Oracle® Communications

Network Analytics Data Director Disaster Recovery Guide

Release 22.0.0
F73001-01
December 2022

Oracle Communications Network Analytics Data Director Disaster Recovery Guide, Release 22.0.0

F73001-01

Contents

My Oracle Support (MOS)

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 provides information about the acronyms and the terminology used in the document.

Table Acronyms

Acronym	Description
CLI	Command Line Interface
MPS	Messages Per Second
NRF	Network Repository Function
OHC	Oracle Help Center
OSDC	Oracle Service Delivery Cloud
SCP	Service Communication Proxy
SVC	Services
URI	Uniform Resource Identifier
KPI	Key Performance Indicator
CNE	Cloud Native Environment
MPS	Messages Per Second

What's New in This Guide

This section lists the documentation updates for Release *22.0.0* in *Oracle Communications Data Director Outbound Interface Specification Document*.

Release 22.0.0 - F52364-03, December 2022

This is the first release of the document.

1 Introduction

This document describes procedures to perform the backup and restore for the Oracle Communications Network Analytics Data Director (OCNADD) deployment. The backup and restore procedures will be used in the disaster recovery of the OCNADD. The OCNADD operators can take only the OCNADD instance specific database and required OCNADD Kafka metadata backup and restore them either on the same or a different Kubernetes cluster.

The backup and restore procedures are helpful in the following scenarios:

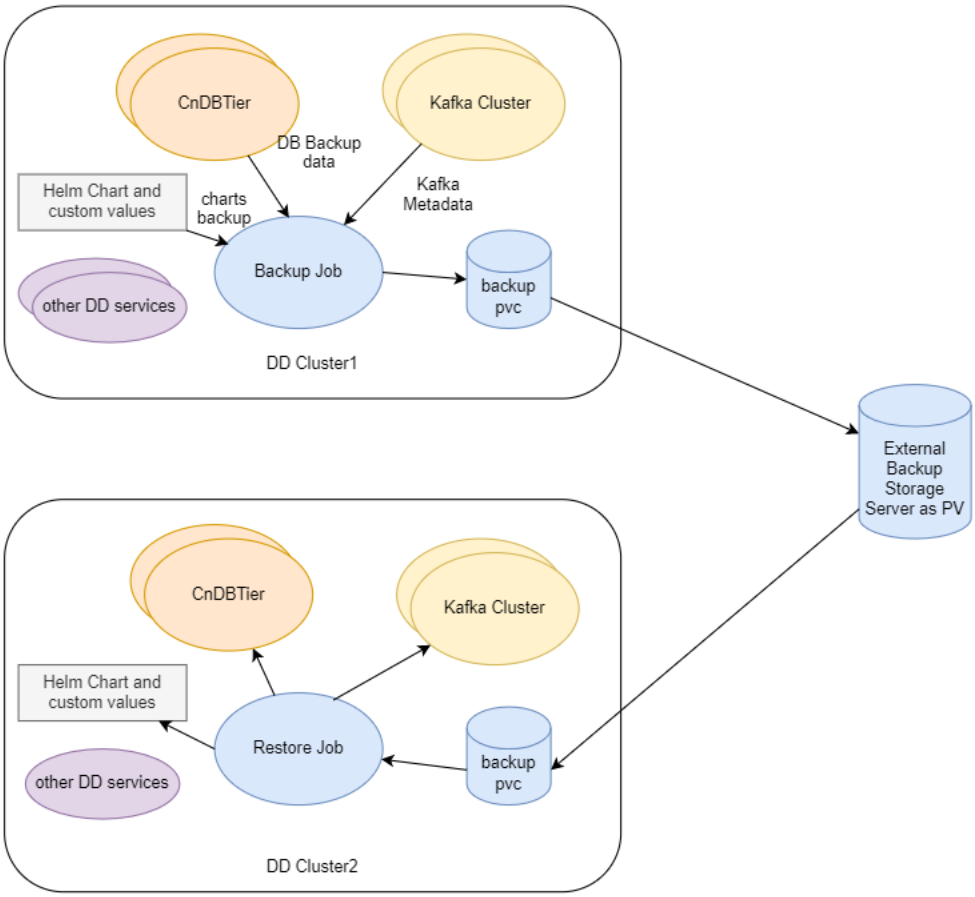
- OCNADD disaster recovery
- OCNADD cluster migration
- OCNADD setup replication from production to development or staging
- OCNADD cluster upgrade to new CNE version or K8s version

The OCNADD backup contains the following data:

- OCNADD database(s) backup
- OCNADD Kafka metadata backup including the topics and partitions information

> **Note:**
>
> If the deployed helm charts and the customized `ocnadd/values.yml` for the current deployment are stored in the customer helm or artifact repository, the helm chart and `values.yml` backup are not required.

Figure 1-1 OCNADD Backup and Restore



OCNADD Database(s) Backup

The OCNADD database consists of the following:

- **Configuration data**: This data is exclusive for the given OCNADD instance. Therefore, an exclusive logical database is created and used by an OCNADD instance to store its configuration data and operator driven configuration. Operators can configure the OCNADD instance specific configurations using the Configuration UI service through the Cloud Native Core (CNC) Console.
- **Alarm configuration data**: This data is also exclusive to the given OCNADD instance. Therefore, an exclusive logical database is created and used by an OCNADD Alarm service instance to store its alarm configuration and alarms.
- **Health monitoring data**: This data is also exclusive to the given OCNADD instance. Therefore, an exclusive logical database is created and used by an OCNADD Health monitoring service instance to store the health profile of various other services.

The database backup job uses the mysqldump utility.

The Scheduled regular backups helps in:

- Restoring the stable version of the data directory databases

- Minimize significant loss of data due to upgrade or rollback failure
- Minimize loss of data due to system failure
- Minimize loss of data due to data corruption or deletion due to external input
- Migration of the database information from one site to another site

OCNADD Kafka Metadata Backup

The OCNADD Kafka metadata backup contains the following information:

- Created topics information
- Created partitions per topic information

Prerequisites

Before you run any disaster recovery procedure, ensure that the following prerequisites are met:

- cnDBTier must be in a healthy state and available on a new or newly installed site where the restore needs to be performed
- Automatic backup should be enabled for OCNADD.
- Docker images used during the last installation or upgrade must be retained in the external data storage or repository
- The `ocnadd/values.yaml` used at the time of OCNADD deployment must be retained. If the `ocnadd/values.yaml` file is not retained, it is required to be recreated manually. This task increases the overall disaster recovery time.

> **Important:**
>
> Do not change DB Secret or CnDBTier MySQL FQDN or IP or PORT configurations during backup and restore.

Disaster Recovery Impact Areas

The following table shares information about impact of OCNADD disaster recovery scenarios:

Table 1-1 OCNADD Disaster Recovery Scenarios Impact Information

Scenario	Requires Disaster Recovery or Reinstallation of CNE?	Requires Disaster Recovery or Reinstallation of cnDBTier?	Requires Disaster Recovery or Reinstallation of Data Director?
Scenario 1: Deployment Failure Recovering OCNADD when its deployment is corrupted	No	No	Yes

Table 1-1 (Cont.) OCNADD Disaster Recovery Scenarios Impact Information

Scenario	Requires Disaster Recovery or Reinstallation of CNE?	Requires Disaster Recovery or Reinstallation of cnDBTier?	Requires Disaster Recovery or Reinstallation of Data Director?
Scenario 2: cnDBTier Corruption	No	Yes	No However, it requires to restore the databases from backup and Helm upgrade of the same OCNADD version to update the OCNADD configuration. For example, change in cnDBTier service information, such as cnDB endpoints, DB credentials, and so on.
Scenario 3: Database Corruption Recovering from corrupted OCNADD configuration database	No	No	No However, it requires to restore the databases from old backup.
Scenario 4: Site Failure Complete site failure due to infrastructure failure, for example, hardware, CNE, and so on.	Yes	Yes	Yes
Scenario 5: Backup Restore in a Different Cluster Obtaining the OCNADD backup from one deployment site and restore it to another site	No	No	No However, it requires to restore the database.

References

While performing disaster recovery procedures, you may refer to the procedures defined in the following existing documents:

- *Oracle Communications Network Analytics Data Director User Guide*
- *Oracle Communications Network Analytics Data Director Troubleshooting Guide*
- *Oracle Communications Network Analytics Data Director Installation Guide*
- *Oracle Communications Cloud Native Environment and Installation Guide*
- *Oracle Communications Cloud Native Core Disaster Recovery Guide*
- *Oracle Communications Cloud Native DBTier Installation Guide*

2
Backup and Restore Flow

Important:

1. It is recommended to keep the backup storage in the external storage that can be shared between different clusters. This is required, so that in an event of a disaster, the backup is accessible on the other clusters. The backup job should create a PV/PVC from the external storage provided for the backup.
2. In case the external storage is not made available for the backup storage, the customer should take care to copy the backups from the associated backup PV in the cluster to the external storage. The security and connectivity to the external storage should be managed by the customer. To copy the backup from the backup PV to the external server, follow Verifying OCNADD Backup.
3. The restore job should have access to the external storage so that the backup from the external storage can be used for the restoration of the OCNADD services. In case the external storage is not available, the backup should be copied from the external storage to the backup PV in the new cluster. For information on the procedure, see Verifying OCNADD Backup.

Note:

At a time, only one among the three backup jobs (ocnaddmanualbackup, ocnaddverify or ocnaddrestore) can be running. If any existing backup job is running, that job needs to be deleted to spawn the new job.

```
kubectl delete job.batch/<ocnadd*> -n <namespace>

where namespace = Namespace of OCNADD deployment
      ocnadd* = Running jobs in the namespace (ocnaddmanualbackup,
ocnaddverify or ocnaddrestore)

Example:
      kubectl delete job.batch/ocnaddverify -n ocnadd-deploy
```

Backup

1. The OCNADD backup is managed using the backup job created at the time of installation. The backup job runs as a cron job and takes the daily backup of the following:
   - OCNADD databases for configuration, alarms, and health monitoring

- OCNADD Kafka metadata including topics and partitions, which are already created

2. The automated backup job spawns as a container and takes the backup at the scheduled time. The backup file in the format `OCNADD_Backup_DD-MM-YYYY_hh-mm-ss.zip` is created and stored in the PV mounted on the path `/work-dir/backup` by the backup container.
3. On-demand backup can also be created by creating the backup container. For more information, see Performing OCNADD Manual Backup.
4. The backup can be stored on external storage.

Restore

1. The OCNADD restore job must have access to the backups from the backup PV/PVC.
2. The restore uses the latest backup file available in the backup storage if the BACKUP_FILE argument is not given.
3. The restore job performs the restore in the following order:
   a. Restore the OCNADD database(s) on the CnDBTier.
   b. Restore the Kafka metadata.

3
OCNADD Backup

The OCNADD Backup is of two types:

- Automated Backup
- Manual Backup

Automated Backup

- This is managed by the automated K8s job configured during the installation of the OCNADD. For more information, see "Create Data Director Backup Job" in *Oracle Communications Network Analytics Data Director Installation Guide*.
- It is a scheduled job and runs daily at the configured time to collect the OCNADD backup and creates the backup file `OCNADD_BACKUP_DD-MM-YYYY_hh-mm-ss.tgz`.

Manual Backup

- This is managed by an on-demand job.
- A new K8s job will be created on executing the Performing OCNADD Manual Backup procedure.
- The job completes after taking the backup. Follow Verifying OCNADD Backup procedure to verify the generated backup.

4
Performing OCNADD Backup Procedures

Performing OCNADD Manual Backup

Perform the following steps to take the manual backup:

1. Go to `custom-templates` folder inside the extracted ocnadd-release package and update the `ocnadd_manualBackup.yaml` file with the following information:
   a. Values for BACKUP_DATABASES can be set to ALL (i.e. healthdb_schema, configuration_schema, and alarm_schema) or the individual DB names can also be passed. By default, the value is 'ALL'.
   b. Update other values as follows:

```
apiVersion:batch/v1
kind:Job
metadata:
  name:ocnaddmanualbackup
  namespace:ocnadd-deploy     #---> update the namespace
  spec:
     template:
           metadata:
              name:ocnaddmanualbackup
           spec:
            volumes:
            -name:backup-vol
            persistentVolumeClaim:
                claimName:backup-mysql-pvc
            -name:config-vol
             configMap:
               name:config-backuprestore-scripts
            serviceAccountName:ocnadd-deploy-gitlab-admin      #--->
update the service account name. Format:<namespace>-gitlab-admin
            containers:
           -name:ocnaddmanualbackup
            image:<repo-path>/ocnaddbackuprestore:1.0.0        #--->
update repository path
            volumeMounts:
              -mountPath:"work-dir"
               name:backup-vol
              -mountPath:"config-backuprestore-scripts"
               name:config-vol
            env:
              -name:HOME
               value:/tmp
              -name:DB_USER
               valueFrom:
```

```
                    secretKeyRef:
                      name:db-secret
                      key:MYSQL_USER

                  -name:DB_PASSWORD
                   valueFrom:
                      secretKeyRef:
                        name:db-secret
                        key:MYSQL_PASSWORD

                   -name:BACKUP_DATABASES
                    value:ALL
            command:
            - /bin/sh
            - -c
            - |
              cp /config-backuprestore-scripts/*.sh /tmp
              chmod +x /tmp/*.sh
              mkdir/work-dir/backup
              touch/work-dir/backup/DB_BACKUP_$(date +%d-%m-%Y_%H-
%M-%S).sql
              echo "Executing manual backup script"
              bash /tmp/backup.sh $BACKUP_DATABASES
             ls -lh /work-dir/backup
   restartPolicy:Never
status:{}
```

2. Execute the below command to run the job:

```
kubectl create -f ocnadd_manualBackup.yaml
```

Verifying OCNADD Backup

> **Caution:**
>
> The connectivity between the external storage via either PV/PVC or network connectivity must be ensured.

To verify the backup, perform the following steps:

1. Go to the `custom-templates` folder inside the extracted ocnadd-release package and update the `ocnadd_verify_backup.yaml` file with the following information:
   a. Sleep time is configurable, update it if required. (default sleep is 10m)
   b. Update other values as follows:

```
apiVersion: batch/v1
kind: Job
metadata:
  name: ocnaddverify
```

```
  namespace: ocnadd-deploy          #---> update the namespace
spec:
  template:
    metadata:
      name: ocnaddverify
    spec:
      volumes:
      - name: backup-vol
        persistentVolumeClaim:
            claimName: backup-mysql-pvc
      - name: config-vol
        configMap:
          name: config-backuprestore-scripts
      serviceAccountName: ocnadd-deploy-gitlab-admin      #--->
update the service account name. Format:<namespace>-gitlab-admin
      containers:
      - name: ocnaddverify
        image: <repo-path>/ocnaddbackuprestore:1.0.0      #--->
update repository path
        volumeMounts:
          - mountPath: "work-dir"
            name: backup-vol
          - mountPath: "config-mysql-scripts"
            name: config-vol
        env:
          - name: HOME
            value: /tmp
          - name: DB_USER
            valueFrom:
              secretKeyRef:
                name: db-secret
                key: MYSQL_USER
          - name: DB_PASSWORD
            valueFrom:
              secretKeyRef:
                name: db-secret
                key: MYSQL_PASSWORD
        command:
        - /bin/sh
        - -c
        - |
          cp /config-backuprestore-scripts/*.sh /tmp
          chmod +x /tmp/*.sh
          echo "Checking backup path"
          ls -lh /work-dir/backup
          sleep 10m
      restartPolicy: Never
status: {}
```

2. Run the following command to run the verify job for verifying the backup generated at the mounted PV by running inside the running container:

```
kubectl create -f ocnadd_verify_backup.yaml
```

3. If the external storage is used as PV/PVC, get inside the ocnaddverify container using the following commands:
   a. `kubectl exec -it <verify_pod> -n <ocnadd namespace> -- bash`
   b. Change the directory to `/work-dir/backup` and inside the latest backup folder OCNADD_BACKUP_DD-MM-YYYY_hh-mm-ss.
   c. Verify the DB backup and Kafka metadata backup files

Obtain the OCNADD Backup files

1. Run the Verifying OCNADD Backup procedure to spawn the `verify_pod`.
2. Get into the running `ocnaddverify` pod to identify and retrieve the desire backup folder with the following commands:
   a. Access the pod

      ```
      kubectl exec -it <ocnaddverify-*> -n <namespace> -- bash
      ```

      where namespace = namespace of ocnadd

      ocnaddverify-* = is the verify pod in the namespace
   b. Change the directory to /work-dir/backup and identify the backup folder "OCNADD_BACKUP_DD-MM-YYYY_hh-mm-ss"
   c. Exit from the ocnaddverify pod
3. Copy the backup from the pod to the local bastion server by copying the OCNADD Database and kafka_metadata files from the ocnaddverify pod with the following commands:
   a. Onbastion host create a folder with the name similar to backup folder "OCNADD_BACKUP_DD-MM-YYYY_hh-mm-ss"
   b. Change to the folder created in step a.
   c. Use below command to copy backup files from the ocnaddverify pod To copy ocnadd Database backup:

      ```
      kubectl exec -n <namespace>  <verify_pod> -- cat/work-dir/backup/
      OCNADD_BACKUP_DD-MM-YYYY_hh-mm-ss/DB_BACKUP_DD-MM-YYYY_hh-mm-
      ss.sql  > DB_BACKUP_DD-MM-YYYY_hh-mm-ss.sql
      ```

      Example:

      ```
      kubectl exec -n ocnadd ocnaddverify-ncczz -- cat/work-dir/backup/
      OCNADD_BACKUP_11-11-2022_16-54-05/
      DB_BACKUP_11-11-2022_16-54-05.sql  >
      DB_BACKUP_11-11-2022_16-54-05.sql
      ```

      To copy ocnadd Kafka metadata backup:

      ```
      kubectl exec -n <namespace>  <verify_pod> -- cat/work-dir/backup/
      ```

```
OCNADD_BACKUP_DD-MM-YYYY_hh-mm-ss/kafka_metadata_DD-MM-YYYY_hh-mm-
ss.txt  > kafka_metadata_DD-MM-YYYY_hh-mm-ss.txt
```

Example:

```
kubectl exec -n ocnadd ocnaddverify-ncczz -- cat/work-dir/backup/
OCNADD_BACKUP_11-11-2022_16-54-05/
kafka_metadata_11-11-2022_16-54-05.txt  >
kafka_metadata_11-11-2022_16-54-05.txt
```

where, namespace = namespace of ocnadd

verify_pod = is the verify pod in the namespace

Copy and Restore the OCNADD backup

1. Get the ocnadd backup folder.
2. Execute the Verifying OCNADD Backup procedure to spawn the verify_pod.
3. Create a folder inside the verify_pod under directory path:/work-dir/backup/ with the name similar to the backup folder "OCNADD_BACKUP_DD-MM-YYYY_hh-mm-ss" .
   a. Access the pod with "kubectl exec -it <ocnaddverify-*> -n <namespace> -- bash".
   b. Create backup folder using command "mkdir -p /work-dir/backup/ OCNADD_BACKUP_DD-MM-YYYY_hh-mm-ss".
4. Copy the backup files from the local bastion server to the running ocnaddverify pod.
   a. Go to the ocnadd backup directory
   b. Use the following command to copy backup files inside the ocnaddverify pod:

      ```
      cat DB_BACKUP_DD-MM-YYYY_hh-mm-ss.sql | kubectl exec -i -n
      <namespace> <verify_pod> -- bash -c "cat > /work-dir/backup/
      OCNADD_BACKUP_DD-MM-YYYY_hh-mm-ss/DB_BACKUP_DD-MM-YYYY_hh-mm-ss.sql
      ```

      Example:

      ```
      cat DB_BACKUP_11-11-2022_16-54-05.sql | kubectl exec -i -n ocnadd
      ocnaddverify-ncczz -- bash
            -c "cat >
            /work-dir/backup/OCNADD_BACKUP_11-11-2022_16-54-05/
      DB_BACKUP_11-11-2022_16-54-05.sql
      ```

   c. To copy ocnadd Kafka metadata backup:

      ```
      cat kafka_metadata_DD-MM-YYYY_hh-mm-ss.txt | kubectl exec -i -n
      <namespace> <verify_pod> -- bash -c "cat  > /work-dir/backup/
      OCNADD_BACKUP_DD-MM-YYYY_hh-mm-ss/kafka_metadata_DD-MM-YYYY_hh-mm-
      ss.txt
      ```

Example:

```
cat kafka_metadata_11-11-2022_16-54-05.txt | kubectl exec -i -n
ocnadd ocnaddverify-ncczz -- bash -c "cat > /work-dir/backup/
OCNADD_BACKUP_11-11-2022_16-54-05/
kafka_metadata_11-11-2022_16-54-05.txt"
```

5. Verify the backup has been copied by getting into the ocnaddverify pod and path: /work-dir/backup/OCNADD_BACKUP_DD-MM-YYYY_hh-mm-ss
6. Restore OCNADD using the procedure defined in Create OCNADD Restore Job.
7. Restart the ocnaddconfiguration, ocnaddalarm, and ocnaddhealthmonitoring pods.

5
Disaster Recovery Scenarios

This chapter describes the disaster recovery procedures for different recovery scenarios.

Scenario 1: Deployment Failure

This section describes how to recover OCNADD when the OCNADD deployment corrupts.

For more information, see Restoring OCNADD.

Scenario 2: cnDBTier Corruption

This section describes how to recover the cnDBTier corruption. For more information, see *Oracle Communication Cloud Native Core DBTier Disaster Recovery Guide*. After the cnDBTier recovery, restore the OCNADD database from the previous backup.

To restore the OCNADD database, execute the procedure Create OCNADD Restore Job by setting BACKUP_ARG to DB.

Scenario 3: Database Corruption

This section describes how to recover from the corrupted OCNADD database.

Perform the following steps to recover the OCNADD configuration database (DB) from the corrupted database:

1. Retain the working ocnadd backup by following Obtain the OCNADD Backup files procedure.
2. Drop the existing Databases by accessing the MySql DB.
3. Perform the Copy and Restore the OCNADD backup procedure to restore the backup.

Scenario 4: Site Failure

This section describes how to perform disaster recovery when the OCNADD site has software failure.

Perform the following steps in case of a complete site failure:

1. Run the Cloud Native Environment (CNE) installation procedure to install a new Kubernetes cluster. For more information, see *Oracle Communications Cloud Native Environment Installation Guide*.
2. Run the cnDBTier installation procedure. For more information, see *Oracle Communications Cloud Native Core DBTier Installation and Upgrade Guide*.
3. For cnDBTier disaster recovery, take a data backup from an older site and restore it to a new site. For more information about cnDBTier backup, see "Create On-demand

Database Backup" and to restore the database to a new site, see "Restore DB with Backup" in *Oracle Communications Cloud Native Core DBTier Disaster Recovery Guide*.

4. Restore OCNADD. For more information, see Restoring OCNADD.

Scenario 5: Backup Restore in a Different Cluster

This section describes how to obtain the OCNADD backup from one deployment site and restore it to another site.

Perform the following steps:

1. Obtain the OCNADD Backup files
2. Copy and Restore the OCNADD backup

6
Restoring OCNADD

Perform this procedure to restore OCNADD when a disaster event has occurred or deployment is corrupted.

> **Note:**
>
> This procedure expects the OCNADD backup folder is retained.

1. Get the retained backup folder "OCNADD_BACKUP_DD-MM-YYYY_hh-mm-ss".
2. Get the helm charts that was used in the earlier deployment.
3. Run the following command to uninstall the corrupted OCNADD deployment:
   a. helm uninstall <release_name> --namespace <namespace>
   b. Where,
      - <release_name> is a name used to track this installation instance.
      - <namespace> is the namespace of OCNADD deployment.

      Example:

      helm uninstall ocnadd --namespace ocnadd-ns
4. Install OCNADD using the helm charts that was used in the earlier deployment. For information about installing OCNADD using Helm, refer to *Oracle Communications Network Analytics Data Director Installation and Upgrade Guide*.
5. To verify whether OCNADD installation is complete, perform the "Verify Data Director Installation" procedure as described in the *Oracle Communications Network Analytics Data Director Installation and Upgrade Guide.*
6. Follow procedure Copy and Restore the OCNADD backup

7
Create OCNADD Restore Job

1. Restore the OCNADD database by executing the following steps:
   a. Go to the custom-templates folder inside the extracted ocnadd-release package and update the ocnadd_restore.yaml file based on the restore requirements:
      i. The value of BACKUP_ARG can be set to DB, KAFKA, and ALL. By default, the value is 'ALL'.
      ii. The value of BACKUP_FILE can be set to folder name which needs to be restored, if not mentioned the latest backup will be used.
      iii. Update other values as below:

```
apiVersion:batch/v1
kind:Job
metadata:
   name:ocnaddrestore
   namespace:ocnadd-deploy    #---> update the namespace
spec:
   template:
       metadata:
          name:ocnaddrestore
       spec:
         volumes:
          -name:backup-vol
           persistentVolumeClaim:
             claimName:backup-mysql-pvc
          -name:config-vol
            configMap:
              name:config-backuprestore-scripts
              serviceAccountName:ocnadd-deploy-gitlab-admin
#---> update the service account name. Format:<namespace>-gitlab-
admin
              containers:
              -name:ocnaddrestore
              image:<repo-path>/ocnaddbackuprestore:1.0.0      #---
>update repository path
              volumeMounts:
                    -mountPath:"work-dir"
                     name:backup-vol
                    -mountPath:"config-backuprestore-
scripts"
                     name:config-vol

                env:
                 -name:HOME
                  value:/tmp
                 -name:DB_USER
```

```
                        valueFrom:
                           secretKeyRef:
                              name:db-secret
                               key:MYSQL_USER

                       -name:DB_PASSWORD
                          valueFrom:
                            secretKeyRef:
                               name:db-secret
                               key:MYSQL_PASSWORD

                        -name:BACKUP_ARG
                            value:ALL
                        -name:BACKUP_FILE
                           value:
                    command:
                    - /bin/sh
                    - -c
                    - |
                    cp /config-backuprestore-scripts/*.sh /tmp
                    chmod +x /tmp/*.sh
                    echo "Executing restore script"
                    ls -lh /work-dir/backup
                    bash /tmp/
restore.sh $BACKUP_ARG $BACKUP_FILE
                    sleep 15m

                restartPolicy:Never
status:{}
```

2. Execute the below command to run the restore job:

   ```
   kubectl create -f ocnadd_restore.yaml
   ```

3. Restart the ocnaddconfiguration, ocnaddalarm, and ocnaddhealthmonitoring pods once the restore job gets completed.

> **Note:**
>
> If the backup is not available for the mentioned date, the pod will be in an error state, notifying the Backup is not available for the given date: $DATE, in such case provide the correct backup dates and repeat the procedure.

8
Configuring Backup and Restore Parameters

To configure backup and restore parameters, configure the parameters listed in the following table:

Table 8-1 Backup and Restore Parameters

Parameter Name	Data Type	Range	Default Value	Mandatory(M)/ Optional(O)/ Conditional(C)	Description
BACKUP_STORAGE	STRING	-	20Gi	M	Persistent Volume storage to keep the OCDD backups
MYSQLDB_NAMESPACE	STRING	-	occne-cndbtierone	M	Mysql Cluster Namespace
BACKUP_CRONEXPRESSION	STRING	-	0 8 * * *	M	Cron expression to schedule backup cronjob
STORAGE_CLASS	STRING	-	standard	M	Dynamically provision of PV in Kubernetes cluster
BACKUP_ARG	STRING	-	ALL	M	Kafka , DB or ALL backup
BACKUP_FILE	STRING	-	-	O	Backup folder name which needs to be restored
BACKUP_DATABASES	STRING	-	ALL	M	Individual databases or all databases backup that need to be taken