Oracle® Communications

Cloud Native Core, cnDBTier User Guide

Release 24.3.1
G13337-03
December 2025

Oracle Communications Cloud Native Core, cnDBTier User Guide, Release 24.3.1

G13337-03

Contents

5 cnDBTier Metrics

6 cnDBTier Alerts

7 Maintenance Procedures

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 24.3.x.

Release 24.3.1 - G13337-03, December 2025

Maintenance Procedures
Updated the Extracting cnDBTier Cluster Script section to correct the CSAR package version.

Release 24.3.1 - G13337-02, March 2025

There are no updates to this document in this release.

Release 24.3.0 - G13337-01, October 2024

Feature Updates:
New Features:
Transparent Data Encryption (TDE):

- Added the Transparent Data Encryption (TDE) section to provide information about encrypting data in the data nodes using Transparent Data Encryption (TDE).

General Updates:

- Added the Service Type column for all APIs in the cnDBTier APIs section to indicate if the APIs provide realtime service or cached service.
- Metrics:
  - Added the following metrics in the cnDBTier Node Status Metrics section to provide details about the metrics used to measure the cluster status:
    * `db_tier_cluster_disconnect`
    * `db_tier_cluster_status`
- Alerts:
  - Added the `MYSQL_NDB_CLUSTER_DISCONNECT` alert in the cnDBTier Cluster Status Alerts section.
  - Added the `GEOREPLICATION_RECOVERY_FAILED` alert in the cnDBTier Replication Alerts section.
  - Updated the SNMP Trap ID value of the `BACKUP_STORAGE_LOW` alert from 2024 to 2014 in the cnDBTier Automated Backup Alerts section.
  - Updated the SNMP Trap ID value of the `REPLICATION_EPOCHS_LOST` alert from 2024 to 2023 in the cnDBTier Replication Error Skip Alerts section.
  - Updated the name of the `PVC_NOT_ACCESSIBLE` alert in the cnDBTier PVC Health Monitoring Alerts section.
  - Updated the condition of the `NODE_DOWN` alert in the cnDBTier Node Status Alerts section.
  - Updated the SNMP Trap ID value of the `HIGH_CPU` (critical) alert from 2002 to 2035 in the cnDBTier CPU Usage Alerts section.
  - Updated the SNMP Trap ID value of the `BINLOG_STORAGE_LOW` (major) alert from 2007 to 2036 in the cnDBTier Bin Log Usage Alerts section.
- Maintenance Procedures:

  - Added a note in the Changing cnDBTier Passwords section to state that any character in the password that doesn't meet the password requirements can break the functionality of the script.
  - Added the Scaling of ndbmgmd Pods section to provide the procedure to scale ndbmgmd pods vertically.
  - Added the Modifying Transparent Data Encryption Password section to provide the procedure to modify TDE password.
  - Added the Modifying HTTPS Certificates section to describe the procedure to modify HTTPS certificates.
- Updated the MySQL version from "8.0.37" to "8.4.2" in the entire document.
- Updated the timestamps in the sample outputs in the entire document.

1

Introduction

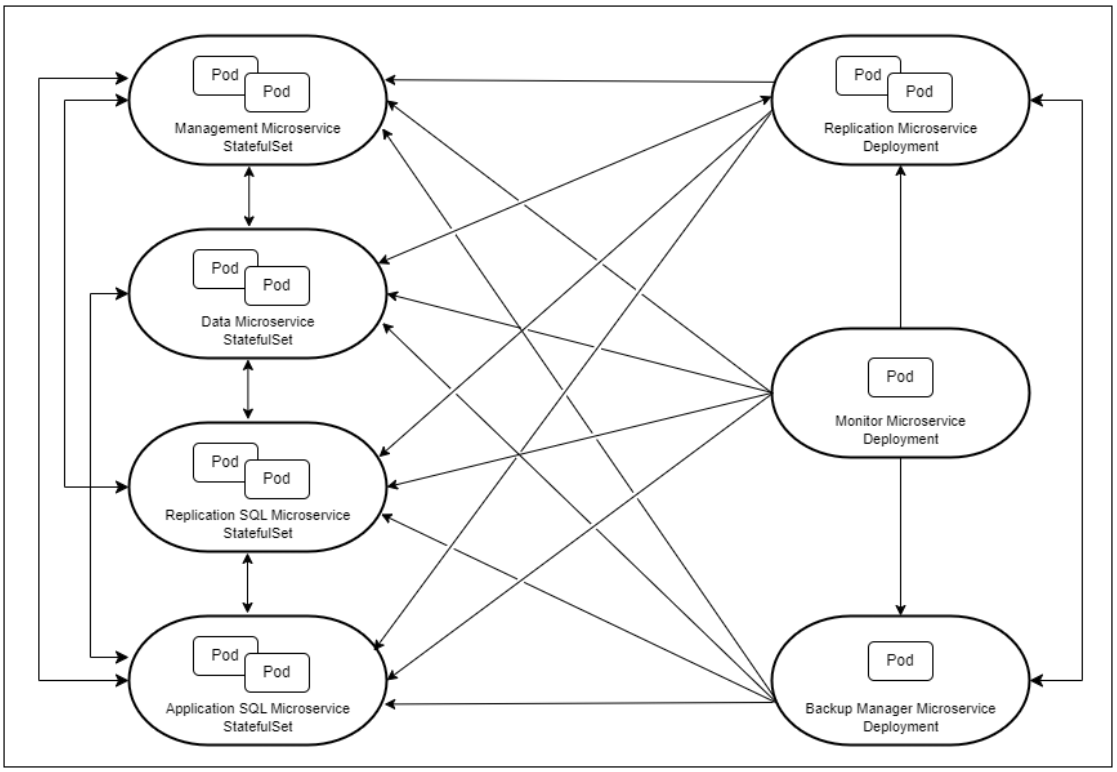
Oracle Communications Cloud Native Core, cnDBTier (cnDBTier) is the geodiverse database layer that provides persistence storage for state and subscriber data on a cloud environment such as Oracle Communications Cloud Native Core, Cloud Native Environment (CNE) or Oracle Cloud Infrastructure (OCI). This 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

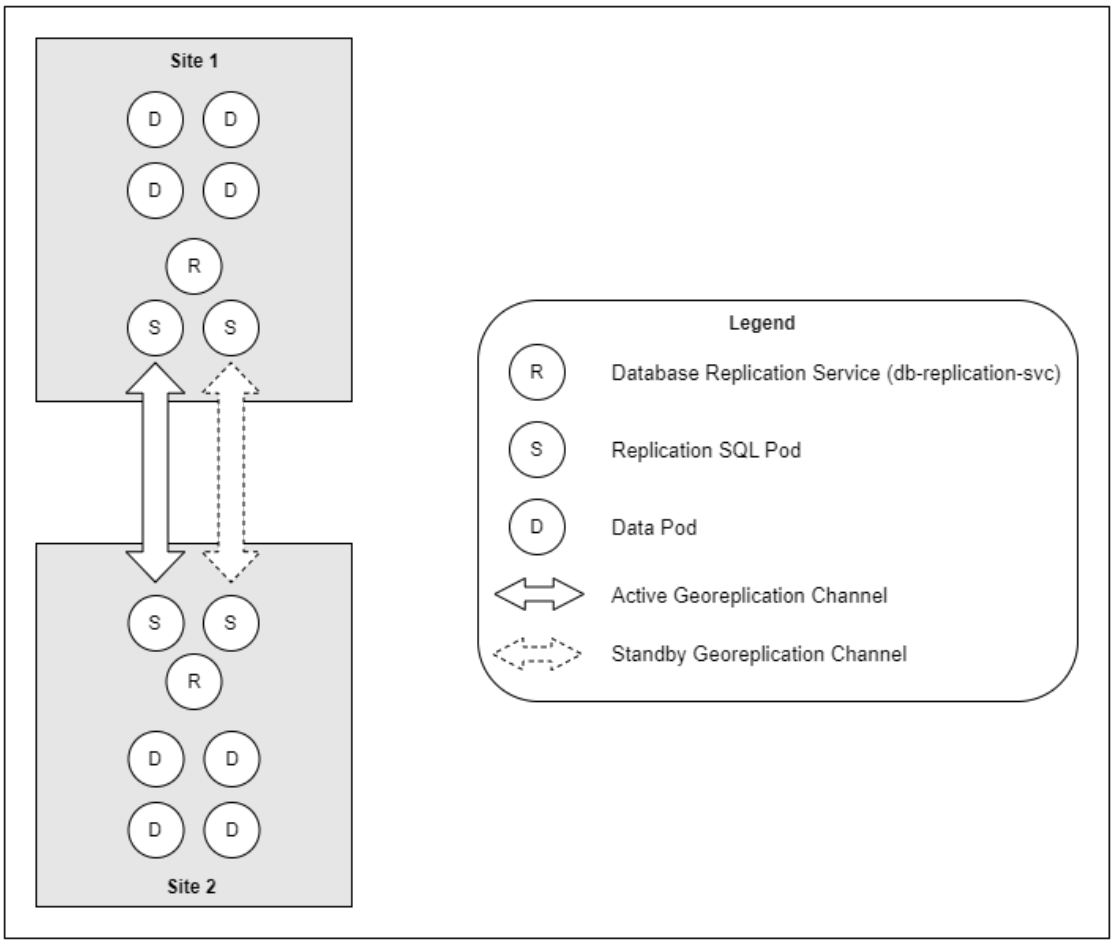
cnDBTier supports two-site, three-site, and four-site georeplication setups. This section provides information and figures to understand the replication setups between cnDBTier sites based on their topology.

> **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

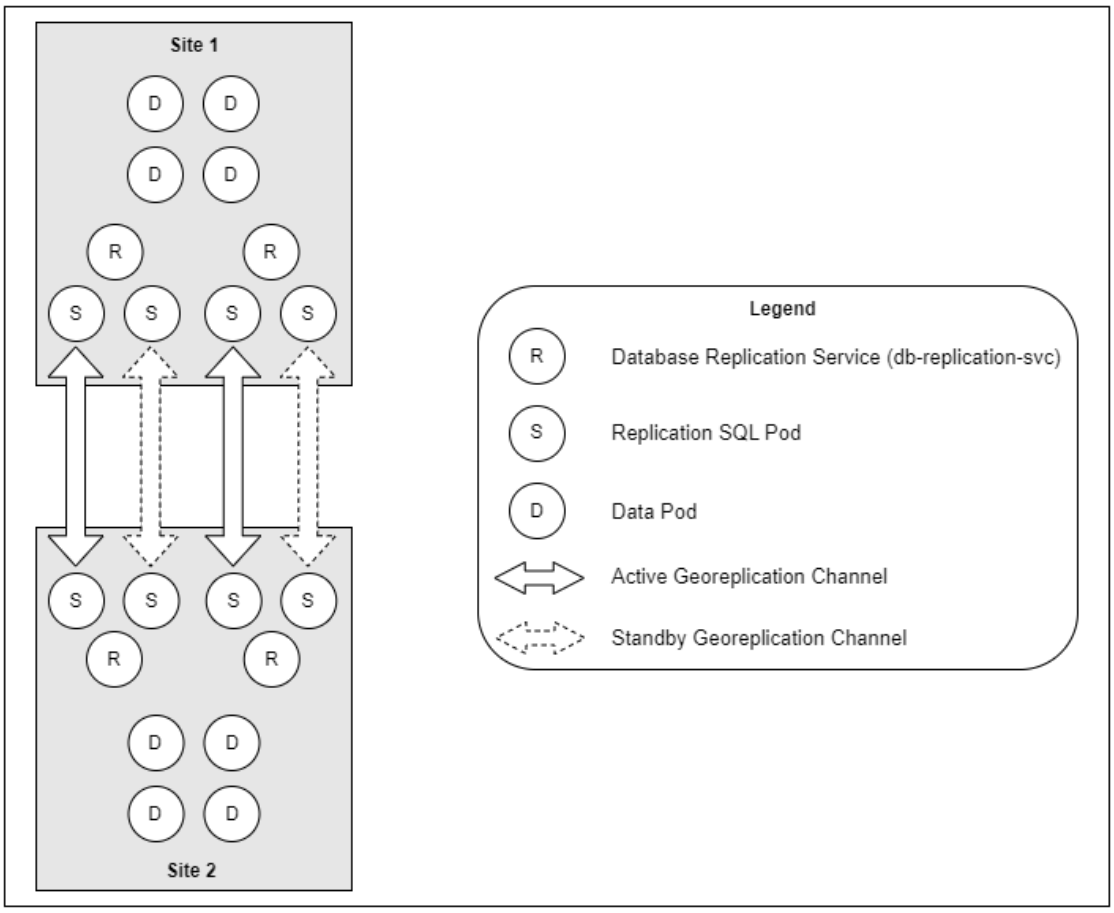
The following images depict the replication SQL pod connection between two cnDBTier sites with single or multiple replication channel groups. They also depict how replication service pods communicate with each other to set up the replication.

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

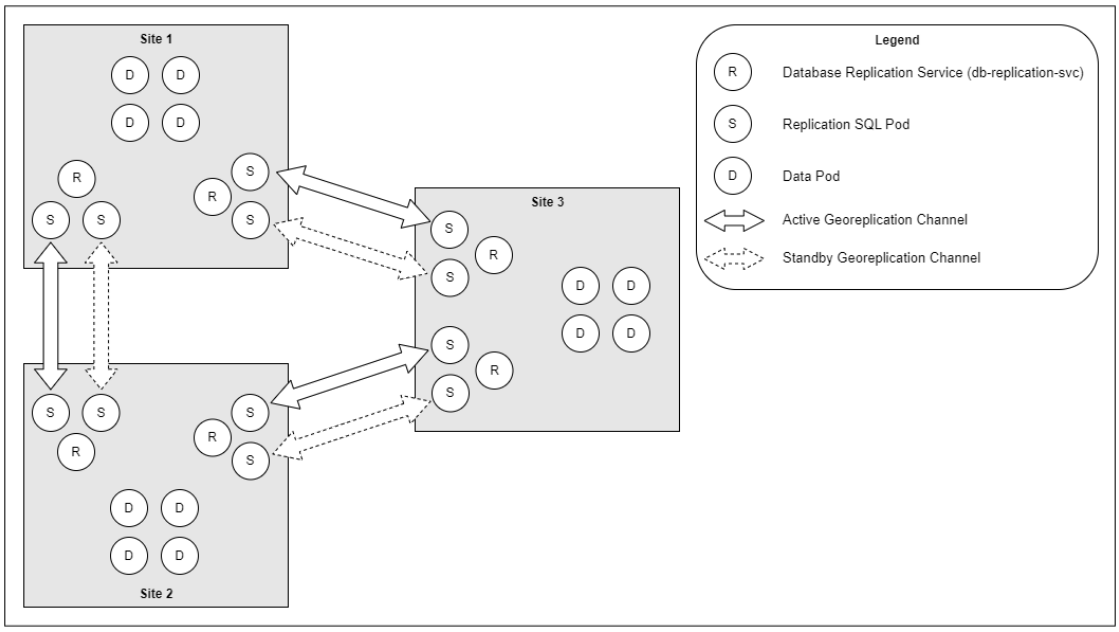


This sample setup contains two cnDBTier sites (Site 1 and Site 2). Each site has two replication channel groups and each replication channel group contains a database replication service pod and two replication SQL pods. One replication channel group on one site (Site a) communicates with one replication 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.

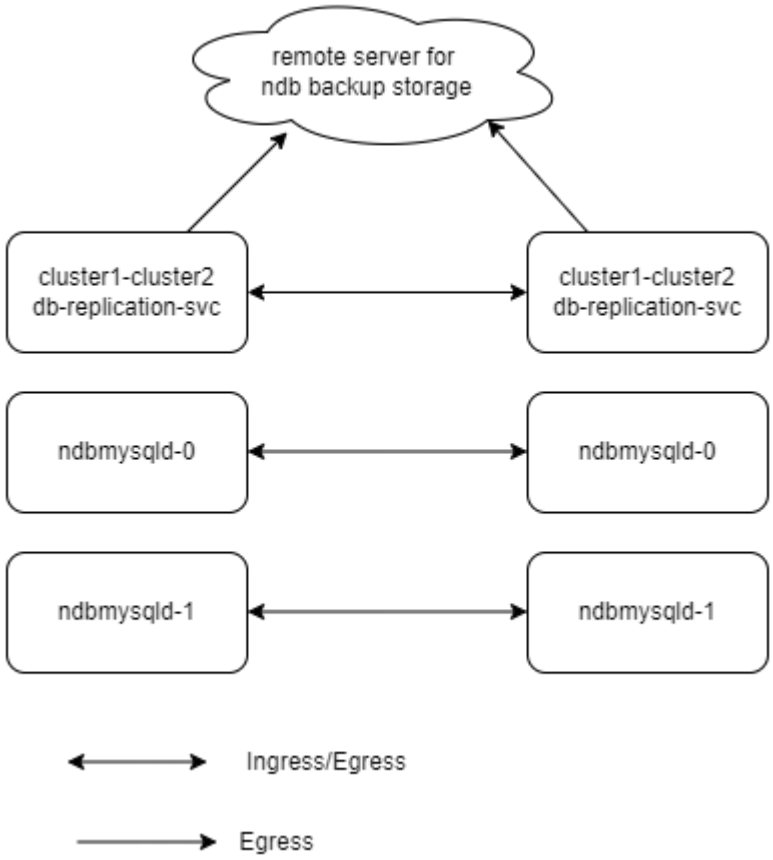
> **Note**
>
> You can extend the three site setup for creating a four-site georeplication setup.

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.

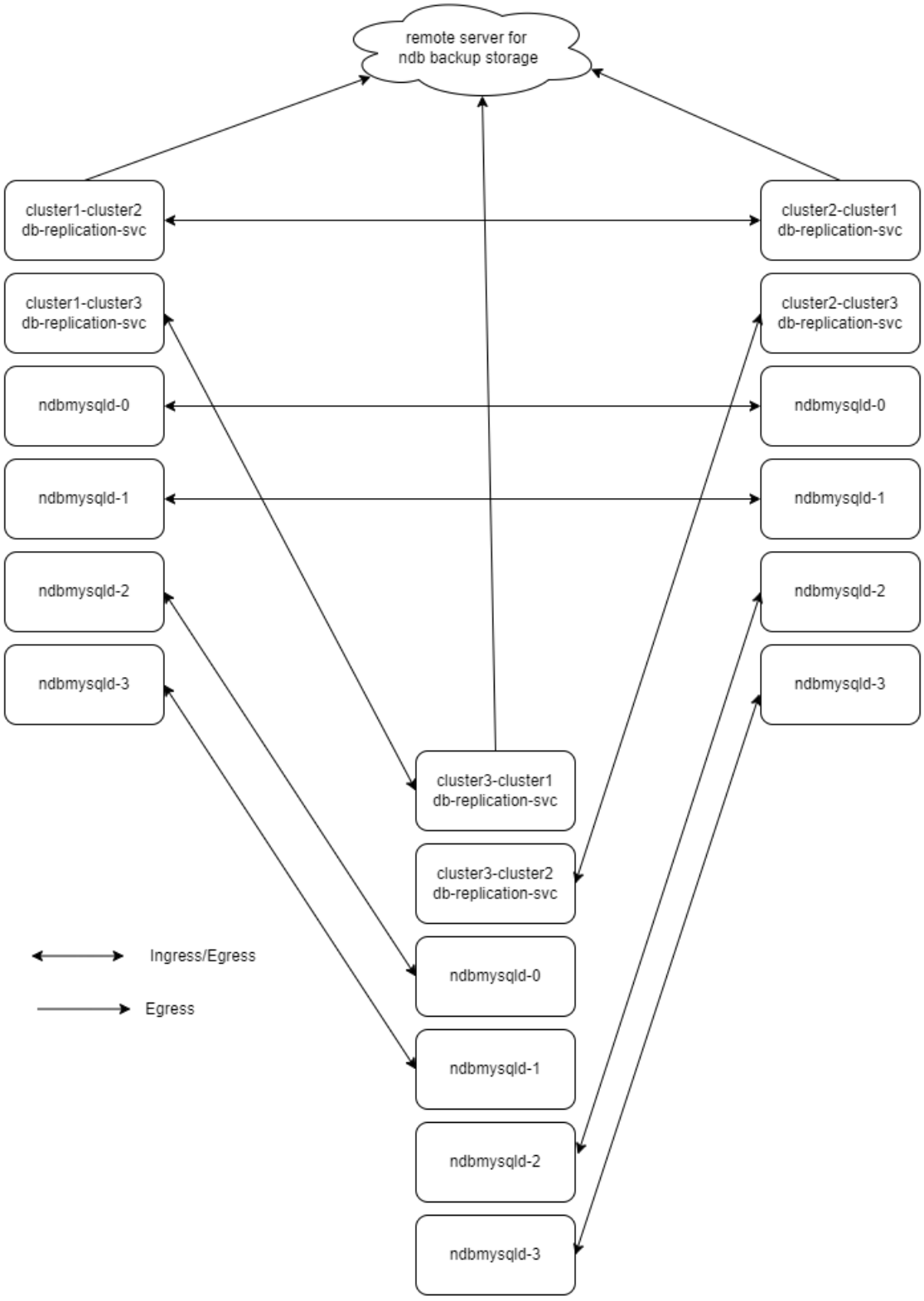
The following diagram depicts the ingress or egress routes in a two-cluster setup:

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 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 3-1 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
37	Enabled	397.5	5.72%

> **Note**
>
> Analysis and testing reveal that enabling TDE increases the restart time of data nodes by an average of 8.33%. Ensure that you 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*.

3.2 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*.

3.3 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:
    mysqld:
      replica_parallel_workers: 4
      binlog_transaction_dependency_tracking: "WRITESET"
      ndb_log_transaction_dependency: 'ON'
```

Sample configuration to disable MTA:

```
global:
  additionalndbconfigurations:
    mysqld:
      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.

3.4 TLS Support for Georeplication

Until now, 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.

On enabling this feature, cnDBTier performes or supports the following operations:

- 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 fault recovery. That is, when a fault recovery completes successfully, the system reestablishes the encrypted connection between the replication channels. This ensures that the TLS is kept intact even after a fault 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.

Managing TLS Support for Georeplication

The following sections provide details on managing TLS support for georeplication:

Enabling TLS Support for Georeplication
You can enable or disable the TLS feature using the `custom_values.yaml` file. While enabling TLS, you must also provide the necessary certificates for the respective ndbmysqld pods in the `custom_values.yaml` file. For the procedure to enable TLS and provide the required certificates, see Enabling TLS for Georeplication.

Modifying cnDBTier Certificates for TLS Support
You can modify the cnDBTier certificates that the system uses to establish an encrypted connection between georeplication sites using TLS. For procedure, see Modifying cnDBTier Certificates to Establish TLS Between Georeplication Sites.

3.5 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*.

3.6 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.

3.7 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.

3.8 cnDBTier Automated Backup

The cnDBTier backup service creates and safely stores DB 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 per MySQL cluster data node basis. Each data node creates a backup of the hosted data and maintains a copy of the backup.

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

Table 3-2 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.

The status of the DB backups is stored in the DB nodes and is accessible to the DB Monitor service. The service includes the status in the metrics.

3.9 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.

3.10 Database Backup Encryption and Secure Transfer

cnDBTier supports creation of database backups at configured intervals. It's important to encrypt the backups as they contain sensitive subscriber data that are transferred to remote servers and sites for performing fault recovery. 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 `<Path of the remote server configured in custom values file (/global/remotetransfer/remoteserverpath) >/<site name of cnDBTier>` path 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, usernname, 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.

3.11 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.

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 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 4-1 cnDBTier APIs for CNC Console

Resource	Rest URL	HTTP Method	Request Payload	Response Code	Response Payload	ServiceType
Site specific real time replication status	http://<base-uri>/ocdbtier/status/replication/realtime/{remoteSiteName}	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: { "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",	Realtime

Table 4-1 (Cont.) cnDBTier APIs for CNC Console

Resource	Rest URL	HTTP Method	Request Payload	Response Code	Response Payload	ServiceType
					"role": "ACTIVE" }, { "remote_replication_ip": "127.0.0.2", "role": "STANDBY" }] }, { "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" }]	

Table 4-1 (Cont.) cnDBTier APIs for CNC Console

Resource	Rest URL	HTTP Method	Request Payload	Response Code	Response Payload	ServiceType

					}
					]
					}
					• 400 Bad Request:
					{
					"title": "Bad Request",
					"status": 400,
					"detail": "<Exception details>",
					"cause": <cause>,
					"invalidParams": <params>
					}
					• 404 Not Found:
					{
					"title": "Not Found",
					"status": 404,
					"detail": "<Exception details>",
					"cause": <cause>,
					"invalidParams": <params>
					}
					• 500 Internal Server Error:
					{
					"title": "Internal Server	

Table 4-1 (Cont.) cnDBTier APIs for CNC Console

Resource	Rest URL	HTTP Method	Request Payload	Response Code	Response Payload	ServiceType

					Error",
					"status": 500,
					"detail":
					"<Exception details>",
					"cause":
					<cause>,
					
					"invalidParams":
					<params>
					}
					• 503 Service Unavailable - Cross-site replication is not enabled:
					
					{
					"title":
					"Service Unavailable",
					"status": 503,
					"detail":
					"<Exception details>",
					"cause":
					<cause>,
					
					"invalidParams":
					<params>
					}	

Table 4-1 (Cont.) cnDBTier APIs for CNC Console

Resource	Rest URL	HTTP Method	Request Payload	Response Code	Response Payload	ServiceType

Real time overall replication status	http://<base-uri>/ocdbtier/status/replication/realtime	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:
					{
					[
					{
					"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,
						Realtime

Table 4-1 (Cont.) cnDBTier APIs for CNC Console

Resource	Rest URL	HTTP Method	Request Payload	Response Code	Response Payload	ServiceType

					secondsBehindRemote: <seconds>
					}]
					},
					{
					
					"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,	

Table 4-1 (Cont.) cnDBTier APIs for CNC Console

Resource	Rest URL	HTTP Method	Request Payload	Response Code	Response Payload	ServiceType

					<pre>secondsBehindRemote: <seconds>
					}]
					}
					]
					}</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>,</pre>	

Table 4-1 (Cont.) cnDBTier APIs for CNC Console

Resource	Rest URL	HTTP Method	Request Payload	Response Code	Response Payload	ServiceType
					"invalidParams": <params> }	
Real time local cluster status	http://<base-uri>/ocdbtier/status/cluster/local/realtime	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: { "clusterName":"<current-cluster-name>", "clusterStatus":"UP/DOWN" } • 500 Internal Server Error: { "title": "Internal Server Error", "status": 500, "detail": "<Exception details>", "cause": <cause>, "invalidParams": <params> }	Realtime

Table 4-1 (Cont.) cnDBTier APIs for CNC Console

Resource	Rest URL	HTTP Method	Request Payload	Response Code	Response Payload	ServiceType

cnDBTier version	http://<base-uri>/ocdbtier/version	GET	NA	• 200 OK
				• 500 Internal Server Error
					• 200 OK:
					{
					"cndbtier_version": "24.3.1",
					"ndb_version": "ndb-8.4.2"
					}
					• 500 Internal Server Error:
					{
					"title": "Internal Server Error",
					"status": 500,
					"detail": "<Exception details>",
					"cause": <cause>,
					"invalidParams": <params>
					}
						Realtime

Table 4-1 (Cont.) cnDBTier APIs for CNC Console

Resource	Rest URL	HTTP Method	Request Payload	Response Code	Response Payload	ServiceType
List completed backups of the current site	http://<base-uri>/ocdbtier/list/backups	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: { "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	Realtime

Table 4-1 (Cont.) cnDBTier APIs for CNC Console

Resource	Rest URL	HTTP Method	Request Payload	Response Code	Response Payload	ServiceType

					identifier>",
					
					"backupSize": "<size of backup>",
					
					"creationTimeStamp": "<timestamp at which backup was initiated>"
					}
					]
					}
					• 404 NOT FOUND Table Missing:
					
					{
					"title": "Not Found",
					"status": 404,
					"detail": "DBTIER_BACKUP_INFO table doesn't exist in the current site so cannot list the backup",
					"cause": null,
					
					"invalidParams": null
					}
					• 500 Internal Server Error:
					
					{
					"title": "Internal Server Error",	

Table 4-1 (Cont.) cnDBTier APIs for CNC Console

Resource	Rest URL	HTTP Method	Request Payload	Response Code	Response Payload	ServiceType
					"status": 500, "detail": "Could not list the backups", "cause": null, "invalidParams": null }	

Table 4-1 (Cont.) cnDBTier APIs for CNC Console

Resource	Rest URL	HTTP Method	Request Payload	Response Code	Response Payload	ServiceType

Database Statistic Reports	http://<base-uri>/ocdbtier/statistics/report/dbinfo	GET	NA	• 200 OK
				• 500 INTERNAL_SERVER_ERROR
				• 503 SERVICE_UNAVAILABLE
					• 200 OK:
					{
					"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>
					}
						Realtime

Table 4-1 (Cont.) cnDBTier APIs for CNC Console

Resource	Rest URL	HTTP Method	Request Payload	Response Code	Response Payload	ServiceType

					]
					},
					{
					"dbName": "<database name>",
					"tables": [
					{
					"tableName": "<table name>",
					"rowCount": <row count>
					},
					{
					"tableName": "<table name>",
					"rowCount": <row count>
					}
					]
					}
					]
					}
					• 500 INTERNAL_SERVER_ERROR:
					{
					"title": "Internal Server Error",
					"status": 500,
					"detail": "<Exception Details>",
					"cause": null,	

Table 4-1 (Cont.) cnDBTier APIs for CNC Console

Resource	Rest URL	HTTP Method	Request Payload	Response Code	Response Payload	ServiceType

					"invalidParams": null
					}
					• 503 SERVICE_UNAVAILABLE:
					{
					"title": "Service Unavailable",
					"status": 503,
					"detail": "<Exception details>",
					"cause": null,
					"invalidParams": null
					}	

Table 4-1 (Cont.) cnDBTier APIs for CNC Console

Resource	Rest URL	HTTP Method	Request Payload	Response Code	Response Payload	ServiceType
Overall HeartBeat Status	http://<base-uri>/ocdbtier/heartbeat/status	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: { "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>", "replicationChanne	Realtime

Table 4-1 (Cont.) cnDBTier APIs for CNC Console

Resource	Rest URL	HTTP Method	Request Payload	Response Code	Response Payload	ServiceType
					lGroupId": <replication channel group id> }], "siteName": "<current_site_name>", } • 503 SERVICE_UNAVAILABLE - Not able to query the Data Base: { "title": "Service Unavailable", "status": 503, "detail": "Not able to query the Data Base", "cause": null, "invalidParams": null } • 500 INTERNAL_SERVER_ERROR - Could Not Check The HeartBeat Status: { "title": "Internal Server Error", "status": 500, "detail": "Could Not Check The HeartBeat	

Table 4-1 (Cont.) cnDBTier APIs for CNC Console

Resource	Rest URL	HTTP Method	Request Payload	Response Code	Response Payload	ServiceType
					Status", "cause": null, "invalidParams": null }	

Table 4-1 (Cont.) cnDBTier APIs for CNC Console

Resource	Rest URL	HTTP Method	Request Payload	Response Code	Response Payload	ServiceType

On-demand Backup	http://<base-uri>/ocdbtier/on-demand/backup/initiate	GET	NA	• 200 OK
				• 404 NOT FOUND
				• 500 INTERNAL_SERVER_ERROR
					• 200 OK:
					{
					"siteName": "<current_site_name>",
					"drStatus": "<drStatus>",
					"backupId": "<backupId>",
					"backupStatus": "<backupStatus>",
					"remoteTransferStatus": "<remoteTransferStatus>",
					"initiateBackup": false/true,
					"transferStatus": "<transferStatus>"
					}
					• 404 NOT FOUND:
					{
					"title": "Not Found",
					"status": 404,
					"detail": "<Exception message>",
					"cause": null,
					"invalidParams": null
					}
						Realtime

Table 4-1 (Cont.) cnDBTier APIs for CNC Console

Resource	Rest URL	HTTP Method	Request Payload	Response Code	Response Payload	ServiceType

					• 500 INTERNAL_SERVER_ERROR:
					
					{
					"title": "Internal Server Error",
					"status": 500,
					"detail": "<Exception message>",
					"cause": null,
					
					"invalidParams": null
					}	

Table 4-1 (Cont.) cnDBTier APIs for CNC Console

Resource	Rest URL	HTTP Method	Request Payload	Response Code	Response Payload	ServiceTyp
Initiate on-demand backup	http://<base-uri>/ocdbtier/on-demand/backup/initiate	POST	{ "siteName": "<current_site_name>", "drStatus": "<drStatus>", "backupId": "<backupId>", "backupStatus": "<backupStatus>", "remoteTransferStatus": "<remoteTransferStatus>", "initiateBackup":false/true, "transferStatus": "<transferStatus>" }	• 200 OK • 400 BAD REQUEST • 404 NOT FOUND • 500 INTERNAL_SERVER_ERROR	• 200 OK: { "siteName": "<current_site_name>", "drStatus": "<drStatus>", "backupId": "<backupId>", "backupStatus": "<backupStatus>", "remoteTransferStatus": "<remoteTransferStatus>", "initiateBackup": false/true, "transferStatus": "<transferStatus>" } • 404 BAD REQUEST: { "title": "Bad Request", "status": 400, "detail": "Couldn't initiate the backup when initiateBackup flag is set to false", "cause": null, "invalidParams": [	Realtime

Table 4-1 (Cont.) cnDBTier APIs for CNC Console

Resource	Rest URL	HTTP Method	Request Payload	Response Code	Response Payload	ServiceType

					{
					"param": "initiateBackup",
					"reason": "Couldn't initiate the backup when initiateBackup flag is set to false"
					}
					]
					}
					• 404 NOT FOUND:
					{
					"title": "Not Found",
					"status": 404,
					"detail": "<Exception message>",
					"cause": null,
					"invalidParams": null
					}
					• 500 INTERNAL_SERVER_ERROR:
					{
					"title": "Internal Server Error",
					"status": 500,
					"detail": "BACKUP_INITIATION_FAILED",	

Table 4-1 (Cont.) cnDBTier APIs for CNC Console

Resource	Rest URL	HTTP Method	Request Payload	Response Code	Response Payload	ServiceType
					"cause": null, "invalidParams": null }	

Table 4-1 (Cont.) cnDBTier APIs for CNC Console

Resource	Rest URL	HTTP Method	Request Payload	Response Code	Response Payload	ServiceType

Health status of replication service	http://<base-uri>/ocdbtier/health-info/replication-svc/status	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:
					{
					"localSiteName": "<siteName>",
					"healthStatusDetails": [
					{
					"serviceName": "<serviceName>",
					"serviceStatus": "UP/DOWN",
					"connectionToDbStatus": "UP/DOWN",
					"overAllReplicationServiceHealth": "UP/DOWN"
					},
					{
					"serviceName": "<serviceName>",
					"serviceStatus": "UP/DOWN",
					"connectionToDbStatus": "UP/DOWN",
					"overAllReplicationServiceHealth": "UP/DOWN"
					}.....
					]
					}
						Realtime

Table 4-1 (Cont.) cnDBTier APIs for CNC Console

Resource	Rest URL	HTTP Method	Request Payload	Response Code	Response Payload	ServiceType

					• 503 SERVICE_UNAVAILABLE - Replication Service IP is empty:
					
					`{ "title": "Service Unavailable", "status": 503, "detail": "Replication Service IP is empty", "cause": null, "invalidParams": null }`
					• 500 INTERNAL_SERVER_ERROR - Could Not Check The Replication Health Status:
					
					`{ "title": "Internal Server Error", "status": 500, "detail": "Could Not Check The Health Status", "cause": null, "invalidParams": null }`	

Table 4-1 (Cont.) cnDBTier APIs for CNC Console

Resource	Rest URL	HTTP Method	Request Payload	Response Code	Response Payload	ServiceType

Health status of monitor service	http://<base-uri>/ocdbtier/health-info/monitor-svc/status	GET	NA	200 OK (Monitor Service is healthy)	200 OK:
					{
					"serviceName": "mysql-cluster-db-monitor-svc",
					"connectionToDbStatus": "UP",
					"metricScrapeStatus": "UP",
					"overAllMonitorServiceHealth": "UP"
					}
						Realtime

Table 4-1 (Cont.) cnDBTier APIs for CNC Console

Resource	Rest URL	HTTP Method	Request Payload	Response Code	Response Payload	ServiceType

Overall health status of NDB data service	http://<base-uri>/ocdbtier/health-info/ndb-svc/status	GET	NA	• 200 OK
				• 404 NOT FOUND - sitename not found
				• 500 INTERNAL_SERVER_ERROR - Could Not Check The Health Status
					• 200 OK:
					{
					"localSiteName": "<siteName>",
					"ndbHealthStatusDetails": [
					{
					"serviceName": "<serviceName>",
					"serviceStatus": "UP/DOWN",
					"pvcHealthStatus": "UP/DOWN"
					},
					{
					"serviceName": "<serviceName>",
					"serviceStatus": "UP/DOWN",
					"pvcHealthStatus": "UP/DOWN"
					}.....
					]
					}
					• 404 NOT FOUND - sitename not found:
					{
					"title": "Not Found",
					"status": 404,
						Realtime

Table 4-1 (Cont.) cnDBTier APIs for CNC Console

Resource	Rest URL	HTTP Method	Request Payload	Response Code	Response Payload	ServiceType
					<pre>"detail": "sitename not found", "cause": null, "invalidParams": null }</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 4-1 (Cont.) cnDBTier APIs for CNC Console

Resource	Rest URL	HTTP Method	Request Payload	Response Code	Response Payload	ServiceType

Health status of backup manager service	http://<base-uri>/ocdbtier/health-info/backup-mgr-svc/status	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
						Realtime

Table 4-1 (Cont.) cnDBTier APIs for CNC Console

Resource	Rest URL	HTTP Method	Request Payload	Response Code	Response Payload	ServiceType

					Check The Health Status:
					
					{
					"title": "Internal Server Error",
					"status": 500,
					"detail": "Could Not Check The Health Status",
					"cause": null,
					
					"invalidParams": null
					}	

Table 4-1 (Cont.) cnDBTier APIs for CNC Console

Resource	Rest URL	HTTP Method	Request Payload	Response Code	Response Payload	ServiceType

cnDBTier Cluster Details	http://<base-uri>/ocdbtier/list/clusterdetails	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,</pre>
						Realtime

Table 4-1 (Cont.) cnDBTier APIs for CNC Console

Resource	Rest URL	HTTP Method	Request Payload	Response Code	Response Payload	ServiceType

					"invalidParams": null
					}
					• 500 INTERNAL SERVER ERROR:
					{
					"title": "INTERNAL_SERVER_ERROR",
					"status": 500,
					"detail": "<Error message>",
					"cause": null,
					"invalidParams": null
					}	

Table 4-1 (Cont.) cnDBTier APIs for CNC Console

Resource	Rest URL	HTTP Method	Request Payload	Response Code	Response Payload	ServiceTyp

Get Failed cnDBTier Clusters	http://<base-uri>/ocdbtier/markcluster/failed	GET	NA	• 200 OK
				• 404 NOT FOUND
				• 500 INTERNAL SERVER ERROR
					• 200 OK:
					{
					"clusterNameTobeMarkedAsFailed": "",
					"failedClusterNames": [
					"<failedcluster1>",
					"<failedcluster2>"
					]
					}
					• 404 NOT FOUND:
					{
					"title": "Not Found",
					"status": 404,
					"detail": "<Error message>",
					"cause": null,
					"invalidParams": null
					}
					• 500 INTERNAL SERVER ERROR:
					{
					"title": "INTERNAL_SERVER_ERROR",
					"status": 500,
					"detail": "<Error message>",
						Realtime

Table 4-1 (Cont.) cnDBTier APIs for CNC Console

Resource	Rest URL	HTTP Method	Request Payload	Response Code	Response Payload	ServiceType
					"cause": null, "invalidParams": null }	

Table 4-1 (Cont.) cnDBTier APIs for CNC Console

Resource	Rest URL	HTTP Method	Request Payload	Response Code	Response Payload	ServiceType

Mark cnDBTier Clusters As Failed	http://<base-uri>/ocdbtier/markcluster/failed	PUT	{
			"clusterNameTobeMarkedAsFailed": "<failedcluster>",
			"failedClusterNames": []
			}
				• 200 OK
				• 400 BAD REQUEST
				• 404 NOT FOUND
				• 500 INTERNAL SERVER ERROR
				• 503 SERVICE_UNAVAILABLE
					• 200 OK:
					{
					"clusterNameTobeMarkedAsFailed": "",
					"failedClusterNames": [
					"<failedcluster1>",
					"<failedcluster2>"
					]
					}
					• 400 BAD REQUEST:
					{
					"title": "BAD_REQUEST",
					"status": 400,
					"detail": "<Error message>",
					"cause": null,
					"invalidParams": null
					}
					• 404 NOT FOUND:
					{
					"title": "Not Found",
					"status": 404,
					"detail": "<Error message>",
					"cause": null,
						Realtime

Table 4-1 (Cont.) cnDBTier APIs for CNC Console

Resource	Rest URL	HTTP Method	Request Payload	Response Code	Response Payload	ServiceType

					"invalidParams": null
					}
					• 500 INTERNAL SERVER ERROR:
					{
					"title": "INTERNAL_SERVER_ERROR",
					"status": 500,
					"detail": "<Error message>",
					"cause": null,
					"invalidParams": null
					}
					• 503 SERVICE_UNAVAILABLE:
					{
					"title": "Service Unavailable",
					"status": 503,
					"detail": "<Error message>",
					"cause": null,
					"invalidParams": null
					}	

Table 4-1 (Cont.) cnDBTier APIs for CNC Console

Resource	Rest URL	HTTP Method	Request Payload	Response Code	Response Payload	ServiceType

Monitor Georeplication Recovery Status	http://<base-uri>/ocdbtier/faultrecovery/status	GET	NA	• 200 OK
				• 404 NOT FOUND
				• 500 INTERNAL SERVER ERROR
					• 200 OK:
					{
					"localClusterName": "<local cluster name>",
					"faultRecoverStatus": [
					{
					"clusterName": "<clusterName>",
					"faultRecoverState": "<faultRecoverState>"
					},
					{
					"clusterName": "<clusterName>",
					"faultRecoverState": "<faultRecoverState>"
					},
					{
					"clusterName": "<clusterName>",
					"faultRecoverState": "<faultRecoverState>"
					}
						Realtime

Table 4-1 (Cont.) cnDBTier APIs for CNC Console

Resource	Rest URL	HTTP Method	Request Payload	Response Code	Response Payload	ServiceType

					]
					}
					• 404 NOT FOUND:
					{
					"title": "Not Found",
					"status": 404,
					"detail": "<Error message>",
					"cause": null,
					"invalidParams": null
					}
					• 500 INTERNAL SERVER ERROR:
					{
					"title": "INTERNAL_SERVER_ERROR",
					"status": 500,
					"detail": "<Error message>",
					"cause": null,
					"invalidParams": null
					}	

Table 4-1 (Cont.) cnDBTier APIs for CNC Console

Resource	Rest URL	HTTP Method	Request Payload	Response Code	Response Payload	ServiceType

Start Georeplication Recovery	http://<base-uri>/ocdbtier/faultrecovery/start	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:
					{
					"backupClusterName": "<backup cluster name>",
					"failedClusterName": "<failed cluster name>"
					}
					• 400 BAD REQUEST:
					{
					"title": "INVALID INPUT",
					"status": 400,
					"detail": "<Error message>",
					"cause": null,
					"invalidParams": null
					}
					• 404 NOT FOUND:
					{
					"title": "Not Found",
					"status": 404,
					"detail": "<Error message>",
					"cause": null,
					"invalidParams": null
					}
						Realtime

Table 4-1 (Cont.) cnDBTier APIs for CNC Console

Resource	Rest URL	HTTP Method	Request Payload	Response Code	Response Payload	ServiceType

					• 409 CONFLICT:
					
					{
					"title": "Conflict",
					"status": 409,
					"detail": "<Error message>",
					"cause": null,
					
					"invalidParams": null
					}
					• 500 INTERNAL SERVER ERROR:
					
					{
					"title": "INTERNAL_SERVER_ERROR",
					"status": 500,
					"detail": "<Error message>",
					"cause": null,
					
					"invalidParams": null
					}
					• 503 SERVICE_UNAVAILABLE:
					
					{
					"title": "Service Unavailable",
					"status": 503,
					"detail": "<Error message>",	

Table 4-1 (Cont.) cnDBTier APIs for CNC Console

Resource	Rest URL	HTTP Method	Request Payload	Response Code	Response Payload	ServiceType
					"cause": null, "invalidParams": null }	

> **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`

4.2 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 4-2 cnDBTier Backup API

Resource	Rest URL	HTTP Method	Request Payload	Response Code	Response Payload	ServiceType

Site Specific On-demand Backup Creation	http://<base-uri>/db-tier/on-demand/backup/initiate	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:
					{
					"backup_id":"<Backup id>",
					"status":"<Backup Initiation Status>"
					"backup_encryption_flag": "<backup encryption enabled/disabled>"
					}
					• 400 BAD REQUEST request is invalid:
					{
					"backup_id":"",
					"status":"BACKUP_INITIATION_FAILED"
					"backup_encryption_flag": "true/false"
					}
					• 409 CONFLICT backup is already going on:
					{
					"backup_id":"",
					"status":"BACKUP_INITIATION_FAILED"
					"backup_encryption
						Realtime

Table 4-2 (Cont.) cnDBTier Backup API

Resource	Rest URL	HTTP Method	Request Payload	Response Code	Response Payload	ServiceType

					_flag": "true/false"
					}
					• 500 INTERNAL SERVER ERROR:
					{
					"error_type":"<error type>",
					"error_message":"<error message>"
					}
					• 503 SERVICE UNAVAILABLE data node is down:
					{
					"backup_id":"",
					"status":"BACKUP_INITIATION_FAILED"
					"backup_encryption_flag": "true/false"
					}	

Table 4-2 (Cont.) cnDBTier Backup API

Resource	Rest URL	HTTP Method	Request Payload	Response Code	Response Payload	ServiceType

Site Specific On-demand Backup Status	http://<base-uri>/db-tier/on-demand/backup/<BACKUP_ID_INITIATED>/status	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 `.Values.global.remotetransfer.enable` is set to *true*:
					`{`
					`"backup_status":"<Backup status>",`
					`"transfer_status":"<Transfer Status>"`
					`}`
					Note: The `"transfer_status"` 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 `.Values.global.remotetransfer.enable` is set to *false*:
					`{`
					`"status":"<Backup status>"`
					`}`
						Realtime

Table 4-2 (Cont.) cnDBTier Backup API

Resource	Rest URL	HTTP Method	Request Payload	Response Code	Response Payload	ServiceType

					• 400 BAD REQUEST Request is invalid:
					{
					"error_description": "<error_description>",
					"error_type": "<error type>"
					}
					• 404 NOT FOUND backup_id does not exist:
					{
					"error_description": "<error_description>",
					"error_type": "<error type>"
					}
					• 500 INTERNAL SERVER ERROR:
					{
					"error_type":"<error type>",
					"error_message":"<error message>"
					}	

Table 4-2 (Cont.) cnDBTier Backup API

Resource	Rest URL	HTTP Method	Request Payload	Response Code	Response Payload	ServiceType

Health Check	http://<base-uri>/db-tier/health	GET	NA	• 200 OK
				• 500 INTERNAL SERVER ERROR
					• 200 OK:
					{
					"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>"
					},
					...
					]
					}
					• 500 INTERNAL SERVER ERROR:
					{
					"error_type":"<err
						Realtime

Table 4-2 (Cont.) cnDBTier Backup API

Resource	Rest URL	HTTP Method	Request Payload	Response Code	Response Payload	ServiceType
					or type>", "error_message":" error message>" }	

> **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 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
  ```

4.3 cnDBTier Replication Service APIs

This section provides details about cnDBTier replication service APIs.

4.3.1 cnDBTier Switchover APIs

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

Table 4-3 cnDBTier Switchover APIs

Resource	Rest URL	HTTP Method	Request Payload	Response Code	Response Payload	ServiceType

Start replication switchover for a site with respect to remote site and group ID	http://<*base-uri*>/ocdbtier/georeplication/switchover/start/sitename/{siteName}/remotesitename/{remoteSiteName}/replchannelgroupid/{replChannelGroupId}	PUT	NA	• 200 OK
				• 404 Not Found
				• 500 Internal Server Error
					• 200 OK:
					`{ "replicationSwitchOver":"start" }`
					• 404 Not Found:
					`{ "title": "Not Found", "status": 404, "detail": "<Exception details>", "cause": <cause>, "invalidParams": <params> }`
					• 500 Internal Server Error:
					`{ "title": "Internal Server Error", "status": 500, "detail": "<Exception details>", "cause": <cause>, "invalidParams": <params> }`
						Realtime

Table 4-3 (Cont.) cnDBTier Switchover APIs

Resource	Rest URL	HTTP Method	Request Payload	Response Code	Response Payload	ServiceType

Stop replication switchover for a site with respect to remote site and group ID	http://<*base-uri*>/ocdbtier/georeplication/switchover/stop/sitename/{siteName}/remotesitename/{remoteSiteName}/replchannelgroupid/{replChannelGroupId}	PUT	NA	• 200 OK
				• 404 Not Found
				• 500 Internal Server Error
					• 200 OK:
					`{ "replicationSwitchOver":"stop" }`
					• 404 Not Found:
					`{ "title": "Not Found", "status": 404, "detail": "<Exception details>", "cause": <cause>, "invalidParams": <params> }`
					• 500 Internal Server Error:
					`{ "title": "Internal Server Error", "status": 500, "detail": "<Exception details>", "cause": <cause>, "invalidParams": <params> }`
						Realtime

Table 4-3 (Cont.) cnDBTier Switchover APIs

Resource	Rest URL	HTTP Method	Request Payload	Response Code	Response Payload	ServiceType

Start replication switchover for a site with respect to remote site name	http://<*base-uri*>/ocdbtier/georeplication/switchover/start/sitename/{siteName}/remotesitename/{remoteSiteName}	PUT	NA	• 200 OK
				• 404 Not Found
				• 500 Internal Server Error
					• 200 OK:
					{
					"replicationSwitchOver":"start"
					}
					• 404 Not Found:
					{
					"title": "Not Found",
					"status": 404,
					"detail": "<Exception details>",
					"cause": <cause>,
					"invalidParams": <params>
					}
					• 500 Internal Server Error:
					{
					"title": "Internal Server Error",
					"status": 500,
					"detail": "<Exception details>",
					"cause": <cause>,
					"invalidParams": <params>
					}
						Realtime

Table 4-3 (Cont.) cnDBTier Switchover APIs

Resource	Rest URL	HTTP Method	Request Payload	Response Code	Response Payload	ServiceType

Stop replication switchover for a site with respect to remote site name	http://<*base-uri*>/ocdbtier/georeplication/switchover/stop/sitename/{siteName}/remotesitename/{remoteSiteName}	PUT	NA	• 200 OK
				• 404 Not Found
				• 500 Internal Server Error
					• 200 OK:
					{
					"replicationSwitchOver":"stop"
					}
					• 404 Not Found:
					{
					"title": "Not Found",
					"status": 404,
					"detail": "<Exception details>",
					"cause": <cause>,
					"invalidParams": <params>
					}
					• 500 Internal Server Error:
					{
					"title": "Internal Server Error",
					"status": 500,
					"detail": "<Exception details>",
					"cause": <cause>,
					"invalidParams": <params>
					}
						Realtime

Table 4-3 (Cont.) cnDBTier Switchover APIs

Resource	Rest URL	HTTP Method	Request Payload	Response Code	Response Payload	ServiceType

Start replication switchover with respect to site name	http://<*base-uri*>/ocdbtier/georeplication/switchover/start/sitename/{siteName}	PUT	NA	• 200 OK
				• 404 Not Found
				• 500 Internal Server Error
					• 200 OK:
					{
					"replicationSwitchOver":"start"
					}
					• 404 Not Found:
					{
					"title": "Not Found",
					"status": 404,
					"detail": "<Exception details>",
					"cause": <cause>,
					"invalidParams": <params>
					}
					• 500 Internal Server Error:
					{
					"title": "Internal Server Error",
					"status": 500,
					"detail": "<Exception details>",
					"cause": <cause>,
					"invalidParams": <params>
					}
						Realtime

Table 4-3 (Cont.) cnDBTier Switchover APIs

Resource	Rest URL	HTTP Method	Request Payload	Response Code	Response Payload	ServiceType

Stop replication switchover for a site with respect to site name	http://<*base-uri*>/ocdbtier/georeplication/switchover/stop/sitename/{siteName}	PUT	NA	• 200 OK
				• 404 Not Found
				• 500 Internal Server Error
					• 200 OK:
					`{ "replicationSwitchOver":"stop" }`
					• 404 Not Found:
					`{ "title": "Not Found", "status": 404, "detail": "<Exception details>", "cause": <cause>, "invalidParams": <params> }`
					• 500 Internal Server Error:
					`{ "title": "Internal Server Error", "status": 500, "detail": "<Exception details>", "cause": <cause>, "invalidParams": <params> }`
						Realtime

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:
  - Create the key PEM file (file.key.pem) and cert PEM file (file.crt.pem) using the p12 certificate (replicationcertificate.p12). Use the same p12 certificate that was used to enable the HTTPS while installing cnDBTier.
  - 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 file.crt.pem --cert-type PEM --key file.key.pem --
    key-type PEM --pass PUT NextGenCne https://$IP:$PORT/ocdbtier/
    georeplication/switchover/start/sitename/{siteName}
    ```

4.3.2 cnDBTier Stop Replica APIs

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

Table 4-4 cnDBTier Stop Replica APIs

Resource	Rest URL	HTTP Method	Request Payload	Response Code	Response Payload	ServiceType

Stop replication on a site with respect to remote site name and replication channel group ID	http://<*base-uri*>/ocdbtier/georeplication/stopreplica/sitename/{siteName}/remotesitename/{remoteSiteName}/replchannelgroupid/{replChannelGroupId}	PUT	NA	• 200 OK
				• 404 Not Found
				• 500 Internal Server Error
					• 200 OK:
					{
					"stopReplica":"stop"
					}
					• 404 Not Found:
					{
					"title": "Not Found",
					"status": 404,
					"detail": "<Exception details>",
					"cause": <cause>,
					"invalidParams": <params>
					}
					• 500 Internal Server Error:
					{
					"title": "Internal Server Error",
					"status": 500,
					"detail": "<Exception details>",
					"cause": <cause>,
					"invalidParams": <params>
					}
						Realtime

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

Resource	Rest URL	HTTP Method	Request Payload	Response Code	Response Payload	ServiceType

Stop replication on a site with respect to remote site names	http://<*base-uri*>/ocdbtier/georeplication/stopreplica/sitename/{siteName}/remotesitename/{remoteSiteName}	PUT	NA	• 200 OK
				• 404 Not Found
				• 500 Internal Server Error
					• 200 OK:
					{
					"stopReplica":"stop"
					}
					• 404 Not Found:
					{
					"title": "Not Found",
					"status": 404,
					"detail": "<Exception details>",
					"cause": <cause>,
					"invalidParams": <params>
					}
					• 500 Internal Server Error:
					{
					"title": "Internal Server Error",
					"status": 500,
					"detail": "<Exception details>",
					"cause": <cause>,
					"invalidParams": <params>
					}
						Realtime

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

Resource	Rest URL	HTTP Method	Request Payload	Response Code	Response Payload	ServiceType

Stop replica on a site with respect to site name	http://<*base-uri*>/ocdbtier/georeplication/stopreplica/sitename/{siteName}	PUT	NA	• 200 OK
				• 404 Not Found
				• 500 Internal Server Error
					• 200 OK:
					{
					"stopReplica":"stop"
					}
					• 404 Not Found:
					{
					"title": "Not Found",
					"status": 404,
					"detail": "<Exception details>",
					"cause": <cause>,
					"invalidParams": <params>
					}
					• 500 Internal Server Error:
					{
					"title": "Internal Server Error",
					"status": 500,
					"detail": "<Exception details>",
					"cause": <cause>,
					"invalidParams": <params>
					}
						Realtime

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/
    stopreplica/sitename/{siteName}
    ```

- When HTTPS is enabled in a cnDBTier cluster:
  - Create the key PEM file (file.key.pem) and cert PEM file (file.crt.pem) using the p12 certificate (replicationcertificate.p12). Use the same p12 certificate that was used to enable the HTTPS while installing cnDBTier.
  - 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 file.crt.pem --cert-type PEM --key file.key.pem --
    key-type PEM --pass NextGenCne PUT https://$IP:$PORT/ocdbtier/
    georeplication/stopreplica/sitename/{siteName}
    ```

4.3.3 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 4-5 cnDBTier Listbackups API

cnDBTier Listbackups API	Rest URL	HTTP Method	Request Payload	Response Code	Response Payload	ServiceType
Site Specific Listing the Backups	http://<*base-uri*>/db-tier/backup/site/{siteName}/listbackups	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: { "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>",	Realtime

Table 4-5 (Cont.) cnDBTier Listbackups API

cnDBTier Listbackups API	Rest URL	HTTP Method	Request Payload	Response Code	Response Payload	ServiceType
					"backupSize":"<size of backup>", "creationTimeStamp":"<timestamp at which backup was initiated>" }] } • 404 Not Found: The response payload can be one of the following: – Site Name is null so cannot list the backup – 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:
  - Create the key PEM file (file.key.pem) and cert PEM file (file.crt.pem) using the p12 certificate (replicationcertificate.p12). Use the same p12 certificate that was used to enable the HTTPS while installing cnDBTier.
  - 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 file.crt.pem --cert-type PEM --key file.key.pem --
    key-type PEM --pass NextGenCne https://$IP:$PORT/db-tier/backup/
    site/cluster1/listbackups
    ```

4.3.4 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 4-6 cnDBTier Replication Service Heartbeat API

cnDBTier Replication Service Heartbeat API	Rest URL	HTTP Method	Request Payload	Response Code	Response Payload	ServiceType
Site Specific Listing the Heartbeat Status	http://<*base-uri*>/db-tier/status/db-replication-svc/realtime	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: { "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":"<H	Realtime

Table 4-6 (Cont.) cnDBTier Replication Service Heartbeat API

cnDBTier Replication Service Heartbeat API	Rest URL	HTTP Method	Request Payload	Response Code	Response Payload	ServiceType
					eart Beat Lag in Seconds>", "replicationChannelGroupId":<replication channel group id> }] } • 503 SERVICE_UNAVAILABLE: Not able to query the Data Base • 500 INTERNAL_SERVER_ERROR: Could not check the replication service 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:
  - Create the key PEM file (file.key.pem) and cert PEM file (file.crt.pem) using the p12 certificate (replicationcertificate.p12). Use the same p12 certificate that was used to enable the HTTPS while installing cnDBTier.
  - 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 file.crt.pem --cert-type PEM --key file.key.pem --
    key-type PEM --pass NextGenCne https://$IP:$PORT/db-tier/
    status/db-replication-svc/realtime
    ```

4.4 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 4-7 cnDBTier Status API

cnDBTier Status API	Rest URL	HTTP Method	Request Payload	Response Code	Response Payload	ServiceType

Site Specific Real Time Replication Status	http://<*base-uri*>/db-tier/status/replication/{mate-site-name}/realtime	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:
					{
					"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
						Realtime

Table 4-7 (Cont.) cnDBTier Status API

cnDBTier Status API	Rest URL	HTTP Method	Request Payload	Response Code	Response Payload	ServiceType

					<pre>},
					{
					
					remote_replication_ip: 127.0.0.2,
					role: STANDBY
					},
					{
					
					remote_replication_ip: 127.0.0.2,
					role: ACTIVE
					},
					{
					
					remote_replication_ip: 127.0.0.3,
					role: STANDBY
					}]
					}</pre> • 404 Not Found: NA
					• 503 Service Unavailable: NA	

Table 4-7 (Cont.) cnDBTier Status API

cnDBTier Status API	Rest URL	HTTP Method	Request Payload	Response Code	Response Payload	ServiceType
Real Time Overall Replication Status	http://<*base-uri*>/db-tier/status/replication/realtime	GET	NA	• 200 OK - Cross-site replication is enabled. • 503 Service Unavailable - Cross-site replication is not enabled and/or DOWN.	• 200 OK: { [{ "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,	Realtime

Table 4-7 (Cont.) cnDBTier Status API

cnDBTier Status API	Rest URL	HTTP Method	Request Payload	Response Code	Response Payload	ServiceType

					secondsBehindRemote: <seconds>
					}]
					},
					{
					
					"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,	

Table 4-7 (Cont.) cnDBTier Status API

cnDBTier Status API	Rest URL	HTTP Method	Request Payload	Response Code	Response Payload	ServiceType
					secondsBehindRemote: <seconds> }] }] } • 503 Service Unavailable: NA	

Table 4-7 (Cont.) cnDBTier Status API

cnDBTier Status API	Rest URL	HTTP Method	Request Payload	Response Code	Response Payload	ServiceTypeCachedNote: This API is deprecated and will be removed in
Local Cluster Status	http://<*base-uri*>/db-tier/status/local	GET	NA	• 200 OK - The local NDB cluster is available. • 503 Service Unavailable - The local NDB cluster is not available.	NA	

Table 4-7 (Cont.) cnDBTier Status API

cnDBTier Status API	Rest URL	HTTP Method	Request Payload	Response Code	Response Payload	ServiceType
						nafutureresease

Table 4-7 (Cont.) cnDBTier Status API

cnDBTier Status API	Rest URL	HTTP Method	Request Payload	Response Code	Response Payload	ServiceType	Note
Site Specific Replication Status	http://<*base-uri*>/db-tier/status/replication/{mate-site-name}	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: `{ "replicationStatus": "UP/DOWN/INITIALIZING", "siteDetails":[ { "remote_replication_ip":"<ip>", "role":"ACTIVE/STANDBY/FAILED" }, { "remote_replication_ip":"<ip>", "role":"STANDBY/STANDBY/FAILED" } ] }` • 404 Not Found: NA • 503 Service Unavailable: NA	Cached	This API is deprecated and will be removed in

Table 4-7 (Cont.) cnDBTier Status API

cnDBTier Status API	Rest URL	HTTP Method	Request Payload	Response Code	Response Payload	ServiceType
						nafutureresease

Table 4-7 (Cont.) cnDBTier Status API

cnDBTier Status API	Rest URL	HTTP Method	Request Payload	Response Code	Response Payload	ServiceType	CachedNote: This API is deprecated and will be removed in
Overall Replication Status	http://<*base-uri*>/db-tier/status/replication	GET	NA	• 200 OK - Cross-site replication is enabled. • 503 Service Unavailable - Cross-site replication is not enabled and/or DOWN.	• 200 OK: `{ [ { "localSiteName":"<current-site-name>", "remoteSiteName":"<remote-site-name>", "replicationStatus":"UP/DOWN/INITIALIZING" }, { "localSiteName":"<current-site-name>", "remoteSiteName":"<remote-site-name>", "replicationStatus":"UP/DOWN/INITIALIZING" } ] }` • 503 Service Unavailable: NA		

Table 4-7 (Cont.) cnDBTier Status API

cnDBTier Status API	Rest URL	HTTP Method	Request Payload	Response Code	Response Payload	ServiceType
						nafutureresease.
Real Time Local Cluster Status	http://<*base-uri*>/db-tier/status/cluster/local/realtime	GET	NA	• 200 OK - The local NDB cluster is available. • 503 Service Unavailable - The local NDB cluster is not available.	NA	Realtime

Table 4-7 (Cont.) cnDBTier Status API

cnDBTier Status API	Rest URL	HTTP Method	Request Payload	Response Code	Response Payload	ServiceType

Database Statistics Reports API	http://<*base-uri*>/db-tier/statistics/dbinfo	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:
					{
					"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>
					}
						Realtime

Table 4-7 (Cont.) cnDBTier Status API

cnDBTier Status API	Rest URL	HTTP Method	Request Payload	Response Code	Response Payload	ServiceType

					]
					},
					{
					"dbName": "<database name>",
					"tables": [
					{
					"tableName": "<table name>",
					"rowCount": <row count>
					},
					{
					"tableName": "<table name>",
					"rowCount": <row count>
					}
					]
					}
					]
					}
					• 503 Service Unavailable: NA
					• 500 INTERNAL SERVER ERROR: NA	

Table 4-7 (Cont.) cnDBTier Status API

cnDBTier Status API	Rest URL	HTTP Method	Request Payload	Response Code	Response Payload	ServiceType
cnDBTier Version	http://<*base-uri*>/db-tier/version	GET	NA	• 200 OK - cnDBTier version retrieved successfully. • 500 INTERNAL SERVER ERROR - failed to get the cnDBTier version.	• 200 OK: `{ "cndbtier_version" : "23.2.0", "ndb_version": "ndb-8.4.2" }` • 500 INTERNAL SERVER ERROR: `"failed to get version"`	Realtime

> **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 kernal 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 threat. Sample Values: ldm, 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: • ndbmtd-0.ndbmtdsvc.occne-cndbtier.svc.cluster.local • ndbmysqld-1.ndbmysqldsvc.occne-cndbtier.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.
master_node_ip	The IP address of the active replication SQL service in the remote site.
slave_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 JVM Heap Metrics

This section provides details about cnDBTier Java Virtual Machine (JVM) heap metrics.

Table 5-4 jvm_max_memory

Field	Details
Description	The total memory designated for a particular service
Type	Gauge

Dimensions	`site_name`
	`namespace`
	`service_name`

Table 5-5 jvm_free_memory

Field	Description
Description	The current free memory allocated for a particular service. This value doesn't indicate the total free available memory.
Type	Gauge

Dimensions	`site_name`
	`namespace`
	`service_name`

Table 5-6 jvm_total_memory

Field	Description
Description	The total memory allocated for a particular service
Type	Gauge

Dimensions	`site_name`
	`namespace`
	`service_name`

5.3 cnDBTier Remote Server Backup Transfer Status Metrics

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

Table 5-7 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`

5.4 cnDBTier Backup Transfer Status Metrics

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

Table 5-8 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

Table 5-8 (Cont.) db_tier_backup_transfer_status

Field	Details
Dimensions	site_name namespace backup_id status

5.5 cnDBTier PVC Health Monitoring Metrics

This section provides details about the cnDBTier PVC health monitoring metrics.

Table 5-9 db_tier_pvc_read_write_speed

Field	Details
Description	The read and write speed of the PVC.
Type	Gauge
Dimensions	site_name namespace mount_path hostname

Table 5-10 db_tier_pvc_is_accesible

Field	Details
Description	Indicates if the PVC is accessible at that point in time: • 0: indicates that the PVC is not accessible. • 1: indicates that the PVC is accessible.
Type	Gauge
Dimensions	site_name namespace mount_path hostname

Table 5-11 db_tier_pvc_failure_count

Field	Details
Description	The number of times the PVC was not accessible during the last scrape.
Type	Gauge
Dimensions	site_name namespace mount_path hostname

5.6 cnDBTier Heartbeat Metrics

This section provides details about the cnDBTier heartbeat metrics.

Table 5-12 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`

5.7 cnDBTier Node Status Metrics

The section provides details about the cnDBTier node status metrics.

Table 5-13 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 5-14 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 5-14 (Cont.) db_tier_cluster_status

Field	Details
Dimensions	site_name namespace

Table 5-15 db_tier_cluster_disconnect

Field	Details
Description	The disconnect status of the cnDBTier cluster. If the value is greater than "0", then the cluster is disconnected.
Type	Gauge
Dimensions	site_name namespace

5.8 cnDBTier Table Read Write Metrics

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

Table 5-16 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 5-17 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 5-18 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 5-19 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 5-20 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 5-21 db_tier_writes

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

Table 5-21 (Cont.) db_tier_writes

Field	Details
Dimensions	node_id block_name block_instance counter_name site_name namespace

Table 5-22 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 5-23 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 5-24 db_tier_table_scans

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

Table 5-24 (Cont.) db_tier_table_scans

Field	Details
Dimensions	node_id block_name block_instance counter_name site_name namespace

Table 5-25 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 5-26 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 5-27 db_tier_scan_slowdown

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

Table 5-27 (Cont.) db_tier_scan_slowdown

Field	Details
Dimensions	node_id block_name block_instance counter_name site_name namespace

5.9 cnDBTier CPU Usage Metrics

This section provides details about the cnDBTier CPU usage metrics.

Table 5-28 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 5-29 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 5-30 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`

5.10 cnDBTier Memory Usage Metrics

This section provides details about the cnDBTier memory usage metrics.

Table 5-31 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 5-32 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`

5.11 cnDBTier Bin Log Usage Metrics

This section provides details about the cnDBTier binlog usage metrics.

Table 5-33 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`

5.12 cnDBTier Replication Metrics

This section provides details about the cnDBTier replication metrics.

Table 5-34 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 5-35 db_tier_replication_slave_delay

Field	Details
Description	Indicates the time (in seconds) by which the last record read by the slave is behind the latest record written by the master.
Type	Gauge

Dimensions	`channel_id`
	`master_node_ip`
	`slave_node_ip`
	`mate_site_name`
	`site_name`
	`namespace`

5.13 cnDBTier Automated Backup Metrics

This section provides details about the cnDBTier automated backup metrics.

Table 5-36 db_tier_backup_used_disk_percentage

Field	Details
Description	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 5-37 db_tier_backup_size

Field	Details
Description	This metric is pegged when a backup completes successfully.
Type	Gauge

Dimensions	`backup_id`
	`node_id`
	`site_name`
	`namespace`

Table 5-38 db_tier_backup

Field	Details
Description	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`

5.14 cnDBTier Fault Recovery State Metrics

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

Table 5-39 db_tier_georeplication_recovery_state

Field	Details
Description	Indicates if the current site is undergoing georeplication recovery.
Type	Gauge
Dimensions	`site_name` (Name of the site that is undergoing georeplication recovery) `namespace`

5.15 cnDBTier Replica Status Metrics

This section provides details about the cnDBTier replica status metrics.

Table 5-40 db_tier_api_trans_commit_count

Field	Details
Description	The number of transactions committed by this replica.
Type	Gauge

Table 5-40 (Cont.) db_tier_api_trans_commit_count

Field	Details

Dimensions	`node_id` (of the DB node)
	`site_name`
	`namespace`

Table 5-41 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 5-42 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 5-43 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`

5.16 cnDBTier ndbinfo Transporters Metrics

This section provides details about the cnDBTier ndbinfo transporters metrics.

Table 5-44 db_tier_node_transporter_bytes_sent

Field	Details
Description	The bytes sent from data node to other nodes.

Table 5-44 (Cont.) db_tier_node_transporter_bytes_sent

Field	Details
Type	Gauge

Dimensions	`node_id`
	`node_type`
	`remote_node_id`
	`site_name`
	`namespace`

Table 5-45 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 5-46 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 5-47 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`

5.17 cnDBTier ndbinfo Threadstat Metrics

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

Table 5-48 db_tier_threadstat_os_time

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

Dimensions	`node_id`,
	`thr_no`
	`thr_nm`
	`site_name`
	`namespace`

Table 5-49 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 5-50 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 5-51 db_tier_threadstat_os_voluntary_context_switches

Field	Details
Description	The number of OS voluntary context switches that happened.
Type	Gauge

Table 5-51 (Cont.) db_tier_threadstat_os_voluntary_context_switches

Field	Details
Dimensions	node_id, thr_no thr_nm site_name namespace

Table 5-52 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

5.18 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 5-53 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 5-54 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 5-55 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 5-56 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 5-57 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 5-58 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 5-59 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 5-60 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 5-61 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 5-62 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 5-63 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 5-64 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`

5.19 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 5-65 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 5-66 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 5-67 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 5-68 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 5-69 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 5-70 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 5-71 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 5-72 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 5-73 db_tier_locks_per_fragment_wait_ok_millis

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

Dimensions	`table_id`
	`fq_name`
	`parent_fq_name`
	`site_name`
	`namespace`

Table 5-74 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`

5.20 cnDBTier ndbinfo Disk Write Speed Aggregate Metrics

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

Table 5-75 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 5-76 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 5-76 (Cont.) db_tier_disk_write_speed_aggregate_redo_speed_last_60sec

Field	Details

Dimensions	`node_id`
	`thr_no`
	`site_name`
	`namespace`

Table 5-77 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 5-78 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 5-79 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 5-80 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`

5.21 cnDBTier Replication Error Skip Info Metrics

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

Table 5-81 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 5-82 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 5-83 db_tier_replication_switchover_due_to_clusterdisconnect

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

Table 5-83 (Cont.) db_tier_replication_switchover_due_to_clusterdisconnect

Field	Details
Dimensions	node_id site_name mate_site_name replchannel_group_id error_number namespace

5.22 cnDBTier BinLog Injector Thread Info Metrics

This section provides details about the cnDBTier replication metrics.

Table 5-84 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 5-85 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

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
Recommended Action	**Cause**: The transfer of backup to remote server failed. **Diagnostic Information**: Check the status of the `db_tier_remote_server_backup_transfer_status` metric (db_tier_remote_server_backup_transfer_status). **Recovery**: This alert is cleared automatically when the backup transfer status is updated from the failed state to other states, that is, the `db_tier_remote_server_backup_transfer_status` metric value is updated to any value other than 1. To recover from this issue: • check if the remote server is in a healthy state • check the network • check if enough space is available For any assistance, collect the logs and contact My Oracle Support.

6.2 cnDBTier PVC Health Monitoring Alerts

This section provides details about the cnDBTier PVC health monitoring alerts.

Table 6-2 PVC_NOT_ACCESSIBLE

Field	Details
Description	This alert is triggered with a critical severity when PVC is not accessible (`db_tier_pvc_is_accesible` metric is 0).
Summary	PVC is not accessible on cnDBTier site {{ $labels.site_name }}
Severity	critical
Condition	db_tier_pvc_is_accesible == 0
Expression Validity	1m
SNMP Trap ID	2029
Affects Service (Y/N)	Y
Recommended Action	**Cause**: The system is unable to access PVC for read or write operation. **Diagnostic Information**: The `db_tier_pvc_is_accesible` metric (db_tier_pvc_is_accesible) provides information if the PVC is accessible or not. **Recovery**: This alert is cleared automatically when the PVC is accessible. Check the PVC and ensure that the PVC is accessible and doesn't have any errors. For any assistance, contact My Oracle Support.

Table 6-3 PVC_FAILURE_COUNT

Field	Details
Description	This alert is triggered with info severity when the value of `db_tier_pvc_failure_count` was greater than 0 during the last scrape. The value of `db_tier_pvc_failure_count` metric indicates the number of times the PVC was not accessible.
Summary	PVC of {{ $labels.hostname }} was unavailable for {{ $value }} times in the last 10m on cnDBTier site {{ $labels.site_name }}
Severity	info
Condition	sum(sum_over_time(db_tier_pvc_failure_count[10m])) by (hostname, site_name, namespace, mount_path) > 0
Expression Validity	5m
SNMP Trap ID	2030
Affects Service (Y/N)	N

Table 6-3 (Cont.) PVC_FAILURE_COUNT

Field	Details
Recommended Action	**Cause**: The system is unable to access the PVC for read or write operation. **Diagnostic Information**: The `db_tier_pvc_failure_count` metric (db_tier_pvc_failure_count) provides information about the number of times the PVC was not accessible during the last scrape. **Recovery**: This alert is cleared automatically when the `db_tier_pvc_failure_count` metric is *zero*. Check the PVC and ensure that the PVC is accessible and doesn't have any errors. For any assistance, contact My Oracle Support.

6.3 cnDBTier Backup Transfer Status Alerts

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

Table 6-4 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).
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
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 `db_tier_backup_transfer_status` metric (db_tier_backup_transfer_status) provides information about the backup transfer status. **Recovery**: This alert is cleared automatically when the `db_tier_backup_transfer_status` metric is updated to a value other than 2. For any assistance, collect the logs and contact My Oracle Support.

Table 6-5 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
Recommended Action	**Cause**: The system failed to transfer a backup from the cnDBTier site to a remote site. **Diagnostic Information**: The `db_tier_backup_transfer_status` metric (db_tier_backup_transfer_status) provides information about the backup transfer status. **Recovery**: This alert is cleared automatically when the `db_tier_backup_transfer_status` metric is updated to a value other than 3. For any assistance, collect the logs and contact My Oracle Support.

Table 6-6 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
Recommended Action	**Cause**: Backup transfer is in progress on the cnDBTier site. **Diagnostic Information**: The `db_tier_backup_transfer_status` metric (db_tier_backup_transfer_status) provides information about the backup transfer status. **Recovery**: This alert is cleared automatically when the `db_tier_backup_transfer_status` metric is updated to a value other than 1. For any assistance, collect the logs and contact My Oracle Support.

6.4 cnDBTier Heartbeat Alerts

This section provides details about cnDBTier heartbeat alerts.

Table 6-7 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 {{ $labels.site_name }} connected to mate site {{ $labels.mate_site_name }} on replication channel group id {{ $labels.replchannel_group_id }} and kubernetes namespace {{ $labels.namespace }}"
Severity	critical
Condition	db_tier_heartbeat_failure == 1
Expression Validity	NA
SNMP Trap ID	2025
Affects Service (Y/N)	Y
Recommended Action	**Cause**: The system is unable to connect to remote site and Heartbeat failed. **Diagnostic Information**: The `db_tier_heartbeat_failure` metric (db_tier_heartbeat_failure) provides information about the heartbeat status and indicates whether the remote site is reachable or not. **Recovery**: This alert is cleared automatically when the `db_tier_heartbeat_failure` metric is *0*. For any assistance, collect the logs and contact My Oracle Support.

6.5 cnDBTier BinLog Injector Thread Alerts

This section provides details about cnDBTier BinLog injector alerts.

Table 6-8 BINLOG_INJECTOR_STOPPED

Field	Details
Description	This alert is triggered with critical severity when Bin Log Injector stops working. The value of `db_tier_binlog_injector_thread` or `db_tier_binlog_injector_thread_latest_epoch` 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 }}

Table 6-8 (Cont.) BINLOG_INJECTOR_STOPPED

Field	Details
Severity	critical
Condition	db_tier_binlog_injector_thread_latest_epoch == 1 or db_tier_binlog_injector_thread == 1
Expression Validity	NA
SNMP Trap ID	2024
Affects Service (Y/N)	Y
Recommended Action	**Cause**: Bin Log Injector thread stalled for the replication SQL node. **Diagnostic Information**: The `db_tier_binlog_injector_thread_latest_epoch` or `db_tier_binlog_injector_thread` 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. **Recovery**: This alert is cleared automatically when the `db_tier_binlog_injector_thread_latest_epoch` or `db_tier_binlog_injector_thread` metric is *0*. For any assistance, collect the logs and contact My Oracle Support.

6.6 cnDBTier Replication Error Skip Alerts

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

Table 6-9 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 6-9 (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 `db_tier_replication_switchover_due_to_clusterdisconnect` metric (Table 5-83) provides information whether a switchover occurred on an API node. **Recovery**: This alert is cleared automatically one hour after the event. For any assistance, collect the logs and contact My Oracle Support.

Table 6-10 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
Recommended Action	**Cause**: Between 10000 and 80000 epochs are lost due to skip errors. **Diagnostic Information**: The `db_tier_epochs_lost_due_to_skiperror` metric (Table 5-82) provides information about the number of epochs lost due to skip errors. **Recovery**: This alert is cleared automatically one hour after the event. For any assistance, collect the logs and contact My Oracle Support.

Table 6-11 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

Table 6-11 (Cont.) REPLICATION_SKIP_ERRORS_LOW

Field	Details
SNMP Trap ID	2021
Affects Service (Y/N)	N
Recommended Action	**Cause**: Replication halted due to less than five skip errors. **Diagnostic Information**: The `db_tier_replication_halted_due_to_skiperror` metric (Table 5-81) provides information about the number of skip errors due to which the replication halted. **Recovery**: This alert is cleared automatically one hour after the event. For any assistance, collect the logs and contact My Oracle Support.

Table 6-12 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.
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
Recommended Action	**Cause**: Replication halted due to more than five skip errors. **Diagnostic Information**: The `db_tier_replication_halted_due_to_skiperror` metric (Table 5-81) provides information about the number of skip errors due to which the replication halted. **Recovery**: This alert is cleared automatically one hour after the event. For any assistance, collect the logs and contact My Oracle Support.

Table 6-13 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.
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

Table 6-13 (Cont.) REPLICATION_EPOCHS_LOST

Field	Details
SNMP Trap ID	2023
Affects Service (Y/N)	N
Recommended Action	**Cause**: Less than 2000 epochs are lost due to skip errors. **Diagnostic Information**: The `db_tier_epochs_lost_due_to_skiperror` metric (Table 5-82) provides information about the number of epochs lost due to skip errors. **Recovery**: This alert is cleared automatically one hour after the event. For any assistance, collect the logs and contact My Oracle Support.

6.7 cnDBTier Georeplication Recovery Status Alerts

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

Table 6-14 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
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 `db_tier_georeplication_recovery_state` metric (db_tier_georeplication_recovery_state) provides information whether georeplication recovery is in progress. **Recovery**: This alert is cleared automatically when the georeplication recovery is complete. For any assistance, collect the logs and contact My Oracle Support.

6.8 cnDBTier Cluster Status Alerts

This section provides details about cnDBTier cluster status alerts.

Table 6-15 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
Recommended Action	**Cause**: • When pod restarts due to Kubernetes liveliness 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: `kubectl -n <namespace> exec -it ndbmgmd-0 -- ndb_mgm -e show` 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. **Recovery**: This alert is cleared automatically when the inactive pod is active. For any assistance, collect the logs and contact My Oracle Support.

Table 6-16 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 disconnect for cnDBTier site {{ $labels.site_name }} and kubernetes namespace {{ $labels.namespace }}
Severity	critical
Condition	sum_over_time(db_tier_cluster_disconnect[5m]) > 0
Expression Validity	1m
SNMP Trap ID	2034
Affects Service (Y/N)	Y

Table 6-16 (Cont.) MYSQL_NDB_CLUSTER_DISCONNECT

Field	Details
Recommended Action	**Cause**: • When pod restarts due to Kubernetes liveliness 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: `kubectl -n <namespace> exec -it ndbmgmd-0 -- ndb_mgm -e show` 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. **Recovery**: This alert is cleared automatically when the inactive pod is active. For any assistance, collect the logs and contact My Oracle Support.

6.9 cnDBTier Automated Backup Alerts

This section provides details about the cnDBTier automated backup alerts.

Table 6-17 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
Affects Service (Y/N)	N

Table 6-17 (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 `db_tier_backup` metric (db_tier_backup) provides information if the backup failed or not. **Recovery**: This alert is cleared automatically when the `db_tier_backup` metric status changes from the FAILED state to other states. For any assistance, collect the logs and contact My Oracle Support.

Table 6-18 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
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 `db_tier_backup` metric (db_tier_backup) provides information if the backup is purged earlier than expected. **Recovery**: This alert is cleared automatically when the `db_tier_backup` metric status changes from the PURGED_EARLY state to other states. For any assistance, collect the logs and contact My Oracle Support.

Table 6-19 BACKUP_SIZE_GROWTH

Field	Details
Description	This alert is triggered with minor severity whenever the current backup size exceeds 5% of the average of the previous backups.

Table 6-19 (Cont.) BACKUP_SIZE_GROWTH

Field	Details
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
Recommended Action	**Cause**: When the current backup size exceeds 5% of the average of the previous backups. **Diagnostic Information**: The `db_tier_backup_used_disk_percentage` metric (db_tier_backup_used_disk_percentage) provides information if the current backup size exceeds 5% of the average. **Recovery**: This alert is cleared automatically when the `db_tier_backup_used_disk_percentage` metric value is reduced to the threshold percentage. For any assistance, collect the logs and contact My Oracle Support.

Table 6-20 BACKUP_STORAGE_LOW

Field	Details
Description	This alert is triggered with minor severity when the total backup size of the data node is >= 70% and < 80% 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	minor
Condition	(avg_over_time(db_tier_backup_used_disk_percentage[5m])>=70) and (avg_over_time(db_tier_backup_used_disk_percentage[5m])<80)
Expression Validity	N/A
SNMP Trap ID	2014
Affects Service (Y/N)	N

Table 6-20 (Cont.) BACKUP_STORAGE_LOW

Field	Details
Recommended Action	**Cause**: When the total backup size of the data node is >= 70% and < 80% of the total data node disk size. **Diagnostic Information**: The `db_tier_backup_used_disk_percentage` metric (db_tier_backup_used_disk_percentage) provides information if the current backup size is >= 70% and < 80% of the total data node disk size. **Recovery**: • This alert is cleared automatically when the `db_tier_backup_used_disk_percentage` metric value is reduced to the threshold percentage. • Increase the disk size by performing the scaling procedure. For any assistance, collect the logs and contact My Oracle Support.

Table 6-21 BACKUP_STORAGE_LOW

Field	Details
Description	This alert is triggered with major severity when the total backup size of the data node is >= 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
Recommended Action	**Cause**: When the total backup size of the data node is >= 80% and < 95% of the total data node disk size. **Diagnostic Information**: The `db_tier_backup_used_disk_percentage` metric (db_tier_backup_used_disk_percentage) provides information if the current backup size is >= 80% and < 95% of the total data node disk size. **Recovery**: • This alert is cleared automatically when the `db_tier_backup_used_disk_percentage` metric value is reduced to the threshold percentage. • Increase the disk size by performing the scaling procedure. For any assistance, collect the logs and contact My Oracle Support.

Table 6-22 BACKUP_STORAGE_FULL

Field	Details
Description	This alert is triggered with critical severity when the total backup size of the data node is >= 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
Recommended Action	**Cause**: When the total backup size of the data node is >= 95% of the total data node disk size. **Diagnostic Information**: The `db_tier_backup_used_disk_percentage` metric (db_tier_backup_used_disk_percentage) provides information if the current backup size is >= 95% of the total data node disk size. **Recovery**: • This alert is cleared automatically when the `db_tier_backup_used_disk_percentage` metric value is reduced to the threshold percentage. • Increase the disk size by performing the scaling procedure. **Note**: Take immediate action to avoid the cnDBTier cluster going out of service. For any assistance, collect the logs and contact My Oracle Support.

6.10 cnDBTier Bin Log Usage Alerts

This section provides details about the cnDBTier binlog usage alerts.

Table 6-23 BINLOG_STORAGE_LOW

Field	Details
Description	This alert is triggered with a minor severity when the total BinLog size of the SQL node is >= 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

Table 6-23 (Cont.) BINLOG_STORAGE_LOW

Field	Details
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
Recommended Action	**Cause**: When the total BinLog size of the SQL node is >= 70% and < 80% of the total SQL node disk size. **Diagnostic Information**: The `db_tier_binlog_used_bytes_percentage` metric (db_tier_binlog_used_bytes_percentage) provides information if the total BinLog size of the SQL node is >=70% and <80% of total SQL node disk size. **Recovery**: This alert is cleared automatically when the value of the `db_tier_binlog_used_bytes_percentage` metric is reduced to the threshold value. For any assistance, contact My Oracle Support.

Table 6-24 BINLOG_STORAGE_LOW

Field	Details
Description	This alert is triggered with major severity when the total BinLog size of the SQL node is >=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
Recommended Action	**Cause**: When the total BinLog size of the SQL node is >=80% and <95% of the total SQL node disk size. **Diagnostic Information**: The `db_tier_binlog_used_bytes_percentage` metric (db_tier_binlog_used_bytes_percentage) provides information if the total BinLog size of the SQL node is >=80% and <95% of total SQL node disk size. **Recovery**: This alert is cleared automatically when the value of the `db_tier_binlog_used_bytes_percentage` metric is reduced to the threshold value. For any assistance, contact My Oracle Support.

Table 6-25 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
Recommended Action	**Cause**: When the total BinLog size of the SQL node is >= 95% of the total SQL node disk size. **Diagnostic Information**: The `db_tier_binlog_used_bytes_percentage` metric (db_tier_binlog_used_bytes_percentage) provides information if the total BinLog size of the SQL node is >= 95% of total SQL node disk size. **Recovery**: This alert is cleared automatically when the value of the `db_tier_binlog_used_bytes_percentage` metric is reduced to the threshold value. **Note**: Take immediate action to avoid the SQL node going into Crashbackloop and becoming inaccessible. For any assistance, contact My Oracle Support.

6.11 cnDBTier Replication Alerts

This section provides details about cnDBTier replication alerts.

Table 6-26 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 6-26 (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: • `db_tier_replication_status{role="failed"} == 0` (db_tier_replication_status{role="failed"} == 0) • `db_tier_replication_status{role="active"} == 0` (db_tier_replication_status{role="active"} == 0) **Recovery**: This alert is cleared automatically when the cross-site replication is UP on the node and ACTIVE. **Note**: Take immediate action to avoid the cnDBTier cluster going out of service. For any assistance, contact My Oracle Support.

Table 6-27 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
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 `db_tier_replication_status` metric (db_tier_replication_status) provides information if the replication failed. **Recovery**: This alert is cleared automatically when the cross-site replication of the cnDBTier site is ACTIVE. For any assistance, contact My Oracle Support.

Table 6-28 SLAVE_REPLICATION_DELAY_HIGH

Field	Details
Description	This alert is triggered when the last record read by the slave is more than five minutes behind the latest record written by the master.
Summary	Slave replication on SQL node at {{ $labels.slave_node_ip }} is {{ $value }} seconds behind the master for cnDBTier site {{ $labels.site_name }} and kubernetes namespace {{ $labels.namespace }}
Severity	major
Condition	avg(avg_over_time(db_tier_replication_slave_delay[5m])) by (master_node_ip,slave_node_ip) >= 300 and avg(avg_over_time(db_tier_replication_slave_delay[5m])) by (master_node_ip,slave_node_ip) < 48*3600
Expression Validity	1m
SNMP Trap ID	2009
Affects Service (Y/N)	N
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 `db_tier_replication_slave_delay` metric (db_tier_replication_slave_delay) provides information if there is a delay in the worker node replication. **Recovery**: This alert is cleared automatically when the when the `db_tier_replication_slave_delay` metric value is reduced below the defined threshold value. For any assistance, collect the logs and contact My Oracle Support.

Table 6-29 SLAVE_REPLICATION_FAILED

Field	Details
Description	This alert is triggered when the last record read by the slave is more than 48 hours behind the latest record written by the master.
Summary	Slave replication has fallen more than 48 hours behind the master. 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_slave_delay[5m])) by (master_node_ip,slave_node_ip) >= 48*3600
Expression Validity	1m
SNMP Trap ID	2010
Affects Service (Y/N)	Y

Table 6-29 (Cont.) SLAVE_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 `db_tier_replication_slave_delay` metric (db_tier_replication_slave_delay) provides information if the worker node replication failed. **Recovery**: Perform georeplication recovery for the DB sync. For procedures, see *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide*. This alert is cleared automatically when the `db_tier_replication_slave_delay` metric value is reduced below the defined threshold value. For any assistance, collect the logs and contact My Oracle Support.

Table 6-30 REPLICATION_SVC_STORAGE_FULL

Field	Details
Description	This alert is triggered with critical severity whenever the PVC consumption of replication service is more than 90% of the overall storage of replication service.
Summary	Disk storage of replication service PVC {{ $labels.persistentvolumeclaim }} is full on kubernetes namespace {{ $labels.namespace }}
Severity	critical
Condition	(kubelet_volume_stats_used_bytes{persistentvolumeclaim=~".*replication.*"}/ kubelet_volume_stats_capacity_bytes{persistentvolumeclaim=~".*replication.*"}) * 100 > 90
Expression Validity	NA
SNMP Trap ID	2032
Affects Service (Y/N)	Y
Recommended Action	**Cause**: When the replication service PVC is filled more than 90% of overall PVC storage. A PVC fill can lead to replication service not functioning properly. **Diagnostic Information**: This alert indicates that PVC of replication service is almost full and requires immediate attention to address the storage issue. **Recovery**: Release storage for the replication service to function properly or scale PVC to accommodate any future data as the PVC is almost full.

Table 6-31 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
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. **Recovery**: Check the configurations like SSH key or disk size. Contact My Oracle Support for additional support.

6.12 cnDBTier Memory Usage Alerts

This section provides details about the cnDBTier memory usage alerts.

Table 6-32 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
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 6-32 (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) **Recovery**: Reduce the incoming service request rate. This alert is cleared automatically when the memory usage of the cnDBTier worker pod is reduced below the defined threshold value. For any assistance, contact My Oracle Support.

Table 6-33 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
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) **Recovery**: Reduce the incoming service request rate. This alert is cleared automatically when the memory usage of the cnDBTier worker pod is reduced below the defined threshold value. **Note**: Take immediate action to avoid the cnDBTier cluster going out of service. For any assistance, contact My Oracle Support.

6.13 cnDBTier CPU Usage Alerts

This section provides details about cnDBTier CPU usage alerts.

Table 6-34 HIGH_CPU

Field	Details
Description	This alert is triggered with major severity when the CPU usage of any 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
Recommended Action	**Cause**: When the CPU utilization of any 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. **Recovery**: Reduce the incoming service request rate. This alert is cleared automatically when the CPU utilization is reduced below the threshold value. For any assistance, contact My Oracle Support.

Table 6-35 HIGH_CPU

Field	Details
Description	This alert is triggered with critical severity when the CPU usage of any 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
Condition	(100 - (avg(avg_over_time(db_tier_cpu_os_idle[10m]))BY (node_id)))>= 90
Expression Validity	1m
SNMP Trap ID	2035
Affects Service (Y/N)	N

Table 6-35 (Cont.) HIGH_CPU

Field	Details
Recommended Action	**Cause**: When the CPU utilization of any node is greater than or equal to 90%. **Diagnostic Information**: Check the CPU threshold level status from the cnDBTier worker pod logs. **Recovery**: Reduce the incoming service request rate. This alert is cleared automatically when the CPU utilization is reduced below the threshold value. **Note**: Take immediate action to avoid the cnDBTier cluster going out of service. For any assistance, contact My Oracle Support.

6.14 cnDBTier Node Status Alerts

The section provides details about cnDBTier node status alerts.

Table 6-36 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 6-36 (Cont.) NODE_DOWN

Field	Details
Recommended Action	**Cause**: • When pod restarts due to Kubernetes liveliness 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: `kubectl -n <namespace> exec -it ndbmgmd-0 -- ndb_mgm -e show` • Check the Kubernetes events for probe failures in the platform logs. • Check if any exception is reported in the cnDBTier application logs. **Recovery**: This alert is cleared automatically when the inactive pod becomes active. For any assistance, collect the application logs and contact My Oracle Support.

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.

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 ndbmgmd --
   replicas=<replica count for MGM pods>
   ```

   Example:

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

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

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

   Example:

   ```
   $ kubectl -n occne-cndbtier scale sts ndbmtd --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.

> **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 ndbappmysqld --replicas=0
   ```

   For example:

   ```
   $ kubectl -n cluster1 scale sts ndbappmysqld --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 ndbmtd --replicas=0
   ```

   For example:

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

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

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

   For example:

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

7.2 Starting or Stopping cnDBTier Georeplication Service

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

7.2.1 Starting cnDBTier Georeplication Between Sites

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

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"
}
```

5. Run the following command to start the replication service 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/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 API response payloads, error codes, and curl commands for starting georeplication service, see cnDBTier Switchover APIs.

7.2.2 Stopping cnDBTier Georeplication Between Sites

This section provides the procedure to stop cnDBTier georeplication service using cnDBTier switchover and stop replica APIs.

To stop cnDBTier georeplication service, perform the following:

1. Perform the following steps to stop the cnDBTier replication service switchover:
   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 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}
      ```

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

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

Sample output:

```
{
    "replicationSwitchOver":"stop"
}
```

d. Run the following command to stop the replication service switchover 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}
```

For example, run the following command to stop the replication service switchover 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"
}
```

e. 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}
```

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"
}
```

> **Note**
>
> For more information about API response payloads, error codes, and curl commands to stop replication service switchover, see cnDBTier Switchover APIs.

2. Perform the following steps to 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 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/stopreplica/
sitename/{siteName}
```

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

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

Sample output:

```
{
    "stopReplica":"stop"
}
```

d. Run the following command to stop 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/stopreplica/
sitename/{siteName}/remotesitename/{remoteSiteName}
```

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

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

Sample output:

```
{
    "stopReplica":"stop"
}
```

e. Run the following command to stop the replication service 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/stopreplica/
sitename/{siteName}/remotesitename/{remoteSiteName}/replchannelgroupid/
{replChannelGroupId}
```

For example, run the following command to stop the replication service in cnDBTier with respect to cluster1, cluster2, and replication channel group ID "1":

```
$ curl -X PUT http://$IP:$PORT/ocdbtier/georeplication/stopreplica/
sitename/cluster1/remotesitename/cluster2/replchannelgroupid/1
```

Sample output:

```
{
    "stopReplica":"stop"
}
```

> **Note**
>
> For more information about API response payloads, error codes, and curl commands to stop replication channel, see cnDBTier Stop Replica APIs.

7.3 cnDBTier Scaling

cnDBTier 24.3.1 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 **ndbappmysqld** and **ndbmtd** pods.

7.3.1.1 Scaling ndbappmysqld Pods

cnDBTier 24.3.1 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 ndbmgmd, 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 ndbappmysqld replica count is increased to the value updated in the `custom_values.yaml` file.

7.3.1.1.2 Scaling ndbappmysqld Pods With Service Account

This section describes the procedure to scale the cnDBTier ndbappmysqld pods when cnDBTier is deployed with service account.
Scaling cnDBTier ndbappmysqld When Autoscaling is Enabled
When cnDBTier is deployed with a service account and autoscaling is enabled, the ndbappmysqld pods are scaled automatically.

Ensure that autoscaling is enabled for cnDBTier ndbappmysqld 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, ndbappmysqld 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 ndbappmysqld When Autoscaling is Disabled

1. Increase the replica count of the ndbappmysqld pods in the `custom_values.yaml` file under global.ndbappReplicaCount:

   ```
   global:
     ndbappReplicaCount: 4
   ```

   > **Note**
   >
   > 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 ndbappmysqld replica count is increased to the value updated in the `custom_values.yaml` file.

7.3.1.2 Scaling ndbmtd 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 ndbmtd pods.

> **Note**
>
> - Before scaling, ensure that the worker nodes have adequate resources to support scaling.
> - Divert the traffic during the horizontal scaling of the ndbmtd 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 ndbmtd, 2 ndbmysqld sql, and 2 ndbappmysqld pods. The ndbmtd 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 2024
   NAMESPACE: occne-cndbtier
   STATUS: deployed
   REVISION: 2
   TEST SUITE:     mysql-cluster-node-connection-test
   Last Started:    Mon May 20 10:03:51 2024
   Last Completed:  Mon May 20 10:04:16 2024
   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-8.4.2 ndb-8.4.2, Nodegroup: 0)
   id=2    @10.233.84.68  (mysql-8.4.2 ndb-8.4.2, Nodegroup: 0, *)
   ```

```
[ndb_mgmd(MGM)] 2 node(s)
id=49   @10.233.73.61  (mysql-8.4.2 ndb-8.4.2)
id=50   @10.233.84.67  (mysql-8.4.2 ndb-8.4.2)

[mysqld(API)]   10 node(s)
id=56   @10.233.84.69  (mysql-8.4.2 ndb-8.4.2)
id=57   @10.233.73.62  (mysql-8.4.2 ndb-8.4.2)
id=70   @10.233.78.70  (mysql-8.4.2 ndb-8.4.2)
id=71   @10.233.72.49  (mysql-8.4.2 ndb-8.4.2)
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 ndbmtd pod in the `custom_values.yaml` file:

> **Note**
>
> - The cnDBTier is deployed with 2 ndbmtd and increased to a replica count of 4 in this step.
> - You can only scale up the ndbmtd 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 ndbmtd replica count:

> **Note**
>
> After the upgrade, the new ndbmtd 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 ndbmtd pods back to their original replica count (the replica count that the ndbmtd 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 ndbmtd -p '{"spec": {"minReplicas":
      2}}'
      $ kubectl -n <namespace> patch hpa ndbmtd -p '{"spec": {"maxReplicas":
      2}}'
      ```

   b. Run the following command to scale down the ndbmtd sts replica count to 2:

      ```
      $ kubectl -n <namespace> scale sts ndbmtd --replicas=2
      ```

   c. After scaling down the ndbmtd sts replica count, terminate the newly added ndbmtd pods. Retain only the existing ndbmtd 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 (ndbmtd-n, ndbmtd-(n-1), ndbmtd-(n-2), and so on).

   For example:

a. Run the following command to delete pod ndbmtd-2:

```
$ kubectl -n <namespace> delete pod ndbmtd-2
```

b. Wait for ndbmtd-2 to return to the running state and connect back to the NDB cluster.

c. Run the following command to delete pod ndbmtd-1:

```
$ kubectl -n <namespace> delete pod ndbmtd-1
```

d. Wait for ndbmtd-1 to return to the running state and connect back to the NDB cluster.

e. Run the following command to delete pod ndbmtd-0:

```
$ kubectl -n <namespace> delete pod ndbmtd-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-8.4.2 ndb-8.4.2, Nodegroup: 0)
id=2    @10.233.84.68  (mysql-8.4.2 ndb-8.4.2, Nodegroup: 0, *)
id=3 (not connected, accepting connect from ndbmtd-2.ndbmtdsvc.occne-
cndbtier.svc.occne1-cgbu-cne-dbtier)
id=4 (not connected, accepting connect from ndbmtd-3.ndbmtdsvc.occne-
cndbtier.svc.occne1-cgbu-cne-dbtier)

[ndb_mgmd(MGM)] 2 node(s)
id=49   @10.233.73.61  (mysql-8.4.2 ndb-8.4.2)
id=50   @10.233.84.67  (mysql-8.4.2 ndb-8.4.2)

[mysqld(API)]   10 node(s)
id=56   @10.233.84.69  (mysql-8.4.2 ndb-8.4.2)
id=57   @10.233.73.62  (mysql-8.4.2 ndb-8.4.2)
id=70   @10.233.78.70  (mysql-8.4.2 ndb-8.4.2)
id=71   @10.233.72.49  (mysql-8.4.2 ndb-8.4.2)
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-8.4.2 ndb-8.4.2, Nodegroup: 0)
    id=2    @10.233.84.68  (mysql-8.4.2 ndb-8.4.2, Nodegroup: 0, *)
    id=3 (not connected, accepting connect from ndbmtd-2.ndbmtdsvc.occne-
    cndbtier.svc.occne1-cgbu-cne-dbtier)
    id=4 (not connected, accepting connect from ndbmtd-3.ndbmtdsvc.occne-
    cndbtier.svc.occne1-cgbu-cne-dbtier)
    ```

```
[ndb_mgmd(MGM)] 2 node(s)
id=49   @10.233.73.61  (mysql-8.4.2 ndb-8.4.2)
id=50   @10.233.84.67  (mysql-8.4.2 ndb-8.4.2)

[mysqld(API)]   10 node(s)
id=56   @10.233.84.69  (mysql-8.4.2 ndb-8.4.2)
id=57   @10.233.73.62  (mysql-8.4.2 ndb-8.4.2)
id=70   @10.233.78.70  (mysql-8.4.2 ndb-8.4.2)
id=71   @10.233.72.49  (mysql-8.4.2 ndb-8.4.2)
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 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-8.4.2 ndb-8.4.2, Nodegroup: 0)
    id=2    @10.233.84.68  (mysql-8.4.2 ndb-8.4.2, Nodegroup: 0, *)
    id=3 (not connected, accepting connect from ndbmtd-2.ndbmtdsvc.occne-
    cndbtier.svc.occne1-cgbu-cne-dbtier)
    id=4 (not connected, accepting connect from ndbmtd-3.ndbmtdsvc.occne-
    cndbtier.svc.occne1-cgbu-cne-dbtier)

    [ndb_mgmd(MGM)] 2 node(s)
    id=49   @10.233.73.61  (mysql-8.4.2 ndb-8.4.2)
    id=50   @10.233.84.67  (mysql-8.4.2 ndb-8.4.2)

    [mysqld(API)]   10 node(s)
    id=56   @10.233.84.69  (mysql-8.4.2 ndb-8.4.2)
    id=70   @10.233.78.70  (mysql-8.4.2 ndb-8.4.2)
    id=71   @10.233.72.49  (mysql-8.4.2 ndb-8.4.2)
    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 ndbmtd 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 ndbmtd -p '{"spec": {"maxReplicas":
    4}}'
    ```

```
$ kubectl -n <namespace> patch hpa ndbmtd -p '{"spec": {"minReplicas":
4}}'
```

b. Scale up the ndbmtd sts replica count to 4:

```
$ kubectl -n <namespace> scale sts ndbmtd --replicas=4
```

c. Wait for the ndbmtd 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 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-8.4.2 ndb-8.4.2, Nodegroup: 0)
id=2    @10.233.84.68  (mysql-8.4.2 ndb-8.4.2, Nodegroup: 0, *)
id=3    @10.233.117.60  (mysql-8.4.2 ndb-8.4.2, starting, no nodegroup)
id=4    @10.233.111.64  (mysql-8.4.2 ndb-8.4.2, starting, no nodegroup)

[ndb_mgmd(MGM)] 2 node(s)
id=49   @10.233.73.61  (mysql-8.4.2 ndb-8.4.2)
id=50   @10.233.84.67  (mysql-8.4.2 ndb-8.4.2)

[mysqld(API)]   10 node(s)
id=56   @10.233.84.69  (mysql-8.4.2 ndb-8.4.2)
id=57   @10.233.73.62  (mysql-8.4.2 ndb-8.4.2)
id=70   @10.233.78.70  (mysql-8.4.2 ndb-8.4.2)
id=71   @10.233.72.49  (mysql-8.4.2 ndb-8.4.2)
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 ndbmtd 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 ndbmtd pod is `<sequence no of the ndbmtd pod + 1>`.

    ```
    ndb_mgm>  CREATE NODEGROUP
    <node_id_for_pod_ndbmtd-2>,<node_id_for_pod_ndbmtd-3>
    ```

    For example:

    ```
    ndb_mgm>  CREATE NODEGROUP <(sequence no of pod ndbmtd-2) + 1>,<(sequence no of pod
    ndbmtd-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 ndbmtd 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 ndbmtd 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_24.3.1/Artifacts/
Scripts/tools/dbtier/lib

Adding /home/cloud-user/user/deploy_cndbtier_sitea_24.3.1/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_24.3.1/Artifacts/Scripts/
tools/dbtier/bin
```

b. Run the dbt_reorg_table_partition script:

```
$ dbt_reorg_table_partition
```

Sample output:

```
dbt_reorg_table_partition 24.3.1
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="ndbmgmd"
2024-01-17T05:11:45Z INFO - MGM_REPLICAS="2"
2024-01-17T05:11:45Z INFO - MGM_PODS:
    ndbmgmd-0
    ndbmgmd-1
2024-01-17T05:11:45Z INFO - Getting NDB sts and sts pod info...
2024-01-17T05:11:45Z INFO - NDB_STS="ndbmtd"
2024-01-17T05:11:45Z INFO - NDB_REPLICAS="2"
2024-01-17T05:11:45Z INFO - NDB_PODS:
    ndbmtd-0
    ndbmtd-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 2024
    NAMESPACE: occne-cndbtier
    STATUS: deployed
    REVISION: 2
    TEST SUITE:     mysql-cluster-node-connection-test
    Last Started:    Mon May 20 10:27:49 2024
    Last Completed:  Mon May 20 10:28:11 2024
    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 ndbmgmd-0 -- ndb_mgm -e show
    ```

    Sample output:

    ```
    Connected to Management Server at: localhost:1186
    Cluster Configuration
    ---------------------
    [ndbd(NDB)]     4 node(s)
    id=1    @10.233.92.53  (mysql-8.4.2 ndb-8.4.2, Nodegroup: 0)
    id=2    @10.233.72.66  (mysql-8.4.2 ndb-8.4.2, Nodegroup: 0)
    id=3    @10.233.117.60  (mysql-8.4.2 ndb-8.4.2, Nodegroup: 1)
    id=4    @10.233.111.64  (mysql-8.4.2 ndb-8.4.2, Nodegroup: 1, *)

    [ndb_mgmd(MGM)] 2 node(s)
    id=49   @10.233.111.62  (mysql-8.4.2 ndb-8.4.2)
    id=50   @10.233.117.59  (mysql-8.4.2 ndb-8.4.2)

    [mysqld(API)]   8 node(s)
    id=56   @10.233.72.65  (mysql-8.4.2 ndb-8.4.2)
    id=57   @10.233.92.51  (mysql-8.4.2 ndb-8.4.2)
    id=70   @10.233.72.64  (mysql-8.4.2 ndb-8.4.2)
    id=71   @10.233.92.52  (mysql-8.4.2 ndb-8.4.2)
    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:
>   replicationskiperrors:
>     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 ndbappmysqld Pods

Perform the following procedure to scale up ndbappmysqld pods vertically.

> **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.

1. Perform the following steps to update the CPU and RAM:
   a. 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
```

b. 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.

2. Perform the following steps to update the PVC value:
   a. Update the `custom_values.yaml` file with the new PVC values for ndbappmysqld pods:

```
global:
  ndbapp:
    ndbdisksize: 10Gi
```

   b. Delete the ndbappmysqld statefulset and make the dependencies as orphan:

```
$ kubectl -n occne-cndbtier delete sts --cascade=orphan ndbappmysqld
```

   c. Delete the ndbappmysqld-1 pod and patch its PVCs with the new PVC value:
      i. Delete the ndbappmysqld-1 pod:

```
$ kubectl -n occne-cndbtier delete pod ndbappmysqld-1
```

      Sample output:

```
pod "ndbappmysqld-1" deleted
```

      ii. 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
```

      iii. Wait for the new PV to attach with the PVC.
   d. Upgrade cnDBTier with the modified `custom_values.yaml` file:

```
$ helm upgrade mysql-cluster occndbtier -f occndbtier/
custom_values.yaml -n occne-cndbtier --no-hooks
```

   e. Repeat steps b through d for ndbappmysqld-0.
   f. 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
```

7.3.2.2 Scaling ndbmysqld Pods

Perform the following procedure to vertically scale up ndbmysqld 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 ndbmysqld, perform a Helm test and ensure that all the cnDBTier services are running smoothly.

1. Perform the following steps to update the CPU and RAM:
   a. 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
   ```

   b. 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*.
2. Perform the following steps to update the PVC value:
   a. Update the `custom_values.yaml` file with the new PVC values for ndbmysqld pods:

   ```
   global:
     api:
       ndbdisksize: 10Gi
   ```

   b. Delete the ndbmysqld statefulset and make the dependencies as orphan:

   ```
   $ kubectl -n occne-cndbtier delete sts --cascade=orphan ndbmysqld
   ```

c. Delete the ndbmysqld-1 pod and patch its PVCs with the new PVC value:

   i. Delete the ndbmysqld-1 pod:

   ```
   $ kubectl -n occne-cndbtier delete pod ndbmysqld-1
   ```

   Sample output:

   ```
   pod "ndbmysqld-1" deleted
   ```

   ii. 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
   ```

d. 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        RWO
   Delete           Bound         occne-cndbtier/pvc-ndbmysqld-ndbmysqld-1
   ```

e. Upgrade cnDBTier with the modified `custom_values.yaml` file:

   ```
   $ helm upgrade mysql-cluster occndbtier -f occndbtier/
   custom_values.yaml -n occne-cndbtier --no-hooks
   ```

f. Repeat steps b through e for ndbmysqld-0.

g. 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
   ```

7.3.2.3 Scaling ndbmtd Pods

Perform the following procedure 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 2024
  NAMESPACE: occne-cndbtier
  STATUS: deployed
  REVISION: 1
  TEST SUITE:     mysql-cluster-node-connection-test
  Last Started:    Mon May 20 04:03:01 2024
  Last Completed:  Mon May 20 04:03:26 2024
  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 ndbmgmd-0 -- ndb_mgm -e show
  ```

  Sample output:

  ```
  Connected to Management Server at: localhost:1186
  Cluster Configuration
  ---------------------
  [ndbd(NDB)]     2 node(s)
  id=1    @10.233.95.87  (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.

1. Perform the following steps to update the CPU and RAM values in the `custom_values.yaml` file:

a. 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
```

b. 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.

2. Perform the following steps to update the PVC value:

a. Update the `custom_values.yaml` file with the new PVC values for ndbmtd pods:
For example:

The following code block shows the old PVC values in the `custom_values.yaml` file:

```
global:
  ndb:
    ndbdisksize: 3Gi
    ndbbackupdisksize: 3Gi
```

The following code block shows the updated PVC values in the `custom_values.yaml` file:

```
global:
  ndb:
    ndbdisksize: 6Gi
    ndbbackupdisksize: 6Gi
```

b. Run the following command to delete the ndbmtd statefulset and set the dependency to *orphan*:

```
$ kubectl -n occne-cndbtier delete sts --cascade=orphan ndbmtd
```

c. Run the following command to delete the ndbmtd-1 pod and patch the pod's PVC values with the values updated in Step a:

```
$ kubectl -n occne-cndbtier delete pod ndbmtd-1

$ kubectl -n occne-cndbtier patch -p '{ "spec": { "resources":
{ "requests": { "storage": "6Gi" }}}}' pvc pvc-ndbmtd-ndbmtd-1
$ kubectl -n occne-cndbtier patch -p '{ "spec": { "resources":
{ "requests": { "storage": "6Gi" }}}}' pvc pvc-backup-ndbmtd-ndbmtd-1
```

d. 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 ndbmtd-1
```

Sample output:

```
pvc-53fe7988-4a81-40d3-a366-fd894de89535  6Gi  RWO  Delete  Bound
occne-cndbtier/pvc-ndbmtd-ndbmtd-1  occne-dbtier-sc  62m
```

e. Upgrade cnDBTier with the modified `custom_values.yaml` file:

```
$ helm upgrade mysql-cluster occndbtier -f occndbtier/
custom_values.yaml -n occne-cndbtier --no-hooks
```

f. Perform Step b through Step e for the ndbmtd-0 pod.

> **Note**
>
> If you have more that two data pods, then follow Steps b to e for each ndbmtd pod in the following order: ndbmtd-n, ndbmtd-(n-1), and so on up to ndbmtd-0.

g. 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 `OCCNE_NAMESPACE` variable in the command with your cnDBTier namespace.

```
helm -n ${OCCNE_NAMESPACE} upgrade ${OCCNE_RELEASE_NAME} occndbtier -f
occndbtier/custom_values.yaml
```

h. 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 2024
NAMESPACE: occne-cndbtier
STATUS: deployed
REVISION: 4
TEST SUITE:     mysql-cluster-node-connection-test
Last Started:    Mon May 20 04:45:15 2024
Last Completed:  Mon May 20 04:45:35 2024
Phase:          Succeeded
```

7.3.2.4 Vertical Scaling of db-replication-svc

This section provides the procedures to scale up the db-replication-svc pods vertically.

> **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.

1. Perform the following steps to update the CPU and RAM:
   a. Configure the required CPU and memory values in the `custom_values.yaml` file. For example:

```
db-replication-svc:
  resources:
    limits:
      cpu: 1
      memory: 2048Mi
      ephemeral-storage: 1Gi
    requests:
      cpu: 0.6
      memory: 1024Mi
      ephemeral-storage: 90Mi
```

b. 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.

2. Perform the following steps to update the PVC value:
   a. Perform the following steps to scale down the replication service pod:
      i. 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
```

      ii. 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
```

   b. 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.

c. 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
```

d. 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        RWO
Delete           Bound    cluster2/pvc-cluster2-cluster1-replication-
svc   occne-dbtier-sc            12m
pvc-e6526259-d007-4868-a4e2-d53a8569edc8   10Gi       RWO
Delete           Bound    cluster1/pvc-cluster1-cluster2-replication-
svc   occne-dbtier-sc            15m
```

e. 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.

7.3.2.5 Scaling of ndbmgmd Pods

Perform the following procedure to scale up ndbmgmd pods vertically.

> **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 ndbmgmd, perform a Helm Test and ensure that all the cnDBTier services are running smoothly.

1. Perform the following steps to update the CPU and RAM:
   a. 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
   ```

   b. 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*.
2. Perform the following steps to update the PVC value:
   a. Update the `custom_values.yaml` file with the new PVC values for ndbmgmd pods:

   ```
   global:
     mgm:
       ndbdisksize: 10Gi
   ```

   b. Delete the ndbmgmd statefulset and make the dependencies as orphan:

   ```
   $ kubectl -n occne-cndbtier delete sts --cascade=orphan ndbmgmd
   ```

   c. Delete the ndbmgmd-1 pod and patch its PVCs with the new PVC value:
      i. Delete the ndbmgmd-1 pod:

      ```
      $ kubectl -n occne-cndbtier delete pod ndbmgmd-1
      ```

      Sample output:

      ```
      pod "ndbmgmd-1" deleted
      ```

ii. Patch the PVCs of ndbmgmd-1 pod with the new PVC values:

```
$ kubectl -n occne-cndbtier patch -p '{ "spec": { "resources":
{ "requests": { "storage": "10Gi" }}}}' pvc pvc-ndbmgmd-ndbmgmd-1
```

d. 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 ndbmgmd-1
```

Sample output:

```
pvc-8a58a7b4-25bb-4b32-a07a-728h8d792835   10Gi        RWO
Delete           Bound         occne-cndbtier/pvc-ndbmgmd-ndbmgmd-1
```

e. Upgrade cnDBTier with the modified `custom_values.yaml` file:

```
$ helm upgrade mysql-cluster occndbtier -f occndbtier/
custom_values.yaml -n occne-cndbtier --no-hooks
```

f. Repeat steps b through e for ndbmgmd-0.

g. 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
```

7.4 Configuring cnDBTier SQL Nodes to Support TLS for Georeplication

This chapter describes the manual procedure to configure cnDBTier SQL nodes to support Transport Layer Security (TLS) for georeplication channels between two mate sites.

> **Note**
>
> - TLS Support between NF applications and cnDBTier is not in the current scope of this procedure.
> - You cannot restore georeplication if TLS is enabled in a cnDBTier setup.

7.4.1 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.

7.4.2 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.
> - Steps 1, 4, and 7 in this procedure are optional. You can skip these steps if you know the list of certificates.

1. [Optional] Get the list of existing CA certificates of the cnDBTier cluster. You can skip this step if you know the list of 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-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, delete the secret using the following command. Otherwise, skip this step and move to Step 4.

   ```
   $ kubectl -n <namespace> delete secret cndbtier-trust-store-secret
   ```

   where, `<namespace>` is the name of the cnDBTier namespace.
   For example:

   ```
   $ kubectl -n occne-cndbtier delete secret cndbtier-trust-store-secret
   ```

3. If you deleted the secret of the existing CA certificates in the previous step, then create the `cndbtier-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-trust-store-secret --from-file=<ca_certificate>
   ```

   For example:

   ```
   $ kubectl -n ${OCCNE_NAMESPACE} create secret generic cndbtier-trust-store-secret --from-file=combine-ca.pem
   ```

4. [Optional] Perform the following steps to get the existing server certificate and the server 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 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
      ```

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-server-secret
   ```

   For example:

   ```
   $ kubectl -n occne-cndbtier delete secret cndbtier-server-secret
   ```

6. 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
   ```

7. [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
      ```

8. 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
```

9. 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
   ```

10. 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.
11. 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.

12. 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=occne-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=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-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.
  ```

13. Repeat Step 12 on all the mate sites where you have updated the certificates.
14. 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
       ndbmtd         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...
```

15. Repeat Step 14 on all the mate sites where you have updated the certificates.

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 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-8.4.2 ndb-8.4.2, Nodegroup: 0)
id=2    @10.233.114.33  (mysql-8.4.2 ndb-8.4.2, Nodegroup: 0, *)

[ndb_mgmd(MGM)] 2 node(s)
id=49   @10.233.65.167  (mysql-8.4.2 ndb-8.4.2)
id=50   @10.233.127.115  (mysql-8.4.2 ndb-8.4.2)

[mysqld(API)]   8 node(s)
id=56   @10.233.114.34   (mysql-8.4.2 ndb-8.4.2)
id=57   @10.233.85.91    (mysql-8.4.2 ndb-8.4.2)
id=70   @10.233.127.117  (mysql-8.4.2 ndb-8.4.2)
id=71   @10.233.85.93    (mysql-8.4.2 ndb-8.4.2)
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 occndbtier/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 occndbtier/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 2024
   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
ndbmtd-0
2/2     Running   0             150m
ndbmtd-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-8.4.2 ndb-8.4.2, Nodegroup: 0)
id=2    @10.233.114.33  (mysql-8.4.2 ndb-8.4.2, Nodegroup: 0, *)

[ndb_mgmd(MGM)] 2 node(s)
id=49   @10.233.65.167  (mysql-8.4.2 ndb-8.4.2)
id=50   @10.233.127.115  (mysql-8.4.2 ndb-8.4.2)

[mysqld(API)]   8 node(s)
id=56   @10.233.114.34  (mysql-8.4.2 ndb-8.4.2)
id=57   @10.233.85.91  (mysql-8.4.2 ndb-88.4.2)
id=70   @10.233.127.117  (mysql-8.4.2 ndb-8.4.2)
id=71   @10.233.85.93  (mysql-8.4.2 ndb-8.4.2)
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-7955b6b6fbdc   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-8.4.2 ndb-8.4.2, Nodegroup: 0)
id=2    @10.233.114.33  (mysql-8.4.2 ndb-8.4.2, Nodegroup: 0, *)

[ndb_mgmd(MGM)] 2 node(s)
id=49   @10.233.65.167  (mysql-8.4.2 ndb-8.4.2)
id=50   @10.233.127.115  (mysql-8.4.2 ndb-8.4.2)

[mysqld(API)]   10 node(s)
id=56   @10.233.114.34  (mysql-8.4.2 ndb-8.4.2)
id=57   @10.233.85.91  (mysql-8.4.2 ndb-8.4.2)
id=58   @10.233.114.34  (mysql-8.4.2 ndb-8.4.2)
id=59   @10.233.85.91  (mysql-8.4.2 ndb-8.4.2)
id=70   @10.233.127.117  (mysql-8.4.2 ndb-8.4.2)
id=71   @10.233.85.93  (mysql-8.4.2 ndb-8.4.2)
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 occndbtier/custom_values.yaml
```

Sample output:

```
 dbreplsvcdeployments:
    ....
    # 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 2024
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
ndbmtd-0
2/2     Running   0              150m
ndbmtd-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-8.4.2 ndb-8.4.2, Nodegroup: 0)
id=2    @10.233.114.33  (mysql-8.4.2 ndb-8.4.2, Nodegroup: 0, *)

[ndb_mgmd(MGM)] 2 node(s)
id=49   @10.233.65.167  (mysql-8.4.2 ndb-8.4.2)
id=50   @10.233.127.115  (mysql-8.4.2 ndb-8.4.2)

[mysqld(API)]   8 node(s)
id=56   @10.233.114.34  (mysql-8.4.2 ndb-8.4.2)
id=57   @10.233.85.91  (mysql-8.4.2 ndb-8.4.2)
id=70   @10.233.127.117  (mysql-8.4.2 ndb-8.4.2)
id=71   @10.233.85.93  (mysql-8.4.2 ndb-8.4.2)
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-7955b6b6fbdc   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-8.4.2 ndb-8.4.2, Nodegroup: 0)
id=2    @10.233.114.33  (mysql-8.4.2 ndb-8.4.2, Nodegroup: 0, *)

[ndb_mgmd(MGM)] 2 node(s)
id=49   @10.233.65.167  (mysql-8.4.2 ndb-8.4.2)
id=50   @10.233.127.115  (mysql-8.4.2 ndb-8.4.2)

[mysqld(API)]   10 node(s)
id=56   @10.233.114.34  (mysql-8.4.2 ndb-8.4.2)
id=57   @10.233.85.91  (mysql-8.4.2 ndb-8.4.2)
id=58   @10.233.114.34  (mysql-8.4.2 ndb-8.4.2)
id=59   @10.233.85.91  (mysql-8.4.2 ndb-8.4.2)
id=70   @10.233.127.117  (mysql-8.4.2 ndb-8.4.2)
id=71   @10.233.85.93  (mysql-8.4.2 ndb-8.4.2)
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 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 2024
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
   ndbmtd-0
   2/2     Running   0              150m
   ndbmtd-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-8.4.2 ndb-8.4.2, Nodegroup: 0)
id=2    @10.233.114.33  (mysql-8.4.2 ndb-8.4.2, Nodegroup: 0, *)

[ndb_mgmd(MGM)] 2 node(s)
id=49   @10.233.65.167  (mysql-8.4.2 ndb-8.4.2)
id=50   @10.233.127.115  (mysql-8.4.2 ndb-8.4.2)

[mysqld(API)]   10 node(s)
id=56   @10.233.124.92  (mysql-8.4.2 ndb-8.4.2)
id=57   @10.233.114.135  (mysql-8.4.2 ndb-8.4.2)
id=58   @10.233.114.34  (mysql-8.4.2 ndb-8.4.2)
id=59   @10.233.85.91  (mysql-8.4.2 ndb-8.4.2)
id=70   @10.233.127.117  (mysql-8.4.2 ndb-8.4.2)
id=71   @10.233.85.93  (mysql-8.4.2 ndb-8.4.2)
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-7955b6b6fbdc   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-8.4.2 ndb-8.4.2, Nodegroup: 0)
id=2    @10.233.114.33  (mysql-8.4.2 ndb-8.4.2, Nodegroup: 0, *)

[ndb_mgmd(MGM)] 2 node(s)
id=49   @10.233.65.167  (mysql-8.4.2 ndb-8.4.2)
id=50   @10.233.127.115  (mysql-8.4.2 ndb-8.4.2)

[mysqld(API)]   12 node(s)
id=56   @10.233.114.34   (mysql-8.4.2 ndb-8.4.2)
id=57   @10.233.85.91    (mysql-8.4.2 ndb-8.4.2)
id=58   @10.233.114.34   (mysql-8.4.2 ndb-8.4.2)
id=59   @10.233.85.91    (mysql-8.4.2 ndb-8.4.2)
id=60   @10.233.62.226   (mysql-8.4.2 ndb-8.4.2)
id=61   @10.233.14.15    (mysql-8.4.2 ndb-8.4.2)
id=70   @10.233.127.117  (mysql-8.4.2 ndb-8.4.2)
id=71   @10.233.85.93    (mysql-8.4.2 ndb-8.4.2)
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 occndbtier/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
        primaryhost: "ndbmysqld-4.ndbmysqldsvc.cluster1.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: "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 2024
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-7955b6b6fbdc
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
ndbmtd-0
2/2     Running   0              150m
ndbmtd-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 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-8.4.2 ndb-8.4.2, Nodegroup: 0)
id=2    @10.233.114.33  (mysql- ndb-8.4.2, Nodegroup: 0, *)

[ndb_mgmd(MGM)] 2 node(s)
id=49   @10.233.65.167  (mysql-8.4.2 ndb-8.4.2)
id=50   @10.233.127.115  (mysql-8.4.2 ndb-8.4.2)
```

```
[mysqld(API)]   12 node(s)
id=56   @10.233.124.92    (mysql-8.4.2 ndb-8.4.2)
id=57   @10.233.114.135   (mysql-8.4.2 ndb-8.4.2)
id=58   @10.233.114.34    (mysql-8.4.2 ndb-8.4.2)
id=59   @10.233.85.91     (mysql-8.4.2 ndb-8.4.2)
id=60   @10.233.15.24     (mysql-8.4.2 ndb-8.4.2)
id=61   @10.233.48.74     (mysql-8.4.2 ndb-8.4.2)
id=70   @10.233.127.117   (mysql-8.4.2 ndb-8.4.2)
id=71   @10.233.85.93     (mysql-8.4.2 ndb-8.4.2)
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-7955b6b6fbdc   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-8.4.2 ndb-8.4.2, Nodegroup: 0)
id=2    @10.233.114.33  (mysql-8.4.2 ndb-8.4.2, Nodegroup: 0, *)

[ndb_mgmd(MGM)] 2 node(s)
id=49   @10.233.65.167  (mysql-8.4.2 ndb-8.4.2)
id=50   @10.233.127.115  (mysql-8.4.2 ndb-8.4.2)

[mysqld(API)]   12 node(s)
id=56   @10.233.114.34    (mysql-8.4.2 ndb-8.4.2)
id=57   @10.233.85.91     (mysql-8.4.2 ndb-8.4.2)
id=58   @10.233.114.34    (mysql-8.4.2 ndb-8.4.2)
id=59   @10.233.85.91     (mysql-8.4.2 ndb-8.4.2)
id=60   @10.233.24.121    (mysql-8.4.2 ndb-8.4.2)
id=61   @10.233.85.22     (mysql-8.4.2 ndb-8.4.2)
id=70   @10.233.127.117   (mysql-8.4.2 ndb-8.4.2)
id=71   @10.233.85.93     (mysql-8.4.2 ndb-8.4.2)
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 2024
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
ndbmtd-0
2/2     Running   0              150m
ndbmtd-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 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-8.4.2 ndb-8.4.2, Nodegroup: 0)
id=2    @10.233.114.33  (mysql-8.4.2 ndb-8.4.2, Nodegroup: 0, *)

[ndb_mgmd(MGM)] 2 node(s)
id=49   @10.233.65.167  (mysql-8.4.2 ndb-8.4.2)
id=50   @10.233.127.115  (mysql-8.4.2 ndb-8.4.2)
```

```
[mysqld(API)]   12 node(s)
id=56   @10.233.124.92  (mysql-8.4.2 ndb-8.4.2)
id=57   @10.233.114.135  (mysql-8.4.2 ndb-8.4.2)
id=58   @10.233.124.111  (mysql-8.4.2 ndb-8.4.2)
id=59   @10.233.110.81  (mysql-8.4.2 ndb-8.4.2)
id=60   @10.233.110.41  (mysql-8.4.2 ndb-8.4.2)
id=61   @10.233.98.102  (mysql-8.4.2 ndb-8.4.2)
id=70   @10.233.127.117  (mysql-8.4.2 ndb-8.4.2)
id=71   @10.233.85.93  (mysql-8.4.2 ndb-8.4.2)
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-7955b6b6fbdc   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 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-8.4.2 ndb-8.4.2, Nodegroup: 0)
id=2    @10.233.114.33  (mysql-8.4.2 ndb-8.4.2, Nodegroup: 0, *)

[ndb_mgmd(MGM)] 2 node(s)
id=49   @10.233.65.167  (mysql-8.4.2 ndb-8.4.2)
id=50   @10.233.127.115  (mysql-8.4.2 ndb-8.4.2)

[mysqld(API)]   12 node(s)
id=56   @10.233.71.24     (mysql-8.4.2 ndb-8.4.2)
id=57   @10.233.119.18    (mysql-8.4.2 ndb-8.4.2)
id=58   @10.233.69.23     (mysql-8.4.2 ndb-8.4.2)
id=59   @10.233.116.21    (mysql-8.4.2 ndb-8.4.2)
id=60   @10.233.70.16     (mysql-8.4.2 ndb-8.4.2)
id=61   @10.233.78.20     (mysql-8.4.2 ndb-8.4.2)
id=70   @10.233.127.117   (mysql-8.4.2 ndb-8.4.2)
id=71   @10.233.85.93     (mysql-8.4.2 ndb-8.4.2)
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 2024
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
ndbmtd-0
2/2     Running   0             150m
ndbmtd-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-8.4.2 ndb-8.4.2, Nodegroup: 0)
id=2    @10.233.114.33  (mysql-8.4.2 ndb-8.4.2, Nodegroup: 0, *)

[ndb_mgmd(MGM)] 2 node(s)
id=49   @10.233.65.167  (mysql-8.4.2 ndb-8.4.2)
id=50   @10.233.127.115  (mysql-8.4.2 ndb-8.4.2)

[mysqld(API)]   12 node(s)
id=56   @10.233.71.24     (mysql-8.4.2 ndb-8.4.2)
id=57   @10.233.119.18    (mysql-8.4.2 ndb-8.4.2)
id=58   @10.233.69.23     (mysql-8.4.2 ndb-8.4.2)
id=59   @10.233.116.21    (mysql-8.4.2 ndb-8.4.2)
id=60   @10.233.70.16     (mysql-8.4.2 ndb-8.4.2)
id=61   @10.233.78.20     (mysql-8.4.2 ndb-8.4.2)
id=70   @10.233.127.117   (mysql-8.4.2 ndb-8.4.2)
id=71   @10.233.85.93     (mysql-8.4.2 ndb-8.4.2)
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_24_3_1_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/rc4.24.3.1/Artifacts/Scripts/
tools/lib

Adding /home/cgbu-cne-dbtier/csar/24.3.1/Artifacts/Scripts/tools/bin to
PATH
PATH=/home/cgbu-cne-dbtier/csar/24.3.1/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
   ```

> **Note**
>
> 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.

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:

      ```
      $ source ./source_me
      $ 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 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 | 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
      $ 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 | 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 3 and restart all the DB replication service deployments on cluster 2:

> **Note**
>
> Replace `cluster1` in the commands 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 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
      $ 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 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 | 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
$ 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 Bastion Host of cluster 2.

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 2:

```
$ 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 occndbtier -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 2024
    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-8.4.2 ndb-8.4.2, Nodegroup: 0)
   id=2    @10.233.114.33  (mysql-8.4.2 ndb-8.4.2, Nodegroup: 0, *)

   [ndb_mgmd(MGM)] 2 node(s)
   id=49   @10.233.65.167  (mysql-8.4.2 ndb-8.4.2)
   id=50   @10.233.127.115  (mysql-8.4.2 ndb-8.4.2)

   [mysqld(API)]   18 node(s)
   ```

```
id=56   @10.233.124.92  (mysql-8.4.2 ndb-8.4.2)
id=57   @10.233.114.135  (mysql-8.4.2 ndb-8.4.2)
id=58   @10.233.113.87  (mysql-8.4.2 ndb-8.4.2)
id=59   @10.233.114.32  (mysql-8.4.2 ndb-8.4.2)
id=60   @10.233.108.33  (mysql-8.4.2 ndb-8.4.2)
id=61   @10.233.78.230  (mysql-8.4.2 ndb-8.4.2)
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-8.4.2 ndb-8.4.2)
id=71   @10.233.85.93  (mysql-8.4.2 ndb-8.4.2)
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 'ndbmtd'
```

Sample output:

```
ndbmtd-0                                                  3/3     Running
0          14m
ndbmtd-1                                                  3/3     Running
0          13m
```

ii. Delete the first data pod of cnDBTier cluster1 (in this example, ndbmtd-0) such that the pod restarts:

```
$ kubectl delete pod ndbmtd-0 --namespace=cluster1
```

Sample output:

```
pod "ndbmtd-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 'ndbmtd'
```

Sample output:

```
ndbmtd-0                                                3/3     Running
0          65s
ndbmtd-1                                                3/3     Running
0          13m
```

iv. Check the status of cnDBTier cluster1:

```
$ kubectl -n cluster1 exec -it ndbmgmd-0 -- ndb_mgm -e show
```

Sample output:

```
---------------------
[ndbd(NDB)]     2 node(s)
id=1    @10.233.85.92  (mysql-8.4.2 ndb-8.4.2, Nodegroup: 0)
id=2    @10.233.114.33  (mysql-8.4.2 ndb-8.4.2, Nodegroup: 0, *)

[ndb_mgmd(MGM)] 2 node(s)
id=49   @10.233.65.167  (mysql-8.4.2 ndb-8.4.2)
id=50   @10.233.127.115  (mysql-8.4.2 ndb-8.4.2)

[mysqld(API)]   18 node(s)
id=56   @10.233.124.92  (mysql-8.4.2 ndb-8.4.2)
id=57   @10.233.114.135  (mysql-8.4.2 ndb-8.4.2)
id=58   @10.233.113.87  (mysql-8.4.2 ndb-8.4.2)
id=59   @10.233.114.32  (mysql-8.4.2 ndb-8.4.2)
id=60   @10.233.108.33  (mysql-8.4.2 ndb-8.4.2)
id=61   @10.233.78.230  (mysql-8.4.2 ndb-8.4.2)
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-8.4.2 ndb-8.4.2)
id=71   @10.233.85.93  (mysql-8.4.2 ndb-8.4.2)
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 ndbmgmd-0 --
ndb_mgm -e show
```

Example:

```
$ kubectl -n cluster1 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    @10.233.92.53  (mysql-8.4.2 ndb-8.4.2, Nodegroup: 0)
id=2    @10.233.72.66  (mysql-8.4.2 ndb-8.4.2, Nodegroup: 0)

[ndb_mgmd(MGM)] 2 node(s)
id=49   @10.233.111.62  (mysql-8.4.2 ndb-8.4.2)
id=50   @10.233.117.59  (mysql-8.4.2 ndb-8.4.2)

[mysqld(API)]  10 node(s)
id=56   @10.233.72.65  (mysql-8.4.2 ndb-8.4.2)
id=57   @10.233.92.51  (mysql-8.4.2 ndb-8.4.2)
id=58   @10.233.81.112  (mysql-8.4.2 ndb-8.4.2)
id=59   @10.233.64.07  (mysql-8.4.2 ndb-8.4.2)
id=60   @10.233.71.16  (mysql-8.4.2 ndb-8.4.2)
id=61   @10.233.114.196  (mysql-8.4.2 ndb-8.4.2)
id=62   @10.233.84.212  (mysql-8.4.2 ndb-8.4.2)
id=63   @10.233.108.21  (mysql-8.4.2 ndb-8.4.2)
id=64   @10.233.121.20  (mysql-8.4.2 ndb-8.4.2)
id=65   @10.233.109.231  (mysql-8.4.2 ndb-8.4.2)
id=66   @10.233.121.37  (mysql-8.4.2 ndb-8.4.2)
id=67   @10.233.84.38  (mysql-8.4.2 ndb-8.4.2)
id=70   @10.233.124.92  (mysql-8.4.2 ndb-8.4.2)
id=71   @10.233.113.109  (mysql-8.4.2 ndb-8.4.2)
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 2024
    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-8.4.2 ndb-8.4.2, Nodegroup: 0)
id=2    @10.233.114.33  (mysql-8.4.2 ndb-8.4.2, Nodegroup: 0, *)

[ndb_mgmd(MGM)] 2 node(s)
id=49   @10.233.65.167  (mysql-8.4.2 ndb-8.4.2)
id=50   @10.233.127.115  (mysql-8.4.2 ndb-8.4.2)

[mysqld(API)]   18 node(s)
id=56   @10.233.124.92  (mysql-8.4.2 ndb-8.4.2)
id=57   @10.233.114.135  (mysql-8.4.2 ndb-8.4.2)
id=58   @10.233.113.87  (mysql-8.4.2 ndb-8.4.2)
id=59   @10.233.114.32  (mysql-8.4.2 ndb-8.4.2)
id=60   @10.233.108.33  (mysql-8.4.2 ndb-8.4.2)
id=61   @10.233.78.230  (mysql-8.4.2 ndb-8.4.2)
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-8.4.2 ndb-8.4.2)
id=71   @10.233.85.93  (mysql-8.4.2 ndb-8.4.2)
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 'ndbmtd'
```

Sample output:

```
ndbmtd-0                                                3/3     Running
0          14m
ndbmtd-1                                                3/3     Running
0          13m
```

ii. Delete first data pod of cnDBTier cluster1 (ndbmtd-0) such that the pod restarts:

```
$ kubectl delete pod ndbmtd-0 --namespace=cluster1
```

Sample output:

```
pod "ndbmtd-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 'ndbmtd'
```

Sample output:

```
ndbmtd-0                                                3/3     Running
0          65s
ndbmtd-1                                                3/3     Running
0          13m
```

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-8.4.2 ndb-8.4.2, Nodegroup: 0)
id=2    @10.233.114.33  (mysql-8.4.2 ndb-8.4.2, Nodegroup: 0, *)
```

```
[ndb_mgmd(MGM)] 2 node(s)
id=49   @10.233.65.167  (mysql-8.4.2 ndb-8.4.2)
id=50   @10.233.127.115  (mysql-8.4.2 ndb-8.4.2)

[mysqld(API)]   18 node(s)
id=56   @10.233.124.92  (mysql-8.4.2 ndb-8.4.2)
id=57   @10.233.114.135  (mysql-8.4.2 ndb-8.4.2)
id=58   @10.233.113.87  (mysql-8.4.2 ndb-8.4.2)
id=59   @10.233.114.32  (mysql-8.4.2 ndb-8.4.2)
id=60   @10.233.108.33  (mysql-8.4.2 ndb-8.4.2)
id=61   @10.233.78.230  (mysql-8.4.2 ndb-8.4.2)
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-8.4.2 ndb-8.4.2)
id=71   @10.233.85.93  (mysql-8.4.2 ndb-8.4.2)
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-8.4.2 ndb-8.4.2, Nodegroup: 0)
    id=2    @10.233.72.66  (mysql-8.4.2 ndb-8.4.2, Nodegroup: 0)

    [ndb_mgmd(MGM)] 2 node(s)
    id=49   @10.233.111.62  (mysql-8.4.2 ndb-8.4.2)
    ```

```
id=50   @10.233.117.59  (mysql-8.4.2 ndb-8.4.2)

[mysqld(API)]  10 node(s)
id=56   @10.233.72.65  (mysql-8.4.2 ndb-8.4.2)
id=57   @10.233.92.51  (mysql-8.4.2 ndb-8.4.2)
id=58   @10.233.113.87  (mysql-8.4.2 ndb-8.4.2)
id=59   @10.233.114.32  (mysql-8.4.2 ndb-8.4.2)
id=60   @10.233.108.33  (mysql-8.4.2 ndb-8.4.2)
id=61   @10.233.78.230  (mysql-8.4.2 ndb-8.4.2)
id=70   @10.233.72.64  (mysql-8.4.2 ndb-8.4.2)
id=71   @10.233.92.52  (mysql-8.4.2 ndb-8.4.2)
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="<password for the user occneuser>"
   export MYSQL_CMD="kubectl -n <namespace> exec <ndbmysqld-0/ndbappmysqld-0
   pod name> -- mysql --binary-as-hex=0 --show-warnings"
   export NEW_ENCRYPTION_KEY="<new_encryption_password>"
   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="<current_encryption_password>"


   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
      ndbmtd         2/2     27h
      ndbmysqld      2/2     27h
      ```

   b. Perform rollout restart of ndbmsqld:

      ```
      $ 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 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-
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...
```

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, occneuser, and occnerepluser 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` scripti 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 7-1 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`
--secrets-only	This option is used to change the password in the secrets only.	`dbtpasswd --secrets-only`

Table 7-1 (Cont.) Configuration Options

Option	Usage	Example
--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 7-1 (Cont.) Configuration Options

Option	Usage	Example

--nf-namespace=<namespace_where_secrets_are>	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:
		`dbtpasswd --secrets-and-mysql-only --nf-namespace=other-namespace-name nf-secret-on-diff-namespace`
		• To discard a non-DBTier old password whose secret is on a different namespace:
		`dbtpasswd --discard-only --nf-namespace=other-namespace-name nf-secret-on-diff-namespace`
		• To change a non-DBTier password in its secret, which is on same namespace as cnDBTier, and in mysql:
		`dbtpasswd --secrets-and-mysql-only nf-secret-on-same-namespace`
		• To discard a non-DBTier old password whose secret is on the same namespace as cnDBTier:
		`dbtpasswd --discard-only nf-secret-on-same-namespace`

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="ndbmgmd"
2022-12-23T21:55:12Z INFO - MGM_REPLICAS="2"
2022-12-23T21:55:12Z INFO -
    ndbmgmd-0
    ndbmgmd-1
2022-12-23T21:55:18Z INFO - NDB_STS="ndbmtd"
2022-12-23T21:55:18Z INFO - NDB_REPLICAS="2"
2022-12-23T21:55:18Z INFO -
    ndbmtd-0
    ndbmtd-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/ndbmgmd-0 not labeled
pod/ndbmgmd-1 not labeled
```

```
pod/ndbmtd-0 not labeled
pod/ndbmtd-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-mysqlndb-root-secret, with new
password
secret/occne-mysqlndb-root-secret patched
2022-12-23T21:56:36Z INFO - Secret, occne-mysqlndb-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 - ndbmtd 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
"kubectl 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="ndbmgmd"
2023-10-12T21:34:07Z INFO - MGM_REPLICAS="2"
2023-10-12T21:34:07Z INFO - MGM_PODS:
    ndbmgmd-0
    ndbmgmd-1
2023-10-12T21:34:07Z INFO - Getting NDB sts and sts pod info...
2023-10-12T21:34:07Z INFO - NDB_STS="ndbmtd"
2023-10-12T21:34:07Z INFO - NDB_REPLICAS="2"
2023-10-12T21:34:07Z INFO - NDB_PODS:
    ndbmtd-0
    ndbmtd-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/ndbmtd-0 labeled
pod/ndbmtd-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-mysqlndb-root-secret, with new
password
secret/occne-mysqlndb-root-secret patched
2023-10-12T21:34:43Z INFO - Secret, occne-mysqlndb-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 ndbmtd 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 ndbmtd-d7bf774f6...
Waiting for 1 pods to be ready...
Waiting for 1 pods to be ready...
statefulset rolling update complete 2 pods at revision ndbmtd-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="ndbmtd"
   2022-12-24T03:41:09Z INFO - NDB_REPLICAS="2"
   2022-12-24T03:41:09Z INFO -
       ndbmtd-0
       ndbmtd-1
   2022-12-24T03:41:09Z INFO - API_STS="ndbmysqld"
   2022-12-24T03:41:09Z INFO - API_REPLICAS="2"
   2022-12-24T03:41:09Z INFO -
       ndbmysqld-0
       ndbmysqld-1
   2022-12-24T03:41:09Z INFO - APP_STS="ndbappmysqld"
   2022-12-24T03:41:09Z INFO - APP_REPLICAS="2"
   2022-12-24T03:41:10Z INFO -
       ndbappmysqld-0
       ndbappmysqld-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/ndbmtd-0 not labeled
pod/ndbmtd-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-mysqlndb-root-secret, with new
password
secret/occne-mysqlndb-root-secret patched
2022-12-24T03:41:15Z INFO - Secret, occne-mysqlndb-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/occne/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 (ndbmtd-0, ndbmtd-1, and so on):

   a. Identify the list of data pods in the cnDBTier cluster:

```
$ kubectl get pods --namespace=<namespace of cnDBTier Cluster> | grep
'ndbmtd'
```

      For example, use the following command to identify the list of data pods in cnDBTier cluster1:

```
$ kubectl get pods --namespace=cluster1 | grep 'ndbmtd'
```

      Sample output:

```
ndbmtd-0                                                  3/3
Running   0          14m
ndbmtd-1                                                  3/3
Running   0          13m
```

   b. Delete the first data pod of the cnDBTier cluster:

```
$ kubectl delete pod ndbmtd-0 --namespace=<namespace of cnDBTier
Cluster>
```

      For example, use the following command to delete the first data pod of cnDBTier cluster1:

```
$ kubectl delete pod ndbmtd-0 --namespace=cluster1
```

      Sample output:

```
pod "ndbmtd-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
'ndbmtd'
```

For example:

```
$ kubectl get pods --namespace=cluster1 | grep 'ndbmtd'
```

Sample output:

```
ndbmtd-0                                                  3/3
Running   0          65s
ndbmtd-1                                                  3/3
Running   0          13m
```

d. Check the status of the cnDBTier cluster:

```
$ kubectl -n <namespace of cnDBTier Cluster> exec -it ndbmgmd-0 --
ndb_mgm -e show
```

For example, use the following command to 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-8.4.2 ndb-8.4.2, Nodegroup: 0)
id=2    @10.233.114.33  (mysql-8.4.2 ndb-8.4.2, Nodegroup: 0, *)

[ndb_mgmd(MGM)] 2 node(s)
id=49   @10.233.65.167  (mysql-8.4.2 ndb-8.4.2)
id=50   @10.233.127.115  (mysql-8.4.2 ndb-8.4.2)

[mysqld(API)]   8 node(s)
id=56   @10.233.120.210  (mysql-8.4.2 ndb-8.4.2)
id=57   @10.233.124.93  (mysql-8.4.2 ndb-8.4.2)
id=70   @10.233.127.117  (mysql-8.4.2 ndb-8.4.2)
id=71   @10.233.85.93  (mysql-8.4.2 ndb-8.4.2)
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 (ndbmtd-1, ndbmtd-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 secrets (keystore and credentials) of all the sites where you want to modify the certificates:

```
$ kubectl -n <cndbtier_namespace> get secret cndbtier-https-cert-file -o
yaml > cndbtier-https-cert-file-backup.yaml
$ 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
```

where, `<cndbtier_namespace>` is the name of the cnDBTier namespace.

For example:

```
$ kubectl -n cluster1 get secret cndbtier-https-cert-file -o yaml >
cndbtier-https-cert-file-backup.yaml
$ kubectl -n cluster1 get secret cndbtier-https-cert-cred -o
jsonpath="{.data.keystorepassword}" | base64 --decode
$ kubectl -n cluster1 get secret cndbtier-https-cert-cred -o
jsonpath="{.data.keystoretype}" | base64 --decode
$ kubectl -n cluster1 get secret cndbtier-https-cert-cred -o
jsonpath="{.data.keyalias}" | base64 --decode
```

3. Delete the old HTTPS secrets of all the sites where you want to modify the certificates:

```
$ kubectl get secrets -n <cndbtier_namespace>
$ kubectl -n <cndbtier_namespace> delete secrets <cndbtier-https-cert-
cred> <cndbtier-https-cert-file>
```

For example:

```
$ export OCCNE_NAMESPACE= "cluster1"
$ kubectl get secrets -n ${OCCNE_NAMESPACE}
$ kubectl delete secrets cndbtier-https-cert-cred cndbtier-https-cert-file
-n ${OCCNE_NAMESPACE}
```

4. Create the new HTTPS secret using the new server certificate created in Step 1, on all the sites where you want to modify the certificates. For more information about creating HTTPs secrets, see the "*Creating Secrets*" section in *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide*.
5. Run the following commands to restart the replication service of each site in which you have modified the certificate:

```
$ kubectl get pods -n <cndbtier_namespace>
$ kubectl delete pod <replication-svc-pod> -n <cndbtier_namespace>
```

For example:

```
$ kubectl get pods -n cluster1
$ kubectl -n cluster1 delete pod mysql-cluster-cluster1-cluster2-
replication-svc-f897657c5-d9vqw
```

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:
      For example,

```
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 `occne-remote-server-username-secret` secret by running the following command:

```
$ kubectl -n <namespace> delete secret occne-remote-server-username-
secret
```

      Where, `<namespace>` is the namespace of cnDBTier.

      For example,

```
$  kubectl  -n occne-cndbtier  delete secret occne-remote-server-
username-secret
```

   b. Recreate the secret with a new username by running the following command:

```
$  kubectl -n <namespace> create secret generic occne-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 occne-cndbtier  create secret generic occne-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 `occne-remoteserver-privatekey-secret` secret by running the following command:

      ```
      $ kubectl -n <namespace> delete secret occne-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        |
ACTIVE  |        NULL | cluster1  |      1000 | 2022-04-04 19:10:44 |
| cluster2         |             2001 |       4597 | 10.233.9.35         |
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
```

4. 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)
```

5. 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
```

4. 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)
```

5. 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 REPLICA 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 'Password';
   mysql> ALTER USER IF EXISTS occnerepluser IDENTIFIED WITH
   'caching_sha2_password' BY 'Password';
   mysql> ALTER USER IF EXISTS root@localhost IDENTIFIED WITH
   'caching_sha2_password' BY 'Password';
   mysql> ALTER USER IF EXISTS healthchecker@localhost IDENTIFIED WITH
   'caching_sha2_password' BY 'Password';
   mysql> ALTER USER IF EXISTS root IDENTIFIED WITH 'caching_sha2_password'
   BY 'Password';
   mysql> ALTER USER IF EXISTS <NF_USER> IDENTIFIED WITH
   'caching_sha2_password' BY 'Password';
   ```

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 'Password';
   mysql> ALTER USER IF EXISTS occnerepluser IDENTIFIED WITH
   'mysql_native_password' BY 'Password';
   mysql> ALTER USER IF EXISTS root@localhost IDENTIFIED WITH
   'mysql_native_password' BY 'Password';
   mysql> ALTER USER IF EXISTS healthchecker@localhost IDENTIFIED WITH
   'mysql_native_password' BY 'Password';
   mysql> ALTER USER IF EXISTS <NF_USER> IDENTIFIED WITH
   'caching_sha2_password' BY 'Password';
   ```