SQL_Running: Yes
      ....
      ....
      ....
      Replica_SQL_Running_State: Replica has read all relay log; waiting for more
updates
      Source_Retry_Count: 86400
      ....
      ....

```

Example to check cluster 2:

```

$ kubectl -n cluster1 exec -it ndbmysqld-1 -- bash
$ mysql -h 127.0.0.1 -uroot -p

```

Sample output:

```

Password:
mysql> SHOW REPLICA STATUS\G;
***** 1. row *****
      Replica_IO_State:
        Source_Host: 10.233.9.35
        Source_User: occnerepluser
        Source_Port: 3306
        Connect_Retry: 60
        Source_Log_File: mysql-bin.000005
        Read_Source_Log_Pos: 26970
        Relay_Log_File: mysql-relay-bin.000002
        Relay_Log_Pos: 2228
        Relay_Source_Log_File: mysql-bin.000005
        Replica_IO_Running: No
        Replica_SQL_Running: No

```

- c. Log in to the cnDBTier cluster2 Bastion Host and check the replication status in the active replication channel:

```

$ kubectl -n cluster2 exec -it ndbmysqld-0 -- bash
$ mysql -h 127.0.0.1 -uroot -p
Password:
mysql> select * from replication_info.DBTIER_REPLICATION_CHANNEL_INFO;

```

Sample output:

```

+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+
| remote_site_name | remote_server_id | channel_id | remote_signaling_ip |
| role            | start_epoch      | site_name  | server_id            | start_ts          |
| replchannel_group_id |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+
| cluster1         | 1000             | 2005601    | 10.233.51.94         |
ACTIVE | NULL | cluster2 | 2000 | 2024-02-19 19:10:45 |
| 1 |
| cluster1         | 1001             | 2005702    | 10.233.10.190        |
STANDBY | NULL | cluster2 | 2001 | 2024-02-19 19:10:44 |
| 1 |
| cluster2         | 2000             | 2005601    | 10.233.8.24          |
ACTIVE | NULL | cluster1 | 1000 | 2024-02-19 19:10:44 |
| 1 |
| cluster2         | 2001             | 2005702    | 10.233.61.239        |
STANDBY | NULL | cluster1 | 1001 | 2024-02-19 19:10:45 |
| 1 |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+
4 rows in set (0.00 sec)

```

- d. Check if the replication is turned on (that is, Replica_IO_Running and Replica_SQL_Running are set to Yes) in the ACTIVE replication channel.
Example for ndbmysqld-0:

```

$ kubectl -n cluster2 exec -it ndbmysqld-0 -- bash
$ mysql -h 127.0.0.1 -uroot -p
Password:
mysql> SHOW REPLICA STATUS\G;

```

Sample output:

```

***** 1. row *****
      Replica_IO_State: Waiting for master to send event
      Source_Host: 10.233.13.41
      Source_User: occnerepluser
      Source_Port: 3306
      Connect_Retry: 60
      Source_Log_File: mysql-bin.000007
      Read_Source_Log_Pos: 32792
      Relay_Log_File: mysql-relay-bin.000002
      Relay_Log_Pos: 17215
      Relay_Source_Log_File: mysql-bin.000007
      Replica_IO_Running: Yes
      Replica_SQL_Running: Yes
      ....
      ....
      ....
      Replica_SQL_Running_State: Replica has read all relay log; waiting for more
updates
      Source_Retry_Count: 86400
      ....
      ....

```

Example for ndbmysqld-1:

```

$ kubectl -n cluster2 exec -it ndbmysqld-1 -- bash
$ mysql -h 127.0.0.1 -uroot -p
Password:
mysql> SHOW REPLICA STATUS\G;

```

Sample output:

```

***** 1. row *****
      Replica_IO_State:
      Source_Host: 10.233.39.110
      Source_User: occnerepluser
      Source_Port: 3306
      Connect_Retry: 60
      Source_Log_File: mysql-bin.000007
      Read_Source_Log_Pos: 9785
      Relay_Log_File: mysql-relay-bin.000002
      Relay_Log_Pos: 203
      Relay_Source_Log_File: mysql-bin.000007
      Replica_IO_Running: No
      Replica_SQL_Running: No

```

Note

As data is replicated to cnDBTier cluster2 and georeplication channels are established between cnDBTier cluster1 and cnDBTier cluster2, the newly added cnDBTier cluster (cluster2) can also be used by the NFs for database operations.

7.5.2 Adding cnDBTier Georedundant Cluster to Two-Site cnDBTier Clusters

This section describes the procedure to add a cnDBTier georedundant cluster (cnDBTier cluster3) to the existing two-site georedundant cnDBTier clusters (cnDBTier cluster1 and cnDBTier cluster2).

1. If TLS was configured on cnDBTier cluster1 and cluster 2, then follow the [Modifying cnDBTier Certificates to Establish TLS Between Georeplication Sites](#) procedure to update the certificates in cnDBTier cluster1 and cluster 2.

Note

Perform this step on those cnDBTier clusters on which you want to reconfigure the CA certificates for the new cnDBTier cluster.

2. Check if georeplication is enabled in cnDBTier cluster1 with respect to cnDBTier cluster3 by performing the following steps:

Note

The following commands and examples are applicable for a single replication channel group only. If multiple replication channel groups are enabled, then check if georeplication is enabled in cnDBTier cluster1 for every replication group with respect to cnDBTier cluster3.

- a. Ensure that the DB replication service in cnDBTier cluster1 is enabled and running successfully with respect to cnDBTier cluster3:

```
$ kubectl -n cluster1 get pods | grep repl
```

Sample output:

```
mysql-cluster-cluster1-cluster2-replication-svc-5bdc46crzzhb 1/1 Running
0 3m11s
mysql-cluster-cluster1-cluster3-replication-svc-7955b6b6fbdcc 1/1 Running
2 3m09s
```

- b. Ensure that at least four georeplication SQL nodes are configured in cluster1:

```
$ kubectl -n cluster1 get pods | grep ndbmysqld
```

Sample output:

```
ndbmysqld-0 3/3
Running 0 4m14s
ndbmysqld-1 3/3
Running 0 3m53s
ndbmysqld-2 3/3
Running 0 3m21s
ndbmysqld-3 3/3
Running 0 3m01s
```

c. Check the status of cnDBTier cluster1:

```
$ kubectl -n cluster1 exec -it ndbmgmd-0 -- ndb_mgm -e show
```

Sample output:

```
Connected to Management Server at: localhost:1186 Cluster Configuration
-----
[ndbd(NDB)] 2 node(s)
id=1 @10.233.85.92 (mysql-8.4.3 ndb-8.4.3, Nodegroup: 0)
id=2 @10.233.114.33 (mysql-8.4.3 ndb-8.4.3, Nodegroup: 0, *)

[ndb_mgmd(MGM)] 2 node(s)
id=49 @10.233.65.167 (mysql-8.4.3 ndb-8.4.3)
id=50 @10.233.127.115 (mysql-8.4.3 ndb-8.4.3)

[mysqld(API)] 10 node(s)
id=56 @10.233.114.34 (mysql-8.4.3 ndb-8.4.3)
id=57 @10.233.85.91 (mysql-8.4.3 ndb-8.4.3)
id=58 @10.233.114.34 (mysql-8.4.3 ndb-8.4.3)
id=59 @10.233.85.91 (mysql-8.4.3 ndb-8.4.3)
id=70 @10.233.127.117 (mysql-8.4.3 ndb-8.4.3)
id=71 @10.233.85.93 (mysql-8.4.3 ndb-8.4.3)
id=222 (not connected, accepting connect from any host)
id=223 (not connected, accepting connect from any host)
id=224 (not connected, accepting connect from any host)
id=225 (not connected, accepting connect from any host)
```

Note

- The node IDs 222 to 225 in the sample output are shown as "not connected" as these nodes are added as empty slot IDs used for georeplication recovery. You can ignore these nodes.
- If cluster1 satisfies all the conditions in Step 1, then georeplication is enabled in cnDBTier cluster1. Otherwise, georeplication is not enabled.

3. If georeplication is not enabled in cluster1 as per [Step 1](#), then perform the following steps to enable georeplication in cnDBTier cluster1 with respect to cnDBTier cluster3. Else, skip to [Step 3](#).

Note

The following commands and examples are applicable for a single replication channel group only. If multiple replication channel groups are enabled, then configure remote mate site name (cluster3) in cnDBTier cluster1 by following the *Configuring Multiple Replication channel groups* procedure in *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide* and skip steps a and b in the following substeps.

- a. Configure the remote mate site name (cluster3) in cnDBTier cluster1 by enabling and configuring the replication service using the custom_values.yaml file:

```
$ vi ocncdbtier/custom_values.yaml
```

Sample output:

```
dbreplsvcd deployments:
  ....
  # if pod prefix is given then use the unique smaller name for this db
  replication service.
  - name: cluster1-cluster3-replication-svc
    enabled: true
    mysql:
      dbtierservice: "mysql-connectivity-service"
      dbtierreplservice: "ndbmysqldsvc"
      # if cndbtier, use CLUSTER-IP from the ndbmysqldsvc-2 LoadBalancer
    service
    primaryhost: "ndbmysqld-2.ndbmysqldsvc.cluster1.svc.occne4-cgbu-cne-
    dbtier"
    port: "3306"
    # if cndbtier, use EXTERNAL-IP from the ndbmysqldsvc-2 LoadBalancer
    service
    primarysignalhost: ""
    # serverid is unique; retrieve it for the site being configured
    primaryhostserverid: "1002"
    # if cndbtier, use CLUSTER-IP from the ndbmysqldsvc-3 LoadBalancer
    service
    secondaryhost: "ndbmysqld-3.ndbmysqldsvc.cluster1.svc.occne4-cgbu-cne-
    dbtier"
    # if cndbtier, use EXTERNAL-IP from the ndbmysqldsvc-3 LoadBalancer
    service
    secondarysignalhost: ""
    # serverid is unique; retrieve it for the site being configured
    secondaryhostserverid: "1003"
    replication:
      # Local site replication service LoadBalancer ip can be
    configured.
    localsiteip: ""
    matesitename: "cluster3"
    remotesiteip: ""
```

- b. Configure the number of SQL pods in cnDBTier cluster1 to at least four in the custom_values.yaml file:

Note

apiReplicaCount is set to "4" as you are adding the third site. The first two ndbmysqld are used for replication between site one and site two only. The two ndbmysqld pods that are newly added will be responsible for replication between site1 and site3.

```
$ vi occndbtier/custom_values.yaml
```

Sample output:

```
apiReplicaCount: 4
```

- c. Upgrade cnDBTier cluster1 by using the CSAR package. For more information, see *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide*.

Note

The following commands and examples are applicable for a single replication channel group only. If multiple replication channel groups are enabled, then upgrade cnDBTier cluster2 by providing appropriate multi-group values file.

```
$ helm upgrade mysql-cluster occndbtier --namespace cluster1 --no-hooks
-f occndbtier/custom_values.yaml
```

Sample output:

```
Release "mysql-cluster" has been upgraded. Happy Helming!
NAME: mysql-cluster
LAST DEPLOYED: Mon May 20 11:10:32 2025
NAMESPACE: cluster1
STATUS: deployed
REVISION: 2
```

- d. Wait until the upgrade is complete and all the pods of cnDBTier cluster 1 are restarted. Verify the status of the cluster by performing the following steps:

- i. Check the status of pods running cluster1:

```
$ kubectl get pods --namespace=cluster1
```

Sample output:

NAME	READY	STATUS	RESTARTS	AGE
mysql-cluster-cluster1-cluster2-replication-svc-7676cc7bd62zssl	1/1	Running	0	153m
mysql-cluster-db-backup-manager-svc-c77df67b7-tqnd2	1/1	Running	0	153m
mysql-cluster-db-monitor-svc-69ff969477-646tz	1/1	Running	0	147m
ndbappmysqld-0	2/2	Running	0	148m
ndbappmysqld-1	2/2	Running	0	147m
ndbmgmd-0	1/1	Running	0	152m
ndbmgmd-1	1/1	Running	0	152m
ndbmt-d-0	2/2	Running	0	150m
ndbmt-d-1	2/2	Running	0	151m
ndbmysqld-0	2/2	Running	0	147m
ndbmysqld-1	2/2	Running	0	147m

ii. Check the status of cnDBTier cluster1:

```
$ kubectl -n cluster1 exec -it ndbmgmd-0 -- ndb_mgm -e show
```

Sample output:

```
Connected to Management Server at: localhost:1186 Cluster
Configuration
-----
[ndbd(NDB)]      2 node(s)
id=1      @10.233.85.92  (mysql-8.4.3 ndb-8.4.3, Nodegroup: 0)
id=2      @10.233.114.33 (mysql-8.4.3 ndb-8.4.3, Nodegroup: 0, *)

[ndb_mgmd(MGM)]  2 node(s)
id=49      @10.233.65.167 (mysql-8.4.3 ndb-8.4.3)
id=50      @10.233.127.115 (mysql-8.4.3 ndb-8.4.3)

[mysqld(API)]    8 node(s)
id=56      @10.233.114.34 (mysql-8.4.3 ndb-8.4.3)
id=57      @10.233.85.91  (mysql-8.4.3 ndb-8.4.3)
id=70      @10.233.127.117 (mysql-8.4.3 ndb-8.4.3)
id=71      @10.233.85.93  (mysql-8.4.3 ndb-8.4.3)
id=222 (not connected, accepting connect from any host)
id=223 (not connected, accepting connect from any host)
id=224 (not connected, accepting connect from any host)
id=225 (not connected, accepting connect from any host)
```

Note

Node IDs 222 to 225 in the sample output are shown as "not connected" as these are added as empty slot IDs that are used for georeplication recovery. You can ignore these node IDs.

4. Check if georeplication is enabled in cnDBTier cluster2 with respect to cnDBTier cluster3 by performing the following steps:

Note

The following commands and examples are applicable for a single replication channel group only. If multiple replication channel groups are enabled, then check if georeplication is enabled in cnDBTier cluster2 for every replication group with respect to cnDBTier cluster3.

a. Ensure that the DB replication service in cnDBTier cluster2 is enabled and running successfully with respect to cnDBTier cluster3:

```
$ kubectl -n cluster2 get pods | grep repl
```

Sample output:

```
mysql-cluster-cluster2-cluster1-replication-svc-5bdc46crzzhb 1/1 Running
0 3m11s
```



```
mysql-cluster-cluster2-cluster3-replication-svc-7955b6b6fbd 1/1 Running
2 3m39s
```

- b. Ensure that at least four georeplication SQL nodes are configured in cluster2:

```
$ kubectl -n cluster2 get pods | grep ndbmysqld
```

Sample output:

```
ndbmysqld-0 3/3
Running 0 4m14s
ndbmysqld-1 3/3
Running 0 3m53s
ndbmysqld-2 3/3
Running 0 3m21s
ndbmysqld-3 3/3
Running 0 3m01s
```

- c. Check the status of cnDBTier cluster2:

```
$ kubectl -n cluster2 exec -it ndbmgmd-0 -- ndb_mgm -e show
```

Sample output:

```
Connected to Management Server at: localhost:1186 Cluster Configuration
```

```
-----
[ndbd(NDB)] 2 node(s)
id=1 @10.233.85.92 (mysql-8.4.3 ndb-8.4.3, Nodegroup: 0)
id=2 @10.233.114.33 (mysql-8.4.3 ndb-8.4.3, Nodegroup: 0, *)

[ndb_mgmd(MGM)] 2 node(s)
id=49 @10.233.65.167 (mysql-8.4.3 ndb-8.4.3)
id=50 @10.233.127.115 (mysql-8.4.3 ndb-8.4.3)

[mysqld(API)] 10 node(s)
id=56 @10.233.114.34 (mysql-8.4.3 ndb-8.4.3)
id=57 @10.233.85.91 (mysql-8.4.3 ndb-8.4.3)
id=58 @10.233.114.34 (mysql-8.4.3 ndb-8.4.3)
id=59 @10.233.85.91 (mysql-8.4.3 ndb-8.4.3)
id=70 @10.233.127.117 (mysql-8.4.3 ndb-8.4.3)
id=71 @10.233.85.93 (mysql-8.4.3 ndb-8.4.3)
id=222 (not connected, accepting connect from any host)
id=223 (not connected, accepting connect from any host)
id=224 (not connected, accepting connect from any host)
id=225 (not connected, accepting connect from any host)
```

① Note

- If cluster2 satisfies all the conditions in Step 3, then georeplication is enabled in cnDBTier cluster2. Otherwise, georeplication is not enabled in cluster2.
- The node IDs 222 to 225 in the sample output are shown as "not connected" as these nodes are added as empty slot IDs used for georeplication recovery. You can ignore these nodes.

5. If georeplication is not enabled in cluster2 as per [Step 3](#), then perform the following steps to enable georeplication in cluster2 with respect to cluster3. Else, skip to Step 5.

Note

The following commands and examples are applicable for a single replication channel group only. If multiple replication channel groups are enabled, then configure remote mate site name (cluster3) in cnDBTier cluster2 by following the *Configuring Multiple Replication channel groups* procedure in *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide* and skip steps a and b in the following substeps.

- a. Configure the remote mate site name (cluster3) in cnDBTier cluster2 by enabling and configuring the replication service using the `custom_values.yaml` file:

```
$ vi occndbtier/custom_values.yaml
```

Sample output:

```
dbreplsvcdeployments:
  ....
  # if pod prefix is given then use the unique smaller name for this db
  replication service.
  - name: cluster2-cluster3-replication-svc
    enabled: true
    mysql:
      dbtierservice: "mysql-connectivity-service"
      dbtierreplservice: "ndbmysqldsvc"
      # if cndbtier, use CLUSTER-IP from the ndbmysqldsvc-2 LoadBalancer
    service
    primaryhost: "ndbmysqld-2.ndbmysqldsvc.cluster2.svc.occne4-cgbu-cne-
dbtier"
    port: "3306"
    # if cndbtier, use EXTERNAL-IP from the ndbmysqldsvc-2 LoadBalancer
    service
    primarysignalhost: ""
    # serverid is unique; retrieve it for the site being configured
    primaryhostserverid: "2002"
    # if cndbtier, use CLUSTER-IP from the ndbmysqldsvc-3 LoadBalancer
    service
    secondaryhost: "ndbmysqld-3.ndbmysqldsvc.cluster2.svc.occne4-cgbu-cne-
dbtier"
    # if cndbtier, use EXTERNAL-IP from the ndbmysqldsvc-3 LoadBalancer
    service
    secondarysignalhost: ""
    # serverid is unique; retrieve it for the site being configured
    secondaryhostserverid: "2003"
    replication:
      # Local site replication service LoadBalancer ip can be configured.
      localsiteip: ""
      matesitename: "cluster3"
      remotesiteip: ""
```

- b. Configure the number of SQL pods in cnDBTier cluster2 to at least four in the `custom_values.yaml` file:

Note

apiReplicaCount is set to "4" as you are adding the third site. The first two ndbmysqld are used for replication between site one and site two only. The two ndbmysqld pods that are newly added will be responsible for replication between site1 and site3.

```
$ vi occndbtier/custom_values.yaml
```

Sample output:

```
apiReplicaCount: 4
```

- c. Upgrade cnDBTier cluster2 by using the CSAR package. For more information, see *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide*.

Note

The following commands and examples are applicable for a single replication channel group only. If multiple replication channel groups are enabled, then upgrade cnDBTier cluster2 by providing appropriate multi-group values file.

```
$ helm upgrade mysql-cluster occndbtier --namespace cluster2 --no-hooks
-f occndbtier/custom_values.yaml
```

Sample output:

```
Release "mysql-cluster" has been upgraded. Happy Helming!
NAME: mysql-cluster
LAST DEPLOYED: Mon May 20 11:10:32 2025
NAMESPACE: cluster2
STATUS: deployed
REVISION: 2
```

- d. Wait until the upgrade is complete and all the pods of cnDBTier cluster 2 are restarted. Verify the status of the cluster by performing the following steps:
 - i. Check the status of pods running cluster2:

```
$ kubectl get pods --namespace=cluster2
```

Sample output:

```
NAME
READY   STATUS    RESTARTS   AGE
mysql-cluster-cluster2-cluster1-replication-svc-578c6578c85d2z5
1/1     Running   0           3m
mysql-cluster-cluster2-cluster3-replication-svc-578c6578c8kpzpv
1/1     Running   0           3m
mysql-cluster-db-backup-manager-svc-688f7cf97d-dxjdt
1/1     Running   0          148m
mysql-cluster-db-monitor-svc-7cdb4f4dd4-l6t87
```

```

1/1      Running  0          148m
ndbappmysqld-0
2/2      Running  0          148m
ndbappmysqld-1
2/2      Running  0          147m
ndbmgmd-0
1/1      Running  0          152m
ndbmgmd-1
1/1      Running  0          152m
ndbmt-d-0
2/2      Running  0          150m
ndbmt-d-1
2/2      Running  0          151m
ndbmysqld-0
2/2      Running  0          147m
ndbmysqld-1
2/2      Running  0          147m
ndbmysqld-2
2/2      Running  0          148m
ndbmysqld-3
2/2      Running  0          149m

```

ii. Check the status of cnDBTier cluster2:

```
$ kubectl -n cluster2 exec -it ndbmgmd-0 -- ndb_mgm -e show
```

Sample output:

```

Connected to Management Server at: localhost:1186 Cluster
Configuration
-----
[ndbd(NDB)]      2 node(s)
id=1      @10.233.85.92  (mysql-8.4.3 ndb-8.4.3, Nodegroup: 0)
id=2      @10.233.114.33 (mysql-8.4.3 ndb-8.4.3, Nodegroup: 0, *)

[ndb_mgmd(MGM)]  2 node(s)
id=49      @10.233.65.167 (mysql-8.4.3 ndb-8.4.3)
id=50      @10.233.127.115 (mysql-8.4.3 ndb-8.4.3)

[mysqld(API)]    10 node(s)
id=56      @10.233.124.92  (mysql-8.4.3 ndb-8.4.3)
id=57      @10.233.114.135 (mysql-8.4.3 ndb-8.4.3)
id=58      @10.233.114.34  (mysql-8.4.3 ndb-8.4.3)
id=59      @10.233.85.91  (mysql-8.4.3 ndb-8.4.3)
id=70      @10.233.127.117 (mysql-8.4.3 ndb-8.4.3)
id=71      @10.233.85.93  (mysql-8.4.3 ndb-8.4.3)
id=222 (not connected, accepting connect from any host)
id=223 (not connected, accepting connect from any host)
id=224 (not connected, accepting connect from any host)
id=225 (not connected, accepting connect from any host)

```

Note

Node IDs 222 to 225 in the sample output are shown as "not connected" as these are added as empty slot IDs that are used for georeplication recovery. You can ignore these node IDs.

6. Install cnDBTier cluster3 by configuring cnDBTier cluster1 as the first mate site and cnDBTier cluster2 as the second mate site with at least four SQL pods. For information on installing the cnDBTier cluster, see *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide*.

Note

- If multiple replication channel groups are enabled, then install cnDBTier cluster3 by configuring cnDBTier cluster1 as first mate site and cnDBTier cluster2 as second mate site with at least eight SQL pods.
- Do not run a Helm test when performing the conversion procedure.
- If you are going to enable encryption, ensure that the new site uses the same encryption key.

7. Check if georeplication channels are established between cnDBTier cluster3 and cnDBTier cluster1, and cnDBTier cluster3 and cnDBTier cluster2 by following the procedures described in the [Checking Georeplication Status Between Clusters](#) section.
8. Restore the data of cnDBTier cluster1 or cnDBTier cluster2 in cnDBTier cluster3 and re-establish the georeplication channels between cnDBTier cluster1 and cnDBTier cluster3, and cnDBTier cluster2 and cnDBTier cluster3. You can perform the georeplication recovery using cnDBTier or CNC Console. For procedures on restoring the cluster data and re-establishing the georeplication channels, refer to the *Georeplication Failure between cnDBTier Clusters in Three Site Replication* section of *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide*.
9. Check if georeplication channels are established between cnDBTier cluster3 and cnDBTier cluster1, and cnDBTier cluster3 and cnDBTier cluster2 by following the procedures described in the [Checking Georeplication Status Between Clusters](#) section.

Note

As data is replicated to cnDBTier cluster3 and georeplication channels are established between cnDBTier cluster1 and cnDBTier cluster3, and cnDBTier cluster2 and cnDBTier cluster3, the newly added cnDBTier cluster (cnDBTier cluster3) can also be used by the NFs for database operations.

7.5.3 Adding cnDBTier Georedundant Cluster to Three-Site cnDBTier Clusters

This section describes the procedure to add a cnDBTier georedundant cnDBTier cluster4 to the existing three-site cnDBTier clusters (cnDBTier cluster1, cnDBTier cluster2, and cnDBTier cluster3).

1. If TLS was configured on cnDBTier cluster 1, cluster 2, and cluster 3, then follow the [Modifying cnDBTier Certificates to Establish TLS Between Georeplication Sites](#) procedure to update the certificates in cnDBTier cluster 1, cluster 2, and cluster 3.

Note

Perform this step on those cnDBTier clusters on which you want to reconfigure the CA certificates for the new cnDBTier cluster.

2. Check if georeplication is enabled in cnDBTier cluster1 with respect to cnDBTier cluster4 by performing the following steps:

Note

The following commands and examples are applicable for a single replication channel group only. If multiple replication channel groups are enabled, then check if georeplication is enabled in cnDBTier cluster1 for every replication group with respect to cnDBTier cluster4.

- a. Ensure that the DB replication service in cnDBTier cluster1 is enabled and running successfully with respect to cnDBTier cluster4:

```
$ kubectl -n cluster1 get pods | grep repl
```

Sample output:

```
mysql-cluster-cluster1-cluster2-replication-svc-5bdc46crzzhb 1/1 Running
0 3m11s
mysql-cluster-cluster1-cluster3-replication-svc-7955b6b6fbd5 1/1 Running
2 3m39s
mysql-cluster-cluster1-cluster4-replication-svc-bd977fc5bnd5 1/1 Running
2 3m11s
```

- b. Ensure that at least six georeplication SQL nodes are configured in cluster1:

```
$ kubectl -n cluster1 get pods | grep ndbmysqld
```

Sample output:

```
ndbmysqld-0 3/3
Running 0 4m14s
ndbmysqld-1 3/3
Running 0 3m53s
ndbmysqld-2 3/3
Running 0 3m21s
ndbmysqld-3 3/3
Running 0 3m01s
ndbmysqld-4 3/3
Running 0 2m48s
ndbmysqld-5 3/3
Running 0 2m31s
```

- c. Check the status of cnDBTier cluster1:

```
$ kubectl -n cluster1 exec -it ndbmgmd-0 -- ndb_mgm -e show
```

Sample output:

Connected to Management Server at: localhost:1186 Cluster Configuration

```
-----
[ndbd(NDB)] 2 node(s)
id=1 @10.233.85.92 (mysql-8.4.3 ndb-8.4.3, Nodegroup: 0)
id=2 @10.233.114.33 (mysql-8.4.3 ndb-8.4.3, Nodegroup: 0, *)

[ndb_mgmd(MGM)] 2 node(s)
id=49 @10.233.65.167 (mysql-8.4.3 ndb-8.4.3)
id=50 @10.233.127.115 (mysql-8.4.3 ndb-8.4.3)

[mysqld(API)] 12 node(s)
id=56 @10.233.114.34 (mysql-8.4.3 ndb-8.4.3)
id=57 @10.233.85.91 (mysql-8.4.3 ndb-8.4.3)
id=58 @10.233.114.34 (mysql-8.4.3 ndb-8.4.3)
id=59 @10.233.85.91 (mysql-8.4.3 ndb-8.4.3)
id=60 @10.233.62.226 (mysql-8.4.3 ndb-8.4.3)
id=61 @10.233.14.15 (mysql-8.4.3 ndb-8.4.3)
id=70 @10.233.127.117 (mysql-8.4.3 ndb-8.4.3)
id=71 @10.233.85.93 (mysql-8.4.3 ndb-8.4.3)
id=222 (not connected, accepting connect from any host)
id=223 (not connected, accepting connect from any host)
id=224 (not connected, accepting connect from any host)
id=225 (not connected, accepting connect from any host)
```

Note

- If cluster1 satisfies all the conditions in Step 1, then georeplication is enabled in cnDBTier cluster1. Otherwise, georeplication is not enabled.
- The node IDs 222 to 225 in the sample output are shown as "not connected" as these nodes are added as empty slot IDs used for georeplication recovery. You can ignore these nodes. This note is applicable to all similar outputs in this procedure.

3. If georeplication is not enabled in cluster1 as per [Step 1](#), then perform the following steps to enable georeplication in cnDBTier cluster1 with respect to cnDBTier cluster4. Else, skip to [Step 3](#).

Note

The following commands and examples are applicable for a single replication channel group only. If multiple replication channel groups are enabled, then configure remote mate site name (cluster4) in cnDBTier cluster1 by following the *Configuring Multiple Replication channel groups* procedure in Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide and skip steps a and b in the following substeps.

- a. Configure the remote mate site name (cluster4) in cnDBTier cluster1 by enabling and configuring the replication service using the custom_values.yaml file:

```
$ vi ocndbtier/custom_values.yaml
```

Sample output:

```

dbreplsvcdeployments:
  ....
  # if pod prefix is given then use the unique smaller name for this db
  replication service.
  - name: cluster1-cluster4-replication-svc
    # Local site replication service LoadBalancer ip can be configured.
    enabled: true
    mysql:
      dbtierservice: "mysql-connectivity-service"
      dbtierreplservice: "ndbmysqldsvc"
      # if cndbtier, use CLUSTER-IP from the ndbmysqldsvc-4 LoadBalancer
    service
    dbtier" primaryhost: "ndbmysqld-4.ndbmysqldsvc.cluster1.svc.occne4-cgbu-cne-
    port: "3306"
    # if cndbtier, use EXTERNAL-IP from the ndbmysqldsvc-4 LoadBalancer
    service
    primarysignalhost: ""
    # serverid is unique; retrieve it for the site being configured
    primaryhostserverid: "1004"
    # if cndbtier, use CLUSTER-IP from the ndbmysqldsvc-5 LoadBalancer
    service
    secondaryhost: "ndbmysqld-5.ndbmysqldsvc.cluster1.svc.occne4-cgbu-cne-
    dbtier"
    # if cndbtier, use EXTERNAL-IP from the ndbmysqldsvc-5 LoadBalancer
    service
    secondarysignalhost: ""
    # serverid is unique; retrieve it for the site being configured
    secondaryhostserverid: "1005"
    replication:
      localsiteip: ""
      matesitename: "cluster4"
      remotesiteip: ""

```

- b. Configure the number of SQL pods in cnDBTier cluster1 to at least six in the `custom_values.yaml`:

Note

`apiReplicaCount` is set to "6" as you are adding the third site. The first four `ndbmysqld` are used for replication between site one, site two, and site three only. The two `ndbmysqld` pods that are newly added will be responsible for replication between site1 and site4.

```
$ vi occndbtier/custom_values.yaml
```

Sample output:

```
apiReplicaCount: 6
```

- c. Upgrade cnDBTier cluster1 by using the CSAR package:

Note

The following command and example are applicable for a single replication channel group only. If multiple replication channel groups are enabled, then upgrade cnDBTier cluster1 by providing appropriate multi-group values file.

```
$ helm upgrade mysql-cluster occndbtier --namespace cluster1 --no-hooks
-f occndbtier/custom_values.yaml
```

Sample output:

```
Release "mysql-cluster" has been upgraded. Happy Helming!
NAME: mysql-cluster
LAST DEPLOYED: Mon May 20 11:10:32 2025
NAMESPACE: cluster1
STATUS: deployed
REVISION: 2
```

- d. Wait until the upgrade is complete and all the pods of cnDBTier cluster 1 are restarted. Verify the status of the cluster by performing the following steps:

- i. Check the status of pods running cluster1:

```
$ kubectl get pods --namespace=cluster1
```

Sample output:

NAME	READY	STATUS	RESTARTS	AGE
mysql-cluster-cluster1-cluster2-replication-svc-5bdc46crzzhb	1/1	Running	0	153m
mysql-cluster-cluster1-cluster3-replication-svc-7955b6b6fbd	1/1	Running	0	153m
mysql-cluster-cluster1-cluster4-replication-svc-bd977fc5bnd5	1/1	Running	2	3m11s
mysql-cluster-db-backup-manager-svc-c77df67b7-tqnd2	1/1	Running	0	153m
mysql-cluster-db-monitor-svc-69ff969477-646tz	1/1	Running	0	147m
ndbappmysqld-0	2/2	Running	0	148m
ndbappmysqld-1	2/2	Running	0	147m
ndbmgmd-0	1/1	Running	0	152m
ndbmgmd-1	1/1	Running	0	152m
ndbmt-d-0	2/2	Running	0	150m
ndbmt-d-1	2/2	Running	0	151m
ndbmysqld-0	2/2	Running	0	147m
ndbmysqld-1	2/2	Running	0	147m

```

ndbmysqld-2
2/2      Running    0              148m
ndbmysqld-3
2/2      Running    0              149m
ndbmysqld-4
2/2      Running    0              147m
ndbmysqld-5
2/2      Running    0              147m

```

ii. Check the status of cnDBTier cluster1:

```
$ kubectl -n cluster1 exec -it ndbmcmd-0 -- ndb_mgm -e show
```

Sample output:

```

Connected to Management Server at: localhost:1186 Cluster
Configuration
-----
[ndbd(NDB)]      2 node(s)
id=1   @10.233.85.92  (mysql-8.4.3 ndb-8.4.3, Nodegroup: 0)
id=2   @10.233.114.33 (mysql- ndb-8.4.3, Nodegroup: 0, *)

[ndb_mgmd(MGM)]  2 node(s)
id=49  @10.233.65.167 (mysql-8.4.3 ndb-8.4.3)
id=50  @10.233.127.115 (mysql-8.4.3 ndb-8.4.3)

[mysqld(API)]    12 node(s)
id=56  @10.233.124.92  (mysql-8.4.3 ndb-8.4.3)
id=57  @10.233.114.135 (mysql-8.4.3 ndb-8.4.3)
id=58  @10.233.114.34  (mysql-8.4.3 ndb-8.4.3)
id=59  @10.233.85.91   (mysql-8.4.3 ndb-8.4.3)
id=60  @10.233.15.24   (mysql-8.4.3 ndb-8.4.3)
id=61  @10.233.48.74   (mysql-8.4.3 ndb-8.4.3)
id=70  @10.233.127.117 (mysql-8.4.3 ndb-8.4.3)
id=71  @10.233.85.93   (mysql-8.4.3 ndb-8.4.3)
id=222 (not connected, accepting connect from any host)
id=223 (not connected, accepting connect from any host)
id=224 (not connected, accepting connect from any host)
id=225 (not connected, accepting connect from any host)

```

Note

Node IDs 222 to 225 in the sample output are shown as "not connected" as these are added as empty slot IDs that are used for georeplication recovery. You can ignore these node IDs.

4. Check if georeplication is enabled in cnDBTier cluster2 with respect to cnDBTier cluster4 by performing the following steps:

Note

The following commands and examples are applicable for a single replication channel group only. If multiple replication channel groups are enabled, then check if georeplication is enabled in cnDBTier cluster2 for every replication group with respect to cnDBTier cluster4.

- a. Ensure that the DB replication service in cnDBTier cluster2 is enabled and running successfully with respect to cnDBTier cluster4:

```
$ kubectl -n cluster2 get pods | grep repl
```

Sample output:

```
mysql-cluster-cluster2-cluster1-replication-svc-5bdc46crzzhb 1/1 Running
0 3m11s
mysql-cluster-cluster2-cluster3-replication-svc-7955b6b6fbdcc 1/1 Running
2 3m39s
mysql-cluster-cluster2-cluster4-replication-svc--556c99fcdzh 1/1 Running
2 3m05s
```

- b. Ensure that at least six georeplication SQL nodes are configured in cluster2:

Note

apiReplicaCount is set to "6" as you are adding the third site. The first four ndbmysqld are used for replication between site one, site two, and site three only. The two ndbmysqld pods that are newly added will be responsible for replication between site1 and site4.

```
$ kubectl -n cluster2 get pods | grep ndbmysqld
```

Sample output:

```
ndbmysqld-0 3/3
Running 0 4m14s
ndbmysqld-1 3/3
Running 0 3m53s
ndbmysqld-2 3/3
Running 0 3m21s
ndbmysqld-3 3/3
Running 0 3m01s
ndbmysqld-4 3/3
Running 0 2m59s
ndbmysqld-5 3/3
Running 0 2m43s
```

- c. Check the status of cnDBTier cluster2:

```
$ kubectl -n cluster2 exec -it ndbmgmd-0 -- ndb_mgm -e show
```

Sample output:

```
Connected to Management Server at: localhost:1186 Cluster Configuration
-----
```

```

[ndbd(NDB)]      2 node(s)
id=1    @10.233.85.92  (mysql-8.4.3 ndb-8.4.3, Nodegroup: 0)
id=2    @10.233.114.33 (mysql-8.4.3 ndb-8.4.3, Nodegroup: 0, *)

[ndb_mgmd(MGM)]  2 node(s)
id=49   @10.233.65.167 (mysql-8.4.3 ndb-8.4.3)
id=50   @10.233.127.115 (mysql-8.4.3 ndb-8.4.3)

[mysqld(API)]    12 node(s)
id=56   @10.233.114.34  (mysql-8.4.3 ndb-8.4.3)
id=57   @10.233.85.91   (mysql-8.4.3 ndb-8.4.3)
id=58   @10.233.114.34  (mysql-8.4.3 ndb-8.4.3)
id=59   @10.233.85.91   (mysql-8.4.3 ndb-8.4.3)
id=60   @10.233.24.121  (mysql-8.4.3 ndb-8.4.3)
id=61   @10.233.85.22   (mysql-8.4.3 ndb-8.4.3)
id=70   @10.233.127.117 (mysql-8.4.3 ndb-8.4.3)
id=71   @10.233.85.93   (mysql-8.4.3 ndb-8.4.3)
id=222  (not connected, accepting connect from any host)
id=223  (not connected, accepting connect from any host)
id=224  (not connected, accepting connect from any host)
id=225  (not connected, accepting connect from any host)

```

Note

- The node IDs 222 to 225 in the sample output are shown as "not connected" as these nodes are added as empty slot IDs used for georeplication recovery. You can ignore these nodes.
- If cluster2 satisfies all the conditions in Step 3, then georeplication is enabled in cnDBTier cluster2. Otherwise, georeplication is not enabled in cluster2.

5. If georeplication is not enabled in cluster2 as per [Step 3](#), then perform the following steps to enable georeplication in cluster2 with respect to cluster4. Else, skip to Step 5.

Note

The following commands and examples are applicable for a single replication channel group only. If multiple replication channel groups are enabled, then configure remote mate site name (cluster4) in cnDBTier cluster2 by following the *Configuring Multiple Replication channel groups* procedure in *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide* and skip steps a and b in the following substeps.

- a. Configure the remote mate site name (cluster4) in cnDBTier cluster2 by enabling and configuring the replication service using the `custom_values.yaml` file:

```
$ vi occndbtier/custom_values.yaml
```

Sample output:

```

dbreplsvcdeployments:
  ....
  # if pod prefix is given then use the unique smaller name for this db
  replication service.
  - name: cluster2-cluster4-replication-svc
    enabled: true

```

```

mysql:
  dbtierservice: "mysql-connectivity-service"
  dbtierreplservice: "ndbmysqldsvc"
  # if cndbtier, use CLUSTER-IP from the ndbmysqldsvc-4 LoadBalancer
service
primaryhost: "ndbmysqld-4.ndbmysqldsvc.cluster2.svc.occne4-cgbu-cne-
dbtier"
port: "3306"
# if cndbtier, use EXTERNAL-IP from the ndbmysqldsvc-4 LoadBalancer
service
primarysignalhost: ""
# serverid is unique; retrieve it for the site being configured
primaryhostserverid: "2004"
# if cndbtier, use CLUSTER-IP from the ndbmysqldsvc-5 LoadBalancer
service
secondaryhost: "ndbmysqld-5.ndbmysqldsvc.cluster2.svc.occne4-cgbu-cne-
dbtier"
# if cndbtier, use EXTERNAL-IP from the ndbmysqldsvc-5 LoadBalancer
service
secondarysignalhost: ""
# serverid is unique; retrieve it for the site being configured
secondaryhostserverid: "2005"
replication:
# Local site replication service LoadBalancer ip can be configured.
localsiteip: ""
matesitename: "cluster4"
remotesiteip: ""

```

- b. Configure the number of SQL pods in cnDBTier cluster2 to at least six in the `custom_values.yaml` file:

Note

`apiReplicaCount` is set to "6" as you are adding the fourth site. The first four `ndbmysqld` are used for replication between site one, site two, and site three only. The two `ndbmysqld` pods that are newly added will be responsible for replication between site1 and site4.

```
$ vi occndbtier/custom_values.yaml
```

Sample output:

```
apiReplicaCount: 6
```

- c. Upgrade cnDBTier cluster2 by using the CSAR package. For more information, see *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide*.

Note

The following commands and examples are applicable for a single replication channel group only. If multiple replication channel groups are enabled, then upgrade cnDBTier cluster2 by providing appropriate multi-group values file.

```
$ helm upgrade mysql-cluster occndbtier --namespace cluster2 --no-hooks
-f occndbtier/custom_values.yaml
```

Sample output:

```
Release "mysql-cluster" has been upgraded. Happy Helming!
NAME: mysql-cluster
LAST DEPLOYED: Mon May 20 11:10:32 2025
NAMESPACE: cluster2
STATUS: deployed
REVISION: 2
```

- d. Wait until the upgrade is complete and all the pods of cnDBTier cluster 2 are restarted. Verify the status of the cluster by performing the following steps:

- i. Check the status of pods running cluster2:

```
$ kubectl get pods --namespace=cluster2
```

Sample output:

NAME	READY	STATUS	RESTARTS	AGE
mysql-cluster-cluster2-cluster1-replication-svc-5bdc46cryyaa	1/1	Running	0	153m
mysql-cluster-cluster2-cluster3-replication-svc-7955b6b6ghht	1/1	Running	0	153m
mysql-cluster-cluster2-cluster4-replication-svc-c574d6cj6mlp	1/1	Running	2	153m
mysql-cluster-db-backup-manager-svc-c77df67b7-tqnd2	1/1	Running	0	153m
mysql-cluster-db-monitor-svc-69ff969477-646tz	1/1	Running	0	147m
ndbappmysqld-0	2/2	Running	0	148m
ndbappmysqld-1	2/2	Running	0	147m
ndbmgmd-0	1/1	Running	0	152m
ndbmgmd-1	1/1	Running	0	152m
ndbmt-d-0	2/2	Running	0	150m
ndbmt-d-1	2/2	Running	0	151m
ndbmysqld-0	2/2	Running	0	147m
ndbmysqld-1	2/2	Running	0	147m

```

ndbmysqld-2
2/2      Running    0              148m
ndbmysqld-3
2/2      Running    0              149m
ndbmysqld-4
2/2      Running    0              147m
ndbmysqld-5
2/2      Running    0              147m

```

ii. Check the status of cnDBTier cluster2:

```
$ kubectl -n cluster2 exec -it ndbmcmd-0 -- ndb_mgm -e show
```

Sample output:

```

Connected to Management Server at: localhost:1186 Cluster
Configuration
-----
[ndbd(NDB)]      2 node(s)
id=1   @10.233.85.92  (mysql-8.4.3 ndb-8.4.3, Nodegroup: 0)
id=2   @10.233.114.33 (mysql-8.4.3 ndb-8.4.3, Nodegroup: 0, *)

[ndb_mgmd(MGM)]  2 node(s)
id=49  @10.233.65.167 (mysql-8.4.3 ndb-8.4.3)
id=50  @10.233.127.115 (mysql-8.4.3 ndb-8.4.3)

[mysqld(API)]    12 node(s)
id=56  @10.233.124.92  (mysql-8.4.3 ndb-8.4.3)
id=57  @10.233.114.135 (mysql-8.4.3 ndb-8.4.3)
id=58  @10.233.124.111 (mysql-8.4.3 ndb-8.4.3)
id=59  @10.233.110.81  (mysql-8.4.3 ndb-8.4.3)
id=60  @10.233.110.41  (mysql-8.4.3 ndb-8.4.3)
id=61  @10.233.98.102  (mysql-8.4.3 ndb-8.4.3)
id=70  @10.233.127.117 (mysql-8.4.3 ndb-8.4.3)
id=71  @10.233.85.93   (mysql-8.4.3 ndb-8.4.3)
id=222 (not connected, accepting connect from any host)
id=223 (not connected, accepting connect from any host)
id=224 (not connected, accepting connect from any host)
id=225 (not connected, accepting connect from any host)

```

Note

Node IDs 222 to 225 in the sample output are shown as "not connected" as these are added as empty slot IDs that are used for georeplication recovery. You can ignore these node IDs.

6. Check if georeplication is enabled in cnDBTier cluster3 with respect to cnDBTier cluster4 by performing the following steps:

Note

The following commands and examples are applicable for a single replication channel group only. If multiple replication channel groups are enabled, then check if georeplication is enabled in cnDBTier cluster3 for every replication group with respect to cnDBTier cluster4.

- a. Ensure that the DB replication service in cnDBTier cluster3 is enabled and running successfully with respect to cnDBTier cluster4:

```
$ kubectl -n cluster3 get pods | grep repl
```

Sample output:

```
mysql-cluster-cluster3-cluster1-replication-svc-5bdc46crzzhb 1/1 Running
0 3m11s
mysql-cluster-cluster3-cluster2-replication-svc-7955b6b6fbdcc 1/1 Running
2 3m39s
mysql-cluster-cluster3-cluster4-replication-svc--556c99fcdzh 1/1 Running
2 3m05s
```

- b. Ensure that at least six georeplication SQL nodes are configured in cluster3:

```
$ kubectl -n cluster3 get pods | grep ndbmysqld
```

Sample output:

```
ndbmysqld-0 3/3
Running 0 4m14s
ndbmysqld-1 3/3
Running 0 3m53s
ndbmysqld-2 3/3
Running 0 3m21s
ndbmysqld-3 3/3
Running 0 3m01s
ndbmysqld-4 3/3
Running 0 2m59s
ndbmysqld-5 3/3
Running 0 2m43s
```

- c. Check the status of cnDBTier cluster3:

```
$ kubectl -n cluster3 exec -it ndbmgsmd-0 -- ndb_mgm -e show
```

Sample output:

```
Connected to Management Server at: localhost:1186 Cluster Configuration
-----
[ndbd(NDB)] 2 node(s)
id=1 @10.233.85.92 (mysql-8.4.3 ndb-8.4.3, Nodegroup: 0)
id=2 @10.233.114.33 (mysql-8.4.3 ndb-8.4.3, Nodegroup: 0, *)

[ndb_mgmd(MGM)] 2 node(s)
id=49 @10.233.65.167 (mysql-8.4.3 ndb-8.4.3)
id=50 @10.233.127.115 (mysql-8.4.3 ndb-8.4.3)

[mysqld(API)] 12 node(s)
id=56 @10.233.71.24 (mysql-8.4.3 ndb-8.4.3)
```



```

id=57 @10.233.119.18 (mysql-8.4.3 ndb-8.4.3)
id=58 @10.233.69.23 (mysql-8.4.3 ndb-8.4.3)
id=59 @10.233.116.21 (mysql-8.4.3 ndb-8.4.3)
id=60 @10.233.70.16 (mysql-8.4.3 ndb-8.4.3)
id=61 @10.233.78.20 (mysql-8.4.3 ndb-8.4.3)
id=70 @10.233.127.117 (mysql-8.4.3 ndb-8.4.3)
id=71 @10.233.85.93 (mysql-8.4.3 ndb-8.4.3)
id=222 (not connected, accepting connect from any host)
id=223 (not connected, accepting connect from any host)
id=224 (not connected, accepting connect from any host)
id=225 (not connected, accepting connect from any host)

```

Note

- The node IDs 222 to 225 in the sample output are shown as "not connected" as these nodes are added as empty slot IDs used for georeplication recovery. You can ignore these nodes.
- If cluster3 satisfies all the conditions in Step 5, then georeplication is enabled in cnDBTier cluster3. Otherwise, georeplication is not enabled in cluster3.

7. If georeplication is not enabled in cluster3 as per [Step 5](#), then perform the following steps to enable georeplication in cluster3 with respect to cluster4. Else, skip to [Step 7](#).

Note

The following commands and examples are applicable for a single replication channel group only. If multiple replication channel groups are enabled, then configure remote mate site name (cluster4) in cnDBTier cluster3 by following the *Configuring Multiple Replication channel groups* procedure in *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide* and skip steps a and b in the following substeps.

- a. Configure the remote mate site name (cluster4) in cnDBTier cluster3 by enabling and configuring the replication service using the `custom_values.yaml` file:

```
$ vi occndbtier/custom_values.yaml
```

Sample output:

```

dbreplsvcdeployments:
  ....
  # if pod prefix is given then use the unique smaller name for this db
  replication service.
  - name: cndbtiercluster3-cndbtiercluster4-replication-svc
    enabled: true
    mysql:
      dbtierservice: "mysql-connectivity-service"
      dbtierreplservice: "ndbmysqldsvc"
      # if cndbtier, use CLUSTER-IP from the ndbmysqldsvc-4 LoadBalancer
    service
    primaryhost: "ndbmysqld-4.ndbmysqldsvc.cluster3.svc.occne4-cgbu-cne-
    dbtier"
    port: "3306"
    # if cndbtier, use EXTERNAL-IP from the ndbmysqldsvc-4 LoadBalancer

```

```

service
  primarysignalhost: ""
  # serverid is unique; retrieve it for the site being configured
  primaryhostserverid: "3004"
  # if cndbtier, use CLUSTER-IP from the ndbmysqldsvc-5 LoadBalancer
service
  secondaryhost: "ndbmysqld-5.ndbmysqldsvc.cluster3.svc.occne4-cgbu-cne-
dbtier"
  # if cndbtier, use EXTERNAL-IP from the ndbmysqldsvc-5 LoadBalancer
service
  secondarysignalhost: ""
  # serverid is unique; retrieve it for the site being configured
  secondaryhostserverid: "3005"
  replication:
    # Local site replication service LoadBalancer ip can be configured.
    localsiteip: ""
    matesitename: "cluster4"
    remotesiteip: ""

```

- b. Configure the number of SQL pods in cnDBTier cluster3 to at least six in the `custom_values.yaml` file:

Note

`apiReplicaCount` is set to "6" as you are adding the third site. The first four `ndbmysqld` are used for replication between site one, site two, and site three only. The two `ndbmysqld` pods that are newly added will be responsible for replication between site1 and site4.

```
$ vi occndbtier/custom_values.yaml
```

Sample output:

```
apiReplicaCount: 6
```

- c. Upgrade cnDBTier cluster3 by using the CSAR package. For more information, see *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide*.

Note

The following commands and examples are applicable for a single replication channel group only. If multiple replication channel groups are enabled, then upgrade cnDBTier cluster3 by providing appropriate multi group values file.

```
$ helm upgrade mysql-cluster occndbtier --namespace cluster3 --no-hooks
-f occndbtier/custom_values.yaml
```

Sample output:

```

Release "mysql-cluster" has been upgraded. Happy Helming!
NAME: mysql-cluster
LAST DEPLOYED: Mon May 20 11:10:32 2025
NAMESPACE: cluster3

```

STATUS: deployed
REVISION: 2

- d. Wait until the upgrade is complete and all the pods of cnDBTier cluster 3 are restarted. Verify the status of the cluster by performing the following steps:

- i. Check the status of pods running cluster3:

```
$ kubectl get pods --namespace=cluster3
```

Sample output:

```
mysql-cluster-cluster3-cluster1-replication-svc-5bdc46dsttic
1/1      Running    0          153m
mysql-cluster-cluster3-cluster2-replication-svc-7955b6c7eced
1/1      Running    0          153m
mysql-cluster-cluster3-cluster4-replication-svc-556c99fdeyah
1/1      Running    2          153m
mysql-cluster-db-backup-manager-svc-8bf9448b8-w8pvf
1/1      Running    0          153m
mysql-cluster-db-monitor-svc-8bf9559b8-v8qwg
1/1      Running    0          147m
ndbappmysqld-0
2/2      Running    0          148m
ndbappmysqld-1
2/2      Running    0          147m
ndbmgmd-0
1/1      Running    0          152m
ndbmgmd-1
1/1      Running    0          152m
ndbmt-d-0
2/2      Running    0          150m
ndbmt-d-1
2/2      Running    0          151m
ndbmysqld-0
2/2      Running    0          147m
ndbmysqld-1
2/2      Running    0          147m
ndbmysqld-2
2/2      Running    0          148m
ndbmysqld-3
2/2      Running    0          149m
ndbmysqld-4
2/2      Running    0          147m
ndbmysqld-5
2/2      Running    0          147m
```

- ii. Check the status of cnDBTier cluster3:

```
$ kubectl -n cluster3 exec -it ndbmgmd-0 -- ndb_mgm -e show
```

Sample output:

```
Connected to Management Server at: localhost:1186 Cluster
Configuration
-----
```

```

[ndbd(NDB)]      2 node(s)
id=1      @10.233.85.92 (mysql-8.4.3 ndb-8.4.3, Nodegroup: 0)
id=2      @10.233.114.33 (mysql-8.4.3 ndb-8.4.3, Nodegroup: 0, *)

[ndb_mgmd(MGM)]  2 node(s)
id=49     @10.233.65.167 (mysql-8.4.3 ndb-8.4.3)
id=50     @10.233.127.115 (mysql-8.4.3 ndb-8.4.3)

[mysqld(API)]    12 node(s)
id=56     @10.233.71.24   (mysql-8.4.3 ndb-8.4.3)
id=57     @10.233.119.18  (mysql-8.4.3 ndb-8.4.3)
id=58     @10.233.69.23   (mysql-8.4.3 ndb-8.4.3)
id=59     @10.233.116.21  (mysql-8.4.3 ndb-8.4.3)
id=60     @10.233.70.16   (mysql-8.4.3 ndb-8.4.3)
id=61     @10.233.78.20   (mysql-8.4.3 ndb-8.4.3)
id=70     @10.233.127.117 (mysql-8.4.3 ndb-8.4.3)
id=71     @10.233.85.93   (mysql-8.4.3 ndb-8.4.3)
id=222    (not connected, accepting connect from any host)
id=223    (not connected, accepting connect from any host)
id=224    (not connected, accepting connect from any host)
id=225    (not connected, accepting connect from any host)

```

Note

Node IDs 222 to 225 in the sample output are shown as "not connected" as these are added as empty slot IDs that are used for georeplication recovery. You can ignore these node IDs.

8. Install cnDBTier cluster4 by configuring cnDBTier cluster1 as the first mate site and cnDBTier cluster2 as the second mate site with at least six SQL pods. For information on installing the cnDBTier cluster, see *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide*.

Note

- The following commands and examples are applicable for a single replication channel group only. If multiple replication channel groups are enabled, then install cnDBTier cluster4 by configuring cnDBTier cluster1 as first mate site, cnDBTier cluster2 as second mate site and cnDBTier cluster3 as third mate site with at least twelve SQL pods.
- Do not run a Helm test when you are performing a conversion procedure.
- If you are going to enable encryption, ensure that the new site uses the same encryption key.

9. Check if georeplication channels are established between cnDBTier cluster1 and cnDBTier cluster4, cnDBTier cluster2 and cnDBTier cluster4, and cnDBTier cluster3 and cnDBTier cluster4 by following the procedures described in the [Checking Georeplication Status Between Clusters](#) section.
10. Restore the data of cnDBTier cluster1, cnDBTier cluster2, or cnDBTier cluster3 in cnDBTier cluster4 and re-establish the georeplication channels between cnDBTier cluster1 and cnDBTier cluster4, cnDBTier cluster2 and cnDBTier cluster4, and cnDBTier cluster3 and cnDBTier cluster4. You can perform the georeplication recovery using cnDBTier or CNC

Console. For procedures on restoring the cluster data and re-establishing the georeplication channels, refer to the *Georeplication Failure between cnDBTier Clusters in Four Site Replication* section of *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide*.

11. Check if georeplication channels are established between cnDBTier cluster1 and cnDBTier cluster4, cnDBTier cluster2 and cnDBTier cluster4, and cnDBTier cluster3 and cnDBTier cluster4 by following the procedures described in the [Checking Georeplication Status Between Clusters](#) section.

Note

As data is replicated to cnDBTier cluster4 and georeplication channels are established between cnDBTier cluster1 and cnDBTier cluster4, cnDBTier cluster2 and cnDBTier cluster4, and cnDBTier cluster3 and cnDBTier cluster4, the newly added cnDBTier cluster (cnDBTier cluster4) can be used by the NFs for database operations.

7.6 Removing a Georedundant cnDBTier Cluster

This section describes the procedures to remove a georedundant cluster from an existing multisite setup (two-site, three-site, or four-site) using the `dbtremovesite` script.

The following terminologies are used in the procedures to refer to a cluster in a multisite:

- cnDBTier Cluster1 (cluster1): The first cloud native cnDBTier cluster in a two-site, three-site, or four-site georeplication setup.
- cnDBTier Cluster2 (cluster2): The second cloud native cnDBTier cluster in a two-site, three-site, or four-site georeplication setup.
- cnDBTier Cluster3 (cluster3): The third cloud native cnDBTier cluster in a two-site, three-site, or four-site georeplication setup.
- cnDBTier Cluster4 (cluster4): The fourth cloud native cnDBTier cluster in a two-site, three-site, or four-site georeplication setup.

Prerequisites

Before performing the following procedures, ensure that you meet the following prerequisites:

- Multiple cnDBTier clusters must be installed.
- The cnDBTier cluster that needs to be removed must be a part of the multisite setup.

Note

- To perform these procedures, georeplication may or may not be established correctly between multiple cnDBTier clusters.
- Ensure that there is no traffic in the cnDBTier cluster that is being removed from the multisite setup.

7.6.1 Extracting cnDBTier Cluster Script

This section describes the procedure to extract the cnDBTier cluster script.

1. Extract CSAR package:

```
$ unzip occndbtier_csar_vzw_25_2_100_0_0.zip
```

2. Run the source_me file:

```
$ cd Artifacts/Scripts/tools
$ source ./source_me
```

Sample output:

NOTE: source_me must be sourced while your current directory is the directory with the source_me file.

```
Enter cndbtier namespace: cluster1
DBTIER_NAMESPACE = "cluster1"
```

```
DBTIER_LIB=/home/cgbu-cne-dbtier/csar/25.2.100/Artifacts/Scripts/tools/lib
```

```
Adding /home/cgbu-cne-dbtier/csar/25.2.100/Artifacts/Scripts/tools/bin to
PATH
PATH=/home/cgbu-cne-dbtier/csar/25.2.100/Artifacts/Scripts/tools/bin:/home/
cgbu-cne-dbtier/.local/bin:/home/cgbu-cne-dbtier/bin:/usr/local/bin:/usr/
bin:/usr/local/sbin:/usr/sbin:/var/occne/cluster/thrust7a/artifacts/
istio-1.15.3/bin:/var/occne/cluster/thrust7a/artifacts:/home/cgbu-cne-
dbtier/luis/bin:/home/cgbu-cne-dbtier/luis/usr/bin
```

3. Check if the dbtremovesite script is present in the lib and bin directory:

```
$ cd Artifacts/Scripts/tools/bin
$ ls -lrth dbtremovesite
```

Sample output:

```
total 104K
-rwx-----. 1 cloud-user cloud-user 120 Apr 23 05:46 dbtremovesite
```

Usage of dbtremovesite

The dbtremovesite script does not take any arguments for removing the cnDBTier cluster details from the other available cnDBTier clusters. You must perform the following procedures to remove the cluster.

1. Uninstall cnDBTier cluster in a site.
2. Run the dbtremovesite script to remove the cnDBTier cluster details in each of the other cnDBTier clusters.
3. Remove the loadbalancer ip address configured in the other cnDBTier clusters and upgrade the cnDBTier clusters.

7.6.2 Removing cnDBTier Cluster 4

This section provides the procedure to remove cnDBTier cluster 4 from a georeplication multisite setup.

Note

The cluster and namespace names used in this procedure may vary depending on your namespace and cluster name. Ensure that you replace the values as per your setup.

1. Perform the following steps to uninstall cnDBTier cluster 4 in a site. For more information, see the "Uninstalling cnDBTier" section in *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide*.

- a. Run the following commands to uninstall cnDBTier cluster 4:

```
$ helm uninstall mysql-cluster -n cluster4
```

- b. Run the following commands to delete the PVCs of cnDBTier cluster 4:

```
$ kubectl -n cluster4 get pvc
$ kubectl -n cluster4 delete pvc <PVC Names>
```

where, <PVC Names> is the PVC names of cnDBTier Cluster 4.

2. Perform the following steps to delete the records related to cluster 4 from all the other available sites (cluster 1, cluster 2, and cluster 3):

- a. Log in to cnDBTier cluster 1 and delete the records related to cluster 4 using the dbtremovesite script:

Note

source_me must be sourced while your current directory is the directory with the source_me file.

```
$ source ./source_me
NOTE: source_me must be sourced while your current directory is the
directory with the source_me file.
```

```
Enter cndbtier namespace: cluster1
```

```
$ dbtremovesite
```

When the dbtremovesite script prompts you to select the cluster that needs to be removed, select cluster 4 by typing *yes* against the cluster 4 option. Type *no* for the rest of the cluster options.

Sample output:

```
Does cluster1 need to be removed (yes/no/exit)? no
Does cluster2 need to be removed (yes/no/exit)? no
Does cluster3 need to be removed (yes/no/exit)? no
Does cluster4 need to be removed (yes/no/exit)? yes
2024-04-23T05:47:00Z INFO - CURRENT_SITE_NAME = cluster1
2024-04-23T05:47:00Z INFO - CURRENT_SITE_ID = 1
2024-04-23T05:47:00Z INFO - Removing cluster4 (Site 4)
```

```

2024-04-23T05:47:00Z INFO - Make sure that cnDBTier cluster(cluster4)
is uninstalled in Site 4
DO YOU WANT TO PROCEED (yes/no/exit)? yes
ARE YOU SURE (yes/no/exit)? yes
-----
-----
-----
2024-04-23T05:47:45Z INFO - Ended timer PHASE 6: 1713851265
2024-04-23T05:47:45Z INFO - PHASE 6 took: 00 hr. 00 min. 02 sec.
2024-04-23T05:47:45Z INFO - Ended timer dbtremovesite: 1713851265
2024-04-23T05:47:45Z INFO - dbtremovesite took: 00 hr. 00 min. 57 sec.
2024-04-23T05:47:45Z INFO - dbtremovesite completed successfully

```

- b. Repeat step a on cluster 2 to delete the records related to cluster 4.
- c. Repeat step a on cluster 3 to delete the records related to cluster 4.
3. Remove the remote site IP address configurations related to cluster 4 and restart all the DB replication service deployments on all the other available sites (cluster 1, cluster 2, and cluster 3):
 - a. Perform the following steps to remove the remote site IP address configurations related to cluster 4 and restart all the DB replication service deployments on cluster 1:
 - i. Log in to the Bastion Host of cluster 1.
 - ii. If fixed IP addresses are not configured to the LoadBalancer services, configure `remotesiteip` with `""` for cluster 4 remote site in the `custom_values.yaml` file of cluster 1:

```
$ vi cndbtier_cluster1_custom_values.yaml
```

```

- name: cluster1-cluster4-replication-svc
  enabled: true
-----
-----
-----

```

```

replication:
  localsiteip: ""
  localsiteport: "80"
  channelgroupid: "1"
  matesitename: "cluster4"
  remotesiteip: ""
  remotesiteport: "80"

```

- iii. If fixed IP addresses are not configured to the LoadBalancer services, upgrade cnDBTier cluster 1:

```
$ helm upgrade mysql-cluster ocndbtier -n cluster1 -f
cndbtier_cluster1_custom_values.yaml
```

- iv. Scale down and scale up all the DB replication service deployments in cluster 1:

```

$ kubectl -n cluster1 scale deploy $(kubectl -n cluster1 get
deployments | awk '{print $1}' | egrep -i 'repl|monitor|backup-
manager') --replicas=0
$ kubectl -n cluster1 scale deploy $(kubectl -n cluster1 get

```



```
deployments | awk '{print $1}' | egrep -i 'repl|monitor|backup-  
manager') --replicas=1
```

- v. Wait until all the pods are up on cluster 1 and verify:

```
$ kubectl -n cluster1 get pods
```

- b. Repeat step a to remove the remote site IP address configurations related to cluster 4 and restart all the DB replication service deployments on cluster 2:

Note

Replace `cluster1` in the commands (in step a) with `cluster2`. However, the values may vary depending on the cluster and namespace names in your setup.

- c. Repeat step a to remove the remote site IP address configurations related to cluster 4 and restart all the DB replication service deployments on cluster 3:

Note

Replace `cluster1` in the commands (in step a) with `cluster3`. However, the values may vary depending on the cluster and namespace names in your setup.

7.6.3 Removing cnDBTier Cluster 3

This section provides the procedure to remove cnDBTier cluster 3 from a georeplication multisite setup.

Note

The cluster and namespace names used in this procedure may vary depending on your namespace and cluster name. Ensure that you replace the values as per your setup.

1. Perform the following steps to uninstall cnDBTier cluster 3 in a site. For more information, see the "Uninstalling cnDBTier" section in *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide*.

- a. Run the following commands to uninstall cnDBTier cluster 3:

```
$ helm uninstall mysql-cluster -n cluster3
```

- b. Run the following commands to delete the PVCs of cnDBTier cluster 3:

```
$ kubectl -n cluster3 get pvc  
$ kubectl -n cluster3 delete pvc <PVC Names>
```

where, <PVC Names> is the PVC names of cnDBTier Cluster 3.

2. Perform the following steps to delete the records related to cluster 3 from all the other sites (cluster 1, cluster 2, and cluster 4):

- a. Log in to cnDBTier cluster 1 and delete the records related to cluster 3 using the dbtremovesite script:

```
$ source ./source_me
```

NOTE: source_me must be sourced while your current directory is the directory with the source_me file.

```
Enter cndbtier namespace: cluster1
```

```
$ dbtremovesite
```

When the dbtremovesite script prompts you to select the cluster that needs to be removed, select cluster 3 by typing *yes* against the cluster 3 option. Type *no* for the rest of the cluster options.

Sample output:

```
Does cluster1 need to be removed (yes/no/exit)? no
Does cluster2 need to be removed (yes/no/exit)? no
Does cluster3 need to be removed (yes/no/exit)? yes
2024-04-23T05:47:00Z INFO - CURRENT_SITE_NAME = cluster1
2024-04-23T05:47:00Z INFO - CURRENT_SITE_ID = 1
2024-04-23T05:47:00Z INFO - Removing cluster3 (Site 3)
2024-04-23T05:47:00Z INFO - Make sure that cnDBTier cluster(cluster3)
is uninstalled in Site 3
DO YOU WANT TO PROCEED (yes/no/exit)? yes
ARE YOU SURE (yes/no/exit)? yes
-----
-----
-----
2024-04-23T05:47:45Z INFO - Ended timer PHASE 6: 1713851265
2024-04-23T05:47:45Z INFO - PHASE 6 took: 00 hr. 00 min. 02 sec.
2024-04-23T05:47:45Z INFO - Ended timer dbtremovesite: 1713851265
2024-04-23T05:47:45Z INFO - dbtremovesite took: 00 hr. 00 min. 57 sec.
2024-04-23T05:47:45Z INFO - dbtremovesite completed successfully
```

- b. Repeat step a on cluster 2 to delete the records related to cluster 3.
- c. Repeat step a on cluster 4 to delete the records related to cluster 3.
3. Remove the remote site IP address configurations related to cluster 3 and restart all the DB replication service deployments on all the other sites (cluster 1, cluster 2, and cluster 4):
 - a. Perform the following steps to remove the remote site IP address configurations related to cluster 3 and restart all the DB replication service deployments on cluster 1:
 - i. Log in to the Bastion Host of cluster 1.

- ii. If fixed IP addresses are not configured to the LoadBalancer services, configure `remotesiteip` with `""` for cluster 3 remote site in the `custom_values.yaml` file of cluster 1:

```
$ vi cndbtier_cluster1_custom_values.yaml

- name: cluster1-cluster3-replication-svc
  enabled: true
  -----
  -----
  -----
  replication:
    localsiteip: ""
    localsiteport: "80"
    channelgroupid: "1"
    matesitename: "cluster3"
    remotesiteip: ""
    remotesiteport: "80"
```

- iii. If fixed IP addresses are not configured to the LoadBalancer services, upgrade cnDBTier cluster 1:

```
$ helm upgrade mysql-cluster occndbtier -n cluster1 -f
cndbtier_cluster1_custom_values.yaml
```

- iv. Scale down and scale up all the DB replication service deployments in cluster 1:

```
$ kubectl -n cluster1 scale deploy $(kubectl -n cluster1 get
deployments -l dbtierapp='dbrepliationsvc' --no-headers | awk
'{print $1}') --replicas=0
```

```
$ kubectl -n cluster1 scale deploy $(kubectl -n cluster1 get
deployments -l dbtierapp='dbmonitorsvc' --no-headers | awk
'{print $1}') --replicas=0
```

```
$ kubectl -n cluster1 scale deploy $(kubectl -n cluster1 get
deployments -l dbtierapp='dbbackupmanagersvc' --no-headers | awk
'{print $1}') --replicas=0
```

```
$ kubectl -n cluster1 scale deploy $(kubectl -n cluster1 get
deployments -l dbtierapp='dbrepliationsvc' --no-headers | awk
'{print $1}') --replicas=1
```

```
$ kubectl -n cluster1 scale deploy $(kubectl -n cluster1 get
deployments -l dbtierapp='dbmonitorsvc' --no-headers | awk
'{print $1}') --replicas=1
```

```
$ kubectl -n cluster1 scale deploy $(kubectl -n cluster1 get
deployments -l dbtierapp='dbbackupmanagersvc' --no-headers | awk
'{print $1}') --replicas=1
```

- v. Wait until all the pods are up on cluster 1 and verify:

```
$ kubectl -n cluster1 get pods
```

- b. Repeat step a to remove the remote site IP address configurations related to cluster 3 and restart all the DB replication service deployments on cluster 2: Perform the following steps to remove the remote site IP address configurations related to cluster 3 and restart all the DB replication service deployments on cluster 1:

Note

Replace `cluster1` in the commands with `cluster3`. However, the values may vary depending on the cluster and namespace names in your setup.

- c. Repeat step a to remove the remote site IP address configurations related to cluster 3 and restart all the DB replication service deployments on cluster 4:

Note

Replace `cluster1` in the commands with `cluster4`. However, the values may vary depending on the cluster and namespace names in your setup.

7.6.4 Removing cnDBTier Cluster 2

This section provides the procedure to remove cnDBTier cluster 2 from a georeplication multisite setup.

Note

The cluster and namespace names used in this procedure may vary depending on your namespace and cluster name. Ensure that you replace the values as per your setup.

1. Perform the following steps to uninstall cnDBTier cluster 2 in a site. For more information, see the "Uninstalling cnDBTier" section in *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide*.

- a. Run the following commands to uninstall cnDBTier cluster 2:

```
$ helm uninstall mysql-cluster -n cluster2
```

- b. Run the following commands to delete the PVCs of cnDBTier cluster 2:

```
$ kubectl -n cluster2 get pvc
$ kubectl -n cluster2 delete pvc <PVC Names>
```

where, <PVC Names> is the PVC names of cnDBTier cluster 2.

2. Perform the following steps to delete the records related to cluster 2 from all the other available sites (cluster 1, cluster 3, and cluster 4):

- a. Log in to cnDBTier cluster 1 and delete the records related to cluster 2 using the `dbtremovesite` script:

```
$ source ./source_me
```

NOTE: source_me must be sourced while your current directory is the directory with the source_me file.

Enter cndbtier namespace: cluster1

\$ dbtremovesite

When the dbtremovesite script prompts you to select the cluster that needs to be removed, select cluster 2 by typing *yes* against the cluster 2 option. Type *no* for the rest of the cluster options.

Sample output:

```
Does cluster1 need to be removed (yes/no/exit)? no
Does cluster2 need to be removed (yes/no/exit)? yes
2024-04-23T05:47:00Z INFO - CURRENT_SITE_NAME = cluster1
2024-04-23T05:47:00Z INFO - CURRENT_SITE_ID = 1
2024-04-23T05:47:00Z INFO - Removing cluster2 (Site 2)
2024-04-23T05:47:00Z INFO - Make sure that cnDBTier cluster(cluster2)
is uninstalled in Site 2
DO YOU WANT TO PROCEED (yes/no/exit)? yes
ARE YOU SURE (yes/no/exit)? yes
-----
-----
-----
2024-04-23T05:47:45Z INFO - Ended timer PHASE 6: 1713851265
2024-04-23T05:47:45Z INFO - PHASE 6 took: 00 hr. 00 min. 02 sec.
2024-04-23T05:47:45Z INFO - Ended timer dbtremovesite: 1713851265
2024-04-23T05:47:45Z INFO - dbtremovesite took: 00 hr. 00 min. 57 sec.
2024-04-23T05:47:45Z INFO - dbtremovesite completed successfully
```

- b. Repeat step a on cluster 3 to delete the records related to cluster 2.
- c. Repeat step a on cluster 4 to delete the records related to cluster 2.
3. Remove the remote site IP address configurations related to cluster 2 and restart all the DB replication service deployments on all the other sites (cluster 1, cluster 3, and cluster 4):
 - a. Perform the following steps to remove the remote site IP address configurations related to cluster 2 and restart all the DB replication service deployments on cluster 1:
 - i. Log in to the Bastion Host of cluster 1.
 - ii. If fixed IP addresses are not configured to the LoadBalancer services, configure remotesiteip with "" for cluster 2 remote site in the custom_values.yaml file of cluster 1:

```
$ vi cndbtier_cluster1_custom_values.yaml

- name: cluster1-cluster2-replication-svc
  enabled: true
  -----
  -----
  -----
  replication:
    localsiteip: ""
    localsiteport: "80"
```

```
channelgroupid: "1"
matesitename: "cluster2"
remotesiteip: ""
remotesiteport: "80"
```

- iii. If fixed IP addresses are not configured to the LoadBalancer services, upgrade cnDBTier cluster 1:

```
$ helm upgrade mysql-cluster ocndbtier -n cluster1 -f
cndbtier_cluster1_custom_values.yaml
```

- iv. Scale down and scale up all the DB replication service deployments in cluster 1:

```
$ kubectl -n cluster1 scale deploy $(kubectl -n cluster1 get
deployments | awk '{print $1}' | egrep -i 'repl|monitor|backup-
manager') --replicas=0
$ kubectl -n cluster1 scale deploy $(kubectl -n cluster1 get
deployments | awk '{print $1}' | egrep -i 'repl|monitor|backup-
manager') --replicas=1
```

- v. Wait until all the pods are up on cluster 1 and verify:

```
$ kubectl -n cluster1 get pods
```

- b. Repeat step a to remove the remote site IP address configurations related to cluster 2 and restart all the DB replication service deployments on cluster 3:

Note

Replace `cluster1` in the commands with `cluster3`. However, the values may vary depending on the cluster and namespace names in your setup.

- c. Repeat step a to remove the remote site IP address configurations related to cluster 2 and restart all the DB replication service deployments on cluster 4:

Note

Replace `cluster1` in the commands with `cluster4`. However, the values may vary depending on the cluster and namespace names in your setup.

7.6.5 Removing cnDBTier Cluster 1

This section provides the procedure to remove cnDBTier cluster 1 from a georeplication multisite setup.

Note

The cluster and namespace names used in this procedure may vary depending on your namespace and cluster name. Ensure that you replace the values as per your setup.

1. Perform the following steps to uninstall cnDBTier cluster 1 in a site. For more information, see the "Uninstalling cnDBTier" section in *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide*.

- a. Run the following commands to uninstall cnDBTier cluster 1:

```
$ helm uninstall mysql-cluster -n cluster1
```

- b. Run the following commands to delete the PVCs of cnDBTier cluster 1:

```
$ kubectl -n cluster1 get pvc
$ kubectl -n cluster1 delete pvc <PVC Names>
```

where, <PVC Names> is the PVC names of cnDBTier cluster 1.

2. Perform the following steps to delete the records related to cluster 1 from all the other available sites (cluster 2, cluster 3, and cluster 4):

- a. Log in to cnDBTier cluster 1 and delete the records related to cluster 1 using the dbtremovesite script:

```
$ source ./source_me
```

NOTE: source_me must be sourced while your current directory is the directory with the source_me file.

```
Enter cndbtier namespace: cluster4
```

```
$ dbtremovesite
```

When the dbtremovesite script prompts you to select the cluster that needs to be removed, select cluster 4 by typing *yes* against the cluster 4 option. Type *no* for the rest of the cluster options.

Sample output:

```
Does cluster1 need to be removed (yes/no/exit)? no
Does cluster2 need to be removed (yes/no/exit)? no
Does cluster3 need to be removed (yes/no/exit)? no
Does cluster4 need to be removed (yes/no/exit)? yes
2024-04-23T05:47:00Z INFO - CURRENT_SITE_NAME = cluster1
2024-04-23T05:47:00Z INFO - CURRENT_SITE_ID = 1
2024-04-23T05:47:00Z INFO - Removing cluster4 (Site 4)
2024-04-23T05:47:00Z INFO - Make sure that cnDBTier cluster(cluster4)
is uninstalled in Site 4
DO YOU WANT TO PROCEED (yes/no/exit)? yes
ARE YOU SURE (yes/no/exit)? yes
-----
-----
-----
2024-04-23T05:47:45Z INFO - Ended timer PHASE 6: 1713851265
2024-04-23T05:47:45Z INFO - PHASE 6 took: 00 hr. 00 min. 02 sec.
2024-04-23T05:47:45Z INFO - Ended timer dbtremovesite: 1713851265
2024-04-23T05:47:45Z INFO - dbtremovesite took: 00 hr. 00 min. 57 sec.
2024-04-23T05:47:45Z INFO - dbtremovesite completed successfully
```

- b. Repeat step a on cluster 3 to delete the records related to cluster 1.
- c. Repeat step a on cluster 4 to delete the records related to cluster 1.
- 3. Remove the remote site IP address configurations related to cluster 1 and restart all the DB replication service deployments on all the other sites (cluster 2, cluster 3, and cluster 4):
 - a. Perform the following steps to remove the remote site IP address configurations related to cluster 1 and restart all the DB replication service deployments on cluster 2:
 - i. Log in to the cnDBTier cluster4.
 - ii. If fixed IP addresses are not configured to the LoadBalancer services, configure `remotesiteip` with `""` for cluster 1 remote site in the `custom_values.yaml` file of cluster 4:

```
$ vi cndbtier_cluster2_custom_values.yaml
```

```
- name: cluster2-cluster1-replication-svc
  enabled: true
-----
-----
-----
  replication:
    localsiteip: ""
    localsiteport: "80"
    channelgroupid: "1"
    matesitename: "cluster1"
    remotesiteip: ""
    remotesiteport: "80"
```

- iii. If fixed IP addresses are not configured to the LoadBalancer services, upgrade cnDBTier cluster 2:

```
$ helm upgrade mysql-cluster ocndbtier -n cluster2 -f
cndbtier_cluster2_custom_values.yaml
```

- iv. Scale down and scale up all the DB replication service deployments in cluster 2:

```
$ kubectl -n cluster2 scale deploy $(kubectl -n cluster2 get
deployments | awk '{print $1}' | egrep -i 'repl|monitor|backup-
manager') --replicas=0
$ kubectl -n cluster2 scale deploy $(kubectl -n cluster2 get
deployments | awk '{print $1}' | egrep -i 'repl|monitor|backup-
manager') --replicas=1
```

- v. Wait until all the pods are up on cluster 2 and verify:

```
$ kubectl -n cluster2 get pods
```

- b. Repeat step a to remove the remote site IP address configurations related to cluster 1 and restart all the DB replication service deployments on cluster 3:

Note

Replace `cluster2` in the commands with `cluster3`. However, the values may vary depending on the cluster and namespace names in your setup.

- c. Repeat step a to remove the remote site IP address configurations related to cluster 1 and restart all the DB replication service deployments on cluster 4:

Note

Replace `cluster2` in the commands with `cluster4`. However, the values may vary depending on the cluster and namespace names in your setup.

7.7 Adding and Removing Replication Channel Group

This section describes the procedures to convert a single replication channel group to multiple replication channel groups and vice versa.

The procedures in this section uses the following terms to identify the clusters:

1. **cnDBTier Cluster1 (a.k.a. cluster1):** The first cloud native based DBTier cluster in a two site, three site, or four site georeplication setup.
2. **cnDBTier Cluster2 (a.k.a. cluster2):** The second cloud native based DBTier cluster in a two site, three site, or four site georeplication setup.
3. **cnDBTier Cluster3 (a.k.a. cluster3):** The third cloud native based DBTier cluster in a two site, three site, or four site georeplication setup.
4. **cnDBTier Cluster4 (a.k.a. cluster4):** The fourth cloud native based DBTier cluster in a two site, three site, or four site georeplication setup.

Prerequisites

1. All the cnDBTier data nodes and SQL nodes that participate in georeplication, and at least one management node in the existing clusters must be up and running.
2. Georeplication must be established between the existing cnDBTier clusters.
3. All the cnDBTier clusters must be installed using cnDBTier 22.3.x or above.
4. All the cnDBTier clusters must have the same number of data nodes and node groups.

Note

- cnDBTier supports two replication groups in each cnDBTier mate sites. To convert an existing single replication group to multi replication group, freshly install cnDBTier 22.3.x or above, or upgrade your existing cnDBTier to 22.3.x or above.
- Before running this procedure, take a backup of the data and download the backup for safety purposes.
- This procedure requires downtime for other cnDBTier sites.
- This procedure is not supported in cnDBTier setups where TLS is enabled.

7.7.1 Converting Single Replication Channel Group to Multiple Replication Channel Groups

This section describes the procedure to convert a single replication channel group to multiple replication channel groups.

Note

This procedure is not supported for cnDBTier setups where TLS is enabled.

To convert a single replication channel group to multiple replication channel groups:

1. Consider any one of the cnDBTier clusters as a leader cluster (cluster1) and move all the NFs' traffic to that cluster. All the pods of leader cluster must be up and running.
2. Wait for replication to be updated in every cnDBTier cluster so that the database is consistent across all the cnDBTier clusters.
3. Scale down the replication service deployments in each site: Log in to Bastion Host of each of the cnDBTier Clusters and scale down the DB replication service deployments:

```
$ kubectl -n <namespace of cnDBTier cluster> get deployments | egrep
'repl' | awk '{print $1}' | xargs -L1 -r kubectl -n <namespace of cnDBTier
cluster> scale deployment --replicas=0
```

Example:

```
# scaling down the DB replication service deployments of cluster1
$ kubectl -n cluster1 get deployments | egrep 'repl' | awk '{print $1}' | xargs -L1 -
r kubectl -n cluster1 scale deployment --replicas=0
deployment.apps/mysql-cluster-cluster1-cluster2-repl-svc scaled
deployment.apps/mysql-cluster-cluster1-cluster3-repl-svc scaled
deployment.apps/mysql-cluster-cluster1-cluster4-repl-svc scaled

# scaling down the DB replication service deployments of cluster2
$ kubectl -n cluster2 get deployments | egrep 'repl' | awk '{print $1}' | xargs -L1 -
r kubectl -n cluster2 scale deployment --replicas=0
deployment.apps/mysql-cluster-cluster2-cluster1-repl-svc scaled
deployment.apps/mysql-cluster-cluster2-cluster3-repl-svc scaled
deployment.apps/mysql-cluster-cluster2-cluster4-repl-svc scaled

# scaling down the DB replication service deployments of cluster3
$ kubectl -n cluster3 get deployments | egrep 'repl' | awk '{print $1}' | xargs -L1 -
r kubectl -n cluster3 scale deployment --replicas=0
deployment.apps/mysql-cluster-cluster3-cluster1-repl-svc scaled
deployment.apps/mysql-cluster-cluster3-cluster2-repl-svc scaled
deployment.apps/mysql-cluster-cluster3-cluster3-repl-svc scaled

# scaling down the DB replication service deployments of cluster4
$ kubectl -n cluster4 get deployments | egrep 'repl' | awk '{print $1}' | xargs -L1 -
r kubectl -n cluster4 scale deployment --replicas=0
deployment.apps/mysql-cluster-cluster4-cluster1-repl-svc scaled
deployment.apps/mysql-cluster-cluster4-cluster2-repl-svc scaled
deployment.apps/mysql-cluster-cluster4-cluster3-repl-svc scaled
```

4. Log in to the Bastion Host of each cnDBTier cluster, and stop and reset replica of all ndbmysqld pods to gracefully stop the replication channels:

```
$ kubectl -n <namespace of cnDBTier cluster> exec -ti <ndbmysqld pod name>
-- mysql -h 127.0.0.1 -uroot -pNextGenCne
mysql> stop replica;
mysql> reset replica all;
```

Example to stop and reset replica of all ndbmysqld pods in cluster1:

```
$ kubectl -n cluster1 exec -ti ndbmysqld-0 -- mysql -h 127.0.0.1 -uroot -pNextGenCne
mysql> stop replica;
Query OK, 0 rows affected (0.01 sec)
mysql> reset replica all;
Query OK, 0 rows affected (0.01 sec)

$ kubectl -n cluster1 exec -ti ndbmysqld-1 -- mysql -h 127.0.0.1 -uroot -pNextGenCne
mysql> stop replica;
Query OK, 0 rows affected (0.01 sec)
mysql> reset replica all;
Query OK, 0 rows affected (0.01 sec)

$ kubectl -n cluster1 exec -ti ndbmysqld-2 -- mysql -h 127.0.0.1 -uroot -pNextGenCne
mysql> stop replica;
Query OK, 0 rows affected (0.01 sec)
mysql> reset replica all;
Query OK, 0 rows affected (0.01 sec)

$ kubectl -n cluster1 exec -ti ndbmysqld-3 -- mysql -h 127.0.0.1 -uroot -pNextGenCne
mysql> stop replica;
Query OK, 0 rows affected (0.01 sec)
mysql> reset replica all;
Query OK, 0 rows affected (0.01 sec)

$ kubectl -n cluster1 exec -ti ndbmysqld-4 -- mysql -h 127.0.0.1 -uroot -pNextGenCne
mysql> stop replica;
Query OK, 0 rows affected (0.01 sec)
mysql> reset replica all;
Query OK, 0 rows affected (0.01 sec)

$ kubectl -n cluster1 exec -ti ndbmysqld-5 -- mysql -h 127.0.0.1 -uroot -pNextGenCne
mysql> stop replica;
Query OK, 0 rows affected (0.01 sec)
mysql> reset replica all;
Query OK, 0 rows affected (0.01 sec)
```

Note

You can use the same example to stop and reset replica of all ndbmysqld pods in cluster2, cluster3, and cluster4 by replacing the cluster names with cluster2, cluster3, and cluster4 respectively.

5. On the leader site, delete all the entries from all the tables of the replication_info database:

```
$ kubectl -n <namespace of leader cnDBTier cluster> exec -ti ndbmysqld-0 --
mysql -h 127.0.0.1 -uroot -pNextGenCne
mysql> DELETE from replication_info.DBTIER_REPLICATION_CHANNEL_INFO;
mysql> DELETE from replication_info.DBTIER_REPL_SITE_INFO;
```

```
mysql> DELETE from replication_info.DBTIER_SITE_INFO;
mysql> DELETE from replication_info.DBTIER_INITIAL_BINLOG_POSTION;
mysql> DELETE from replication_info.DBTIER_REPL_ERROR_SKIP_INFO;
mysql> DELETE from replication_info.DBTIER_REPL_EVENT_INFO;
```

Example with output:

```
$kubectl -n cluster1 exec -ti ndbmysqld-0 -- mysql -h 127.0.0.1 -uroot -pNextGenCne
mysql> DELETE from replication_info.DBTIER_REPLICATION_CHANNEL_INFO;
Query OK, 24 rows affected (0.01 sec)
mysql> DELETE from replication_info.DBTIER_REPL_SITE_INFO;
Query OK, 12 rows affected (0.01 sec)
mysql> DELETE from replication_info.DBTIER_SITE_INFO;
Query OK, 4 rows affected (0.01 sec)
mysql> DELETE from replication_info.DBTIER_INITIAL_BINLOG_POSTION;
Query OK, 32 rows affected (0.01 sec)
mysql> DELETE from replication_info.DBTIER_REPL_ERROR_SKIP_INFO;
Query OK, 0 rows affected (0.00 sec)
mysql> DELETE from replication_info.DBTIER_REPL_EVENT_INFO;
Query OK, 0 rows affected (0.00 sec)
```

6. Uninstall cnDBTier Cluster2 if exists.
7. Uninstall cnDBTier Cluster3 if exists.
8. Uninstall cnDBTier Cluster4 if exists.

Note

To uninstall cnDBTier clusters in steps 6, 7 and 8, follow the "Uninstalling DBTier" procedure in *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide*.

9. Make changes specific to multiple replication channel groups in the `custom_values.yaml` file by following the "Configuring Multiple Replication Channel Groups" procedure in *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide*.
10. On updating the `custom_values.yaml` file, upgrade the cnDBTier leader site (cluster1) by running the following command:

```
$ helm upgrade mysql-cluster --namespace <namespace of leader cnDBTier
cluster> occndbtier -f occndbtier/custom_values.yaml --no-hooks
```

Example:

```
$ helm upgrade mysql-cluster --namespace cluster1 occndbtier -f
occndbtier/custom_values.yaml --no-hooks
```

Sample output:

```
Release "mysql-cluster" has been upgraded. Happy Helming!
NAME: mysql-cluster
LAST DEPLOYED: Mon May 20 10:19:42 2025
NAMESPACE: cluster1
STATUS: deployed
REVISION: 2
```

11. Restart the management, data, and SQL pods of cnDBTier cluster by performing the following steps:

- a. Log in to the Bastion Host of cnDBTier cluster and scale down the DB replication service deployments of the cnDBTier leader site (cluster1):

```
$ kubectl -n <namespace of cnDBTier cluster> get deployments | egrep
'repl' | awk '{print $1}' | xargs -L1 -r kubectl -n <namespace of
cnDBTier cluster> scale deployment --replicas=0
```

Example:

```
# scale down the db replication service deployments of cluster1
$ kubectl -n cluster1 get deployments | egrep 'repl' | awk '{print $1}'
| xargs -L1 -r kubectl -n cluster1 scale deployment --replicas=0
```

Sample output:

```
deployment.apps/mysql-cluster-cluster1-cluster2-repl-grp1 scaled
deployment.apps/mysql-cluster-cluster1-cluster2-repl-grp2 scaled
deployment.apps/mysql-cluster-cluster1-cluster3-repl-grp1 scaled
deployment.apps/mysql-cluster-cluster1-cluster3-repl-grp2 scaled
deployment.apps/mysql-cluster-cluster1-cluster4-repl-grp1 scaled
deployment.apps/mysql-cluster-cluster1-cluster4-repl-grp2 scaled
```

- b. Restart all the management pods of cnDBTier leader site (cluster1) by performing the following steps:

- i. Identify the list of management pods at cnDBTier cluster1:

```
$ kubectl get pods --namespace=cluster1 | grep 'mgmd'
```

Sample output:

```
ndbmgmd-0                                2/2
Running                                0          9m34s
ndbmgmd-1                                2/2
Running                                0          9m4s
```

- ii. Delete all the management pods of cnDBTier cluster1 to restart the management pods:

```
$ kubectl delete pod ndbmgmd-0 ndbmgmd-1 --namespace=cluster1
```

Sample output:

```
pod "ndbmgmd-0" deleted
pod "ndbmgmd-1" deleted
```

- iii. Wait for the management pods to restart and run the following command to check if the management pods are up and running:

```
$ kubectl get pods --namespace=cluster1 | grep 'mgmd'
```

Sample output:

```
ndbmgmd-0                                2/2      Running
0          4m29s
```

```
ndbmgmd-1          2/2    Running
0                4m12s
```

iv. Check the status of cnDBTier cluster1.

```
$ kubectl -n cluster1 exec -it ndbmgmd-0 -- ndb_mgm -e show
```

Sample output:

```
Connected to Management Server at: localhost:1186 Cluster Configuration
```

```
-----
[ndbd(NDB)]      2 node(s)
id=1   @10.233.85.92 (mysql-8.4.3 ndb-8.4.3, Nodegroup: 0)
id=2   @10.233.114.33 (mysql-8.4.3 ndb-8.4.3, Nodegroup: 0, *)

[ndb_mgmd(MGM)]  2 node(s)
id=49   @10.233.65.167 (mysql-8.4.3 ndb-8.4.3)
id=50   @10.233.127.115 (mysql-8.4.3 ndb-8.4.3)

[mysqld(API)]   18 node(s)
id=56   @10.233.124.92 (mysql-8.4.3 ndb-8.4.3)
id=57   @10.233.114.135 (mysql-8.4.3 ndb-8.4.3)
id=58   @10.233.113.87 (mysql-8.4.3 ndb-8.4.3)
id=59   @10.233.114.32 (mysql-8.4.3 ndb-8.4.3)
id=60   @10.233.108.33 (mysql-8.4.3 ndb-8.4.3)
id=61   @10.233.78.230 (mysql-8.4.3 ndb-8.4.3)
id=62   (not connected, accepting connect from
ndbmysqld-2.ndbmysqldsvc.cluster1.svc.occn4-cgbu-cne-dbtier)
id=63   (not connected, accepting connect from
ndbmysqld-3.ndbmysqldsvc.cluster1.svc.occn4-cgbu-cne-dbtier)
id=64   (not connected, accepting connect from
ndbmysqld-2.ndbmysqldsvc.cluster1.svc.occn4-cgbu-cne-dbtier)
id=65   (not connected, accepting connect from
ndbmysqld-3.ndbmysqldsvc.cluster1.svc.occn4-cgbu-cne-dbtier)
id=66   (not connected, accepting connect from
ndbmysqld-2.ndbmysqldsvc.cluster1.svc.occn4-cgbu-cne-dbtier)
id=67   (not connected, accepting connect from
ndbmysqld-3.ndbmysqldsvc.cluster1.svc.occn4-cgbu-cne-dbtier)
id=70   @10.233.127.117 (mysql-8.4.3 ndb-8.4.3)
id=71   @10.233.85.93 (mysql-8.4.3 ndb-8.4.3)
id=222  (not connected, accepting connect from any host)
id=223  (not connected, accepting connect from any host)
id=224  (not connected, accepting connect from any host)
id=225  (not connected, accepting connect from any host)
```

Note

- The API pods with node IDs 222 to 225 in the sample output are shown as "not connected" as they are added as empty API slots for georeplication recovery. You can ignore these nodes.
- The SQL pods with node IDs 62 to 67 in the sample output are the newly added pods. These pods remain in the not connected state until all the data nodes are restarted.

c. Restart the data pods sequentially by performing the following steps:

- i. Identify the list of data pods at cnDBTier cluster1:

```
$ kubectl get pods --namespace=cluster1 | grep 'ndbmt-d'
```

Sample output:

```
ndbmt-d-0          3/3      Running
0                14m
ndbmt-d-1          3/3      Running
0                13m
```

- ii. Delete the first data pod of cnDBTier cluster1 (in this example, ndbmt-d-0) such that the pod restarts:

```
$ kubectl delete pod ndbmt-d-0 --namespace=cluster1
```

Sample output:

```
pod "ndbmt-d-0" deleted
```

- iii. Wait for the first data pod to restart and run the following command to check if the first pod is up and running:

```
$ kubectl get pods --namespace=cluster1 | grep 'ndbmt-d'
```

Sample output:

```
ndbmt-d-0          3/3      Running
0                65s
ndbmt-d-1          3/3      Running
0                13m
```

- iv. Check the status of cnDBTier cluster1:

```
$ kubectl -n cluster1 exec -it ndbm-gmd-0 -- ndb_mgm -e show
```

Sample output:

```
-----
[ndbd(NDB)] 2 node(s)
id=1 @10.233.85.92 (mysql-8.4.3 ndb-8.4.3, Nodegroup: 0)
id=2 @10.233.114.33 (mysql-8.4.3 ndb-8.4.3, Nodegroup: 0, *)

[ndb_mgm(MGM)] 2 node(s)
id=49 @10.233.65.167 (mysql-8.4.3 ndb-8.4.3)
id=50 @10.233.127.115 (mysql-8.4.3 ndb-8.4.3)

[mysqld(API)] 18 node(s)
id=56 @10.233.124.92 (mysql-8.4.3 ndb-8.4.3)
id=57 @10.233.114.135 (mysql-8.4.3 ndb-8.4.3)
id=58 @10.233.113.87 (mysql-8.4.3 ndb-8.4.3)
id=59 @10.233.114.32 (mysql-8.4.3 ndb-8.4.3)
id=60 @10.233.108.33 (mysql-8.4.3 ndb-8.4.3)
id=61 @10.233.78.230 (mysql-8.4.3 ndb-8.4.3)
id=62 (not connected, accepting connect from
ndbmysqld-2.ndbmysqldsvc.cluster1.svc.occn4-cgbu-cne-dbtier)
id=63 (not connected, accepting connect from
ndbmysqld-3.ndbmysqldsvc.cluster1.svc.occn4-cgbu-cne-dbtier)
id=64 (not connected, accepting connect from
ndbmysqld-2.ndbmysqldsvc.cluster1.svc.occn4-cgbu-cne-dbtier)
```

```

id=65 (not connected, accepting connect from
ndbmysqld-3.ndbmysqldsvc.cluster1.svc.occne4-cgbu-cne-dbtier)
id=66 (not connected, accepting connect from
ndbmysqld-2.ndbmysqldsvc.cluster1.svc.occne4-cgbu-cne-dbtier)
id=67 (not connected, accepting connect from
ndbmysqld-3.ndbmysqldsvc.cluster1.svc.occne4-cgbu-cne-dbtier)
id=70 @10.233.127.117 (mysql-8.4.3 ndb-8.4.3)
id=71 @10.233.85.93 (mysql-8.4.3 ndb-8.4.3)
id=222 (not connected, accepting connect from any host)
id=223 (not connected, accepting connect from any host)
id=224 (not connected, accepting connect from any host)
id=225 (not connected, accepting connect from any host)

```

Note

- The API pods with node IDs 222 to 225 in the sample output are shown as "not connected" as they are added as empty API slots for georeplication recovery. You can ignore these nodes.
- The SQL pods with node IDs 62 to 67 in the sample output are the newly added pods. These pods remain in the not connected state until all the data nodes are restarted.

- v. Follow [Step ii](#) through [Step iv](#) to delete and restart each of the remaining data pods of cnDBTier cluster1.
- d. Log in to Bastion Host of the cnDBTier cluster and scale up the db replication service deployments of cnDBTier leader site (cluster1).

```

$ kubectl -n <namespace of cnDBTier cluster> get deployments | egrep
'repl' | awk '{print $1}' | xargs -L1 -r kubectl -n <namespace of
cnDBTier cluster> scale deployment --replicas=1

```

Example:

```

# scale up the db replication service deployments of cluster1
$ kubectl -n cluster1 get deployments | egrep 'repl' | awk '{print $1}'
| xargs -L1 -r kubectl -n cluster1 scale deployment --replicas=1

```

Sample output:

```

deployment.apps/mysql-cluster-cluster1-cluster2-repl-grp1 scaled
deployment.apps/mysql-cluster-cluster1-cluster2-repl-grp2 scaled
deployment.apps/mysql-cluster-cluster1-cluster3-repl-grp1 scaled
deployment.apps/mysql-cluster-cluster1-cluster3-repl-grp2 scaled
deployment.apps/mysql-cluster-cluster1-cluster4-repl-grp1 scaled
deployment.apps/mysql-cluster-cluster1-cluster4-repl-grp2 scaled

```

- e. Perform the following steps to restart the monitor service pod of the leader site (cnDBTier cluster1):
 - i. Identify the monitor service pod at cnDBTier cluster1:

```

$ kubectl get pods --namespace=cluster1 | grep 'monitor'

```

Sample output:


```
mysql-cluster-db-monitor-svc-8bf9448b8-4cghv    1/1    Running
0                                                13h
```

- ii. Delete the monitor service pod of cnDBTier cluster1:

```
$ kubectl delete pod mysql-cluster-db-monitor-svc-8bf9448b8-4cghv --
namespace=cluster1
```

Sample output:

```
pod "mysql-cluster-db-monitor-svc-8bf9448b8-4cghv" deleted
```

- iii. Wait until the monitor service pod is up and running. You can check the status of the pod by running the following command:

```
$ kubectl get pods --namespace=cluster1 | grep 'monitor'
```

Sample output:

```
mysql-cluster-db-monitor-svc-8bf9448b8-w8pvf    1/1    Running
0                                                109s
```

12. Wait until all the pods of the leader site are up and running. Verify if the data node, API nodes, and management nodes are connected to the cnDBTier cluster by running the following command:

```
# Checking the status of MySQL NDB Cluster in cnDBTier cluster
$ kubectl -n <namespace of leader cnDBTier Cluster> exec -it ndbmcmd-0 --
ndb_mgm -e show
```

Example:

```
$ kubectl -n cluster1 exec ndbmcmd-0 -- ndb_mgm -e show
```

Sample output:

```
Connected to Management Server at: localhost:1186
Cluster Configuration
```

```
-----
[ndbd(NDB)] 2 node(s)
id=1 @10.233.92.53 (mysql-8.4.3 ndb-8.4.3, Nodegroup: 0)
id=2 @10.233.72.66 (mysql-8.4.3 ndb-8.4.3, Nodegroup: 0)
```

```
[ndb_mgmd(MGM)] 2 node(s)
id=49 @10.233.111.62 (mysql-8.4.3 ndb-8.4.3)
id=50 @10.233.117.59 (mysql-8.4.3 ndb-8.4.3)
```

```
[mysqld(API)] 10 node(s)
id=56 @10.233.72.65 (mysql-8.4.3 ndb-8.4.3)
id=57 @10.233.92.51 (mysql-8.4.3 ndb-8.4.3)
id=58 @10.233.81.112 (mysql-8.4.3 ndb-8.4.3)
id=59 @10.233.64.07 (mysql-8.4.3 ndb-8.4.3)
id=60 @10.233.71.16 (mysql-8.4.3 ndb-8.4.3)
id=61 @10.233.114.196 (mysql-8.4.3 ndb-8.4.3)
id=62 @10.233.84.212 (mysql-8.4.3 ndb-8.4.3)
id=63 @10.233.108.21 (mysql-8.4.3 ndb-8.4.3)
id=64 @10.233.121.20 (mysql-8.4.3 ndb-8.4.3)
id=65 @10.233.109.231 (mysql-8.4.3 ndb-8.4.3)
id=66 @10.233.121.37 (mysql-8.4.3 ndb-8.4.3)
```

```

id=67 @10.233.84.38 (mysql-8.4.3 ndb-8.4.3)
id=70 @10.233.124.92 (mysql-8.4.3 ndb-8.4.3)
id=71 @10.233.113.109 (mysql-8.4.3 ndb-8.4.3)
id=222 (not connected, accepting connect from any host)
id=223 (not connected, accepting connect from any host)
id=224 (not connected, accepting connect from any host)
id=225 (not connected, accepting connect from any host)

```

Note

Node IDs 222 to 225 in the sample output are shown as "not connected" as they are added as empty API slots for georeplication recovery. You can ignore these nodes.

13. Upgrade the leader cnDBTier cluster using the custom_values.yaml file that you updated in [Step 9](#). For more information on upgrading cnDBTier, see *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide*.
14. Scale down the replication services of cnDBTier leader site (cluster1). Log in to Bastion Host of cnDBTier Cluster and scale down the DB replication service deployments:

```

$ kubectl -n <namespace of cnDBTier cluster> get deployments | egrep
'repl' | awk '{print $1}' | xargs -L1 -r kubectl -n <namespace of cnDBTier
cluster> scale deployment --replicas=0

```

For example, run the following command to scale down the DB replication service deployments of cluster1:

```

$ kubectl -n cluster1 get deployments | egrep 'repl' | awk '{print $1}' |
xargs -L1 -r kubectl -n cluster1 scale deployment --replicas=0

```

Sample output:

```

deployment.apps/mysql-cluster-cluster1-cluster2-repl-grp1 scaled
deployment.apps/mysql-cluster-cluster1-cluster2-repl-grp2 scaled
deployment.apps/mysql-cluster-cluster1-cluster3-repl-grp1 scaled
deployment.apps/mysql-cluster-cluster1-cluster3-repl-grp2 scaled
deployment.apps/mysql-cluster-cluster1-cluster4-repl-grp1 scaled
deployment.apps/mysql-cluster-cluster1-cluster4-repl-grp2 scaled

```

15. Delete all entries from all the tables of replication_info database on the leader site:

```

$ kubectl -n <namespace of leader cnDBTier cluster> exec -ti ndbmysqld-0 --
mysql -h 127.0.0.1 -uroot -pNextGenCne
mysql> DELETE from replication_info.DBTIER_REPLICATION_CHANNEL_INFO;
mysql> DELETE from replication_info.DBTIER_REPL_SITE_INFO;
mysql> DELETE from replication_info.DBTIER_SITE_INFO;
mysql> DELETE from replication_info.DBTIER_INITIAL_BINLOG_POSTION;
mysql> DELETE from replication_info.DBTIER_REPL_ERROR_SKIP_INFO;
mysql> DELETE from replication_info.DBTIER_REPL_EVENT_INFO;

```

Example with output:

```

$ kubectl -n cluster1 exec -ti ndbmysqld-0 -- mysql -h 127.0.0.1 -uroot -pNextGenCne
mysql> DELETE from replication_info.DBTIER_REPLICATION_CHANNEL_INFO;
Query OK, 24 rows affected (0.01 sec)
mysql> DELETE from replication_info.DBTIER_REPL_SITE_INFO;

```

```
Query OK, 12 rows affected (0.01 sec)
mysql> DELETE from replication_info.DBTIER_SITE_INFO;
Query OK, 4 rows affected (0.01 sec)
mysql> DELETE from replication_info.DBTIER_INITIAL_BINLOG_POSTION;
Query OK, 32 rows affected (0.01 sec)
mysql> DELETE from replication_info.DBTIER_REPL_ERROR_SKIP_INFO;
Query OK, 0 rows affected (0.00 sec)
mysql> DELETE from replication_info.DBTIER_REPL_EVENT_INFO;
Query OK, 0 rows affected (0.00 sec)
```

16. Scale up the replication services of cnDBTier leader site (cluster1). Log in to Bastion Host of cnDBTier cluster and scale up the DB replication service deployments:

```
$ kubectl -n <namespace of cnDBTier cluster> get deployments | egrep
'repl' | awk '{print $1}' | xargs -L1 -r kubectl -n <namespace of cnDBTier
cluster> scale deployment --replicas=1
```

For example, run the following command to scale up the DB replication service deployments of cluster1:

```
$ kubectl -n cluster1 get deployments | egrep 'repl' | awk '{print $1}' | xargs -L1 -
r kubectl -n cluster1 scale deployment --replicas=1
```

Sample output:

```
deployment.apps/mysql-cluster-cluster1-cluster2-repl-grp1 scaled
deployment.apps/mysql-cluster-cluster1-cluster2-repl-grp2 scaled
deployment.apps/mysql-cluster-cluster1-cluster3-repl-grp1 scaled
deployment.apps/mysql-cluster-cluster1-cluster3-repl-grp2 scaled
deployment.apps/mysql-cluster-cluster1-cluster4-repl-grp1 scaled
deployment.apps/mysql-cluster-cluster1-cluster4-repl-grp2 scaled
```

17. Reinstall cnDBTier Cluster2 by following the [Adding cnDBTier Georedundant Cluster to Single Site cnDBTier Cluster](#) procedure.
18. Reinstall cnDBTier Cluster3 by following the [Adding cnDBTier Georedundant Cluster to Two Site cnDBTier Clusters](#) procedure.
19. Reinstall cnDBTier Cluster4 by following the [Adding cnDBTier Georedundant Cluster to Four Site cnDBTier Clusters](#) procedure.

7.7.2 Converting Multiple Replication Channel Groups to Single Replication Channel Group

This section describes the procedure to convert multiple replication channel groups to single replication channel group.

Note

This procedure is not supported for cnDBTier setups where TLS is enabled.

This procedure can be used if multiple replication channel groups are enabled in the cnDBTier cluster and if you want to convert it to a single replication group due to a rollback to the previous versions or any other reasons.

To convert multiple replication channel groups to single replication channel group:

1. Consider any one of the cnDBTier clusters as a leader cluster (cluster1) and move all the NFs' traffic to that cluster. All the pods of leader cluster must be up and running.
2. Wait for replication to be updated in every cnDBTier cluster so that the database is consistent across all the cnDBTier clusters.
3. Scale down the replication service deployments in each site: Log in to Bastion Host of each of the cnDBTier Clusters and scale down the DB replication service deployments:

```
$ kubectl -n <namespace of cnDBTier cluster> get deployments | egrep
'repl' | awk '{print $1}' | xargs -L1 -r kubectl -n <namespace of cnDBTier
cluster> scale deployment --replicas=0
```

Example:

```
# scaling down the DB replication service deployments of cluster1
$ kubectl -n cluster1 get deployments | egrep 'repl' | awk '{print $1}' | xargs -L1 -
r kubectl -n cluster1 scale deployment --replicas=0
deployment.apps/mysql-cluster-cluster1-cluster2-repl-grp1 scaled
deployment.apps/mysql-cluster-cluster1-cluster2-repl-grp2 scaled
deployment.apps/mysql-cluster-cluster1-cluster3-repl-grp1 scaled
deployment.apps/mysql-cluster-cluster1-cluster3-repl-grp2 scaled
deployment.apps/mysql-cluster-cluster1-cluster4-repl-grp1 scaled
deployment.apps/mysql-cluster-cluster1-cluster4-repl-grp2 scaled
```

```
# scaling down the DB replication service deployments of cluster2 If exists
$ kubectl -n cluster2 get deployments | egrep 'repl' | awk '{print $1}' | xargs -L1 -
r kubectl -n cluster2 scale deployment --replicas=0
deployment.apps/mysql-cluster-cluster2-cluster1-repl-grp1 scaled
deployment.apps/mysql-cluster-cluster2-cluster1-repl-grp2 scaled
deployment.apps/mysql-cluster-cluster2-cluster3-repl-grp1 scaled
deployment.apps/mysql-cluster-cluster2-cluster3-repl-grp2 scaled
deployment.apps/mysql-cluster-cluster2-cluster4-repl-grp1 scaled
deployment.apps/mysql-cluster-cluster2-cluster4-repl-grp2 scaled
```

```
# scaling down the DB replication service deployments of cluster3 If exists
$ kubectl -n cluster3 get deployments | egrep 'repl' | awk '{print $1}' | xargs -L1 -
r kubectl -n cluster3 scale deployment --replicas=0
deployment.apps/mysql-cluster-cluster3-cluster1-repl-grp1 scaled
deployment.apps/mysql-cluster-cluster3-cluster1-repl-grp2 scaled
deployment.apps/mysql-cluster-cluster3-cluster2-repl-grp1 scaled
deployment.apps/mysql-cluster-cluster3-cluster2-repl-grp2 scaled
deployment.apps/mysql-cluster-cluster3-cluster4-repl-grp1 scaled
deployment.apps/mysql-cluster-cluster3-cluster4-repl-grp2 scaled
```

```
# scaling down the DB replication service deployments of cluster4 If exists
$ kubectl -n cluster4 get deployments | egrep 'repl' | awk '{print $1}' | xargs -L1 -
r kubectl -n cluster4 scale deployment --replicas=0
deployment.apps/mysql-cluster-cluster4-cluster1-repl-grp1 scaled
deployment.apps/mysql-cluster-cluster4-cluster1-repl-grp2 scaled
deployment.apps/mysql-cluster-cluster4-cluster2-repl-grp1 scaled
deployment.apps/mysql-cluster-cluster4-cluster2-repl-grp2 scaled
deployment.apps/mysql-cluster-cluster4-cluster3-repl-grp1 scaled
deployment.apps/mysql-cluster-cluster4-cluster3-repl-grp2 scaled
```

4. Log in to the Bastion Host of each cnDBTier cluster, and stop and reset replica of all ndbmysqld pods to gracefully stop the replication channels:

```
$ kubectl -n <namespace of cnDBTier cluster> exec -ti <ndbmysqld pod name>
-- mysql -h 127.0.0.1 -uroot -pNextGenCne
```

```
mysql> stop replica;  
mysql> reset replica all;
```

Example to stop and reset replica of all ndbmysqld pods in cluster1:

```
$ kubectl -n cluster1 exec -ti ndbmysqld-0 -- mysql -h 127.0.0.1 -uroot -pNextGenCne  
mysql> stop replica;  
Query OK, 0 rows affected (0.01 sec)  
mysql> reset replica all;  
Query OK, 0 rows affected (0.01 sec)
```

```
$ kubectl -n cluster1 exec -ti ndbmysqld-1 -- mysql -h 127.0.0.1 -uroot -pNextGenCne  
mysql> stop replica;  
Query OK, 0 rows affected (0.01 sec)  
mysql> reset replica all;  
Query OK, 0 rows affected (0.01 sec)
```

```
$ kubectl -n cluster1 exec -ti ndbmysqld-2 -- mysql -h 127.0.0.1 -uroot -pNextGenCne  
mysql> stop replica;  
Query OK, 0 rows affected (0.01 sec)  
mysql> reset replica all;  
Query OK, 0 rows affected (0.01 sec)
```

```
$ kubectl -n cluster1 exec -ti ndbmysqld-3 -- mysql -h 127.0.0.1 -uroot -pNextGenCne  
mysql> stop replica;  
Query OK, 0 rows affected (0.01 sec)  
mysql> reset replica all;  
Query OK, 0 rows affected (0.01 sec)
```

```
$ kubectl -n cluster1 exec -ti ndbmysqld-4 -- mysql -h 127.0.0.1 -uroot -pNextGenCne  
mysql> stop replica;  
Query OK, 0 rows affected (0.01 sec)  
mysql> reset replica all;  
Query OK, 0 rows affected (0.01 sec)
```

```
$ kubectl -n cluster1 exec -ti ndbmysqld-5 -- mysql -h 127.0.0.1 -uroot -pNextGenCne  
mysql> stop replica;  
Query OK, 0 rows affected (0.01 sec)  
mysql> reset replica all;  
Query OK, 0 rows affected (0.01 sec)
```

```
$ kubectl -n cluster1 exec -ti ndbmysqld-6 -- mysql -h 127.0.0.1 -uroot -pNextGenCne  
mysql> stop replica;  
Query OK, 0 rows affected (0.01 sec)  
mysql> reset replica all;  
Query OK, 0 rows affected (0.01 sec)
```

```
$ kubectl -n cluster1 exec -ti ndbmysqld-7 -- mysql -h 127.0.0.1 -uroot -pNextGenCne  
mysql> stop replica;  
Query OK, 0 rows affected (0.01 sec)  
mysql> reset replica all;  
Query OK, 0 rows affected (0.01 sec)
```

```
$ kubectl -n cluster1 exec -ti ndbmysqld-8 -- mysql -h 127.0.0.1 -uroot -pNextGenCne  
mysql> stop replica;  
Query OK, 0 rows affected (0.01 sec)  
mysql> reset replica all;  
Query OK, 0 rows affected (0.01 sec)
```

```
$ kubectl -n cluster1 exec -ti ndbmysqld-9 -- mysql -h 127.0.0.1 -uroot -pNextGenCne  
mysql> stop replica;  
Query OK, 0 rows affected (0.01 sec)
```

```
mysql> reset replica all;
Query OK, 0 rows affected (0.01 sec)

$ kubectl -n cluster1 exec -ti ndbmysqld-10 -- mysql -h 127.0.0.1 -uroot -pNextGenCne
mysql> stop replica;
Query OK, 0 rows affected (0.01 sec)
mysql> reset replica all;
Query OK, 0 rows affected (0.01 sec)

$ kubectl -n cluster1 exec -ti ndbmysqld-11 -- mysql -h 127.0.0.1 -uroot -pNextGenCne
mysql> stop replica;
Query OK, 0 rows affected (0.01 sec)
mysql> reset replica all;
Query OK, 0 rows affected (0.01 sec)
```

Note

You can use the same example to stop and reset replica of all ndbmysqld pods in cluster2, cluster3, and cluster4 by replacing the cluster names with cluster2, cluster3, and cluster4 respectively.

5. On the leader site, delete all the entries from all the tables of the replication_info database:

```
$kubectl -n <namespace of leader cnDBTier cluster> exec -ti ndbmysqld-0 --
mysql -h 127.0.0.1 -uroot -pNextGenCne
mysql> DELETE from replication_info.DBTIER_REPLICATION_CHANNEL_INFO;
mysql> DELETE from replication_info.DBTIER_REPL_SITE_INFO;
mysql> DELETE from replication_info.DBTIER_SITE_INFO;
mysql> DELETE from replication_info.DBTIER_INITIAL_BINLOG_POSTION;
mysql> DELETE from replication_info.DBTIER_REPL_ERROR_SKIP_INFO;
mysql> DELETE from replication_info.DBTIER_REPL_EVENT_INFO;
```

Example with output:

```
$kubectl -n cluster1 exec -ti ndbmysqld-0 -- mysql -h 127.0.0.1 -uroot -pNextGenCne
mysql> DELETE from replication_info.DBTIER_REPLICATION_CHANNEL_INFO;
Query OK, 4 rows affected (0.01 sec)
mysql> DELETE from replication_info.DBTIER_REPL_SITE_INFO;
Query OK, 2 rows affected (0.01 sec)
mysql> DELETE from replication_info.DBTIER_SITE_INFO;
Query OK, 2 rows affected (0.01 sec)
mysql> DELETE from replication_info.DBTIER_INITIAL_BINLOG_POSTION;
Query OK, 8 rows affected (0.01 sec)
mysql> DELETE from replication_info.DBTIER_REPL_ERROR_SKIP_INFO;
Query OK, 0 rows affected (0.00 sec)
mysql> DELETE from replication_info.DBTIER_REPL_EVENT_INFO;
Query OK, 0 rows affected (0.00 sec)
```

6. Uninstall cnDBTier Cluster2 if exists.
7. Uninstall cnDBTier Cluster3 if exists.
8. Uninstall cnDBTier Cluster4 if exists.

Note

To uninstall cnDBTier clusters in steps 6, 7 and 8, follow the "Uninstalling DBTier" procedure in *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide*.

9. Remove or disable the configurations specific to multiple replication channel groups from the `custom_values.yaml` file.
10. On updating the `custom_values.yaml` file, upgrade the cnDBTier leader site (cluster1) by running the following command:

```
$ helm upgrade mysql-cluster --namespace <namespace of leader cnDBTier cluster> occndbtier -f occndbtier/custom_values.yaml --no-hooks
```

Example:

```
$ helm upgrade mysql-cluster --namespace cluster1 occndbtier -f occndbtier/custom_values.yaml --no-hooks
```

Sample output:

```
Release "mysql-cluster" has been upgraded. Happy Helming!
NAME: mysql-cluster
LAST DEPLOYED: Mon May 20 10:19:42 2025
NAMESPACE: cluster1
STATUS: deployed
REVISION: 2
```

11. Restart the management, data, and SQL pods of cnDBTier cluster by performing the following steps:
 - a. Log in to the Bastion Host of cnDBTier cluster and scale down the DB replication service deployments of the cnDBTier leader site (cluster1):

```
$ kubectl -n <namespace of cnDBTier cluster> get deployments | egrep 'repl' | awk '{print $1}' | xargs -L1 -r kubectl -n <namespace of cnDBTier cluster> scale deployment --replicas=0
```

Example:

```
# scale down the db replication service deployments of cluster1
$ kubectl -n cluster1 get deployments | egrep 'repl' | awk '{print $1}' | xargs -L1 -r kubectl -n cluster1 scale deployment --replicas=0
```

Sample output:

```
deployment.apps/mysql-cluster-cluster1-cluster2-repl-svc scaled
deployment.apps/mysql-cluster-cluster1-cluster3-repl-svc scaled
deployment.apps/mysql-cluster-cluster1-cluster4-repl-svc scaled
```

- b. Restart all the management pods of cnDBTier leader site (cluster1) by performing the following steps:

- i. Identify the list of management pods at cnDBTier cluster1:

```
$ kubectl get pods --namespace=cluster1 | grep 'mgmd'
```

Sample output:

```
ndbmgmd-0                2/2
Running                  0          9m34s
ndbmgmd-1                2/2
Running                  0          9m4s
```

- ii. Delete all the management pods of cnDBTier cluster1 to restart the management pods:

```
$ kubectl delete pod ndbmgmd-0 ndbmgmd-1 --namespace=cluster1
```

Sample output:

```
pod "ndbmgmd-0" deleted
pod "ndbmgmd-1" deleted
```

- iii. Wait for the management pods to restart and run the following command to check if the management pods are up and running:

```
$ kubectl get pods --namespace=cluster1 | grep 'mgmd'
```

Sample output:

```
ndbmgmd-0                2/2      Running
0                4m29s
ndbmgmd-1                2/2      Running
0                4m12s
```

- iv. Check the status of cnDBTier cluster1.

```
$ kubectl -n cluster1 exec -it ndbmgmd-0 -- ndb_mgm -e show
```

Sample output:

Connected to Management Server at: localhost:1186 Cluster Configuration

```
-----
[ndbd(NDB)] 2 node(s)
id=1  @10.233.85.92 (mysql-8.4.3 ndb-8.4.3, Nodegroup: 0)
id=2  @10.233.114.33 (mysql-8.4.3 ndb-8.4.3, Nodegroup: 0, *)
```

```
[ndb_mgmd(MGM)] 2 node(s)
id=49  @10.233.65.167 (mysql-8.4.3 ndb-8.4.3)
id=50  @10.233.127.115 (mysql-8.4.3 ndb-8.4.3)
```

```
[mysqld(API)] 18 node(s)
id=56  @10.233.124.92 (mysql-8.4.3 ndb-8.4.3)
id=57  @10.233.114.135 (mysql-8.4.3 ndb-8.4.3)
id=58  @10.233.113.87 (mysql-8.4.3 ndb-8.4.3)
id=59  @10.233.114.32 (mysql-8.4.3 ndb-8.4.3)
id=60  @10.233.108.33 (mysql-8.4.3 ndb-8.4.3)
id=61  @10.233.78.230 (mysql-8.4.3 ndb-8.4.3)
id=62  (not connected, accepting connect from
ndbmysqld-2.ndbmysqldsvc.cluster1.svc.occn4-cgbu-cne-dbtier)
id=63  (not connected, accepting connect from
ndbmysqld-3.ndbmysqldsvc.cluster1.svc.occn4-cgbu-cne-dbtier)
```



```

id=64 (not connected, accepting connect from
ndbmysqld-2.ndbmysqldsvc.cluster1.svc.occne4-cgbu-cne-dbtier)
id=65 (not connected, accepting connect from
ndbmysqld-3.ndbmysqldsvc.cluster1.svc.occne4-cgbu-cne-dbtier)
id=66 (not connected, accepting connect from
ndbmysqld-2.ndbmysqldsvc.cluster1.svc.occne4-cgbu-cne-dbtier)
id=67 (not connected, accepting connect from
ndbmysqld-3.ndbmysqldsvc.cluster1.svc.occne4-cgbu-cne-dbtier)
id=70 @10.233.127.117 (mysql-8.4.3 ndb-8.4.3)
id=71 @10.233.85.93 (mysql-8.4.3 ndb-8.4.3)
id=222 (not connected, accepting connect from any host)
id=223 (not connected, accepting connect from any host)
id=224 (not connected, accepting connect from any host)
id=225 (not connected, accepting connect from any host)

```

Note

- The API pods with node IDs 222 to 225 in the sample output are shown as "not connected" as they are added as empty API slots for georeplication recovery. You can ignore these nodes.
- The SQL pods with node IDs 62 to 67 in the sample output are the newly added pods. These pods remain in the not connected state until all the data nodes are restarted.

c. Restart the data pods sequentially by performing the following steps:

i. Identify the list of data pods at cnDBTier cluster1:

```
$ kubectl get pods --namespace=cluster1 | grep 'ndbmt-d'
```

Sample output:

```

ndbmt-d-0                                3/3      Running
0                                         14m
ndbmt-d-1                                3/3      Running
0                                         13m

```

ii. Delete first data pod of cnDBTier cluster1 (ndbmt-d-0) such that the pod restarts:

```
$ kubectl delete pod ndbmt-d-0 --namespace=cluster1
```

Sample output:

```
pod "ndbmt-d-0" deleted
```

iii. Wait for the first data pod to restart and run the following command to check if the first pod is up and running:

```
$ kubectl get pods --namespace=cluster1 | grep 'ndbmt-d'
```

Sample output:

```

ndbmt-d-0                                3/3      Running
0                                         65s
ndbmt-d-1                                3/3      Running
0                                         13m

```

iv. Check the status of cnDBTier cluster1:

```
$ kubectl -n cluster1 exec -it ndbmcmd-0 -- ndb_mgm -e show
```

Sample output:

Connected to Management Server at: localhost:1186 Cluster Configuration

```
-----
[ndbd(NDB)] 2 node(s)
id=1 @10.233.85.92 (mysql-8.4.3 ndb-8.4.3, Nodegroup: 0)
id=2 @10.233.114.33 (mysql-8.4.3 ndb-8.4.3, Nodegroup: 0, *)

[ndb_mgmd(MGM)] 2 node(s)
id=49 @10.233.65.167 (mysql-8.4.3 ndb-8.4.3)
id=50 @10.233.127.115 (mysql-8.4.3 ndb-8.4.3)

[mysqld(API)] 18 node(s)
id=56 @10.233.124.92 (mysql-8.4.3 ndb-8.4.3)
id=57 @10.233.114.135 (mysql-8.4.3 ndb-8.4.3)
id=58 @10.233.113.87 (mysql-8.4.3 ndb-8.4.3)
id=59 @10.233.114.32 (mysql-8.4.3 ndb-8.4.3)
id=60 @10.233.108.33 (mysql-8.4.3 ndb-8.4.3)
id=61 @10.233.78.230 (mysql-8.4.3 ndb-8.4.3)
id=62 (not connected, accepting connect from
ndbmysqld-2.ndbmysqldsvc.cluster1.svc.occne4-cgbu-cne-dbtier)
id=63 (not connected, accepting connect from
ndbmysqld-3.ndbmysqldsvc.cluster1.svc.occne4-cgbu-cne-dbtier)
id=64 (not connected, accepting connect from
ndbmysqld-2.ndbmysqldsvc.cluster1.svc.occne4-cgbu-cne-dbtier)
id=65 (not connected, accepting connect from
ndbmysqld-3.ndbmysqldsvc.cluster1.svc.occne4-cgbu-cne-dbtier)
id=66 (not connected, accepting connect from
ndbmysqld-2.ndbmysqldsvc.cluster1.svc.occne4-cgbu-cne-dbtier)
id=67 (not connected, accepting connect from
ndbmysqld-3.ndbmysqldsvc.cluster1.svc.occne4-cgbu-cne-dbtier)
id=70 @10.233.127.117 (mysql-8.4.3 ndb-8.4.3)
id=71 @10.233.85.93 (mysql-8.4.3 ndb-8.4.3)
id=222 (not connected, accepting connect from any host)
id=223 (not connected, accepting connect from any host)
id=224 (not connected, accepting connect from any host)
id=225 (not connected, accepting connect from any host)
```

Note

- The API pods with node IDs 222 to 225 in the sample output are shown as "not connected" as they are added as empty API slots for georeplication recovery. You can ignore these nodes.
- The SQL pods with node IDs 62 to 67 in the sample output are the newly added pods. These pods remain in the not connected state until all the data nodes are restarted.

v. Follow [Step ii](#) through [Step iv](#) to delete and restart the second data pod of cnDBTier cluster1 (ndbmtd-1).

- d. Log in to Bastion Host of the cnDBTier cluster and scale up the db replication service deployments of cnDBTier leader site (cluster1).

```
$ kubectl -n <namespace of cnDBTier cluster> get deployments | egrep
'repl' | awk '{print $1}' | xargs -L1 -r kubectl -n <namespace of
cnDBTier cluster> scale deployment --replicas=1
```

Example:

```
# scale up the db replication service deployments of cluster1
$ kubectl -n cluster1 get deployments | egrep 'repl' | awk '{print $1}'
| xargs -L1 -r kubectl -n cluster1 scale deployment --replicas=1
```

Sample output:

```
deployment.apps/mysql-cluster-cluster1-cluster2-repl-svc scaled
deployment.apps/mysql-cluster-cluster1-cluster3-repl-svc scaled
deployment.apps/mysql-cluster-cluster1-cluster4-repl-svc scaled
```

- e. Perform the following steps to restart the monitor service pod of the leader site (cnDBTier cluster1):

- i. Identify the monitor service pod at cnDBTier cluster1:

```
$ kubectl get pods --namespace=cluster1 | grep 'monitor'
```

Sample output:

```
mysql-cluster-db-monitor-svc-8bf9448b8-4cghv      1/1      Running
0                                                  13h
```

- ii. Delete the monitor service pod of cnDBTier cluster1:

```
$ kubectl delete pod mysql-cluster-db-monitor-svc-8bf9448b8-4cghv --
namespace=cluster1
```

Sample output:

```
pod "mysql-cluster-db-monitor-svc-8bf9448b8-4cghv" deleted
```

- iii. Wait until the monitor service pod is up and running. You can check the status of the pod by running the following command:

```
$ kubectl get pods --namespace=cluster1 | grep 'monitor'
```

Sample output:

```
mysql-cluster-db-monitor-svc-8bf9448b8-w8pvf      1/1      Running
0                                                  109s
```

12. Wait until all the pods of the leader site are up and running. Verify if the data node, API nodes, and management nodes are connected to the cnDBTier cluster by running the following command:

```
$ kubectl -n <namespace of leader cnDBTier Cluster> exec -it ndbmgmd-0 --
ndb_mgm -e show
```

Example:

```
$ kubectl -n cluster1 exec -it ndbmgmd-0 -- ndb_mgm -e show
```

Sample output:

```
Connected to Management Server at: localhost:1186
Cluster Configuration
-----
[ndbd(NDB)] 2 node(s)
id=1 @10.233.92.53 (mysql-8.4.3 ndb-8.4.3, Nodegroup: 0)
id=2 @10.233.72.66 (mysql-8.4.3 ndb-8.4.3, Nodegroup: 0)

[ndb_mgmd(MGM)] 2 node(s)
id=49 @10.233.111.62 (mysql-8.4.3 ndb-8.4.3)
id=50 @10.233.117.59 (mysql-8.4.3 ndb-8.4.3)

[mysqld(API)] 10 node(s)
id=56 @10.233.72.65 (mysql-8.4.3 ndb-8.4.3)
id=57 @10.233.92.51 (mysql-8.4.3 ndb-8.4.3)
id=58 @10.233.113.87 (mysql-8.4.3 ndb-8.4.3)
id=59 @10.233.114.32 (mysql-8.4.3 ndb-8.4.3)
id=60 @10.233.108.33 (mysql-8.4.3 ndb-8.4.3)
id=61 @10.233.78.230 (mysql-8.4.3 ndb-8.4.3)
id=70 @10.233.72.64 (mysql-8.4.3 ndb-8.4.3)
id=71 @10.233.92.52 (mysql-8.4.3 ndb-8.4.3)
id=222 (not connected, accepting connect from any host)
id=223 (not connected, accepting connect from any host)
id=224 (not connected, accepting connect from any host)
id=225 (not connected, accepting connect from any host)
```

Note

Node IDs 222 to 225 in the sample output are shown as "not connected" as they are added as empty API slots for georeplication recovery. You can ignore these nodes.

13. Upgrade the leader cnDBTier cluster using the custom_values.yaml file that you updated in [Step 9](#). For more information on upgrading cnDBTier, see *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide*.
14. Scale down the replication services of cnDBTier leader site (cluster1). Log in to Bastion Host of cnDBTier Cluster and scale down the DB replication service deployments:

```
$ kubectl -n <namespace of cnDBTier cluster> get deployments | egrep
'repl' | awk '{print $1}' | xargs -L1 -r kubectl -n <namespace of cnDBTier
cluster> scale deployment --replicas=0
```

For example, run the following command to scale down the DB replication service deployments of cluster1:

```
$ kubectl -n cluster1 get deployments | egrep 'repl' | awk '{print $1}' |
xargs -L1 -r kubectl -n cluster1 scale deployment --replicas=0
```

Sample output:

```
deployment.apps/mysql-cluster-cluster1-cluster2-repl-svc scaled
deployment.apps/mysql-cluster-cluster1-cluster3-repl-svc scaled
deployment.apps/mysql-cluster-cluster1-cluster4-repl-svc scaled
```

15. Delete all entries from all the tables of replication_info database on the leader site:

```
$kubectl -n <namespace of leader cnDBTier cluster> exec -ti ndbmysqld-0 --
mysql -h 127.0.0.1 -uroot -pNextGenCne
mysql> DELETE from replication_info.DBTIER_REPLICATION_CHANNEL_INFO;
mysql> DELETE from replication_info.DBTIER_REPL_SITE_INFO;
mysql> DELETE from replication_info.DBTIER_SITE_INFO;
mysql> DELETE from replication_info.DBTIER_INITIAL_BINLOG_POSTION;
mysql> DELETE from replication_info.DBTIER_REPL_ERROR_SKIP_INFO;
mysql> DELETE from replication_info.DBTIER_REPL_EVENT_INFO;
```

Example with output:

```
$kubectl -n cluster1 exec -ti ndbmysqld-0 -- mysql -h 127.0.0.1 -uroot -pNextGenCne
mysql> DELETE from replication_info.DBTIER_REPLICATION_CHANNEL_INFO;
Query OK, 4 rows affected (0.01 sec)
mysql> DELETE from replication_info.DBTIER_REPL_SITE_INFO;
Query OK, 2 rows affected (0.01 sec)
mysql> DELETE from replication_info.DBTIER_SITE_INFO;
Query OK, 2 rows affected (0.01 sec)
mysql> DELETE from replication_info.DBTIER_INITIAL_BINLOG_POSTION;
Query OK, 8 rows affected (0.01 sec)
mysql> DELETE from replication_info.DBTIER_REPL_ERROR_SKIP_INFO;
Query OK, 0 rows affected (0.00 sec)
mysql> DELETE from replication_info.DBTIER_REPL_EVENT_INFO;
Query OK, 0 rows affected (0.00 sec)
```

16. Scale up the replication services of cnDBTier leader site (cluster1). Log in to Bastion Host of cnDBTier cluster and scale up the DB replication service deployments:

```
$ kubectl -n <namespace of cnDBTier cluster> get deployments | egrep
'repl' | awk '{print $1}' | xargs -L1 -r kubectl -n <namespace of cnDBTier
cluster> scale deployment --replicas=1
```

For example, run the following command to scale up the DB replication service deployments of cluster1:

```
$ kubectl -n cluster1 get deployments | egrep 'repl' | awk '{print $1}' | xargs -L1 -
r kubectl -n cluster1 scale deployment --replicas=1
```

Sample output:

```
deployment.apps/mysql-cluster-cluster1-cluster2-repl-svc scaled
deployment.apps/mysql-cluster-cluster1-cluster3-repl-svc scaled
deployment.apps/mysql-cluster-cluster1-cluster4-repl-svc scaled
```

17. Reinstall cnDBTier Cluster2 by following the [Adding cnDBTier Georedundant Cluster to Single Site cnDBTier Cluster](#) procedure.
18. Reinstall cnDBTier Cluster3 by following the [Adding cnDBTier Georedundant Cluster to Two Site cnDBTier Clusters](#) procedure.
19. Reinstall cnDBTier Cluster4 by following the [Adding cnDBTier Georedundant Cluster to Three Site cnDBTier Clusters](#) procedure.

7.8 Managing Passwords and Secrets

This section provides the procedures to manage cnDBTier passwords and secrets.

7.8.1 Modifying cnDBTier Password Encryption Key

Encryption key is used to encrypt the replication username and password stored in the database using the Advanced Encryption Standard (AES) algorithm. This section provides the procedure to modify the encryption key used to encrypt the replication username and password.

Note

If you update an encryption key on one site, ensure that you use the same encryption key across all the other mate sites.

1. Run the `hooks.sh` script with the `--update-encryption-key` flag to change the encryption secret and encrypt the replication username and password using the new encryption key.

Note

Replace the values of the environment variables in the commands with the values corresponding to your cluster.

```
export OCCNE_NAMESPACE="occne-cndbtier"
export MYSQL_CONNECTIVITY_SERVICE="mysql-connectivity-service"
export MYSQL_USERNAME="occneuser"
export MYSQL_PASSWORD=""
export MYSQL_CMD="kubectl -n <namespace> exec <ndbmysqld-0/ndbappmysqld-0
pod name> -- mysql --binary-as-hex=0 --show-warnings"
export NEW_ENCRYPTION_KEY=""
export DBTIER_REPLICATION_SVC_DATABASE="replication_info"
export APP_STS_NAME="ndbappmysqld"
# Optional: Uncomment the line below to decrypt the encrypted credentials
with a custom key.
# export CURRENT_ENCRYPTION_KEY=""
```

```
occndbtier/files/hooks.sh --update-encryption-key
```

2. Perform rollout restart of `ndbmysqld` on the cnDBTier cluster:
 - a. Run the following command to fetch the statefulset name of `ndbmysqld`:

```
$ kubectl get statefulset --namespace=<namespace of cnDBTier cluster>
```

For example:

```
$ kubectl get statefulset --namespace=occne-cndbtier
```

Sample output:

NAME	READY	AGE
ndbappmysqld	2/2	27h
ndbmgmd	2/2	27h
ndbmtl	2/2	27h
ndbmysqld	2/2	27h

- b. Perform rollout restart of ndbmysqld:

```
$ kubectl rollout restart statefulset <statefulset name of ndbmysqld> --
namespace=<namespace of cnDBTier cluster>
```

For example:

```
$ kubectl rollout restart statefulset ndbmysqld --namespace=occne-
cnbdtier
```

Sample output:

```
statefulset.apps/ndbmysqld restarted
```

- c. Wait until the rollout restart of ndbmysqld pod completes. Run the following command to check the status:

```
$ kubectl rollout status statefulset <statefulset name of ndbmysqld> --
namespace=<namespace of cnDBTier cluster>
```

For example:

```
$ kubectl rollout status statefulset ndbmysqld --namespace=occne-
cnbdtier
```

Sample output:

```
Waiting for 1 pods to be ready...
waiting for statefulset rolling update to complete 1 pods at revision
ndbmysqld-7c6b9c9f84...
Waiting for 1 pods to be ready...
Waiting for 1 pods to be ready...
statefulset rolling update complete 2 pods at revision
ndbmysqld-7c6b9c9f84...
```

3. After the rollout restart, run the following command to ensure that the replication is UP on all cnDBTier sites:

```
$ kubectl -n <namespace of cnDBTier cluster> exec -it ndbmysqld-0 -- curl
http://mysql-cluster-db-monitor-svc.<namespace of cnDBTier
cluster>:8080/db-tier/status/replication/realtime
```

where, <namespace> is the namespace name of the of cnDBTier cluster.

The value of `replicationStatus` in the output indicates if the local site is able to replicate data from that remote site:

- "UP": Indicates that the local site is able to replicate data from that remote site.
- "DOWN": Indicates that the local site is not able to replicate data from the respective remote site.

For example, run the following command to check the georeplication status of `cnDBTier cluster2` configured with other remote sites:

```
$ kubectl -n cluster2 exec -it ndbmysqld-0 -- curl http://mysql-cluster-db-monitor-svc.cluster2:8080/db-tier/status/replication/realtime
```

Sample output:

```
[
  {
    "localSiteName": "cluster2",
    "remoteSiteName": "cluster1",
    "replicationStatus": "UP",
    "secondsBehindRemote": 0,
    "replicationGroupDelay": [
      {
        "replchannel_group_id": "1",
        "secondsBehindRemote": 0
      }
    ]
  },
  {
    "localSiteName": "cluster2",
    "remoteSiteName": "cluster3",
    "replicationStatus": "UP",
    "secondsBehindRemote": 0,
    "replicationGroupDelay": [
      {
        "replchannel_group_id": "1",
        "secondsBehindRemote": 0
      }
    ]
  },
  {
    "localSiteName": "cluster2",
    "remoteSiteName": "cluster4",
    "replicationStatus": "UP",
    "secondsBehindRemote": 0,
    "replicationGroupDelay": [
      {
        "replchannel_group_id": "1",
        "secondsBehindRemote": 0
      }
    ]
  }
]
```


In the sample output, the value of `remotesiteName` is "UP" for the `localSiteName` cluster1, cluster3, and cluster4. This indicates that the cluster2 for `localSiteName` cluster2 is able to replicate data from `remotesiteName` cluster1, cluster3, and cluster4.

4. Repeat steps 1, 2, and 3 on the other `cnDBTier` mate sites.

7.8.2 Changing `cnDBTier` Passwords

This section provides the procedures to change `cnDBTier` passwords such as `root` user, `occnreuser`, and `occnrepluser` passwords in a stand-alone and multisite georeplication setup using the `dbtpasswd` bash script.

Note

- The password changes are applicable only to the current site and are not replicated in the mate sites.
- Ensure that your password meets the following password policy requirements:
 - Password must be between 20 and 32 characters in length.
 - Password must contain at least one lower case letter.
 - Password must contain at least one upper case letter.
 - Password must include at least one digit.
 - Password must include at least one of the following special characters: `,%~+. :_/-`
- Any character that does not meet the complexity requirements mentioned in the previous note is not supported as it may break the functionality of the `dbtpasswd` script.

Changing a `cnDBTier` password involves the following steps:

1. Adding a new password in MySQL.

Note

The old password is active until the pods are restarted and the transitional operations are complete.

2. Replacing the current or old password with the new password in Kubernetes secret.
3. Restarting the configured `cnDBTier` pods to use the new Kubernetes secret (This step is not applicable while changing an NF password).
4. Discarding the old password in MySQL.

The `dbtpasswd` bash script automates these steps and provides a single script to change all `cnDBTier` passwords at once. The script also provides options to run selective steps as changing passwords often requires running selective transitional operations, such as restarting pods, while both passwords are still valid.

You can use `dbtpasswd` to:

- change one or more `cnDBTier` passwords in a single site or cluster. Changing a `cnDBTier` password on a site includes changing the password in MySQL and Kubernetes secret,

restarting all required cnDBTier pods, and updating passwords on cnDBTier database tables if necessary.

- change passwords on a live cluster with no service interruption.
- change NF passwords. However, when changing an NF password, *dbtpasswd* can change the password in the Kubernetes secret and database only. You have to manually restart NF pods as a separate user action.

To provide flexibility and support for changing non-cnDBTier passwords (that is, NF passwords), *dbtpasswd* provides a range of options for controlling the phases of execution. For example, you can configure *dbtpasswd* to only add the new password to the database, change it in the Kubernetes secret, and then stop. After running the other operations, you can configure the script and run it again to discard the old password only.

MySQL Dual Password

The *dbtpasswd* script uses the MySQL Dual Password feature to first add the new password and then discard the old MySQL password after database values and secrets are changed and the pods are restarted. Both the passwords are valid until the old password is discarded.

Note

MySQL Dual Password is not supported for changing the root password.

Support for Non-cnDBTier Secrets on Different Namespace

To support changing NF passwords, that have their secrets on a namespace different from the cnDBTier namespace, *dbtpasswd* provides the `--nf-namespace` to configure the namespace in which the secrets are stored. For more information about this option, see [Configuration Options](#).

7.8.2.1 Prerequisites

Before changing the cnDBTier passwords ensure that the following prerequisites are met:

- The *dbtpasswd* script must be properly installed. For more information, see [Installing and Using dbtpasswd Script](#).
- If you are changing the replication password in a multisite setup, then ensure that the DB Replication is up and running between all of the sites in the system.

7.8.2.2 Installing and Using dbtpasswd Script

This section provides details about installing and using the *dbtpasswd* script to change cnDBTier passwords.

Installing the dbtpasswd Script

Run the following commands to source and add the bin directory containing the *dbtpasswd* program to the user's path.

```
cd Artifacts/Scripts/tools
source ./source_me
```

The system prompts you to enter the namespace and uses the same to set the DBTIER_NAMESPACE. It also sets the DBTIER_LIB environment variable with the path to the directory containing the libraries needed by *dbtpasswd*.

Using the dbtpasswd Script

You can use the dbtpasswd script in the following ways:

```
dbtpasswd [-h | --help]
dbtpasswd [OPTIONS] [SECRET...]
```

7.8.2.3 Configuration Options

This section describes the list of options that are available to configure the dbtpasswd script.

Table 7-2 Configuration Options

Option	Usage	Example
-h --help	This option is used to print the help message and exit.	dbtpasswd --help
-v --version	This option is used to print script's version.	dbtpasswd --version
--debug	This option is used to output DEBUG log message to standard error, stderr.	dbtpasswd --debug
--skip-namespace-test	This option is used to skip testing that the namespace in DBTIER_NAMESPACE exists in the current cluster.	dbtpasswd --skip-namespace-test
--rollout-watch-timeout=0s	This option is used to define the time to wait before ending the rollout status watch. Set this value to zero if you donot want to wait before ending the rollout status watch. Use a corresponding time unit to set other values. For example: 1s, 2m, 3h.	dbtpasswd --rollout-watch-timeout=0s
--no-checks	This option is used to skip verifying if replication to mate sites is UP, when updating a replication password.	dbtpasswd --no-checks
--no-discard	This option is used to change a password without discarding the old password.	To change all cnDBTier passwords, but retain the old passwords: dbtpasswd --no-discard
--discard-only	This option is used to only discard the old password.	dbtpasswd --discard-only

Table 7-2 (Cont.) Configuration Options

Option	Usage	Example
--secrets-only	This option is used to change the password in the secrets only.	dbtpasswd --secrets-only
--mysql-only	This option is used to change the passwords in MySQL only. Note: The current secret password must be the current MySQL password. Therefore, if you are using the --mysql-only and --secrets-only options to change the passwords, you must first change the MySQL password and then the secret password.	dbtpasswd --mysql-only
--secrets-and-mysql-only	This option is used to change the passwords in the secrets and MySQL only.	dbtpasswd --secrets-and-mysql-only
--restart-only	This option is used to only restart the pods configured to use cnDBTier secrets and doesn't change the passwords.	dbtpasswd --restart-only

Table 7-2 (Cont.) Configuration Options

Option	Usage	Example
<code>--nf-namespace=<namespace_where_secrets_are></code>	Non-cnDBTier secrets may be stored on a namespace different from the cnDBTier namespace. This option is used to provide the details of the namespace where the non-cnDBTier secrets are stored to support changing NF passwords. Note: All non-cnDBTier secrets must be stored in the NF namespace. All cnDBTier secrets must be stored in DBTIER_NAMESPACE.	- To change a non-DBTier password in its secret, which is on a different namespace, and in mysql: <pre>dbtpasswd --secrets-and-mysql-only --nf-namespace=other-namespace-name nf-secret-on-diff-namespace</pre> - To discard a non-DBTier old password whose secret is on a different namespace: <pre>dbtpasswd --discard-only --nf-namespace=other-namespace-name nf-secret-on-diff-namespace</pre> - To change a non-DBTier password in its secret, which is on same namespace as cnDBTier, and in mysql: <pre>dbtpasswd --secrets-and-mysql-only nf-secret-on-same-namespace</pre> - To discard a non-DBTier old password whose secret is on the same namespace as cnDBTier: <pre>dbtpasswd --discard-only nf-secret-on-same-namespace</pre>

Note

Use the `dbtpasswd --help` command to refer to more examples.

7.8.2.4 Changing All cnDBTier Passwords in a Site

Run the following command to change all the cnDBTier passwords in a cnDBTier site:

```
$ dbtpasswd
```

Sample output:

```
2022-12-23T21:54:24Z INFO - Changing password for user root
Current password:
```

```

Enter new password:
Enter new password again:
2022-12-23T21:54:39Z INFO - Changing password for user occneuser
Current password:
Enter new password:
Enter new password again:
2022-12-23T21:54:58Z INFO - Changing password for user occnerepluser
Current password:
Enter new password:
Enter new password again:
2022-12-23T21:55:05Z INFO - Getting sts and sts pod info...
2022-12-23T21:55:12Z INFO - MGM_STS="ndbmcmd"
2022-12-23T21:55:12Z INFO - MGM_REPLICAS="2"
2022-12-23T21:55:12Z INFO -
    ndbmcmd-0
    ndbmcmd-1
2022-12-23T21:55:18Z INFO - NDB_STS="ndbmt"
2022-12-23T21:55:18Z INFO - NDB_REPLICAS="2"
2022-12-23T21:55:18Z INFO -
    ndbmt-0
    ndbmt-1
2022-12-23T21:55:25Z INFO - API_STS="ndbmysqld"
2022-12-23T21:55:25Z INFO - API_REPLICAS="2"
2022-12-23T21:55:25Z INFO -
    ndbmysqld-0
    ndbmysqld-1
2022-12-23T21:55:29Z INFO - APP_STS="ndbappmysqld"
2022-12-23T21:55:29Z INFO - APP_REPLICAS="2"
2022-12-23T21:55:29Z INFO -
    ndbappmysqld-0
    ndbappmysqld-1
2022-12-23T21:55:29Z INFO - Getting deployment pod info...
2022-12-23T21:55:29Z INFO - grepping for backup-man (BAK_CHART_NAME)...
2022-12-23T21:55:36Z INFO -
    mysql-cluster-db-backup-manager-svc-7bb947f5f9-nc45s
2022-12-23T21:55:36Z INFO -
    mysql-cluster-db-backup-manager-svc
2022-12-23T21:55:36Z INFO - grepping for db-mon (MON_CHART_NAME)...
2022-12-23T21:55:42Z INFO -
    mysql-cluster-db-monitor-svc-5cff4bf789-g86rr
2022-12-23T21:55:42Z INFO -
    mysql-cluster-db-monitor-svc
2022-12-23T21:55:42Z INFO - grepping for replicat (REP_CHART_NAME)...
2022-12-23T21:55:47Z INFO -
    mysql-cluster-lfg-site-1-lfg-site-2-replication-svc-b4f8d9g6hbc
2022-12-23T21:55:47Z INFO -
    mysql-cluster-lfg-site-1-lfg-site-2-replication-svc
2022-12-23T21:55:47Z INFO - Labeling pods with dbtier-app...
pod/ndbmcmd-0 not labeled
pod/ndbmcmd-1 not labeled
pod/ndbmt-0 not labeled
pod/ndbmt-1 not labeled
pod/ndbappmysqld-0 not labeled
pod/ndbappmysqld-1 not labeled
pod/ndbmysqld-0 not labeled
pod/ndbmysqld-1 not labeled
pod/mysql-cluster-db-backup-manager-svc-7bb947f5f9-nc45s not labeled
pod/mysql-cluster-db-monitor-svc-5cff4bf789-g86rr not labeled
pod/mysql-cluster-lfg-site-1-lfg-site-2-replication-svc-b4f8d9g6hbc not labeled
2022-12-23T21:56:08Z INFO - Pods labeled with dbtier-app
2022-12-23T21:56:08Z INFO - Verifying Geo Replication to mates...
2022-12-23T21:56:18Z INFO - Geo Replication to mates is UP

```

```

2022-12-23T21:56:18Z INFO - Changing mysql password for 'root'@'localhost'...
mysql: [Warning] Using a password on the command line interface can be insecure.
2022-12-23T21:56:29Z INFO - Mysql password changed
2022-12-23T21:56:33Z INFO - Patching secret, occne-mysqldb-root-secret, with new
password
secret/occne-mysqldb-root-secret patched
2022-12-23T21:56:36Z INFO - Secret, occne-mysqldb-root-secret, patched with new password
2022-12-23T21:56:36Z INFO - Adding new mysql password for 'occneuser'@'%'...
mysql: [Warning] Using a password on the command line interface can be insecure.
2022-12-23T21:56:48Z INFO - New mysql password added
2022-12-23T21:56:54Z INFO - Patching secret, occne-secret-db-monitor-secret, with new
password
secret/occne-secret-db-monitor-secret patched
2022-12-23T21:56:58Z INFO - Secret, occne-secret-db-monitor-secret, patched with new
password
2022-12-23T21:56:58Z INFO - Adding new mysql password for 'occnerepluser'@'%'...
mysql: [Warning] Using a password on the command line interface can be insecure.
2022-12-23T21:57:10Z INFO - New mysql password added
2022-12-23T21:57:10Z INFO - Changing password in replication_info.DBTIER_REPL_SITE_INFO
table...
2022-12-23T21:57:16Z INFO - Using replication pod: mysql-cluster-lfg-site-1-lfg-site-2-
replication-svc-b4f8d9g6hbc
2022-12-23T21:57:16Z INFO - Using replication pod container: lfg-site-1-lfg-site-2-
replication-svc
2022-12-23T21:57:16Z INFO - MYSQL_REPLICATION_SITE_NAME = lfg-site-1
mysql: [Warning] Using a password on the command line interface can be insecure.
2022-12-23T21:57:28Z INFO - Password changed in replication_info.DBTIER_REPL_SITE_INFO
table
2022-12-23T21:57:35Z INFO - Patching secret, occne-replication-secret-db-replication-
secret, with new password
secret/occne-replication-secret-db-replication-secret patched
2022-12-23T21:57:38Z INFO - Secret, occne-replication-secret-db-replication-secret,
patched with new password
2022-12-23T21:57:38Z INFO - Starting rollover restarts at 2022-12-23T21:57:38Z
2022-12-23T21:57:38Z INFO - number of db-replication-svc deployments: 1
2022-12-23T21:57:38Z INFO - Patching deployment 0: mysql-cluster-lfg-site-1-lfg-site-2-
replication-svc...
deployment.apps/mysql-cluster-lfg-site-1-lfg-site-2-replication-svc patched (no change)
2022-12-23T21:57:41Z INFO - Rollout restarting mysql-cluster-lfg-site-1-lfg-site-2-
replication-svc...
deployment.apps/mysql-cluster-lfg-site-1-lfg-site-2-replication-svc restarted
...
2022-12-23T22:01:53Z INFO - ndbmt pods rollout restarted
...
2022-12-23T22:02:11Z INFO - ndbappmysqld pods rollout restarted
...
2022-12-23T22:02:27Z INFO - ndbmysqld pods rollout restarted
...
2022-12-23T22:02:48Z INFO - db-backup-manager-svc pods rollout restarted
...
2022-12-23T22:03:08Z INFO - db-monitor-svc pods rollout restarted
2022-12-23T22:03:08Z INFO - Discarding old mysql password for 'occneuser'@'%'...
mysql: [Warning] Using a password on the command line interface can be insecure.
2022-12-23T22:03:20Z INFO - Old mysql password discarded
2022-12-23T22:03:20Z INFO - Discarding old mysql password for 'occnerepluser'@'%'...
mysql: [Warning] Using a password on the command line interface can be insecure.
2022-12-23T22:03:30Z INFO - Old mysql password discarded
2022-12-23T22:03:30Z INFO - Password(s) updated successfully

```

7.8.2.5 Changing All Passwords in a Stand-Alone or Georeplication Site

This section provides the command to change all cnDBTier passwords in a stand-alone site or a site where Georeplication has been configured but the mate sites are not configured.

```
$ dbtpasswd
```

Sample output:

```
2023-10-12T21:34:05Z INFO - DBTIER_NAMESPACE = occne-cndbtier
2023-10-12T21:34:05Z INFO - Testing namespace, occne-cndbtier, exists...
2023-10-12T21:34:05Z INFO - Should be able to see namespace, occne-cndbtier, with
"kubecttl get ns -o name occne-cndbtier" - PASSED
2023-10-12T21:34:05Z INFO - Namespace, occne-cndbtier, exists
2023-10-12T21:34:05Z INFO - Changing password for user root 2023-01-03T19:06:31Z INFO -
Changing password for user root
Current password:
Enter new password:
Enter new password again:
2023-10-12T21:34:06Z INFO - Changing password for user occneuser
Current password:
Enter new password:
Enter new password again:
2023-10-12T21:34:06Z INFO - Changing password for user occnerepluser
Current password:
Enter new password:
Enter new password again: 2023-10-12T21:34:06Z INFO - Getting sts and sts pod info...
2023-10-12T21:34:06Z INFO - Getting MGM sts and sts pod info...
2023-10-12T21:34:07Z INFO - MGM_STS="ndbmcmd"
2023-10-12T21:34:07Z INFO - MGM_REPLICAS="2"
2023-10-12T21:34:07Z INFO - MGM_PODS:
    ndbmcmd-0
    ndbmcmd-1
2023-10-12T21:34:07Z INFO - Getting NDB sts and sts pod info...
2023-10-12T21:34:07Z INFO - NDB_STS="ndbmttd"
2023-10-12T21:34:07Z INFO - NDB_REPLICAS="2"
2023-10-12T21:34:07Z INFO - NDB_PODS:
    ndbmttd-0
    ndbmttd-1
2023-10-12T21:34:07Z INFO - Getting API sts and sts pod info...
2023-10-12T21:34:07Z INFO - API_STS="ndbmysqld"
2023-10-12T21:34:07Z INFO - API_REPLICAS="6"
2023-10-12T21:34:07Z INFO - API_PODS:
    ndbmysqld-0
    ndbmysqld-1
    ndbmysqld-2
    ndbmysqld-3
    ndbmysqld-4
    ndbmysqld-5
2023-10-12T21:34:07Z INFO - Getting APP sts and sts pod info...
2023-10-12T21:34:07Z INFO - APP_STS="ndbappmysqld"
2023-10-12T21:34:07Z INFO - APP_REPLICAS="2"
2023-10-12T21:34:07Z INFO - APP_PODS:
    ndbappmysqld-0
    ndbappmysqld-1
2023-10-12T21:34:07Z INFO - Getting deployment pod info...
2023-10-12T21:34:07Z INFO - grepping for backup-man (BAK_CHART_NAME)...
2023-10-12T21:34:07Z INFO - BAK_PODS:
    mysql-cluster-db-backup-manager-svc-78fdcdfd98-gpml2
2023-10-12T21:34:07Z INFO - BAK_DEPLOY:
```



```

mysql-cluster-db-backup-manager-svc
2023-10-12T21:34:07Z INFO - grepping for db-mon (MON_CHART_NAME)...
2023-10-12T21:34:07Z INFO - MON_PODS:
    mysql-cluster-db-monitor-svc-ccc9bfbfd-5z45b
2023-10-12T21:34:07Z INFO - MON_DEPLOY:
    mysql-cluster-db-monitor-svc
2023-10-12T21:34:07Z INFO - grepping for repl (REP_CHART_NAME)...
2023-10-12T21:34:08Z INFO - REP_PODS:
    mysql-cluster-lfg-site-1-lfg-site-2-replication-svc-67d96bg9z5m
    mysql-cluster-lfg-site-1-lfg-site-3-replication-svc-8f77b9xrrqt
    mysql-cluster-lfg-site-1-lfg-site-4-replication-svc-ddc647dt78q
2023-10-12T21:34:08Z INFO - REP_DEPLOY:
    mysql-cluster-lfg-site-1-lfg-site-2-replication-svc
    mysql-cluster-lfg-site-1-lfg-site-3-replication-svc
    mysql-cluster-lfg-site-1-lfg-site-4-replication-svc
2023-10-12T21:34:08Z INFO - Labeling pods with dbtier-app...
pod/ndbmgmd-0 labeled
pod/ndbmgmd-1 labeled
pod/ndbmt-d-0 labeled
pod/ndbmt-d-1 labeled
pod/ndbappmysqld-0 labeled
pod/ndbappmysqld-1 labeled
pod/ndbmysqld-0 labeled
pod/ndbmysqld-1 labeled
pod/ndbmysqld-2 labeled
pod/ndbmysqld-3 labeled
pod/ndbmysqld-4 labeled
pod/ndbmysqld-5 labeled
pod/mysql-cluster-db-backup-manager-svc-78fdcdfd98-gpml2 labeled
pod/mysql-cluster-db-monitor-svc-ccc9bfbfd-5z45b labeled
pod/mysql-cluster-lfg-site-1-lfg-site-2-replication-svc-67d96bg9z5m labeled
pod/mysql-cluster-lfg-site-1-lfg-site-3-replication-svc-8f77b9xrrqt labeled
pod/mysql-cluster-lfg-site-1-lfg-site-4-replication-svc-ddc647dt78q labeled
2023-10-12T21:34:41Z INFO - Pods labeled with dbtier-app
2023-10-12T21:34:42Z INFO - DBTIER_REPL_SITE_INFO table is empty (num_of_recs=0);
indicating SINGLE SITE SETUP...
2023-10-12T21:34:42Z INFO - Changing mysql password for 'root'@'localhost'...
2023-10-12T21:34:43Z INFO - Mysql password changed
2023-10-12T21:34:43Z INFO - Patching secret, occne-mysqldb-root-secret, with new
password
secret/occne-mysqldb-root-secret patched
2023-10-12T21:34:43Z INFO - Secret, occne-mysqldb-root-secret, patched with new password
2023-10-12T21:34:43Z INFO - Adding new mysql password for 'occneuser'@'%'...
2023-10-12T21:34:43Z INFO - New mysql password added
2023-10-12T21:34:43Z INFO - Patching secret, occne-secret-db-monitor-secret, with new
password
secret/occne-secret-db-monitor-secret patched
2023-10-12T21:34:44Z INFO - Secret, occne-secret-db-monitor-secret, patched with new
password
2023-10-12T21:34:44Z INFO - Adding new mysql password for 'occnerepluser'@'%'...
2023-10-12T21:34:44Z INFO - New mysql password added
2023-10-12T21:34:44Z INFO - Patching secret, occne-replication-secret-db-replication-
secret, with new password
secret/occne-replication-secret-db-replication-secret patched
2023-10-12T21:34:44Z INFO - Secret, occne-replication-secret-db-replication-secret,
patched with new password
2023-10-12T21:34:44Z INFO - Starting rollover restarts at 2023-10-12T21:34:44Z
2023-10-12T21:34:44Z INFO - number of db-replication-svc deployments: 3
2023-10-12T21:34:44Z INFO - Patching deployment 0: mysql-cluster-lfg-site-1-lfg-site-2-
replication-svc...
2023-10-12T21:34:47Z INFO - Patching deployment 1: mysql-cluster-lfg-site-1-lfg-site-3-
replication-svc...

```

```

2023-10-12T21:34:47Z INFO - Patching deployment 2: mysql-cluster-lfg-site-1-lfg-site-4-
replication-svc...
2023-10-12T21:34:48Z INFO - Waiting for deployment mysql-cluster-lfg-site-1-lfg-site-2-
replication-svc to rollout restart...
Waiting for deployment "mysql-cluster-lfg-site-1-lfg-site-2-replication-svc" rollout to
finish: 0 out of 1 new replicas have been updated...
Waiting for deployment "mysql-cluster-lfg-site-1-lfg-site-2-replication-svc" rollout to
finish: 0 out of 1 new replicas have been updated...
Waiting for deployment "mysql-cluster-lfg-site-1-lfg-site-2-replication-svc" rollout to
finish: 0 out of 1 new replicas have been updated...
Waiting for deployment "mysql-cluster-lfg-site-1-lfg-site-2-replication-svc" rollout to
finish: 0 of 1 updated replicas are available...
deployment "mysql-cluster-lfg-site-1-lfg-site-2-replication-svc" successfully rolled out
2023-10-12T21:36:19Z INFO - Waiting for deployment mysql-cluster-lfg-site-1-lfg-site-3-
replication-svc to rollout restart...
deployment "mysql-cluster-lfg-site-1-lfg-site-3-replication-svc" successfully rolled out
2023-10-12T21:36:19Z INFO - Waiting for deployment mysql-cluster-lfg-site-1-lfg-site-4-
replication-svc to rollout restart...
deployment "mysql-cluster-lfg-site-1-lfg-site-4-replication-svc" successfully rolled out
2023-10-12T21:36:22Z INFO - number of db-backup-manager-svc deployments: 1
2023-10-12T21:36:22Z INFO - Patching deployment 0: mysql-cluster-db-backup-manager-svc...
2023-10-12T21:36:23Z INFO - number of db-monitor-svc deployments: 1
2023-10-12T21:36:23Z INFO - Patching deployment 0: mysql-cluster-db-monitor-svc...
2023-10-12T21:36:23Z INFO - Waiting for statefulset ndbmttd to rollout restart...
Waiting for 1 pods to be ready...
Waiting for 1 pods to be ready...
waiting for statefulset rolling update to complete 1 pods at revision ndbmttd-d7bf774f6...
Waiting for 1 pods to be ready...
Waiting for 1 pods to be ready...
statefulset rolling update complete 2 pods at revision ndbmttd-d7bf774f6...
2023-10-12T21:37:49Z INFO - Waiting for statefulset ndbappmysqld to rollout restart...
statefulset rolling update complete 2 pods at revision ndbappmysqld-5b8948d47d...
2023-10-12T21:37:49Z INFO - Waiting for statefulset ndbmysqld to rollout restart...
Waiting for 1 pods to be ready...
waiting for statefulset rolling update to complete 4 pods at revision
ndbmysqld-94dd7fcbb...
Waiting for 1 pods to be ready...
Waiting for 1 pods to be ready...
waiting for statefulset rolling update to complete 5 pods at revision
ndbmysqld-94dd7fcbb...
Waiting for 1 pods to be ready...
Waiting for 1 pods to be ready...
statefulset rolling update complete 6 pods at revision ndbmysqld-94dd7fcbb...
2023-10-12T21:38:32Z INFO - Waiting for deployment mysql-cluster-db-backup-manager-svc
to rollout restart...
deployment "mysql-cluster-db-backup-manager-svc" successfully rolled out
2023-10-12T21:38:33Z INFO - Waiting for deployment mysql-cluster-db-monitor-svc to
rollout restart...
deployment "mysql-cluster-db-monitor-svc" successfully rolled out
2023-10-12T21:38:33Z INFO - Discarding old mysql password for 'occneuser'@'%'.
2023-10-12T21:38:33Z INFO - Old mysql password discarded
2023-10-12T21:38:33Z INFO - Discarding old mysql password for 'occnerepluser'@'%'.
2023-10-12T21:38:34Z INFO - Old mysql password discarded
2023-10-12T21:38:34Z INFO - Password(s) updated successfully

```

7.8.2.6 Changing All cnDBTier Passwords in Phases

This section provides the procedure to change all cnDBTier Passwords in Phases.

1. Run the following command to add the new passwords to MySQL (retain the old passwords) and change the passwords in the Kubernetes secret:

```
$ dbtpasswd --secrets-and-mysql-only
```

Sample output:

```
2022-12-24T03:41:08Z INFO - Changing password for user root
Current password:
Enter new password:
Enter new password again:
2022-12-24T03:41:08Z INFO - Changing password for user occneuser
Current password:
Enter new password:
Enter new password again:
2022-12-24T03:41:09Z INFO - Changing password for user occnerepluser
Current password:
Enter new password:
Enter new password again:
2022-12-24T03:41:09Z INFO - Getting sts and sts pod info...
2022-12-24T03:41:09Z INFO - MGM_STS="ndbmgmd"
2022-12-24T03:41:09Z INFO - MGM_REPLICAS="2"
2022-12-24T03:41:09Z INFO -
    ndbmgmd-0
    ndbmgmd-1
2022-12-24T03:41:09Z INFO - NDB_STS="ndbmttd"
2022-12-24T03:41:09Z INFO - NDB_REPLICAS="2"
2022-12-24T03:41:09Z INFO -
    ndbmttd-0
    ndbmttd-1
2022-12-24T03:41:09Z INFO - API_STS="ndbmysql"
2022-12-24T03:41:09Z INFO - API_REPLICAS="2"
2022-12-24T03:41:09Z INFO -
    ndbmysql-0
    ndbmysql-1
2022-12-24T03:41:09Z INFO - APP_STS="ndbappmysql"
2022-12-24T03:41:09Z INFO - APP_REPLICAS="2"
2022-12-24T03:41:10Z INFO -
    ndbappmysql-0
    ndbappmysql-1
2022-12-24T03:41:10Z INFO - Getting deployment pod info...
2022-12-24T03:41:10Z INFO - grepping for backup-man (BAK_CHART_NAME)...
2022-12-24T03:41:10Z INFO -
    mysql-cluster-db-backup-manager-svc-c4648f6bc-jpkt9
2022-12-24T03:41:10Z INFO -
    mysql-cluster-db-backup-manager-svc
2022-12-24T03:41:10Z INFO - grepping for db-mon (MON_CHART_NAME)...
2022-12-24T03:41:10Z INFO -
    mysql-cluster-db-monitor-svc-7d684c7c6f-gvv76
2022-12-24T03:41:10Z INFO -
    mysql-cluster-db-monitor-svc
2022-12-24T03:41:10Z INFO - grepping for replicat (REP_CHART_NAME)...
2022-12-24T03:41:10Z INFO -
    mysql-cluster-lfg-site-1-lfg-site-2-replication-svc-7b689frvfr2
2022-12-24T03:41:10Z INFO -
    mysql-cluster-lfg-site-1-lfg-site-2-replication-svc
2022-12-24T03:41:10Z INFO - Labeling pods with dbtier-app...
pod/ndbmgmd-0 not labeled
pod/ndbmgmd-1 not labeled
pod/ndbmttd-0 not labeled
pod/ndbmttd-1 not labeled
```

```

pod/ndbappmysqld-0 not labeled
pod/ndbappmysqld-1 not labeled
pod/ndbmysqld-0 not labeled
pod/ndbmysqld-1 not labeled
pod/mysql-cluster-db-backup-manager-svc-c4648f6bc-jpkt9 not labeled
pod/mysql-cluster-db-monitor-svc-7d684c7c6f-gvv76 not labeled
pod/mysql-cluster-lfg-site-1-lfg-site-2-replication-svc-7b689frvfr2 not labeled
2022-12-24T03:41:11Z INFO - Pods labeled with dbtier-app
2022-12-24T03:41:11Z INFO - Verifying Geo Replication to mates...
2022-12-24T03:41:14Z INFO - Geo Replication to mates is UP
2022-12-24T03:41:14Z INFO - Changing mysql password for 'root'@'localhost'...
mysql: [Warning] Using a password on the command line interface can be insecure.
2022-12-24T03:41:15Z INFO - Mysql password changed
2022-12-24T03:41:15Z INFO - Patching secret, occne-mysqldb-root-secret, with new
password
secret/occne-mysqldb-root-secret patched
2022-12-24T03:41:15Z INFO - Secret, occne-mysqldb-root-secret, patched with new
password
2022-12-24T03:41:15Z INFO - Adding new mysql password for 'occneuser'@'%'...
mysql: [Warning] Using a password on the command line interface can be insecure.
2022-12-24T03:41:15Z INFO - New mysql password added
2022-12-24T03:41:16Z INFO - Patching secret, occne-secret-db-monitor-secret, with
new password
secret/occne-secret-db-monitor-secret patched
2022-12-24T03:41:16Z INFO - Secret, occne-secret-db-monitor-secret, patched with new
password
2022-12-24T03:41:16Z INFO - Adding new mysql password for 'occnerepluser'@'%'...
mysql: [Warning] Using a password on the command line interface can be insecure.
2022-12-24T03:41:16Z INFO - New mysql password added
2022-12-24T03:41:16Z INFO - Changing password in
replication_info.DBTIER_REPL_SITE_INFO table...
2022-12-24T03:41:16Z INFO - Using replication pod: mysql-cluster-lfg-site-1-lfg-
site-2-replication-svc-7b689frvfr2
2022-12-24T03:41:16Z INFO - Using replication pod container: lfg-site-1-lfg-site-2-
replication-svc
2022-12-24T03:41:16Z INFO - MYSQL_REPLICATION_SITE_NAME = lfg-site-1
mysql: [Warning] Using a password on the command line interface can be insecure.
2022-12-24T03:41:17Z INFO - Password changed in
replication_info.DBTIER_REPL_SITE_INFO table
2022-12-24T03:41:17Z INFO - Patching secret, occne-replication-secret-db-replication-
secret, with new password
secret/occne-replication-secret-db-replication-secret patched
2022-12-24T03:41:17Z INFO - Secret, occne-replication-secret-db-replication-secret,
patched with new password
2022-12-24T03:41:17Z INFO - Password(s) updated successfully

```

2. Run the following command to restart the appropriate cnDBTier pods:

```
$ dbtpasswd --restart-only
```

Sample output:

```

2022-12-24T03:58:36Z INFO - Changing password for user root
Current password:
2022-12-24T03:58:41Z INFO - Changing password for user occneuser
Current password:
2022-12-24T03:58:46Z INFO - Changing password for user occnerepluser
Current password:
2022-12-24T03:58:49Z INFO - Getting sts and sts pod info...
...
2022-12-24T04:01:39Z INFO - db-monitor-svc pods rollout restarted
2022-12-24T04:01:39Z INFO - Password(s) updated successfully

```

3. Once the necessary transition is done, run the following command to discard the old MySQL cnDBTier passwords:

```
$ dbtpasswd --discard-only
```

Sample output:

```
2022-12-24T03:51:40Z INFO - Changing password for user root
Current password:
2022-12-24T03:52:04Z INFO - Changing password for user occneuser
Current password:
2022-12-24T03:52:07Z INFO - Changing password for user occnerepluser
Current password:
2022-12-24T03:52:09Z INFO - Getting sts and sts pod info...
...
2022-12-24T03:52:12Z INFO - Discarding old mysql password for 'occneuser'@'%'...
mysql: [Warning] Using a password on the command line interface can be insecure.
2022-12-24T03:52:12Z INFO - Old mysql password discarded
2022-12-24T03:52:12Z INFO - Discarding old mysql password for 'occnerepluser'@'%'...
mysql: [Warning] Using a password on the command line interface can be insecure.
2022-12-24T03:52:13Z INFO - Old mysql password discarded
2022-12-24T03:52:13Z INFO - Password(s) updated successfully
```

7.8.2.7 Changing an NF Password

This section provides a sample procedure to change an NF password when the secret is stored in an NF namespace that is different from the cnDBTier namespace.

1. Run the following commands to change the password on the secret and add a new password to MySQL.

Note

When the output prompts for the current password, enter the current password in the NF secret.

```
$ export DBTIER_NAMESPACE="dbtier_namespace"
$ dbtpasswd --secrets-and-mysql-only --nf-namespace=name-of-nf-namespace
nf-secret-in-nf-namespace
```

Sample output:

```
2022-12-15T23:27:19Z INFO - Changing password for user luis
Current password:
Enter new password:
Enter new password again:
...
2022-12-15T23:27:37Z INFO - Adding new mysql password for 'luis'@'%'...
mysql: [Warning] Using a password on the command line interface can be insecure.
2022-12-15T23:27:37Z INFO - New mysql password added
2022-12-15T23:27:37Z INFO - Patching secret, nf-secret-in-nf-namespace, with new
password
secret/nf-secret-in-nf-namespace patched
2022-12-15T23:27:37Z INFO - Secret, nf-secret-in-nf-namespace, patched with new
password
2022-12-15T23:27:37Z INFO - Password(s) updated successfully
```

2. Run the following command to discard the old password in MySQL, after restarting NF pods, adding new NF passwords on mate sites, or both.

Note

- When the output prompts for the current password, enter the current password in the NF secret.
- The NF secret must be present on nf-namespace and the corresponding MySQL user must be present with the corresponding password.

```
$ dbtpasswd --discard-only --nf-namespace=name-of-nf-namespace nf-secret-
in-nf-namespace
```

Sample output:

```
2022-12-15T23:28:48Z INFO - Changing password for user luis
Current password:
...
2022-12-15T23:28:53Z INFO - Discarding old mysql password for 'luis'@'%'...
mysql: [Warning] Using a password on the command line interface can be
insecure.
2022-12-15T23:28:54Z INFO - Old mysql password discarded
2022-12-15T23:28:54Z INFO - Password(s) updated successfully
```

7.8.3 Modifying cnDBTier Backup Encryption Password

This section provides the procedure to modify the cnDBTier backup encryption password.

1. Get the existing backup encryption password (occne-backup-encryption-secret):

```
$ kubectl get secret occne-backup-encryption-secret -n
<cndbtier_namespace> -o jsonpath="{.data.backup_encryption_password}" |
base64 --decode
```

For example,

```
$ kubectl get secret occne-backup-encryption-secret -n occne-cndbtier -o
jsonpath="{.data.backup_encryption_password}" | base64 --decode
```

Note

Skip this step if you already know the existing occne-backup-encryption-secret password.

2. Delete the existing backup encryption secret:

```
$ kubectl -n <cndbtier_namespace> delete secret occne-backup-encryption-
secret
```

For example,

```
$ kubectl -n occne-cndbtier delete secret occne-backup-encryption-secret
```

3. Run the following command to create secret with a new password. For information about the password policies, see the "Creating Secrets" section in *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide*.

```
$ kubectl -n <cndbtier_namespace> create secret generic occne-backup-encryption-secret --from-literal="backup_encryption_password=<new_password>"
```

For example,

```
$ kubectl -n occne-cndbtier create secret generic occne-backup-encryption-secret --from-literal="backup_encryption_password=NextGenCne"
```

4. Scale down the replication service and backup manager service deployments. After scaling down, wait until all the replication service and backup manager pods are DOWN:

```
$ kubectl -n <cndbtier_namespace> get deployments | egrep 'repl' | awk '{print $1}' | xargs -L1 -r kubectl -n <cndbtier_namespace> scale deployment --replicas=0
$ kubectl -n <cndbtier_namespace> get deployments | egrep 'db-backup-manager-svc' | awk '{print $1}' | xargs -L1 -r kubectl -n <cndbtier_namespace> scale deployment --replicas=0
```

For example,

```
$ kubectl -n occne-cndbtier get deployments | egrep 'repl' | awk '{print $1}' | xargs -L1 -r kubectl -n occne-cndbtier scale deployment --replicas=0
$ kubectl -n occne-cndbtier get deployments | egrep 'db-backup-manager-svc' | awk '{print $1}' | xargs -L1 -r kubectl -n occne-cndbtier scale deployment --replicas=0
```

5. Scale up the replication service and backup manager service deployments. After scaling up, wait until all the replication service and backup manager pods are UP:

```
$ kubectl -n <cndbtier_namespace> get deployments | egrep 'repl' | awk '{print $1}' | xargs -L1 -r kubectl -n <cndbtier_namespace> scale deployment --replicas=1
$ kubectl -n <cndbtier_namespace> get deployments | egrep 'db-backup-manager-svc' | awk '{print $1}' | xargs -L1 -r kubectl -n <cndbtier_namespace> scale deployment --replicas=1
```

For example,

```
$ kubectl -n occne-cndbtier get deployments | egrep 'repl' | awk '{print $1}' | xargs -L1 -r kubectl -n occne-cndbtier scale deployment --replicas=1
$ kubectl -n occne-cndbtier get deployments | egrep 'db-backup-manager-
```



```
svc' | awk '{print $1}' | xargs -L1 -r kubectl -n occne-cndbtier scale
deployment --replicas=1
```

7.8.4 Modifying SSH Keys for Transferring Backups

This section provides the procedure to modify Secure Shell (SSH) keys for securely transferring cnDBTier backups.

1. Perform the following steps to move the existing SSH keys to the backup directory and delete the existing SSH secrets:

- a. Perform the following steps to move the existing SSH keys to the backup directory:

- i. Identify the location of the existing SSH keys:

```
$ ls /var/occne/cluster/${OCCNE_CLUSTER}/cndbtierssh/
```

For example:

```
$ ls /var/occne/cluster/${OCCNE_CLUSTER}/cndbtierssh/
```

Sample output:

```
cndbtier_id_rsa cndbtier_id_rsa.pub
```

- ii. Move the existing SSH keys from the location identified in the previous step to the backup directory:

```
$ sshBackupDateTime=$(date +%m-%d-%y-%H-%M-%S")
$ mkdir -p /var/occne/cluster/${OCCNE_CLUSTER}/cndbtierssh/${sshBackupDateTime}
$ mv <SSH private key file name> /var/occne/cluster/${OCCNE_CLUSTER}/cndbtierssh/${sshBackupDateTime}/
$ mv <SSH public key file name> /var/occne/cluster/${OCCNE_CLUSTER}/cndbtierssh/${sshBackupDateTime}/
```

For example:

```
$ sshBackupDateTime=$(date +%m-%d-%y-%H-%M-%S")
$ mkdir -p /var/occne/cluster/${OCCNE_CLUSTER}/cndbtierssh/${sshBackupDateTime}
$ mv /var/occne/cluster/${OCCNE_CLUSTER}/cndbtierssh/cndbtier_id_rsa /var/occne/cluster/${OCCNE_CLUSTER}/cndbtierssh/${sshBackupDateTime}/
$ mv /var/occne/cluster/${OCCNE_CLUSTER}/cndbtierssh/cndbtier_id_rsa.pub /var/occne/cluster/${OCCNE_CLUSTER}/cndbtierssh/${sshBackupDateTime}/
```

- b. Perform the following steps to delete the existing SSH or Secure File Transfer Protocol (SFTP) secrets from the current cnDBTier cluster:

- i. Identify the SSH or SFTP secrets from the current cluster by running the following command:

```
$ kubectl get secrets --namespace=<namespace of cnDBTier Cluster> |
grep ssh
```


For example:

```
$ kubectl get secrets --namespace=cluster1 | grep ssh
```

Sample output:

```
cndbtier-ssh-private-key
Opaque                1          7d
cndbtier-ssh-public-key
Opaque                1          7d
```

- ii. Delete the SSH or SFTP secrets from the current cluster by running the following commands:

```
$ kubectl delete secret cndbtier-ssh-private-key --
namespace=<namespace of cnDBTier Cluster>
$ kubectl delete secret cndbtier-ssh-public-key --
namespace=<namespace of cnDBTier Cluster>
```

Example to delete a private key:

```
$ kubectl delete secret cndbtier-ssh-private-key --
namespace=cluster1
```

Sample output:

```
secret "cndbtier-ssh-private-key" deleted
```

Example to delete a public key:

```
$ kubectl delete secret cndbtier-ssh-public-key --namespace=cluster1
```

Sample output:

```
secret "cndbtier-ssh-public-key" deleted
```

2. Create the SSH keys by running the following commands:

```
$ mkdir -p -m 0700 /var/occne/cluster/${OCCNE_CLUSTER}/cndbtierssh
$ ssh-keygen -b 4096 -t rsa -C "cndbtier key" -f "/var/occne/cluster/${OCCNE_CLUSTER}/cndbtierssh/cndbtier_id_rsa" -q -N ""
```

For more information about creating SSH secrets, see "Creating SSH Keys" in *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide*.

3. Create the SFTP secrets by running the following commands:

```
$ kubectl create secret generic cndbtier-ssh-private-key --from-
file=id_rsa=/var/occne/cluster/${OCCNE_CLUSTER}/cndbtierssh/
cndbtier_id_rsa -n ${OCCNE_NAMESPACE}
$ kubectl create secret generic cndbtier-ssh-public-key --from-
```

```
file=id_rsa.pub=/var/ocne/cluster/${OCCNE_CLUSTER}/cndbtierssh/
cndbtier_id_rsa.pub -n ${OCCNE_NAMESPACE}
```

For more information about creating SFTP secrets, see step 2 of "Creating Secrets" in *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide*.

4. Perform the following steps to restart the data nodes sequentially in an ascending order (ndbmt-d-0, ndbmt-d-1, and so on):
 - a. Identify the list of data pods in the cnDBTier cluster:

```
$ kubectl get pods --namespace=<namespace of cnDBTier Cluster> | grep
'ndbmt-d'
```

For example, use the following command to identify the list of data pods in cnDBTier cluster1:

```
$ kubectl get pods --namespace=cluster1 | grep 'ndbmt-d'
```

Sample output:

```
ndbmt-d-0                                3/3
Running   0                14m
ndbmt-d-1                                3/3
Running   0                13m
```

- b. Delete the first data pod of the cnDBTier cluster:

```
$ kubectl delete pod ndbmt-d-0 --namespace=<namespace of cnDBTier
Cluster>
```

For example, use the following command to delete the first data pod of cnDBTier cluster1:

```
$ kubectl delete pod ndbmt-d-0 --namespace=cluster1
```

Sample output:

```
pod "ndbmt-d-0" deleted
```

- c. Wait for the first data pod to come up and verify if the pod is up by running the following command:

```
$ kubectl get pods --namespace=<namespace of cnDBTier Cluster> | grep
'ndbmt-d'
```

For example:

```
$ kubectl get pods --namespace=cluster1 | grep 'ndbmt-d'
```

Sample output:

```
ndbmt-d-0          3/3
Running   0        65s
ndbmt-d-1          3/3
Running   0        13m
```

- d. Check the status of the cnDBTier cluster:

```
$ kubectl -n <namespace of cnDBTier Cluster> exec -it ndbm-gmd-0 --
ndb_mgm -e show
```

For example, use the following command to check the status of cnDBTier cluster1:

```
$ kubectl -n cluster1 exec -it ndbm-gmd-0 -- ndb_mgm -e show
```

Sample output:

```
Connected to Management Server at: localhost:1186 Cluster Configuration
-----
[ndbd(NDB)]      2 node(s)
id=1   @10.233.85.92  (mysql-8.4.3 ndb-8.4.3, Nodegroup: 0)
id=2   @10.233.114.33 (mysql-8.4.3 ndb-8.4.3, Nodegroup: 0, *)

[ndb_mgm(MGM)]  2 node(s)
id=49   @10.233.65.167 (mysql-8.4.3 ndb-8.4.3)
id=50   @10.233.127.115 (mysql-8.4.3 ndb-8.4.3)

[mysqld(API)]   8 node(s)
id=56   @10.233.120.210 (mysql-8.4.3 ndb-8.4.3)
id=57   @10.233.124.93  (mysql-8.4.3 ndb-8.4.3)
id=70   @10.233.127.117 (mysql-8.4.3 ndb-8.4.3)
id=71   @10.233.85.93   (mysql-8.4.3 ndb-8.4.3)
id=222  (not connected, accepting connect from any host)
id=223  (not connected, accepting connect from any host)
id=224  (not connected, accepting connect from any host)
id=225  (not connected, accepting connect from any host)
```

Note

Node IDs 222 to 225 in the sample output are shown as "not connected" as these are added as empty slot IDs that are used for georeplication recovery.

- e. Repeat steps a through d to delete the remaining data pods (ndbmt-d-1, ndbmt-d-2, and so on).
5. Perform the following steps to restart all the replication services of the current cluster:
- a. Identify the replication service running in the current cluster:

```
$ kubectl get pods --namespace=<namespace of cnDBTier Cluster>
```

For example:

```
$ kubectl get pods --namespace=cluster1
```

Sample output:

```
NAME
READY   STATUS    RESTARTS      AGE
mysql-cluster-cluster1-cluster2-replication-svc-5d4b8fd685tshzd
1/1     Running   1 (13h ago)    13h
```

- b. Delete all the replication service pods of the current cluster:

```
$ kubectl delete pod <replication service pod name> --
namespace=<namespace of cnDBTier Cluster>
```

For example,

```
$ kubectl delete pod mysql-cluster-cluster1-cluster2-replication-
svc-5d4b8fd685tshzd --namespace=cluster1
```

Sample output:

```
pod "mysql-cluster-cluster1-cluster2-replication-svc-5d4b8fd685tshzd"
delete
```

6. Follow steps 1 through 5 to replace the SSH or SFTP keys and secrets on the other georeplication cnDBTier clusters.

Note

The SSH or SFTP secrets must be the same across all the georeplication cnDBTier clusters.

7.8.5 Modifying Transparent Data Encryption Password

This section provides the procedure to modify the Transparent Data Encryption (TDE) password.

1. Get the existing TDE password (occne-tde-encrypted-filesystem-secret):

```
$ kubectl -n <cnDbTier_namespace> get secret occne-tde-encrypted-
filesystem-secret -o jsonpath="{.data.filesystem-password}" | base64 --
decode
```

For example,

```
$ kubectl -n occne-cnDbTier get secret occne-tde-encrypted-filesystem-
secret -o jsonpath="{.data.filesystem-password}" | base64 --decode
```

2. Delete the existing TDE secret:

```
$ kubectl -n <cnDbTier_namespace> delete secret occne-tde-encrypted-filesystem-secret
```

For example,

```
$ kubectl -n occne-cnDbTier delete secret occne-tde-encrypted-filesystem-secret
```

3. Run the following command to create secret with a new password. For information about the password policies, see the "Creating Secrets" section in *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide*.

```
$ kubectl -n <cnDbTier_namespace> create secret generic occne-tde-encrypted-filesystem-secret --from-literal="filesystem-password=<new_tde-encryption-password>"
```

For example,

```
$ kubectl -n occne-cnDbTier create secret generic occne-tde-encrypted-filesystem-secret --from-literal="filesystem-password=NextGenCne"
```

4. Perform a cnDBTier upgrade by following the procedure in *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide*.

7.9 Modifying HTTPS Certificates

This section provides the procedure to modify HTTPS certificates.

1. Create a new certificate by following the sample procedure provided in the "Creating HTTPS or TLS Certificates for Encrypted Connection" section of *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide*.
2. Run the following commands to take a backup of the existing HTTPS CA certificates from secret of all the sites where you want to modify the certificates:

Note

If the file name has a dot either in the extension or in its name, use an escape character \ before the dot.

```
$ kubectl get secret cnDbTier-https-trust-store-secret -n <cnDbTier_namespace> -o jsonpath="{.data.<ca_certificate_file_name>}" | base64 --decode
```

where, <cnDbTier_namespace> is the name of the cnDBTier namespace.
For example:

```
$ kubectl get secret cnDbTier-https-trust-store-secret -n occne-cnDbTier -o jsonpath="{.data.combine-ca\\.pem}" | base64 --decode
```

3. Update the old HTTPS CA certificate(s) of the cnDBTier cluster where you want to modify the certificates:

```
$ TLS_CA_CERT=$(base64 < "<updated-certificate-name>" | tr -d '\n')
$ kubectl -n <namespace> patch secret cndbtier-https-trust-store-secret -p
"{\"data\":{\"<ca_certificate>\":\"${TLS_CA_CERT}\"}}"
```

For example:

```
$ TLS_CA_CERT=$(base64 < "combine-ca.pem" | tr -d '\n')
$ kubectl -n occne-cndbtier patch secret cndbtier-https-trust-store-secret
-p "{\"data\":{\"combine-ca.pem\":\"${TLS_CA_CERT}\"}}"
```

4. Take the backup of the existing HTTPS client certificate & client key of the cnDBTier cluster.

Note

If the file name has a dot either in the extension or in its name, use an escape character \ before the dot.

- a. Get the HTTPS client certificate.

```
kubectl get secret cndbtier-https-client-secret -n
<cndbtier_namespace> -o
jsonpath="{.data.<client_certificate_file_name>}" | base64 --decode
```

- b. Get the client certificate key.

```
kubectl get secret cndbtier-https-client-secret -n
<cndbtier_namespace> -o
jsonpath="{.data.<client_certificate_key_file_name>}" | base64 --decode
```

For example:

```
$ kubectl get secret cndbtier-https-client-secret -n occne-cndbtier -o
jsonpath="{.data.client-cert\\.pem}" | base64 --decode
$ kubectl get secret cndbtier-https-client-secret -n occne-cndbtier -o
jsonpath="{.data.client-key\\.pem}" | base64 --decode
```

5. Update the existing HTTPS client certificate & client key secret of cnDBTier cluster where you want to modify the certificates:

```
$ CLIENT_CERT=$(base64 < "<updated-certificate-name>" | tr -d '\n')
$ CLIENT_KEY=$(base64 < "<updated-certificate-key-name>" | tr -d '\n')
$ kubectl -n <namespace> patch secret cndbtier-https-client-secret -p
"{\"data\":{\"<client_certificate>\":\"${CLIENT_CERT}\"
\", \"<client_certificate_key>\":\"${CLIENT_KEY}\"}}"
```

For example:

```
$ CLIENT_CERT=$(base64 < "client_cert.pem" | tr -d '\n')
$ CLIENT_KEY=$(base64 < "client_key.pem" | tr -d '\n')
$ kubectl -n <namespace> patch secret cndbtier-https-client-secret -p
"{\"data\":{\"client_cert.pem\":\"${CLIENT_CERT}\",\"client_key.pem\":\"${CLIENT_KEY}\"}}"
```

6. Take the backup of the existing HTTPS secret cndbtier-https-cert-cred which hosts the Certificate Type, Alias for Server Certificate and Client Certificate, as well as password for generating the keystores.

```
$ kubectl -n <cndbtier_namespace> get secret cndbtier-https-cert-cred -o
jsonpath="{.data.keystorepassword}" | base64 --decode
$ kubectl -n <cndbtier_namespace> get secret cndbtier-https-cert-cred -o
jsonpath="{.data.keystoretype}" | base64 --decode
$ kubectl -n <cndbtier_namespace> get secret cndbtier-https-cert-cred -o
jsonpath="{.data.keyalias}" | base64 --decode
$ kubectl -n <cndbtier_namespace> get secret cndbtier-https-cert-cred -o
jsonpath="{.data.clientkeyalias}" | base64 --decode
```

For example:

```
$ kubectl -n occne-cndbtier get secret cndbtier-https-cert-cred -o
jsonpath="{.data.keystorepassword}" | base64 --decode
$ kubectl -n occne-cndbtier get secret cndbtier-https-cert-cred -o
jsonpath="{.data.keystoretype}" | base64 --decode
$ kubectl -n occne-cndbtier get secret cndbtier-https-cert-cred -o
jsonpath="{.data.keyalias}" | base64 --decode
$ kubectl -n occne-cndbtier get secret cndbtier-https-cert-cred -o
jsonpath="{.data.clientkeyalias}" | base64 --decode
```

7. Delete the existing HTTPS secret cndbtier-https-cert-cred which hosts the Certificate Type, Alias for Server Certificate and Client Certificate, as well as password for generating the keystores.

```
$ KEYALIAS=$(echo -n '<keyalias>' | base64)"
$ KEYCLIENTALIAS=$(echo -n '<clientalias>' | base64)"
$ KEYSTORE_TYPE=$(echo -n '<keystoretype>' | base64)"
$ KEYSTORE_PASSWORD=$(echo -n '<keystorepassword>' | base64)"
$ kubectl -n <namespace> patch secret cndbtier-https-cert-cred -p
"{\"data\":{\"<keyalias>\":\"${KEYALIAS}\",\"<keystoretype>\":\"${KEYSTORE_TYPE}\",\"<keystorepassword>\":\"${KEYSTORE_PASSWORD}\",\"<clientkeyalias>\":\"${CLIENT_ALIAS}\"}}"
```

For example:

```
$ KEYALIAS=$(echo -n 'serverkeyalias' | base64)"
$ KEYCLIENTALIAS=$(echo -n 'clientalias' | base64)"
$ KEYSTORE_TYPE=$(echo -n 'PKCS12' | base64)"
$ KEYSTORE_PASSWORD=$(echo -n 'NextGenCne' | base64)"
$ kubectl -n cluster1 patch secret cndbtier-https-cert-cred -p "{\"data\":{\"keyalias\":\"${KEYALIAS}\",\"keystoretype\":\"${KEYSTORE_TYPE}\"}}
```

```
\", \"keystorepassword\": \"${KEYSTORE_PASSWORD}\", \"clientkeyalias\": \"${KEYSTORE_PASSWORD}\"}]\"}
```

Note

- The server-cert.pem, client-cert.pem files can be modified when required.
- Modifying ca-cert.pem file must be done in a maintenance window and after creating the new secrets with new ca-cert.pem file, restart the replication service and monitor service of each site where the certificates are modified.

7.10 Modifying Remote Server Configurations for Secure Transfer of Backups

cnDBTier requires the following remote server configurations to securely transfer backups:

- IP address or Fully Qualified Domain Name (FQDN)
- Secure File Transfer Protocol (SFTP) port
- The path where the backups are to be securely transferred
- SSH key and username that are configured in the form of secrets

This section provides the procedure to modify remote server configurations in cnDBTier.

1. Perform the following steps to update the remote server configurations such as remoteserverip, remoteserverport, and remoteserverpath:
 - a. Locate the custom_values.yaml file.
 - b. Edit the configurations as per your requirement.

Note

The faultrecoverybackuptransfer parameter is set to *false* by default in the custom_values.yaml file. Therefore, even if the /global/remotetransfer/enable parameter is set to *true*, the system will not transfer the backups taken during fault recovery to the remote server. If you want the backups taken for fault recovery to be transferred to the remote server, set the faultrecoverybackuptransfer parameter to *true*.

The following code block provides a template to configure the remote transfer parameters:

```
global:
  ....
  ....
  remotetransfer:
    enable: true
    faultrecoverybackuptransfer: true
    remoteserverip: "< IP address of the remote server>"
    remoteserverport: "<SFTP Port of the remote server>"
```



```
remoteserverpath: "<path where cndbtier backups will be stored>"
```

For example:

```
global:
....
....
remotetransfer:
  enable: true
  faultrecoverybackuptransfer: true
  remoteserverip: "10.75.216.8"
  remoteserverport: "2022"
  remoteserverpath: "/var/occnedb"
```

2. Perform the following steps to change the username of the remote server:

- a. Delete the `ocne-remote-server-username-secret` secret by running the following command:

```
$ kubectl -n <namespace> delete secret ocne-remote-server-username-secret
```

Where, `<namespace>` is the namespace of `cnDBTier`.

For example,

```
$ kubectl -n ocne-cndbtier delete secret ocne-remote-server-username-secret
```

- b. Recreate the secret with a new username by running the following command:

```
$ kubectl -n <namespace> create secret generic ocne-remote-server-username-secret --from-literal="remote_server_user_name=<user_name>"
```

Where,

- `<namespace>` is the namespace of `cnDBTier`.
- `<user_name>` is the new username of the remote server.

For example,

```
$ kubectl -n ocne-cndbtier create secret generic ocne-remote-server-username-secret --from-literal="remote_server_user_name=cndbtierbackupserver"
```

As per the example, the username of the remote server is `cndbtierbackupserver`.

3. Perform the following steps to update the SSH key of the remote server:

- a. Delete the `ocne-remoteserver-privatekey-secret` secret by running the following command:

```
$ kubectl -n <namespace> delete secret ocne-remoteserver-privatekey-secret
```

Where, <namespace> is the namespace of cnDBTier.

For example,

```
$ kubectl -n occne-cndbtier delete secret occne-remoteserver-privatekey-secret
```

- b. Copy the private SSH key of the remote server and save it in a file.
- c. Create the secret by using the file created in the previous step:

```
$ kubectl -n <namespace> create secret generic occne-remoteserver-privatekey-secret --from-file=id_rsa=<private key path>
```

Where,

- <namespace> is the namespace of cnDBTier
- <private key path> is the path to the private key file created in step b. The file path must include the file name.

For example,

```
$ kubectl -n occne-cndbtier create secret generic occne-remoteserver-privatekey-secret --from-file=id_rsa=/var/occne/cluster/dbtier/remoteserver_id_rsa
```

As per the example, the private key of the remote server is saved in the /var/occne/cluster/dbtier/ directory with the file name remoteserver_id_rsa.

4. Upgrade the cnDBTier cluster to update all the configurations changed in steps 1, 2, and 3:

```
$ helm upgrade mysql-cluster occndbtier -f occndbtier/custom_values.yaml -n <namespace>
```

Where, <namespace> is the namespace of cnDBTier.

5. If the replication service pods are not restarted after helm upgrade in the previous step, run the following command to manually delete the replication service pods:

```
$ kubectl -n <cndbtier_namespace> delete pod <replication_pod_name>
```

7.11 Checking the Georeplication Status Between Clusters

This section describes the following procedures to check the georeplication status of a cnDBTier cluster in a site with a remote site:

1. [Checking the Georeplication Status of cnDBTier cluster1 with Remote Sites](#)
2. [Checking the Georeplication Status of cnDBTier cluster2 with Remote Sites](#)
3. [Checking the Georeplication Status of cnDBTier cluster3 with Remote Sites](#)
4. [Checking the Georeplication Status of cnDBTier cluster4 with Remote Sites](#)

Note

Replace the name of the cnDBTier cluster namespaces(cluster1, cluster2, cluster3, and cluster4) with the actual name of your namespaces in each of the commands in this section.

Checking the Georeplication Status of cnDBTier cluster1 with Remote Sites

Perform the following steps to check the georeplication status of cnDBTier cluster1 with remote sites (cnDBTier cluster2, cnDBTier cluster3, and cnDBTier cluster4):

Note

The following commands and examples are applicable for a single replication channel group only. If multiple replication channel groups are enabled, then check the georeplication status in cnDBTier cluster1 with remote sites (cnDBTier cluster2, cnDBTier cluster3, and cnDBTier cluster4) for every replication group.

1. Run the following commands to check the georeplication status of cnDBTier cluster1 with respect to cnDBTier cluster2:

```
$ kubectl -n cluster1 exec -it ndbmysqld-0 -- bash
$ mysql -h 127.0.0.1 -uroot -p
Password:
mysql> select * from replication_info.DBTIER_REPLICATION_CHANNEL_INFO
where remote_server_id in (2000, 2001) AND remote_site_name = "cluster2";
```

Sample output:

```
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
| remote_site_name | remote_server_id | channel_id | remote_signaling_ip |
| role            | start_epoch      | site_name  | server_id           | start_ts          |
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
| cluster2        | 2000             | 8970       | 10.233.0.147        | 2022-04-04 19:10:44 |
ACTIVE | NULL | cluster1 | 1000 | 2022-04-04 19:10:44 |
| cluster2        | 2001             | 4597       | 10.233.9.35         | 2022-04-04 19:10:45 |
STANDBY | NULL | cluster1 | 1001 | 2022-04-04 19:10:45 |
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

2. Run the following commands to check if replication is turned on in the ACTIVE Replication channel of cnDBTier cluster1 with respect to cnDBTier cluster2.

Note

When replication is turned on, the Replica_IO_Running and Replica_SQL_Running parameters are set to Yes.

The following example describes the commands to check the replication status of the ACTIVE replication channel:

```
$ kubectl -n cluster1 exec -it ndbmysqld-0 -- bash
$ mysql -h 127.0.0.1 -uroot -p
Password:
mysql> SHOW REPLICA STATUS\G;
```

Sample output:

```
***** 1. row *****
      Replica_IO_State: Waiting for master to send event
      Source_Host: 10.233.0.147
      Source_User: occnerepluser
      Source_Port: 3306
      Connect_Retry: 60
      Source_Log_File: mysql-bin.000007
      Read_Source_Log_Pos: 16442
      Relay_Log_File: mysql-relay-bin.000002
      Relay_Log_Pos: 11203
      Relay_Source_Log_File: mysql-bin.000007
      Replica_IO_Running: Yes
      Replica_SQL_Running: Yes
      ....
      ....
      ....
      Replica_SQL_Running_State: Replica has read all relay log; waiting for
more updates
      Source_Retry_Count: 86400
      ....
      ....
```

3. Run the following commands to check if replication is turned off in the STANDBY Replication channel of cnDBTier cluster1 with respect to cnDBTier cluster2.

Note

When replication is turned off, the `Replica_IO_Running` and `Replica_SQL_Running` parameters are set to No.

The following example describes the commands to check the replication status of the STANDBY replication channel:

```
$ kubectl -n cluster1 exec -it ndbmysqld-1 -- bash
$ mysql -h 127.0.0.1 -uroot -p
Password:
mysql> SHOW REPLICA STATUS\G;
```

Sample output:

```
***** 1. row *****
      Replica_IO_State:
      Source_Host: 10.233.9.35
```

```

Source_User: occnerepluser
Source_Port: 3306
Connect_Retry: 60
Source_Log_File: mysql-bin.000005
Read_Source_Log_Pos: 26970
Relay_Log_File: mysql-relay-bin.000002
Relay_Log_Pos: 2228
Relay_Source_Log_File: mysql-bin.000005
Replica_IO_Running: No
Replica_SQL_Running: No

```

- Run the following commands to check the georeplication status of cnDBTier cluster1 with respect to cnDBTier cluster3:

```

$ kubectl -n cluster1 exec -it ndbmysqld-2 -- bash
$ mysql -h 127.0.0.1 -uroot -p
Password:
mysql> select * from replication_info.DBTIER_REPLICATION_CHANNEL_INFO
where remote_server_id in (3000, 3001) AND remote_site_name = "cluster3";

```

Sample output:

```

+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
| remote_site_name | remote_server_id | channel_id | remote_signaling_ip |
role      | start_epoch | site_name | server_id | start_ts          |
+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
| cluster3        |                  | 3000      | 5533      | 10.233.16.34      |
ACTIVE | NULL | cluster1 | 1002      | 2022-04-04 19:10:44 |
| cluster3        |                  | 3001      | 7918      | 10.233.41.15      |
STANDBY | NULL | cluster1 | 1003      | 2022-04-04 19:10:45 |
+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

```

- Run the following commands to check if replication is turned on in the ACTIVE Replication channel of cnDBTier cluster1 with respect to cnDBTier cluster3.

Note

When replication is turned on, the `Replica_IO_Running` and `Replica_SQL_Running` parameters are set to Yes.

The following example describes the commands to check the replication status of the ACTIVE replication channel:

```

$ kubectl -n cluster1 exec -it ndbmysqld-2 -- bash
$ mysql -h 127.0.0.1 -uroot -p
Password:
mysql> SHOW REPLICA STATUS\G;

```

Sample output:

```
***** 1. row *****
      Replica_IO_State: Waiting for master to send event
      Source_Host: 10.233.16.34
      Source_User: occnerepluser
      Source_Port: 3306
      Connect_Retry: 60
      Source_Log_File: mysql-bin.000007
      Read_Source_Log_Pos: 18542
      Relay_Log_File: mysql-relay-bin.000002
      Relay_Log_Pos: 11203
      Relay_Source_Log_File: mysql-bin.000007
      Replica_IO_Running: Yes
      Replica_SQL_Running: Yes
      ....
      ....
      ....
      Replica_SQL_Running_State: Replica has read all relay log; waiting for
more updates
      Source_Retry_Count: 86400
      ....
      ....
```

6. Run the following commands to check if replication is turned off in the STANDBY Replication channel of cnDBTier cluster1 with respect to cnDBTier cluster3.

Note

When replication is turned off, the `Replica_IO_Running` and `Replica_SQL_Running` parameters are set to No.

The following example describes the commands to check the replication status of the STANDBY replication channel:

```
$ kubectl -n cluster1 exec -it ndbmysqld-3 -- bash
$ mysql -h 127.0.0.1 -uroot -p
Password:
mysql> SHOW REPLICA STATUS\G;
```

Sample output:

```
***** 1. row *****
      Replica_IO_State:
      Source_Host: 10.233.41.15
      Source_User: occnerepluser
      Source_Port: 3306
      Connect_Retry: 60
      Source_Log_File: mysql-bin.000006
      Read_Source_Log_Pos: 26421
      Relay_Log_File: mysql-relay-bin.000002
      Relay_Log_Pos: 2228
      Relay_Source_Log_File: mysql-bin.000005
```

Replica_IO_Running: No
Replica_SQL_Running: No

7. Run the following commands to check the georeplication status of cnDBTier cluster1 with respect to cnDBTier cluster4:

```
$ kubectl -n cluster1 exec -it ndbmysqld-4 -- bash
$ mysql -h 127.0.0.1 -uroot -p
Password:
mysql> select * from replication_info.DBTIER_REPLICATION_CHANNEL_INFO
where remote_server_id in (4000, 4001) AND remote_site_name = "cluster4";
```

Sample output:

```
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
| remote_site_name | remote_server_id | channel_id | remote_signaling_ip |
role      | start_epoch | site_name | server_id | start_ts          |
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
| cluster4        |                  | 4000      | 5909      | 10.233.57.61      |
ACTIVE | NULL | cluster1 | 1004      | 2022-04-04 19:10:44 |
| cluster4        |                  | 4001      | 1937      | 10.233.54.145     |
STANDBY | NULL | cluster1 | 1005      | 2022-04-04 19:10:45 |
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

8. Run the following commands to check if replication is turned on in the ACTIVE Replication channel of cnDBTier cluster1 with respect to cnDBTier cluster4.

Note

When replication is turned on, the Replica_IO_Running and Replica_SQL_Running parameters are set to Yes

The following example describes the commands to check the replication status of the ACTIVE replication channel::

```
$ kubectl -n cluster1 exec -it ndbmysqld-4 -- bash
$ mysql -h 127.0.0.1 -uroot -p
Password:
mysql> SHOW REPLICA STATUS\G;
```

Sample output:

```
***** 1. row *****
      Replica_IO_State: Waiting for master to send event
      Source_Host: 10.233.57.61
      Source_User: occnerepluser
      Source_Port: 3306
      Connect_Retry: 60
      Source_Log_File: mysql-bin.000004
```

```

Read_Source_Log_Pos: 185
Relay_Log_File: mysql-relay-bin.000002
Relay_Log_Pos: 11203
Relay_Source_Log_File: mysql-bin.000007
Replica_IO_Running: Yes
Replica_SQL_Running: Yes
....
....
....
Replica_SQL_Running_State: Replica has read all relay log; waiting for
more updates
Source_Retry_Count: 86400
....
....

```

9. Run the following commands to check if replication is turned off in the STANDBY Replication channel of cnDBTier cluster1 with respect to cnDBTier cluster4.

Note

When replication is turned off, the Replica_IO_Running and Replica_SQL_Running parameters are set to No.

The following example describes the commands to check the replication status of the STANDBY replication channel:

```

$ kubectl -n cluster1 exec -it ndbmysqld-5 -- bash
$ mysql -h 127.0.0.1 -uroot -p
Password:
mysql> SHOW REPLICA STATUS\G;

```

Sample output:

```

***** 1. row *****
Replica_IO_State:
Source_Host: 10.233.54.145
Source_User: occnerepluser
Source_Port: 3306
Connect_Retry: 60
Source_Log_File: mysql-bin.000005
Read_Source_Log_Pos: 26420
Relay_Log_File: mysql-relay-bin.000002
Relay_Log_Pos: 2228
Relay_Source_Log_File: mysql-bin.000005
Replica_IO_Running: No
Replica_SQL_Running: No

```

Checking the Georeplication Status of cnDBTier cluster2 with Remote Sites

Perform the following steps to check the georeplication status of cnDBTier cluster2 with remote sites (cnDBTier cluster1, cnDBTier cluster3, and cnDBTier cluster4):

Note

The following commands and examples are applicable for a single replication channel group only. If multiple replication channel groups are enabled, then check the georeplication status in cnDBTier cluster2 with remote sites (cnDBTier cluster1, cnDBTier cluster3, and cnDBTier cluster4) for every replication group.

1. Run the following commands to check the georeplication status of cnDBTier cluster2 with respect to cnDBTier cluster1:

```
$ kubectl -n cluster2 exec -it ndbmysqld-0 -- bash
$ mysql -h 127.0.0.1 -uroot -p
Password:
mysql> select * from replication_info.DBTIER_REPLICATION_CHANNEL_INFO
where remote_server_id in (1000, 1001) AND remote_site_name = "cluster1";
```

Sample output:

```
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| remote_site_name | remote_server_id | channel_id | remote_signaling_ip |
role      | start_epoch | site_name | server_id | start_ts          |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| cluster1        | 1000 | 8970 | 10.233.13.41 |
ACTIVE | NULL | cluster2 | 2000 | 2022-04-04 19:10:44 |
| cluster1        | 1001 | 4597 | 10.233.39.110 |
STANDBY | NULL | cluster2 | 2001 | 2022-04-04 19:10:45 |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

2. Run the following commands to check if replication is turned on in the ACTIVE Replication channel of cnDBTier cluster2 with respect to cnDBTier cluster1.

Note

When replication is turned on, the Replica_IO_Running and Replica_SQL_Running parameters are set to Yes.

The following example describes the commands to check the replication status of the ACTIVE replication channel:

```
$ kubectl -n cluster2 exec -it ndbmysqld-0 -- bash
$ mysql -h 127.0.0.1 -uroot -p
Password:
mysql> SHOW REPLICA STATUS\G;
```

Sample output:

```
***** 1. row *****
      Replica_IO_State: Waiting for master to send event
      Source_Host: 10.233.13.41
      Source_User: occnerepluser
      Source_Port: 3306
      Connect_Retry: 60
      Source_Log_File: mysql-bin.000009
      Read_Source_Log_Pos: 16480
      Relay_Log_File: mysql-relay-bin.000002
      Relay_Log_Pos: 11203
      Relay_Source_Log_File: mysql-bin.000007
      Replica_IO_Running: Yes
      Replica_SQL_Running: Yes
      ....
      ....
      ....
      Replica_SQL_Running_State: Replica has read all relay log; waiting for
more updates
      Source_Retry_Count: 86400
      ....
      ....
```

3. Run the following commands to check if replication is turned off in the STANDBY Replication channel of cnDBTier cluster2 with respect to cnDBTier cluster1.

Note

When replication is turned off, the Replica_IO_Running and Replica_SQL_Running parameters are set to No.

The following example describes the commands to check the replication status of the STANDBY replication channel:

```
$ kubectl -n cluster2 exec -it ndbmysqld-1 -- bash
$ mysql -h 127.0.0.1 -uroot -p
Password:
mysql> SHOW REPLICA STATUS\G;
```

Sample output:

```
***** 1. row *****
      Replica_IO_State:
      Source_Host: 10.233.39.110
      Source_User: occnerepluser
      Source_Port: 3306
      Connect_Retry: 60
      Source_Log_File: mysql-bin.000007
      Read_Source_Log_Pos: 13270
      Relay_Log_File: mysql-relay-bin.000002
      Relay_Log_Pos: 2228
      Relay_Source_Log_File: mysql-bin.000005
```

Replica_IO_Running: No
Replica_SQL_Running: No

- Run the following commands to check the georeplication status of cnDBTier cluster2 with respect to cnDBTier cluster3:

```
$ kubectl -n cluster2 exec -it ndbmysqld-2 -- bash
$ mysql -h 127.0.0.1 -uroot -p
Password:
mysql> select * from replication_info.DBTIER_REPLICATION_CHANNEL_INFO
where remote_server_id in (3002, 3003) AND remote_site_name = "cluster3";
```

Sample output:

```
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
| remote_site_name | remote_server_id | channel_id | remote_signaling_ip |
role      | start_epoch | site_name | server_id | start_ts          |
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
| cluster3        |          3002 |      9981 | 10.233.62.226      |
ACTIVE    |      NULL | cluster2 |      2002 | 2022-04-04 19:10:44 |
| cluster3        |          3003 |      9319 | 10.233.14.15       |
STANDBY   |      NULL | cluster2 |      2003 | 2022-04-04 19:10:45 |
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

- Run the following commands to check if replication is turned on in the ACTIVE Replication channel of cnDBTier cluster2 with respect to cnDBTier cluster3.

Note

When replication is turned on, the Replica_IO_Running and Replica_SQL_Running parameters are set to Yes.

The following example describes the commands to check the replication status of the ACTIVE Replication channel:

```
$ kubectl -n cluster2 exec -it ndbmysqld-2 -- bash
$ mysql -h 127.0.0.1 -uroot -p
Password:
mysql> SHOW REPLICATION STATUS\G;
```

Sample output:

```
***** 1. row *****
      Replica_IO_State: Waiting for master to send event
      Source_Host: 10.233.62.226
      Source_User: occnerepluser
      Source_Port: 3306
      Connect_Retry: 60
      Source_Log_File: mysql-bin.000008
```

```

Read_Source_Log_Pos: 16485
Relay_Log_File: mysql-relay-bin.000002
Relay_Log_Pos: 11203
Relay_Source_Log_File: mysql-bin.000007
Replica_IO_Running: Yes
Replica_SQL_Running: Yes
....
....
....
Replica_SQL_Running_State: Replica has read all relay log; waiting for
more updates
Source_Retry_Count: 86400
....
....

```

6. Run the following commands to check if replication is turned off in the STANDBY Replication channel of cnDBTier cluster2 with respect to cnDBTier cluster3.

Note

When replication is turned off, the Replica_IO_Running and Replica_SQL_Running parameters are set to No.

The following example describes the commands to check the replication status of the STANDBY Replication channel:

```

$ kubectl -n cluster2 exec -it ndbmysqld-3 -- bash
$ mysql -h 127.0.0.1 -uroot -p
Password:
mysql> SHOW REPLICA STATUS\G;

```

Sample output:

```

***** 1. row *****
Replica_IO_State:
Source_Host: 10.233.14.15
Source_User: occnerepluser
Source_Port: 3306
Connect_Retry: 60
Source_Log_File: mysql-bin.000008
Read_Source_Log_Pos: 13670
Relay_Log_File: mysql-relay-bin.000002
Relay_Log_Pos: 2228
Relay_Source_Log_File: mysql-bin.000005
Replica_IO_Running: No
Replica_SQL_Running: No

```

7. Run the following commands to check the georeplication status of cnDBTier cluster2 with respect to cnDBTier cluster4:

```

$ kubectl -n cluster2 exec -it ndbmysqld-4 -- bash
$ mysql -h 127.0.0.1 -uroot -p
Password:

```

```
mysql> select * from replication_info.DBTIER_REPLICATION_CHANNEL_INFO
where remote_server_id in (4002, 4003) AND remote_site_name = "cluster4";
```

Sample output:

```
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
| remote_site_name | remote_server_id | channel_id | remote_signaling_ip |
role      | start_epoch | site_name | server_id | start_ts      |
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
| cluster4        |          4002 |      4500 | 10.233.42.33        |
ACTIVE |      NULL | cluster2 |      2004 | 2022-04-04 19:10:44 |
| cluster4        |          4003 |      9921 | 10.233.48.56        |
STANDBY |      NULL | cluster2 |      2005 | 2022-04-04 19:10:45 |
+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

8. Run the following commands to check if replication is turned on in the ACTIVE Replication channel of cnDBTier cluster2 with respect to cnDBTier cluster4.

Note

When replication is turned on, the `Replica_IO_Running` and `Replica_SQL_Running` parameters are set to Yes.

The following example describes the commands to check the replication status of the ACTIVE Replication channel:

```
$ kubectl -n cluster2 exec -it ndbmysqld-4 -- bash
$ mysql -h 127.0.0.1 -uroot -p
Password:
mysql> SHOW REPLICA STATUS\G;
```

Sample output:

```
***** 1. row *****
      Replica_IO_State: Waiting for master to send event
        Source_Host: 10.233.42.33
        Source_User: occnerepluser
        Source_Port: 3306
        Connect_Retry: 60
        Source_Log_File: mysql-bin.000005
        Read_Source_Log_Pos: 34765
        Relay_Log_File: mysql-relay-bin.000002
        Relay_Log_Pos: 11203
        Relay_Source_Log_File: mysql-bin.000007
        Replica_IO_Running: Yes
        Replica_SQL_Running: Yes
          ....
          ....
          ....
```

```

Replica_SQL_Running_State: Replica has read all relay log; waiting for
more updates
Source_Retry_Count: 86400
....
....

```

9. Run the following commands to check if replication is turned off in the STANDBY Replication channel of cnDBTier cluster2 with respect to cnDBTier cluster4.

Note

When replication is turned off, the Replica_IO_Running and Replica_SQL_Running parameters are set to No.

The following example describes the commands to check the replication status of the STANDBY Replication channel:

```

$ kubectl -n cluster2 exec -it ndbmysqld-5 -- bash
$ mysql -h 127.0.0.1 -uroot -p
Password:
mysql> SHOW REPLICA STATUS\G;

```

Sample output:

```

***** 1. row *****
Replica_IO_State:
Source_Host: 10.233.48.56
Source_User: occnerepluser
Source_Port: 3306
Connect_Retry: 60
Source_Log_File: mysql-bin.000005
Read_Source_Log_Pos: 24672
Relay_Log_File: mysql-relay-bin.000002
Relay_Log_Pos: 2228
Relay_Source_Log_File: mysql-bin.000005
Replica_IO_Running: No
Replica_SQL_Running: No

```

Checking the Georeplication Status of cnDBTier cluster3 with Remote Sites

Perform the following steps to check the georeplication status of cnDBTier cluster3 with remote sites (cnDBTier cluster1, cnDBTier cluster2, and cnDBTier cluster4):

Note

The following commands and examples are applicable for a single replication channel group only. If multiple replication channel groups are enabled, then check the georeplication status in cnDBTier cluster3 with remote sites (cnDBTier cluster1, cnDBTier cluster2, and cnDBTier cluster4) for every replication group.

1. Run the following commands to check the georeplication status of cnDBTier cluster3 with respect to cnDBTier cluster1:

```
$ kubectl -n cluster3 exec -it ndbmysqld-0 -- bash
$ mysql -h 127.0.0.1 -uroot -p
Password:
mysql> select * from replication_info.DBTIER_REPLICATION_CHANNEL_INFO
where remote_server_id in (1002, 1003) AND remote_site_name = "cluster1";
```

Sample output:

```
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
| remote_site_name | remote_server_id | channel_id | remote_signaling_ip |
role      | start_epoch | site_name | server_id | start_ts      |
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
| cluster1        |          1002    |          5533 | 10.233.15.214      |
ACTIVE    |      NULL    | cluster3    |          3000 | 2022-04-04 19:10:44 |
| cluster1        |          1003    |          7918 | 10.233.24.249      |
STANDBY   |      NULL    | cluster3    |          3001 | 2022-04-04 19:10:45 |
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

2. Run the following commands to check if replication is turned on in the ACTIVE Replication channel of cnDBTier cluster3 with respect to cnDBTier cluster1.

Note

When replication is turned on, the `Replica_IO_Running` and `Replica_SQL_Running` parameters are set to Yes.

The following example describes the commands to check the replication status of the ACTIVE Replication channel:

```
$ kubectl -n cluster3 exec -it ndbmysqld-0 -- bash
$ mysql -h 127.0.0.1 -uroot -p
Password:
mysql> SHOW REPLICA STATUS\G;
```

Sample output:

```
***** 1. row *****
      Replica_IO_State: Waiting for master to send event
        Source_Host: 10.233.15.214
        Source_User: occnerepluser
        Source_Port: 3306
        Connect_Retry: 60
        Source_Log_File: mysql-bin.000005
  Read_Source_Log_Pos: 34765
        Relay_Log_File: mysql-relay-bin.000002
        Relay_Log_Pos: 11203
```

```

Relay_Source_Log_File: mysql-bin.000007
Replica_IO_Running: Yes
Replica_SQL_Running: Yes
    ....
    ....
    ....
Replica_SQL_Running_State: Replica has read all relay log; waiting for
more updates
Source_Retry_Count: 86400
    ....
    ....

```

3. Run the following commands to check if replication is turned off in the STANDBY Replication channel of cnDBTier cluster3 with respect to cnDBTier cluster1.

Note

When replication is turned off, the Replica_IO_Running and Replica_SQL_Running parameters are set to No.

The following example describes the commands to check the replication status of the STANDBY Replication channel:

```

$ kubectl -n cluster3 exec -it ndbmysqld-1 -- bash
$ mysql -h 127.0.0.1 -uroot -p
Password:
mysql> SHOW REPLICA STATUS\G;

```

Sample output:

```

***** 1. row *****
      Replica_IO_State:
        Source_Host: 10.233.24.249
        Source_User: occnerepluser
        Source_Port: 3306
        Connect_Retry: 60
        Source_Log_File: mysql-bin.000005
        Read_Source_Log_Pos: 24672
        Relay_Log_File: mysql-relay-bin.000002
        Relay_Log_Pos: 2228
        Relay_Source_Log_File: mysql-bin.000005
        Replica_IO_Running: No
        Replica_SQL_Running: No

```

4. Run the following commands to check the georeplication status of cnDBTier cluster3 with respect to cnDBTier cluster2:

```

$ kubectl -n cluster3 exec -it ndbmysqld-2 -- bash
$ mysql -h 127.0.0.1 -uroot -p
Password:
mysql> select * from replication_info.DBTIER_REPLICATION_CHANNEL_INFO
where remote_server_id in (2002, 2003) AND remote_site_name = "cluster2";

```


Sample output:

```
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| remote_site_name | remote_server_id | channel_id | remote_signaling_ip |
| role            | start_epoch      | site_name  | server_id | start_ts          |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| cluster2        |                2002 |          9981 | 10.233.34.252      |
ACTIVE |          NULL | cluster3 |          3002 | 2022-04-04 19:10:44 |
| cluster2        |                2003 |          9319 | 10.233.15.228      |
STANDBY |          NULL | cluster3 |          3003 | 2022-04-04 19:10:45 |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

5. Run the following commands to check if replication is turned on in the ACTIVE Replication channel of cnDBTier cluster3 with respect to cnDBTier cluster2.

Note

When replication is turned on, the `Replica_IO_Running` and `Replica_SQL_Running` parameters are set to Yes.

The following example describes the commands to check the replication status of the ACTIVE Replication channel:

```
$ kubectl -n cluster3 exec -it ndbmysqld-2 -- bash
$ mysql -h 127.0.0.1 -uroot -p
Password:
mysql> SHOW REPLICA STATUS\G;
```

Sample output:

```
***** 1. row *****
      Replica_IO_State: Waiting for master to send event
      Source_Host: 10.233.34.252
      Source_User: occnerepluser
      Source_Port: 3306
      Connect_Retry: 60
      Source_Log_File: mysql-bin.000005
      Read_Source_Log_Pos: 34765
      Relay_Log_File: mysql-relay-bin.000002
      Relay_Log_Pos: 11203
      Relay_Source_Log_File: mysql-bin.000007
      Replica_IO_Running: Yes
      Replica_SQL_Running: Yes
      ....
      ....
      ....
      Replica_SQL_Running_State: Replica has read all relay log; waiting for
      more updates
      Source_Retry_Count: 86400
```

....
....

6. Run the following commands to check if replication is turned off in the STANDBY Replication channel of cnDBTier cluster3 with respect to cnDBTier cluster2.

Note

When replication is turned off, the Replica_IO_Running and Replica_SQL_Running parameters are set to No.

The following example describes the commands to check the replication status of the STANDBY Replication channel:

```
$ kubectl -n cluster3 exec -it ndbmysqld-3 -- bash
$ mysql -h 127.0.0.1 -uroot -p
Password:
mysql> SHOW REPLICA STATUS\G;
```

Sample output:

```
***** 1. row *****
      Replica_IO_State:
        Source_Host: 10.233.15.228
        Source_User: occnerepluser
        Source_Port: 3306
        Connect_Retry: 60
        Source_Log_File: mysql-bin.000005
        Read_Source_Log_Pos: 24672
        Relay_Log_File: mysql-relay-bin.000002
        Relay_Log_Pos: 2228
        Relay_Source_Log_File: mysql-bin.000005
        Replica_IO_Running: No
        Replica_SQL_Running: No
```

7. Run the following commands to check the georeplication status of cnDBTier cluster3 with respect to cnDBTier cluster4:

```
$ kubectl -n cluster3 exec -it ndbmysqld-4 -- bash
$ mysql -h 127.0.0.1 -uroot -p
Password:
mysql> select * from replication_info.DBTIER_REPLICATION_CHANNEL_INFO
where remote_server_id in (4004, 4005) AND remote_site_name = "cluster4";
```

Sample output:

```
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| remote_site_name | remote_server_id | channel_id | remote_signaling_ip |
| role            | start_epoch      | site_name  | server_id | start_ts          |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| cluster4        |                  | 4004       | 3625 | 10.233.63.121    |
```

```

ACTIVE |          NULL | cluster3 |          3004 | 2022-04-04 19:10:44 |
| cluster4          |          4005 |          3837 | 10.233.40.173      |
STANDBY |          NULL | cluster3 |          3005 | 2022-04-04 19:10:45 |
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

```

8. Run the following commands to check if replication is turned on in the ACTIVE Replication channel of cnDBTier cluster3 with respect to cnDBTier cluster4.

Note

When replication is turned on, the `Replica_IO_Running` and `Replica_SQL_Running` parameters are set to Yes.

The following example describes the commands to check the replication status of the ACTIVE Replication channel:

```

$ kubectl -n cluster3 exec -it ndbmysqld-4 -- bash
$ mysql -h 127.0.0.1 -uroot -p
Password:
mysql> SHOW REPLICA STATUS\G;

```

Sample output:

```

***** 1. row *****
      Replica_IO_State: Waiting for master to send event
      Source_Host: 10.233.63.121
      Source_User: occnerepluser
      Source_Port: 3306
      Connect_Retry: 60
      Source_Log_File: mysql-bin.000005
      Read_Source_Log_Pos: 34765
      Relay_Log_File: mysql-relay-bin.000002
      Relay_Log_Pos: 11203
      Relay_Source_Log_File: mysql-bin.000007
      Replica_IO_Running: Yes
      Replica_SQL_Running: Yes
      ....
      ....
      ....
      Replica_SQL_Running_State: Replica has read all relay log; waiting for
more updates
      Source_Retry_Count: 86400
      ....
      ....

```

9. Run the following commands to check if replication is turned off in the STANDBY Replication channel of cnDBTier cluster3 with respect to cnDBTier cluster4.

Note

When replication is turned off, the `Replica_IO_Running` and `Replica_SQL_Running` parameters are set to No.

The following example describes the commands to check the replication status of the STANDBY Replication channel:

```
$ kubectl -n cluster3 exec -it ndbmysqld-5 -- bash
$ mysql -h 127.0.0.1 -uroot -p
Password:
mysql> SHOW REPLICA STATUS\G;
```

Sample output:

```
***** 1. row *****
      Replica_IO_State:
        Source_Host: 10.233.40.173
        Source_User: occnerepluser
        Source_Port: 3306
        Connect_Retry: 60
        Source_Log_File: mysql-bin.000005
        Read_Source_Log_Pos: 24672
        Relay_Log_File: mysql-relay-bin.000002
        Relay_Log_Pos: 2228
        Relay_Source_Log_File: mysql-bin.000005
        Replica_IO_Running: No
        Replica_SQL_Running: No
```

Checking the Georeplication Status of cnDBTier cluster4 with Remote Sites

Perform the following steps to check the georeplication status of cnDBTier cluster4 with remote sites (cnDBTier cluster1, cnDBTier cluster2, and cnDBTier cluster3):

Note

The following commands and examples are applicable for a single replication channel group only. If multiple replication channel groups are enabled, then check the georeplication status in cnDBTier cluster4 with remote sites (cnDBTier cluster1, cnDBTier cluster2, and cnDBTier cluster3) for every replication group.

1. Run the following commands to check the georeplication status of cnDBTier cluster4 with respect to cnDBTier cluster1:

```
$ kubectl -n cluster4 exec -it ndbmysqld-0 -- bash
$ mysql -h 127.0.0.1 -uroot -p
Password:
mysql> select * from replication_info.DBTIER_REPLICATION_CHANNEL_INFO
where remote_server_id in (1004, 1005) AND remote_site_name = "cluster1";
```

Sample output:

```
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| remote_site_name | remote_server_id | channel_id | remote_signaling_ip |
| role            | start_epoch      | site_name  | server_id  | start_ts          |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| cluster1        | 1004             | 5909       | 10.233.37.108      |
ACTIVE | NULL | cluster4 | 4000 | 2022-04-04 19:10:44 |
| cluster1        | 1005             | 1937       | 10.233.11.31       |
STANDBY | NULL | cluster4 | 4001 | 2022-04-04 19:10:45 |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

2. Run the following commands to check if replication is turned on in the ACTIVE Replication channel of cnDBTier cluster4 with respect to cnDBTier cluster1.

Note

When replication is turned on, the `Replica_IO_Running` and `Replica_SQL_Running` parameters are set to Yes.

The following example describes the commands to check the replication status of the ACTIVE Replication channel:

```
$ kubectl -n cluster4 exec -it ndbmysqld-0 -- bash
$ mysql -h 127.0.0.1 -uroot -p
Password:
mysql> SHOW REPLICA STATUS\G;
```

Sample output:

```
***** 1. row *****
      Replica_IO_State: Waiting for master to send event
      Source_Host: 10.233.37.108
      Source_User: occnerepluser
      Source_Port: 3306
      Connect_Retry: 60
      Source_Log_File: mysql-bin.000005
      Read_Source_Log_Pos: 34765
      Relay_Log_File: mysql-relay-bin.000002
      Relay_Log_Pos: 11203
      Relay_Source_Log_File: mysql-bin.000007
      Replica_IO_Running: Yes
      Replica_SQL_Running: Yes
      ....
      ....
      ....
      Replica_SQL_Running_State: Replica has read all relay log; waiting for
      more updates
      Source_Retry_Count: 86400
```

....
....

3. Run the following commands to check if replication is turned off in the STANDBY Replication channel of cnDBTier cluster4 with respect to cnDBTier cluster1.

Note

When replication is turned off, the `Replica_IO_Running` and `Replica_SQL_Running` parameters are set to No.

The following example describes the commands to check the replication status of the STANDBY Replication channel:

```
$ kubectl -n cluster4 exec -it ndbmysqld-1 -- bash
$ mysql -h 127.0.0.1 -uroot -p
Password:
mysql> SHOW REPLICA STATUS\G;
```

Sample output:

```
***** 1. row *****
      Replica_IO_State:
        Source_Host: 10.233.11.31
        Source_User: occnerepluser
        Source_Port: 3306
        Connect_Retry: 60
        Source_Log_File: mysql-bin.000005
        Read_Source_Log_Pos: 24672
        Relay_Log_File: mysql-relay-bin.000002
        Relay_Log_Pos: 2228
        Relay_Source_Log_File: mysql-bin.000005
        Replica_IO_Running: No
        Replica_SQL_Running: No
```

4. Run the following commands to check the georeplication status of cnDBTier cluster4 with respect to cnDBTier cluster2:

```
$ kubectl -n cluster4 exec -it ndbmysqld-2 -- bash
$ mysql -h 127.0.0.1 -uroot -p
Password:
mysql> select * from replication_info.DBTIER_REPLICATION_CHANNEL_INFO
where remote_server_id in (2004, 2005) AND remote_site_name = "cluster2";
```

Sample output:

```
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| remote_site_name | remote_server_id | channel_id | remote_signaling_ip |
| role           | start_epoch      | site_name  | server_id | start_ts          |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| cluster2        |                  | 2004       | 4500      | 10.233.15.245    |
```

```

ACTIVE |          NULL | cluster4 |          4002 | 2022-04-04 19:10:44 |
| cluster2          |          2005 |          9921 | 10.233.48.74          |
STANDBY |          NULL | cluster4 |          4003 | 2022-04-04 19:10:45 |
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

```

5. Run the following commands to check if replication is turned on in the ACTIVE Replication channel of cnDBTier cluster4 with respect to cnDBTier cluster2.

Note

When replication is turned on, the `Replica_IO_Running` and `Replica_SQL_Running` parameters are set to Yes.

The following example describes the commands to check the replication status of the ACTIVE Replication channel:

```

$ kubectl -n cluster4 exec -it ndbmysqld-2 -- bash
$ mysql -h 127.0.0.1 -uroot -p
Password:
mysql> SHOW REPLICA STATUS\G;

```

Sample output:

```

***** 1. row *****
      Replica_IO_State: Waiting for master to send event
      Source_Host: 10.233.15.245
      Source_User: occnerepluser
      Source_Port: 3306
      Connect_Retry: 60
      Source_Log_File: mysql-bin.000005
      Read_Source_Log_Pos: 34765
      Relay_Log_File: mysql-relay-bin.000002
      Relay_Log_Pos: 11203
      Relay_Source_Log_File: mysql-bin.000007
      Replica_IO_Running: Yes
      Replica_SQL_Running: Yes
      ....
      ....
      ....
      Replica_SQL_Running_State: Replica has read all relay log; waiting for
more updates
      Source_Retry_Count: 86400
      ....
      ....

```

6. Run the following commands to check if replication is turned off in the STANDBY Replication channel of cnDBTier cluster4 with respect to cnDBTier cluster2.

Note

When replication is turned off, the `Replica_IO_Running` and `Replica_SQL_Running` parameters are set to No.

The following example describes the commands to check the replication status of the STANDBY Replication channel:

```
$ kubectl -n cluster4 exec -it ndbmysqld-3 -- bash
$ mysql -h 127.0.0.1 -uroot -p
Password:
mysql> SHOW REPLICA STATUS\G;
```

Sample output:

```
***** 1. row *****
      Replica_IO_State:
        Source_Host: 10.233.48.74
        Source_User: occnerepluser
        Source_Port: 3306
        Connect_Retry: 60
        Source_Log_File: mysql-bin.000005
        Read_Source_Log_Pos: 24672
        Relay_Log_File: mysql-relay-bin.000002
        Relay_Log_Pos: 2228
        Relay_Source_Log_File: mysql-bin.000005
        Replica_IO_Running: No
        Replica_SQL_Running: No
```

7. Run the following commands to check the georeplication status of cnDBTier cluster4 with respect to cnDBTier cluster3:

```
$ kubectl -n cluster4 exec -it ndbmysqld-4 -- bash
$ mysql -h 127.0.0.1 -uroot -p
Password:
mysql> select * from replication_info.DBTIER_REPLICATION_CHANNEL_INFO
where remote_server_id in (3004, 3005) AND remote_site_name = "cluster3";
```

Sample output:

```
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| remote_site_name | remote_server_id | channel_id | remote_signaling_ip |
| role            | start_epoch      | site_name  | server_id | start_ts          |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| cluster3        |                  | 3004       | 3625 | 10.233.3.32      |
ACTIVE | NULL | cluster4 | 4004 | 2022-04-04 19:10:44 |
| cluster3        |                  | 3005       | 3837 | 10.233.14.89     |
STANDBY | NULL | cluster4 | 4005 | 2022-04-04 19:10:45 |
+-----+-----+-----+-----+
```



```
+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

8. Run the following commands to check if replication is turned on in the ACTIVE Replication channel of cnDBTier cluster4 with respect to cnDBTier cluster3.

Note

When replication is turned on, the `Replica_IO_Running` and `Replica_SQL_Running` parameters are set to Yes.

The following example describes the commands to check the replication status of the ACTIVE Replication channel:

```
$ kubectl -n cluster4 exec -it ndbmysqld-4 -- bash
$ mysql -h 127.0.0.1 -uroot -p
Password:
mysql> SHOW REPLICA STATUS\G;
```

Sample output:

```
***** 1. row *****
      Replica_IO_State: Waiting for master to send event
      Source_Host: 10.233.3.32
      Source_User: occnerepluser
      Source_Port: 3306
      Connect_Retry: 60
      Source_Log_File: mysql-bin.000005
      Read_Source_Log_Pos: 34765
      Relay_Log_File: mysql-relay-bin.000002
      Relay_Log_Pos: 11203
      Relay_Source_Log_File: mysql-bin.000007
      Replica_IO_Running: Yes
      Replica_SQL_Running: Yes
      ....
      ....
      ....
      Replica_SQL_Running_State: Replica has read all relay log; waiting for
      more updates
      Source_Retry_Count: 86400
      ....
      ....
```

9. Run the following commands to check if replication is turned off in the STANDBY Replication channel of cnDBTier cluster4 with respect to cnDBTier cluster3.

Note

When replication is turned off, the `Replica_IO_Running` and `Replica_SQL_Running` parameters are set to No.

The following example describes the commands to check the replication status of the STANDBY Replication channel :

```
$ kubectl -n cluster4 exec -it ndbmysqld-5 -- bash
$ mysql -h 127.0.0.1 -uroot -p
Password:
mysql> SHOW REPLICA STATUS\G;
```

Sample output:

```
***** 1. row *****
      Replica_IO_State:
        Source_Host: 10.233.14.89
        Source_User: occnerepluser
        Source_Port: 3306
        Connect_Retry: 60
        Source_Log_File: mysql-bin.000005
        Read_Source_Log_Pos: 24672
        Relay_Log_File: mysql-relay-bin.000002
        Relay_Log_Pos: 2228
        Relay_Source_Log_File: mysql-bin.000005
        Replica_IO_Running: No
        Replica_SQL_Running: No
```

7.12 Changing Authentication Plugin on cnDBTier Sites

Users created on cnDBTier setups older than 23.4.x (that is, 23.2.x, and 23.3.0) use the `mysql_native_password` plugin for authentication. As this plugin is deprecated in mysql version 8.0.34 (cnDBTier 23.3.0), you must use the `caching_sha2_password` plugin for user authentication. This section provides the steps to change the authentication plugin for users created on cnDBTier setups older than 23.3.1.

Note

- Perform this procedure on the `ndbappmysqld` pod of one site only.
- If you have upgraded to 24.1.x from a cnDBTier version older than 23.3.1, you must perform this procedure to change the authentication plugin after all the sites are upgraded.

Perform the following steps to alter a user to use the `caching_sha2_password` authentication plugin:

1. Log in to the `ndbappmysqld` pod:

```
$ kubectl -n <namespace> exec -it ndbappmysqld-0 -- mysql -h:::1 -uroot -p<password>
```

where,

- `<namespace>` is the namespace name
- `<password>` is the password to access the `ndbappmysqld` pod

For example:

```
$ kubectl -n occne-cndbtier exec -it ndbappmysqld-0 -- mysql -h::1 -uroot -p<password>
```

2. Run the following MySQL command to change the authentication plugin:

```
mysql> ALTER USER IF EXISTS <USER_NAME> IDENTIFIED WITH
'caching_sha2_password' BY '<password>';
```

where,

- <USER NAME> is the name of the user.
- <password> is the password of user.

For example:

```
mysql> ALTER USER IF EXISTS occneuser IDENTIFIED WITH
'caching_sha2_password' BY 'NextGenCne';
mysql> ALTER USER IF EXISTS occnerepluser IDENTIFIED WITH
'caching_sha2_password' BY 'NextGenCne';
mysql> ALTER USER IF EXISTS root@localhost IDENTIFIED WITH
'caching_sha2_password' BY 'NextGenCne';
mysql> ALTER USER IF EXISTS root IDENTIFIED WITH 'caching_sha2_password'
BY 'NextGenCne';
mysql> ALTER USER IF EXISTS <NF_USER> IDENTIFIED WITH
'caching_sha2_password' BY 'NextGenCne';
```

If you want to roll back to cnDBTier 23.2.x, 23.3.x, 23.4.x, or 24.1.x after performing the above procedure, then you must change the authentication plugin to `mysql_native_password` before performing a rollback:

1. Log in to the `ndbappmysqld` pod:

```
$ kubectl -n <namespace> exec -it ndbappmysqld-0 -- mysql -h::1 -uroot -p<password>
```

where,

- <namespace> is the namespace name.
- <password> is the password to access the `ndbappmysqld` pod.

For example:

```
$ kubectl -n occne-cndbtier exec -it ndbappmysqld-0 -- mysql -h::1 -uroot -p<password>
```

2. Run the following `mysql` command to change the authentication plugin to `mysql_native_password`:

```
mysql> ALTER USER IF EXISTS <USER_NAME> IDENTIFIED WITH
'mysql_native_password' BY '<password>';
```

where,

- <USER NAME> is the name of the user.
- <password> is the password of the user.

For example:

```
mysql> ALTER USER IF EXISTS occneuser IDENTIFIED WITH
'mysql_native_password' BY 'NextGenCne';
mysql> ALTER USER IF EXISTS occnerepluser IDENTIFIED WITH
'mysql_native_password' BY 'NextGenCne';
mysql> ALTER USER IF EXISTS root@localhost IDENTIFIED WITH
'mysql_native_password' BY 'NextGenCne';
mysql> ALTER USER IF EXISTS <NF_USER> IDENTIFIED WITH
'caching_sha2_password' BY 'NextGenCne';
```

7.13 LCM based Automation

This feature optimizes the deployment or upgrade steps. This is achieved by automating service account creation that enables you to create user-defined service account automatically without performing any manual steps. The automation includes:

- **Creating Service Account:** An automated resource service account creation has been introduced in this release to streamline the management of Kubernetes resources through Helm charts.
- **Applying NF Alert Configuration rules:** The cnDBTier leverages the oso-alr-config Helm chart, introduced as part of the cnDBTier package, to apply alert rules using Helm upgrades. While the oso-alr-config Helm chart is deployed automatically during the cnDBTier installation, the cnDBTier performs Helm upgrades to apply or update the required alert rules. Both manual and automated configurations are supported.

Managing LCM-based Automation

This section explains the procedure to enable and configure the feature.

Enable

You can enable this feature using the Helm.

Automating Service Account, Role and Role Binding

To automate the creation of service account, role, and role binding, enable the following parameters:

- **Global Flag (global.autoCreateResources.enabled):** Controls the overall automation of resource creation. This flag is enabled by default.
- **Resource-Specific Flag (serviceAccounts.create):** Controls ServiceAccount creation at the resource level. This flag is enabled by default and make sure it is set to true alongside the global flag to enable ServiceAccount automation. This flag is conditional on the global flag and will only take effect if the global flag is set to true.

Following snippet is a sample from custom_values.yaml file:

```
autoCreateResources:
  enabled: true
  serviceAccounts:
    create: true
    accounts:
      - serviceAccountName: ""
```

```

    type: APP
    create: true
  - serviceAccountName: "cndbtier-upgrade-serviceaccount"
    type: UPGRADE
    create: true
  - serviceAccountName: "cndbtier-multus-serviceaccount"
    type: MULTUS
    create: false

```

For more information about the configuration parameters, see the "Global Parameters" section in the *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide*.

Automating NF Alert Configuration rules

Alerts are automated by using Helm upgrade command with the Helm chart provided as part of OSO software package. The alert automation process is as follows:

1. A new oso-alr-config Helm chart is provided as part of OSO software package from 25.1.200 release onwards. For information to download OSO software package, see OSO Install Guide.
2. The oso-alr-config Helm chart must be deployed once OSO is installed.
3. This separate Helm chart allows the Helm install command to run without an input alert file.

```

helm install oso-alr-config ocoso_csar_abc_25_1_200_0_0-rc.1_alert_config_charts.tgz -f ocoso_csar_abc_25_1_200_0_0-rc.1_alert_config_custom_values.yaml -n <oso deployed namespace>

```

4. When the oso-alr-config Helm chart installation is completed then the oso-alr-config is ready to use.
5. Run Helm upgrade, if you are enabling this feature after cnDBTier deployment. Run the following Helm upgrade command in oso-alr-config file to apply cnDBTier alert file.

```

helm upgrade oso-alr-config oso-alr-config/ -f ocoso_csar_abc_25_1_200_0_0-rc.1_alert_config_custom_values.yaml -f ocsf_alerts_non_haprom_25.1.200.yaml -n <oso deployed namespace>

```

6. Once the Helm upgrade is completed, you can view the alerts file that is applied to OSO Prometheus ConfigMap. This can be viewed in the Prometheus Graphical User Interface (GUI).
7. You can also update the changes in the same alert file and perform a Helm upgrade. The alert file will be updated with the latest changes.

7.14 Support for Dual Stack

This section explains the procedure to migrate from single stack cnDBTier site to dual stack cnDBTier site.

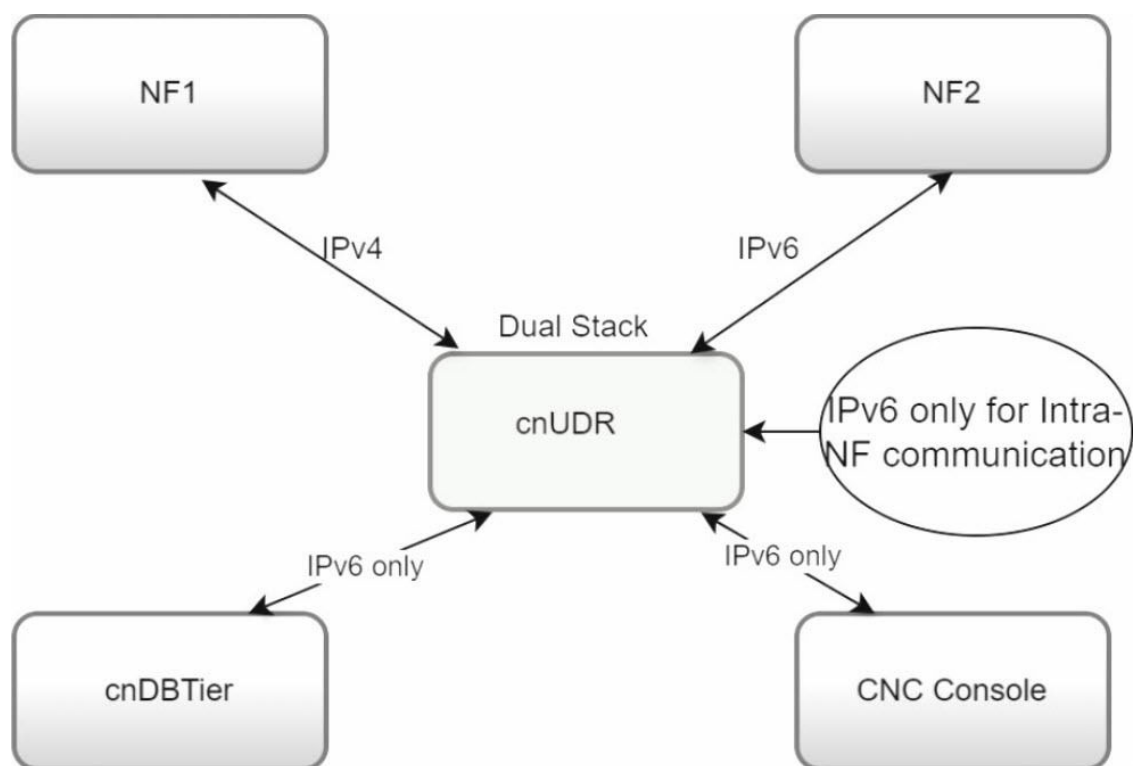
Dual Stack feature enables cnDBTier to operate seamlessly in a Kubernetes environment configured for both IPv4 and IPv6 protocols. This feature facilitates the coexistence and interoperability of IPv4 and IPv6 within Kubernetes clusters, allowing simultaneous communication using either of the protocols based on the deployment configuration.

Dual stack provides:

- **Coexistence of IPv4 and IPv6:** Hosts and services within the Kubernetes cluster can operate on both protocols, enabling communication with IPv4 and IPv6 compatible entities.
- **Flexible Deployment Modes:** cnDBTier supports multiple deployment modes, such as IPv4 only, IPv6 only, and dual stack configurations with either IPv4 or IPv6 preferred.
- **Dynamic IP Allocation:** IP address allocation for pods and services adapts to the cluster's preferred IP family configuration, ensuring consistency across deployments. This allocation is applicable for all Kubernetes resources unless explicitly specified during cluster creation.
- **Edge and Backend Microservices:** Dual stack support is extended to the Ingress Gateway (edge microservices), while backend microservices continue to operate as single stack with either IPv4 or IPv6.

cnDBTier communicates with the entities that support IPv4 and IPv6.

Figure 7-1 Dual Stack



cnDBTier supports:

- dual stack to communicate with the external entities. For example, NFs and provisioning client etc.
- only IPv6 communication within the cnDBTier microservices.
- only IPv6 communication with cnDBTier, CNC Console, and Kubernetes APIs.

Note

Single to multi-replication channels are not supported when dual stack is enabled.

The communication with the different entities is segregated as follows:

- External pods and services
- Internal pods and services

External Pods and Services

The expected behavior of external pods and services are as follows:

- **For Ingress Gateway traffic:**
 - Pods are dual stack and must be reachable on both IPv6 and IPv4 interfaces.
 - Services are dual stack and must be reachable on both IPv6 and IPv4 services. The preferred IP address must be IPv6.
- **For Egress Gateway traffic:**
 - Pods are dual stack and must be exposed to both IPv6 and IPv4 interfaces. The preferred IP must be IPv6 interface. Use IPv4 interface depending on DNS resolution as follows:
 - * If DNS has both A and AAAA records, then you must use the IPv6 interface on the pod.
 - * If DNS has only A records, then you must use the IPv4 interface on the pod.

Internal Pods and Services

The expected behavior of internal pods and services are as follows:

- **For Ingress Gateway traffic:**
 - Pods are single stack and must be exposed to only IPv6 interfaces.
 - Services are single stack and must be exposed to only IPv6 services.
- **For Egress Gateway traffic:**
 - Pods are single stack and must be exposed to only IPv6 interfaces. The preferred IP must be IPv6 interface.

IP Address Allocation to Pods

IP address allocation to all the cnDBTier pods depends on the IP address preference set in the Kubernetes cluster. Pods do not have the privilege to choose an IP address. Consider the following example of a pod deployed in an IPv4 preferred infrastructure. Here, since the Kubernetes cluster has IPv4 preferred configuration, both IPv4 and IPv6 are allocated to the pod, but the primary IP address is IPv4.

Example:

```
IP:                10.xxx.xxx.xxx
IPs:
  IP:                10.xxx.xxx.xxx
  IP:                fd00::1:cxxx:bxxx:8xxx:xxxx
```

IP Address Allocation to Services

IP address allocation to all the cnDBTier services depends on the `global.serviceMode.internal` and `global.serviceMode.external` Helm parameter configuration. The Helm parameters automatically configure IP Family Policy and IP Families attributes for the cnDBTier services. For more information about `global.serviceMode.internal` and `global.serviceMode.external` parameters, see the

"Customizing cnDBTier" section in the *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide*.

Services route the traffic to the destination endpoints based on this configuration. If the `global.serviceMode.internal` and `global.serviceMode.external` Helm parameters are set to IPv4, then IPv4 is allocated to services, and services use IPv4 pod IPs to send the traffic to endpoints.

The following table describes how IP address allocation, IP Family Policy, and IP Families vary based on the `global.serviceMode.internal` and `global.serviceMode.external` Helm parameter configuration for services:

Table 7-3 IP Address Allocation

Infrastructure Preference	Application Preference (global.deploy mentMode serviceMode Helm Parameter)	IP Family Policy Attribute	IP Families Attribute	Pod IP	Service IP	Endpoints
IPv4 Preferred	IPv4	SingleStack	IPv4	IPv4, IPv6	IPv4	IPv4
IPv6 Preferred	IPv4	SingleStack	IPv4	IPv6, IPv4	IPv4	IPv4
IPv4 Preferred	IPv6	SingleStack	IPv6	IPv4, IPv6	IPv6	IPv6
IPv6 Preferred	IPv6	SingleStack	IPv6	IPv6, IPv4	IPv6	IPv6
IPv4 Preferred	IPv6_IPv4 (IPv6Preferred)	PreferDualStack	IPv6 Preferred	IPv4, IPv6	IPv4, IPv6	IPv6
IPv6 Preferred	IPv6_IPv4 (IPv6Preferred)	PreferDualStack	IPv6 Preferred	IPv6, IPv4	IPv6, IPv4	IPv6

Managing Dual Stack Enable

You do not need to enable or disable this feature.

Configure

You can configure this feature using Helm.

- You can configure the feature by setting the `global.serviceMode.internal` and `global.serviceMode.external` parameters to indicate the deployment mode of the cnDBTier in the global section of the `custom_values.yaml` file. The default value is set as IPv4. The possible values are as follows:
 - IPv4
 - IPv6
 - IPv6_IPv4

```
# When k8s is single or dual-stack mode, then service mode provide the
preferences as below:
# ServiceMode external possible values are "IPv6_IPv4", "IPv4" and "IPv6".
# All the DBTier external services with SingleStack /PreferDualStack
ipFamilyPolicy
# ServiceMode internal possible values are "IPv4" and "IPv6".
# All the DBTier internal services with SingleStack ipFamilyPolicy.
serviceMode:
```



```

internal: "IPv4"
external:
  ndbmysqldsvc: "IPv4"
  connectivityService: "IPv4"
  dbMonitorService: "IPv4"

```

cnDBTier must be deployed on a dual stack Kubernetes infrastructure with IPv4 or IPv6 preferred CNE to use dual stack feature. The following table list deployment mode and its behavior.

Service Mode	Description
IPv4	Only single stack and IPv4 preferred for all the services. If the deployment mode is set to IPv4, then all the communication between microservices are through IPv4. In this case, all the internal single stack and IP families will be IPv4.
IPv6	Only single stack and IPv6 preferred for all the services. If the deployment mode is set to IPv6, then all the communication between microservices are through IPv6. In this case, all the internal microservices and external services IP family policy are single stack and IP families will be IPv6.
IPv6_IPv4	Dual Stack and IPv6 preferred for all the services. If the deployment mode is set to IPv6_IPv4, then all the communication between microservices are through IPv6, since IPv6 is preferred. In this case, all the internal microservices IP family policy are single stack and IP families will be IPv6. The IP family policies of the external facing services are PreferDualStack and IP families will be IPv4 and IPv6.

Migrating cnDBTier to Dual Stack

Enabling cnDBTier for Dual Stack

Dual Stack support in cnDBTier allows internal communication to occur over IPv6 while external communication to occur over IPv4 or vice-versa, or any other IP families as required.

To enable this functionality, the following parameters must be configured in the `custom_values.yaml` file:

- `serviceMode` - This parameter determines whether the Kubernetes services are deployed in **Single Stack** or **Dual Stack** mode.
- `preferredIpFamily` - In case of a Dual Stack deployment, this parameter specifies the preferred IP family for communication.

Table 7-4 serviceMode Configuration Parameters

Parameter Name	Description	Supported Values
global.serviceMode.internal	Defines the IP family (IPv4 or IPv6) to be used for internal communication between DBTier components.	- IPv4 - IPv6
global.serviceMode.external.ndbmysqlsvc	Specifies the IP family for the external service associated with ndbmysqlsvc, which exposes the MySQL server outside the Kubernetes cluster. Example: ndbmysqlsvc-0 ndbmysqlsvc-1	- IPv4 - IPv6 - IPv6_IPv4
global.serviceMode.external.connectivityService	Specifies the IP family for the mysql-connectivity-service which NF uses to communicate with the database.	- IPv4 - IPv6
global.serviceMode.external.dbMonitorService	Specifies the IP family for the db-monitor-svc which NF uses to fetch cluster status.	- IPv4 - IPv6 - IPv6_IPv4
global.db-replication-svc.dbreplsvcdeployments[].serviceMode.external	Specifies the IP family for the db-replication-svc external service, which facilitates communication between db-replication-svc pods deployed across different sites.	- IPv4 - IPv6 - IPv6_IPv4

The value of `global.serviceMode.internal` can be independently set to either **IPv4** or **IPv6**, based on the internal communication protocol.

For `global.serviceMode.external.connectivityService` or `global.serviceMode.external.dbMonitorService` parameter, the choice of IP family must consider one of the following capabilities of the deployed NF.

- if the NF supports only IPv4, then the cnDBTier service must be deployed in **IPv4** mode.
- the IP family must be selected based on the NF's deployment mode.

Configuring primarysignalhost, secondarysignalhost, localsiteip, and remotesiteip Parameters in Dual Stack Mode

In Dual Stack mode, for the parameter `global.serviceMode.external.ndbmysqlsvc`, replication SQL pods will have both IPv4 and IPv6 addresses configured in their associated Kubernetes service.

To accommodate this, the `db-replication-svc` service allows specifying multiple IP addresses for both the `primarysignalhost` and `secondarysignalhost` parameters.

Supported Format:

Each host entry can include both IPv4 and IPv6 addresses using the following syntax:

```
[IPv4|IPv6|IPv46|IPv64][:]<IPv4 address>|<IPv6 address>
```

Where:

Note

Alternatively, you can specify the IP version explicitly (ipv4:, ipv6:, ipv64:, ipv46:). If not specified, cnDBTier will interpret the addresses.

Configuration Examples:

The following table provides example configurations for the `primarysignalhost` parameter based on different scenarios. Similarly, you can configure the `secondarysignalhost`, `localsiteip`, and `remotesiteip` parameters.

Note

If the `db-replication-svc.enableInitContainerForIpDiscovery` parameter is set to `true`, then these values will be auto populated.

Table 7-5 primarysignalhost Configuration Parameters

Example	Description
<code>primarysignalhost:</code> <code>"ipv4:fqdn_for_ipv4,ipv6:fqdn_for_ipv6"</code>	Uses different FQDNs for IPv4 and IPv6 addresses. In this example, FQDNs specifically have IPv4 and IPV6 in their names.
<code>primarysignalhost:</code> <code>"fqdn_for_ipv4,fqdn_for_ipv6"</code>	Uses different FQDNs for IPv4 and IPv6 addresses.
<code>primarysignalhost:</code> <code>"ipv46:ndbmysqld-0.ndbmysqldsvc.occne-cndbtier.svc.cluster.local"</code>	FQDN points to both IPv4 and IPv6.
<code>primarysignalhost:</code> <code>"ipv4:10.75.23.45,ipv6:2606:b400:605:b819"</code>	Specifies direct IP addresses for both IPv4 and IPv6 explicitly.
<code>primarysignalhost:</code> <code>"10.23.45.66,2606:b400:605:b819"</code>	Provides direct IP addresses without specifying the IP version.
<code>primarysignalhost: "10.75.23.45"</code>	Provides single IPv4 IP address.
<code>primarysignalhost: "2606:b400:605:b819"</code>	Provides single IPv6 IP address.

Configuring preferredIpFamily Parameter in Dual Stack Mode

The `preferredIpFamily` parameter configuration in the `custom_values.yaml` file determines which IP family must be preferred for external communication when cnDBTier is deployed in Dual Stack mode. If cnDBTier is deployed in a Single Stack mode, this configuration is ignored, as only a single IP family is available.

Table 7-6 preferredIpFamily Configuration Parameter

Configuration	Description	Supported Values
<code>global.db-replication-svc.dbreplsvcdeployments[].replication.preferredIpFamily</code>	Determines the preferred IP family for replication traffic between the configured sites, specifically for communication over replication SQL pods and between db-replication-svc pods.	- IPv4 - IPv6

Note

This is a local configuration between each pair of configured local and remote sites. This means that if there are more than two sites, the preferredIpFamily can be configured independently for each site-to-site pairing.

Example:

Site 1 and Site 2 can have preferredIpFamily set to IPv4, Site 1 and Site 3 can have preferredIpFamily set to IPv6.

Result:

- ndbmysqld-0 and ndbmysqld-1 will replicate over IPv4 (Site 1 ↔ Site 2).
- ndbmysqld-2 and ndbmysqld-3 will replicate over IPv6 (Site 1 ↔ Site 3).

Similarly, the db-replication-svc pod communication between Site 1 and Site 2 will occur over IPv4, while the communication between Site 1 and Site 3 will occur over IPv6.

Migrating a cnDBTier Site from IPv4 to IPv6 in a Multisite Replication Setup

This section explains the procedure to migrate a site from IPv4 to IPv6 in a multi-site replication setup.

Note

cnDBTier does not allow upgrading from IPv4 to IPv6 directly. Before migration, uninstall the site and reinstall it with the updated IP family setting.

Procedure to migrate a cnDBTier site from IPv4 to IPv6

Perform the following steps to migrate a cnDBTier site from IPv4 to IPv6:

1. Upgrading the Schema:

If the site you want to migrate to IPv6 is running a version earlier than 25.1.2xx, you must first update the schema on that site to 25.1.2xx.

Note

Once the schema is updated to version 25.1.2xx, it will be replicated across other sites. This is an important step because during reinstallation of the cnDBTier site with IPv6 settings, the site uses 25.1.2xx version. To ensure smooth replication, all sites must be running the schema version 25.1.2xx. Else, replication will fail.

cnDBTier is compatible with the higher schema version (25.1.2xx). This ensures that existing deployed cnDBTier clusters continues to function after the schema is upgraded to version 25.1.2xx.

To do the schema update, run the following commands:

replace the values for the environment variables below with the correct ones for your cluster

```
export OCCNE_NAMESPACE="occne-cndbtier"
export MYSQL_CONNECTIVITY_SERVICE="mysql-connectivity-service"
export MYSQL_USERNAME="occneuser"
export MYSQL_PASSWORD="<password for the user occneuser>"
export DBTIER_REPLICATION_SVC_DATABASE="replication_info"
export DBTIER_BACKUP_SVC_DATABASE="backup_info"
export DBTIER_HBREPLICAGROUP_DATABASE="hbreplica_info"
export DBTIER_CLUSTER_INFO_DATABASE="cluster_info"
export REPLCHANNEL_GROUP_COUNT=<configured number of replication channel groups i.e either 1/2/3>
export MYSQL_CMD="kubectl -n <namespace> exec <ndbmysqld-0/ndbappmysqld-0 pod name> -- mysql --binary-as-hex=0 --show-warnings"
```

```
# To enable or disable password encryption, uncomment the below line and set the variable to true or false to enable or disable it.
# export ENABLE_ENCRYPTION="<true/false>"
```

```
occndbtier/files/hooks.sh --schema-upgrade
```

Note

In the above commands hooks.sh file must be obtained from the 25.1.2xx csar package.

Once the schema upgrade is completed on any site, log in to the MySQL prompt of all sites and verify that the schema upgrade was successful and replicated across all georeplication sites. To verify, run the following command:

```
mysql> SHOW CREATE TABLE replication_info.DBTIER_REPL_SITE_INFO;
```

Sample output:

```
| DBTIER_REPL_SITE_INFO | CREATE TABLE `DBTIER_REPL_SITE_INFO` (
  `site_name` varchar(64) NOT NULL,
```

```

`mate_site_name` varchar(64) NOT NULL,
`mate_db_replication_svc_ip` varchar(500) DEFAULT NULL,
`mate_db_replication_svc_port` varchar(100) DEFAULT NULL,
`mate_db_username` varchar(100) DEFAULT NULL,
`mate_db_password` varchar(100) DEFAULT NULL,
`dr_state` varchar(64) DEFAULT NULL,
`replchannel_group_id` int NOT NULL,
`stop_repl_switchover` int NOT NULL,
PRIMARY KEY (`site_name`, `mate_site_name`, `replchannel_group_id`)
) ENGINE=ndbcluster DEFAULT CHARSET=utf8mb3 |

```

In the above output, ensure that the column `mate_db_replication_svc_ip` has a length of 500 characters. This confirms that the schema change has been successfully replicated to all the sites.

2. Removing the Site:

Uninstall the `cnDBTier` site and remove it from the existing replication setup. To ensure proper uninstallation and site removal, perform the steps given in the "Remove `cnDBTier` Geo-Redundant Site" section in the *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide*.

3. Preparing the `custom_values.yaml` file for the New Site:

After the site is removed, reinstall the site with DualStack and the appropriate preferred IP family configurations. Then, add the site back to the existing replication setup. Before adding the new site, the `custom_values.yaml` file for the current site must be updated with the correct values for the Dual Stack changes.

Perform the following configurations:

a. Setting the ServiceMode Parameters:

```

global.serviceMode.internal: Set to IPv6.
global.serviceMode.external.ndbmysqldsvc: Set to IPv6_IPv4 (DualStack
with IPv6 preferred).
global.serviceMode.external.connectivityService: Set to IPv6
global.serviceMode.external.dbMonitorService: Set to IPv6_IPv4
global.db-replication-
svc.dbreplsvcdeployments[*].serviceMode.external: Set to IPv6_IPv4
(DualStack with IPv6 preferred).

```

b. Setting the preferredIpFamily Parameter:

```

global.db-replication-
svc.dbreplsvcdeployments[*].replication.preferredIpFamily: Set to
IPv4

```

c. Setting the primaryhost and secondaryhost Parameters:

```

db-replication-svc.dbreplsvcdeployments[*].mysql.primaryhost must
always set to "<prefix-if-there>-ndbmysqld-<pod-
index>.ndbmysqldsvc.<namespace>.svc.<cluster-domain>"
db-replication-svc.dbreplsvcdeployments[*].mysql.secondaryhost must
always set to "<prefix-if-there>-ndbmysqld-<pod-
index>.ndbmysqldsvc.<namespace>.svc.<cluster-domain>"

```

Note

For Dual Stack migration, the values of the `primaryhost` and `secondaryhost` parameters must not be modified. The default values must be retained.

d. Setting the `primarysignalhost`, `secondarysignalhost`, and `localsiteip`**Parameters:**

Scenario: `db-replication-svc.enableInitContainerForIpDiscovery` is enabled:

```
if db-replication-svc.enableInitContainerForIpDiscovery == true
```

In this case, the `db-replication-svc` will auto fetch the IP addresses for the parameters `primarysignalhost`, `secondarysignalhost` and `localsiteip`, if they are empty (empty string ""). If the value of the parameter is anything other than "", then it is considered as non-empty.

If any of the parameters are not empty, then `db-replication-svc` will not try to auto populate that configurations. The value provided in that configuration will be used.

Hence, if you want the `db-replication-svc` service to auto populate the IP addresses, then keep the parameter values empty ("").

```
db-replication-svc.dbreplsvcdeployments[*].mysql.primarysignalhost
db-replication-svc.dbreplsvcdeployments[*].mysql.secondarysignalhost
db-replication-svc.dbreplsvcdeployments[*].replication.localsiteip
```

Scenario: `db-replication-svc.enableInitContainerForIpDiscovery` is disabled:

```
if db-replication-svc.enableInitContainerForIpDiscovery == false
```

In this case, the `db-replication-svc` will not auto fetch the IP addresses and will use the value assigned to them. Ensure to set a proper value for the below configurations:

```
db-replication-svc.dbreplsvcdeployments[*].mysql.primarysignalhost
db-replication-svc.dbreplsvcdeployments[*].mysql.secondarysignalhost
db-replication-svc.dbreplsvcdeployments[*].replication.localsiteip
```

The accepted value for the `primarysignalhost`, `secondarysignalhost`, and `localsiteip` parameters are as below:

IP addresses:

```
"10.75.10.12" – IPv4
"4001:ab8::1" – IPv6
"10.75.10.12,4001:ab8::1" – Dual (comma-separated)
```

With Explicit Type:

```
"ipv4:10.75.10.12"
"ipv6:4001:ab8::1"
"ipv4:10.75.10.12,ipv6:4001:ab8::1"
"10.75.10.12,ipv6:4001:ab8::1"
"ipv4:10.75.10.12,4001:ab8::1"
```

FQDN (DNS Names)

```
"ndbmysqld-0.ndbmysqldsvc.namespace.svc.cluster.local"
"ipv4.fqdn.cluster.local,ipv6.fqdn.cluster.local"
"ipv4:ipv4.fqdn.cluster.local"
"ipv6:ipv6.fqdn.cluster.local"
"ipv4:ipv4.fqdn.cluster.local,ipv6:ipv6.fqdn.cluster.local"
"ipv4.fqdn.cluster.local,ipv6:ipv6.fqdn.cluster.local"
"ipv4:ipv4.fqdn.cluster.local,ipv6.fqdn.cluster.local"
```

DualStack FQDN:

```
"ipv6.fqdn.cluster.local" – FQDN resolves both IPv4 and IPv6 (auto-
detect IP types)
"ipv6:ipv6.fqdn.cluster.local" – Explicitly marked as dual-stack; no
resolution performed
```

e. **Setting the `remotesiteip` Parameter:**

Before configuring the parameter, check the deployment type of the remote site:

If the remote site is on Single Stack IPv4, then set the `remotesiteip` to IPv4 loadbalancer IP/FQDN of the remote site.

Example:

```
db-replication-svc.dbreplsvcdeployments[*].replication.remotesiteip
= "10.75.12.45"
```

If the remote site is on Dual Stack IPv4, then check both IPv4 and IPv6 load balancer IPs/FQDN of the remote site `db-replication-svc` service and assign both the values as comma separated strings.

```
db-replication-svc.dbreplsvcdeployments[*].replication.remotesiteip
= "10.75.12.45,4001:ab8::1"
```

Note

The `primarysignalhost`, `secondarysignalhost`, `localsiteip`, and `remotesiteip` are loadbalancer/external IPs. You can not assign them to a localhost or any private IP which is not accessible from the remote site.

4. **Adding the Site:**

Install the site with the `custom_values.yaml` file configured as per the procedure given in the [Adding a Georedundant cnDBTier Cluster](#) section. Perform the site addition procedure to add the site back to the existing replication sites.

5. **Changing the `preferredIpFamily`:**

Once both the sites are deployed with Dual Stack support, update the `preferredIpFamily` setting between the local and remote sites.

When Site1 is migrated to DualStack but Site2 is still on IPv4, the **`preferredIpFamily`** must remain as IPv4 towards Site2. However, after Site2 is also migrated to Dual Stack, the `preferredIpFamily` between Site1 and Site2 can be updated accordingly.

When both the sites are configured with both IPv4 and IPv6 addresses, communication uses the IP family specified by the preferredIpFamily configuration.

Updating the preferredIpFamily using the REST API:

You can use the following REST API to update the preferredIpFamily configuration for all the external communication between the sites.

Note

The preferredIpFamily change is local to each site. To fully update the preferred IP family between the two sites, call the same REST API from both the local and the remote sites.

Update the preferredIpFamily between two sites using the following REST API:

```
curl -X POST http://<base-url>:8080/db-tier/api/v1/ip-family/switch/local/  
/{localSiteName}/remote/{remoteSiteName}/preferred/{preferredIpFamily}/switch-  
over/{doSwitchOver}
```

Example:

```
curl -X POST http://<base-url>:8080/db-tier/api/v1/ip-family/switch/local/  
cluster1/remote/cluster2/preferred/IPv6/switch-over/true
```

The following table explains the REST API parameters and their descriptions:

Table 7-7 REST API Parameters and Descriptions

Parameter	Description
base-url	Indicates the db-monitor-svc Kubernetes service name of the local site.
localSiteName	Indicates the name of the local site from which the REST API is called.
remoteSiteName	Indicates the name of the remote site for which preferredIpFamily must be updated.
preferredIpFamily	Indicates the new IP family to prefer for communication. Supported values: IPv4 or IPv6
doSwitchOver	This parameter is a boolean flag. If set to true, the replication channel switches over to use the new preferred IP family. If set to false, the preferred IP family will be updated, but the replication channel will continue to use the existing connection until a manual switch over is performed later.

Updating the preferredIpFamily for all the configured remote sites:

```
curl -X POST http://<base-url>:8080/db-tier/api/v1/ip-family/switch/local/  
/{localSiteName}/preferred/{preferredIpFamily}/switch-over/{doSwitchOver}
```

Example:

```
curl -X POST http://<base-url>:8080/db-tier/api/v1/ip-family/switch/local/cluster1/preferred/IPv6/switch-over/true
```

Checking the status of the preferredIpFamily between the two sites:

```
curl -X GET http://<base-url>:8080/db-tier/api/v1/ip-family/preference/status/local-site/{localSiteName}/cmd-ids/{cmdIds}
```

Example:

```
curl -X GET http://<base-url>:8080/db-tier/api/v1/ip-family/preference/status/local-site/cluster1/cmd-ids/1
curl -X GET http://<base-url>:8080/db-tier/api/v1/ip-family/preference/status/local-site/cluster1/cmd-ids/1,2
```

Table 7-8 REST API Parameters and Descriptions

Parameter	Description
base-url	Indicates the db-monitor-svc Kubernetes service name of the local site.
localSiteName	Indicates the name of the local site from which the REST API is called.
cmdIds	The comma separated cmdIds provide as above change preferredIpFamily api will give response cmdIds .

7.15 Migrating cnDBTier to Use ASM for External Communication

Aspen Service Mesh (ASM) is a service mesh solution built on top of Istio, an open-source service mesh that enables secure, observable, and controlled communication between services in a Kubernetes cluster.

In cnDBTier, pods can be deployed with ASM integration, where the istio-proxy sidecar is automatically injected into each pod. This enables encrypted communication (through mTLS) for internal and external service-to-service traffic.

Until version 25.2.100, ASM support in cnDBTier was a simple binary configuration - either fully enabled or completely disabled.

Starting from 25.2.100, the configuration has been enhanced to support three distinct ASM sidecar injection modes. The boolean useasm flag is replaced with istioSidecarInject flag to support fine-grained control over sidecar injection.

- none (default):
 - No sidecar is injected.
 - cnDBTier pods operate outside of ASM.
 - All communication is unencrypted.

- all:
 - Sidecar is injected into all cnDBTier pods.
 - Both internal and external communication is encrypted and routed through ASM.
- external:
 - Sidecar is injected only to support external communication.
 - Internal communication remains outside of ASM.
 - Only the following traffic is routed through ASM:
 - * Georeplication: ndbmysqld (local) → ndbmysqld (remote)
 - * HTTP REST: db-replication-svc (local) → db-replication-svc (remote)

Enables ASM sidecar injection only for externally facing cnDBTier components.

Once Istio sidecar injection is migrated from `istioSidecarInject: all` to `istioSidecarInject: external`, rolling back requires a full reinstall with the `istioSidecarInject: all` configuration.

Prerequisites:

The following are the prerequisites to use this feature:

- A Kubernetes cluster with ASM or Istio must be installed and enabled.
- cnDBTier is deployed with ASM integration.

Assumption: Traffic will be diverted into other cnDBTier sites during the procedure.

Managing ASM for External Communication

This section explains the procedure to enable and configure the feature.

Enable:

You can enable this feature using the Helm. Following are some sample configurations:

```
global:
  istioSidecarInject:
    mode: none
```

```
global:
  istioSidecarInject:
    mode: all
```

To inject the sidecar to support external communication, enable the following parameters:

```
global:
  istioSidecarInject:
    mode: external
```

Migrating cnDBTier to Support ASM for External Communication

If you have previously deployed cnDBTier in an ASM-enabled namespace with the Istio sidecar (istio-proxy) injected into all cnDBTier pods, which means encrypted traffic for both internal and external service communication are routed through ASM, and now if you want to remove ASM

from internal communication while retaining it for external communication, perform the following migration procedure:

① Note

An in-service upgrade is not supported for this configuration change.

Before proceeding, you must:

- Divert traffic to another active site to maintain service availability.
- Isolate the current site from georeplication, ensuring no data sync or active traffic flows through the site during the update.

1. **Change the Schema:** This step is required only if you are upgrading from any release before 25.2.100. If you are using 25.2.100, then skip this step.

- a. Run the following commands after replacing the environment variables with values specific to your cluster:

```
export OCCNE_NAMESPACE="occne-cndbtier"
export MYSQL_CONNECTIVITY_SERVICE="mysql-connectivity-service"
export MYSQL_USERNAME="occneuser"
export MYSQL_PASSWORD="<password for the user occneuser>"
export DBTIER_REPLICATION_SVC_DATABASE="replication_info"
export DBTIER_BACKUP_SVC_DATABASE="backup_info"
export DBTIER_HBREPLICAGROUP_DATABASE="hbreplica_info"
export DBTIER_CLUSTER_INFO_DATABASE="cluster_info"
export REPLCHANNEL_GROUP_COUNT=<configured replication channel group count: 1/2/3>
export MYSQL_CMD="kubectl -n <namespace> exec <ndbmysqld-0 or ndbappmysqld-0 pod> -- mysql --binary-as-hex=0 --show-warnings"
export NDB_STS_NAME="<ndbmtl sts name>"
```

- b. (This step is optional) Enable or disable the password encryption.

```
# export ENABLE_ENCRYPTION="<true/false>"
```

- c. Run the schema upgrade.

```
occndbtier/files/hooks.sh --schema-upgrade
```

① Note

The hooks . sh file must be obtained from the 25.2.100 Helm chart only.

2. **Divert the traffic:**

Redirect the NF (Network Function) traffic away from the current site to another active cnDBTier site to safely remove ASM from internal communication, ensure the current site is not serving live traffic.

3. **Isolate the Site from Georeplication:**

- a. Gracefully stop all the georeplication:

- from the current cnDBTier site to other sites
- from other sites to the current site

This step is important to prevent disruption and ensures data consistency while modifying ASM configuration.

b. Perform the steps below to isolate the current site from the georeplication setup:

- i.** Use the namespace of the site where you are changing `istioSidecarInject.mode` to external. Run the following commands:

```
$ cd tools
$ source source_me
$ ./bin/dbtreplmgr --stop-repl-to-from=<current-site-id>
```

- ii.** Replace the `<current-site-id>` with the site ID of the current site.

This stops georeplication from the current site to all the other configured sites. For example:

If you are changing `istioSidecarInject.mode` to external for site id 1, run the following command:

```
$ ./bin/dbtreplmgr --stop-repl-to-from=1
```

If you are changing `istioSidecarInject.mode` to external for site id 2, run the following command:

```
$ ./bin/dbtreplmgr --stop-repl-to-from=2
```

If you are changing `istioSidecarInject.mode` to external for site id 3, run the following command:

```
$ ./bin/dbtreplmgr --stop-repl-to-from=3
```

If you are changing `istioSidecarInject.mode` to external for site id 4, run the following command:

```
$ ./bin/dbtreplmgr --stop-repl-to-from=4
```

4. Update the Custom Values File with the ASM Configuration:

- a.** Modify the `custom_values.yaml` file to set the ASM mode to external. This configuration ensures that only external communication (example, georeplication and REST calls between sites) is routed through ASM. Internal communication within the site will bypass ASM and occurs outside of the Istio mesh.

```
global:
  istioSidecarInject:
    mode: external
```

This step only includes ASM-specific configuration. For all other upgrade-related changes in the `custom_values.yaml`, refer to the "Upgrading cnDBTier Clusters" section in *Oracle Communications Cloud Native Core, cnDBTier User Guide*.

Note

- During this migration, do not change the topology of the dbtier. For example, the replica count, PVC values and so on.
- Do not modify the values of the following NDB variables, as changing any of them will require the ndbmttd pods to perform an initial restart.
- Do not change the following values since the helm upgrade will be run with the --no-hooks flag. The initial restart step will be skipped:

```
global:
  ndbConfigsThatNeedRestartTypeInitial:
    - DataDir
    - FileSystemPath
    - BackupDataDir
    - FragmentLogFileSize
    - InitFragmentLogFiles
    - NoOfFragmentLogFiles
    - EncryptedFileSystem
    - NoOfFragmentLogParts
    - FileSystemPathDD
    - FileSystemPathDataFiles
    - FileSystemPathUndoFiles
    - IndexStatSaveSize
    - IndexStatSaveScale
    - IndexStatTriggerPct
    - IndexStatTriggerScale
    - IndexStatUpdateDelay
    - RedoBuffer
```

5. Helm Upgrade with --no-hooks Flag:

Upgrade the cnDBTier Helm chart with --no-hooks to skip automated hook execution:

```
export OCCNE_RELEASE_NAME="mysql-cluster"
helm -n ${OCCNE_NAMESPACE} upgrade ${OCCNE_RELEASE_NAME} occndbtier -f
occndbtier/custom_values.yaml --no-hooks
```

6. Scale Down DBTier Pods:

Scale down all the cnDBTier pods to zero replicas. Perform the procedure to gracefully stop cnDBTier from the [Stopping cnDBTier](#) section. Wait until all the pods are fully terminated before proceeding.

7. Patch the updateStrategy:

Patch the ndbmttd, ndbmysqld, and ndbappmysqld stateful sets with an upgrade strategy as a rolling restart.

```
$ kubectl -n <namespace> patch sts <ndbmttd sts name> -p '{"spec":
{"updateStrategy":{"type":"RollingUpdate"}}}'
$ kubectl -n <namespace> patch sts <ndbmysqld sts name> -p '{"spec":
{"updateStrategy":{"type":"RollingUpdate"}}}'
$ kubectl -n <namespace> patch sts <ndbappmysqld sts name> -p '{"spec":
{"updateStrategy":{"type":"RollingUpdate"}}}'
```

8. Scale Up DBTier Pods:

Scale up the cnDBTier pods back to their original replica count. Perform the procedure to gracefully start cnDBTier as per the [Starting cnDBTier](#) section. Wait until all cnDBTier pods are in the running state before proceeding.

9. Start Georeplication Monitoring:

Before proceeding for georeplication, run the following script to enable monitoring of replication channel health as it was disabled in step 3:

- a. Use the namespace of the site where you are changing `istioSidecarInject.mode` to external. Run the following commands:

```
$ cd tools
$ source source_me
$ ./bin/dbtreplmgr --start-repl-to-from=<current-site-id>
```

- b. Replace the `<current-site-id>` with the site ID of the current site.

This stops georeplication from the current site to all the other configured sites.

For example:

If you are changing `istioSidecarInject.mode` to external for site id 1, run the following command:

```
$ ./bin/dbtreplmgr --start-repl-to-from=1
```

If you are changing `istioSidecarInject.mode` to external for site id 2, run the following command:

```
$ ./bin/dbtreplmgr --start-repl-to-from=2
```

If you are changing `istioSidecarInject.mode` to external for site id 3, run the following command:

```
$ ./bin/dbtreplmgr --start-repl-to-from=3
```

If you are changing `istioSidecarInject.mode` to external for site id 4, run the following command:

```
$ ./bin/dbtreplmgr --start-repl-to-from=4
```

10. Perform Georeplication Recovery (GRR):

After the pods are up, perform the [Starting cnDBTier Georeplication Between Sites Manually](#) procedure to recover the current site from a known good site in the georeplication group.

11. Start Georeplication Monitoring:

Once the georeplication is completed, run the following script to enable monitor of replication channel since it might have stored monitoring because of getting older records restored as part of georeplication.

- a. Use the namespace of the site where you are changing `istioSidecarInject.mode` to external. Run the following commands:

```
$ cd tools
$ source source_me
$ ./bin/dbtreplmgr --start-repl-to-from=<current-site-id>
```

- b. Replace the <current-site-id> with the site ID of the current site.

This stops georeplication from the current site to all the other configured sites.
For example:

If you are changing istioSidecarInject.mode to external for site id 1, run the following command:

```
$ ./bin/dbtreplmgr --start-repl-to-from=1
```

If you are changing istioSidecarInject.mode to external for site id 2, run the following command:

```
$ ./bin/dbtreplmgr --start-repl-to-from=2
```

If you are changing istioSidecarInject.mode to external for site id 3, run the following command:

```
$ ./bin/dbtreplmgr --start-repl-to-from=3
```

If you are changing istioSidecarInject.mode to external for site id 4, run the following command:

```
$ ./bin/dbtreplmgr --start-repl-to-from=4
```