Oracle® Communications

Operations Services Overlay Installation and Upgrade Guide

Release 24.3.1
G10884-03
March 2025

Oracle Communications Operations Services Overlay Installation and Upgrade Guide, Release 24.3.1

G10884-03

Contents

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:

1. Select **2** for New Service Request.
2. Select **3** for Hardware, Networking and Solaris Operating System Support.
3. Select one of the following options:
    - 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**.

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

Acronym	Description
alm	AlertManager
cm-r	Configmap-reload
CNC Console	Cloud Native Configuration Console
CNE	Oracle Communications Cloud Native Core, Cloud Native Environment
CSAR	Cloud Service Archive
GUI	Graphical User Interface
ONAP	Open Network Automation Platform
OSO	Operations Services Overlay
OSDC	Oracle Software Download Center
prom	Prometheus
svr	Server
TSDB	Time Series Database

What's New in This Guide

This section introduces the documentation updates for release 24.3.x.

Release 24.3.1- G10884-03, March 2025

General Updates

- Removed the Time Series Database (TSDB) Snapshots section as this feature is not supported in this release.
- Removed the `extraFlags` parameter from the Prometheus Configuration Parameters section as the Time Series Database (TSDB) Snapshots feature is not supported in this release.
- Updated the Oracle Error Correction Policy section with the latest patch details.

Release 24.3.1- G10884-02, January 2025

General Updates

- Updated the release version to 24.3.1 in the entire document.
- Added Time Series Database (TSDB) Snapshots section with the procedure for capturing snapshots in TSDB.

Installation Updates

- Changed the common_pod version from `24_3_common_pod:24.3.0` to `24_3_common_oso:24.3.0` in the Prerequisites section.
- Replaced the version of common_pod from `24_3_common_pod` to `24_3_common_oso` in the OSO Configuration Parameters section.
- Added the `extraFlags` parameter in the Prometheus Configuration Parameters section.

Upgrade, Rollback, and Uninstall Updates

- Added a note in the Upgrading OSO section to mention that OSO does not support upgrade in this release.
- Added a note in the Rolling Back OSO section to mention that OSO does not support rollback in this release.

Release 24.3.0- G10884-01, October 2024

General Updates

Updated the release version to 24.3.0 in the entire document.

Added the procedure for prometheus vertical scaling in the Prometheus Vertical Scaling section.

Renamed the existing `ocoso_csar_24_3_1_0_0_custom_values.yaml` file with the `ocoso_csar_24_3_1_0_0_prom_custom_values.yaml` and `ocoso_csar_24_3_1_0_0_alm_custom_values.yaml` in the following sections:

- Installing OSO
- Installing OSO Using CSAR

- Customizing OSO
- Naming Conventions and Limits

Installation Updates

- Updated the Prerequisites section with the following:
  - Changed the common_pod version from `24_2_common_pod` to `24_3_common_pod`.
  - Added the resource requirements in the Prerequisites section.
- Replaced the version of common_pod from `24_2_common_pod` to `24_3_common_pod` in the OSO Configuration Parameters section.
- Added the following sections for categorizing the alertmanager and Prometheus parameters:
  - Alertmanager Configuration Parameters
  - Prometheus Configuration Parameters

Upgrade, Rollback, and Uninstall Updates

- Added a note in the Upgrading OSO section to mention that OSO does not support upgrade in this release.
- Added a note in the Rolling Back OSO section to mention that OSO does not support rollback in this release.

1 Introduction

This guide describes how to install or upgrade Oracle Communications Operations Services Overlay (OSO) in a cloud native environment.

> ⚠ **Caution**
>
> User, computer and applications, and character encoding settings may cause an issue when copy-pasting commands or any content from PDF. PDF reader version also affects the copy-pasting functionality. It is recommended to verify the pasted content especially when the hyphens or any special characters are part of the copied content.

1.1 Overview

Oracle Communications Operations Services Overlay is a key component of the 5G Service Based Architecture (SBA).

Oracle Communications Operations Services Overlay installs and configures common operation services. For example, you can install and configure Prometheus and its components like AlertManager in a previously installed Kubernetes cluster.

Oracle Communications Operations Services Overlay is an independent deliverable distinct from Oracle Communications Cloud Native Core, Cloud Native Environment (CNE).

1.2 References

Following are the reference documents while deploying OSO:

- *Oracle Communications Operations Services Overlay User Guide*
- *Oracle Communications Operations Services Overlay Network Impact Report*

1.3 Oracle Error Correction Policy

The table below outlines the key details for the current and past releases, their General Availability (GA) dates, the latest patch versions, and the end dates for the Error Correction Grace Period.

Table 1-1 Error Correction Policy

Release Number	General Availability (GA) Date	Latest Patch Version	Error Correction Grace Period End Date
3.24.3	October 2024	24.3.1	October 2025
3.24.2	July 2024	24.2.0	July 2025
3.24.1	April 2024	24.1.0	April 2025
3.23.4	December 2023	23.4.0	December 2024

> **Note**
>
> For a release, Sev1 and Critical Patch Unit (CPU) patches are supported for 12 months. For more information, see Oracle Communications Cloud Native Core and Network Analytics Error Correction Policy.

1.4 Oracle Open Source Support Policies

Oracle Communications Cloud Native Core uses open source technology governed by the Oracle Open Source Support Policies. For more information, see Oracle Open Source Support Policies.

2
Installing OSO

This chapter provides information about installing Oracle Communications Operations Services Overlay (OSO) in a cloud native environment.

> **Note**
>
> OSO 24.3.0 is an installation only release. OSO does not support upgrade from 24.2.x and 24.1.x to 24.3.0

OSO, packaged as a Cloud Service Archive (CSAR) file, is a ZIP file that contains the following components:

- All the required OSO images, including opensource software as a tar file.
- All the required OSO Helm charts.
- The following custom values.yaml files:
  - `ocoso_csar_24_3_1_0_0_prom_custom_values.yaml` is used for configuring the Prometheus parameters
  - `ocoso_csar_24_3_1_0_0_alm_custom_values.yaml` is used for configuring the Alertmanager parameters.

> **Note**
>
> The README doc contains details to populate mandatory values in the `ocoso_csar_24_3_1_0_0_prom_custom_values.yaml` and `ocoso_csar_24_3_1_0_0_alm_custom_values.yaml` files. For more information about the configuration parameters, see OSO Configuration Parameters.

2.1 Prerequisites

Before installing and configuring OSO, ensure that the following prerequisites are met:

1. CSAR package is downloaded.
2. Unzip and TAR utilities are installed.
3. Docker or Podman is installed, and you must be able to run the docker or podman commands.
4. Helm is installed with the minimum supported Helm version (v3.15.2).
5. kubectl is installed.
6. A central repository is made available for all images, binaries, helm charts, and so on, before running this procedure.
7. The following images are populated in the registry on the repo server:
   - `occne.io/oso/prometheus:v2.52.0`

- `occne.io/oso/alertmanager:v0.27.0`
- `occne.io/oso/configmapreload:v0.13.0`
- `occne.io/occne/24_3_common_oso:24.3.1`

All the above images are packed into tar format and will be present in OSO CSAR under (Artifacts/Images) folder. You can use the following commands to load these images into your cluster's registry.

```
$ docker/podman load -i <image-name>.tar
$ docker/podman tag <image-url> <registry-address>:<port>/image-url
$ docker/podman push <registry-address>:<port>/image-url
```

2.2 Resource Utilization

The use of resources such as CPU and RAM by the OSO services are constrained to ensure that the CPU and RAM do not consume excess resources that could be used by other applications.

During the deployment of the services, each service gets an initial CPU and RAM allocation. Each service is allowed to consume the resources to a specified upper limit while it continues to run. The initial allocation limit and the upper limit are set to the same value for services where the resource consumption limit remains the same as the initial allocation or in a case where increasing the CPU or RAM limits underneath a running application can cause service disruption. The resource requests and limits are as follows:

Table 2-1 CPU and RAM Resource Requests and Limits

Service	CPU Initial Request(m)	CPU Limit(m)	RAM Initial Request(Mi)	RAM Limit(Mi)	Instances
Prometheus	2000	2000	4096	4096	2
Prometheus AlertManager	20	20	64	64	2

The overall observability services resource usage varies on each worker node. The observability services listed in the table are distributed evenly across all worker nodes in the Kubernetes cluster.

2.3 Installing OSO Using CSAR

Open Network Automation Platform (ONAP) compliant orchestrator uses CSAR format to onboard, validate, and install OSO. However, in the absence of an orchestrator, manual installation is possible using the CSAR file contents.

Perform the following steps to install OSO using CSAR artifacts:

1. Download the OSO CSAR zip file from My Oracle Support (MOS).
2. Extract the CSAR zip file:

   ```
   $ unzip <OSO CSAR package>
   ```

3. Upload all the artifacts present in the `Artifacts/Images` folder to the configured repository. For more information about the artifacts, see the Prerequisites section.

4. From 24.3.0 onwards, the existing `ocoso_csar_24_3_1_0_0_custom_values.yaml` file is split into two different files:
   - `ocoso_csar_24_3_1_0_0_prom_custom_values.yaml` is used for configuring the Prometheus parameters
   - `ocoso_csar_24_3_1_0_0_alm_custom_values.yaml` is used for configuring the Alertmanager parameters.

   These files are required to be populated and installed separately. Update the custom_values.yaml files inside `Artifacts/Images` directory with the required values as mentioned in OSO Configuration Parameters.
5. Enable IPv6 Dualstack.
   a. **To enable IPv6 DualStack in Prometheus**
      - Enable IPv6 Dualstack in the `ocoso_csar_24_3_1_0_0_prom_custom_values.yaml`. Search for the following comment in the yaml file and uncomment the following four lines after this comment.

        ```
        # Custom section to enable IPV6, Uncomment below section in order
        to enable Dualstack OSO
              having both Ipv4 and Ipv6 addresses
        #ipFamilies:
        #- IPv4
        #- IPv6
        #ipFamilyPolicy: PreferDualStack
        ```

      - Change the service type from ClusterIP to LoadBalancer to assign IPv6.
      - Save the file and proceed with the normal installation.

   b. **To enable IPv6 DualStack in AlertManager** Replace "ENABLE_DUAL_STACK" with "true" in below section inside the `ocoso_csar_24_3_1_0_0_alm_custom_values.yaml` file before installing alertmanager.

      ```
      # ip dual stack
        ipDualStack:
          enabled: ENABLE_DUAL_STACK
          ipFamilies: ["IPv6", "IPv4"]
          ipFamilyPolicy: "PreferDualStack"
      ```

6. Save the file and proceed with the normal installation.
7. Install OSO using the following steps:
   a. Extract the Helm charts tgz file available in the `Artifacts/Scripts` directory.

      ```
      $ cd Artifacts/Scripts
      $ tar -xvzf ocoso_csar_24_3_1_0_0_alertmanager_charts.tgz
      $ tar -xvzf ocoso_csar_24_3_1_0_0_prometheus_charts.tgz
      ```

   b. Install Prometheus and Alertmanager using Helm charts provided and updated `ocoso_csar_24_3_1_0_0_prom_custom_values.yaml` and `ocoso_csar_24_3_1_0_0_alm_custom_values.yaml`file with the following command. Use the following command for installation using Helm, if custom labels are given:

      ```
      $ kubectl create namespace <deployment-namespace-name>
      ```

```
$ helm install -f <ocoso_csar_24_3_1_0_0_prom_custom_values.yaml> --
namespace=<deployment-namespace-name> --name-template=<deployment-
name> ./prometheus
$ helm install -f <ocoso_csar_24_3_1_0_0_alm_custom_values.yaml> --
namespace=<deployment-namespace-name> --name-template=<deployment-
name> ./alertmanager
```

For example:

```
$ kubectl create namespace ns1
## Prometheus
$helm install -f ocoso_csar_24_3_1_0_0_prom_custom_values.yaml --
namespace=ns1 --name-template=oso-p ./
ocoso_csar_24_3_1_0_0_prometheus_charts.tgz
##  Alertmanager
$ helm install -f ocoso_csar_24_3_1_0_0_alm_custom_values.yaml --
namespace=ns1 --name-template=oso-a ./
ocoso_csar_24_3_1_0_0_alertmanager_charts.tgz
```

> **Note**
>
> Skip the flag (`--disable-openapi-validation`) if custom labels aren't given.

c. Run the following commands to bring Prometheus up in the Prometheus GUI.

   i. Take a backup of current `configmap` for restoration.

   ```
   $ kubectl -n <oso-namespace> get cm <oso-prom-configmap-name> -
   oyaml > prom-cm-backup.yaml
   ```

   For example:

   ```
   [cloud-user@occne4-xyz-bastion-1 Scripts]$ o get cm oso-p-prom-svr -
   oyaml >  prom-backup.yaml
   [cloud-user@occne4-xyz-bastion-1 Scripts]$
   [cloud-user@occne4-xyz-bastion-1 Scripts]$
   [cloud-user@occne4-xyz-bastion-1 Scripts]$ ls -lrth prom-backup.yaml
   -rw-------. 1 cloud-user cloud-user 5.5K Apr  3 18:48 prom-
   backup.yaml
   ```

   ii. Take a copy of the current `configmap` for further modifications.

   ```
   $ kubectl -n <oso-namespace> get cm <oso-prom-configmap-name> -
   oyaml > /tmp/prom-cm.yaml
   ```

   For example:

   ```
   [cloud-user@occne4-xyz-bastion-1 Scripts]$ o get cm oso-p-prom-svr -
   oyaml > /tmp/oso-p-cm.yaml
   [cloud-user@occne4-xyz-bastion-1 Scripts]$
   ```

```
[cloud-user@occne4-xyz-bastion-1 Scripts]$ ls -lrth /tmp/oso-p-
cm.yaml
-rw-------. 1 cloud-user cloud-user 5.5K Apr  3 18:48 /tmp/oso-p-
cm.yaml
```

iii. Add the `metrics_path` in Prometheus job.

```
$ sed -i '/ job_name: prometheus/a \ \ \ \ \ \ metrics_path: /
<cluster-name-prefix>/prometheus/metrics' /tmp/prom-cm.yaml
```

For example:

```
[cloud-user@occne4-xyz-bastion-1 Scripts]$ sed -i '/ job_name:
prometheus/a \ \ \ \ \ \ metrics_path: /cluster1/prometheus/
metrics' /tmp/oso-p-cm.yaml
```

iv. Verify if the `metrics_path` was applied appropriately in Prometheus job.

```
$ cat /tmp/prom-cm.yaml | grep metrics_path: -A 2 -B 2
```

For example:

```
[cloud-user@occne4-xyz-bastion-1 Scripts]$ cat /tmp/oso-p-cm.yaml |
grep metrics_path: -A 2 -B 2
    scrape_configs:
    - job_name: prometheus
      metrics_path: /cluster1/prometheus/metrics
      static_configs:
      - targets:
```

v. Replace the existing OSO Prometheus configmap (`oso-p-prom-svr` taken as example here) as shown below:

```
$ kubectl -n <oso-namespace> replace configmap <oso-prom-configmap-
name> -f /tmp/prom-cm.yaml
```

For example:

```
[cloud-user@occne4-xyz-bastion-1 Scripts]$ o replace configmap oso-
p-prom-svr
      -f /tmp/oso-p-cm.yaml configmap/oso-p-prom-svr replaced
```

vi. Verify if the `metrics_path` is updated correctly in the OSO prometheus configmap.

```
$ $ kubectl -n <oso-namespace> get cm <oso-prom-configmap-name> -
oyaml | grep metrics_path: -A 2 -B 2
    scrape_configs:
    - job_name: prometheus
      metrics_path: /<cluster-name-prefix>/prometheus/metrics
      static_configs:
      - targets:
```

For example:

```
[cloud-user@occne4-xyz-bastion-1 Scripts]$ o get cm oso-p-prom-svr -
oyaml | grep metrics_path: -A 2 -B 2
    scrape_configs:
    - job_name: prometheus
      metrics_path: /cluster1/prometheus/metrics
      static_configs:
      - targets:
```

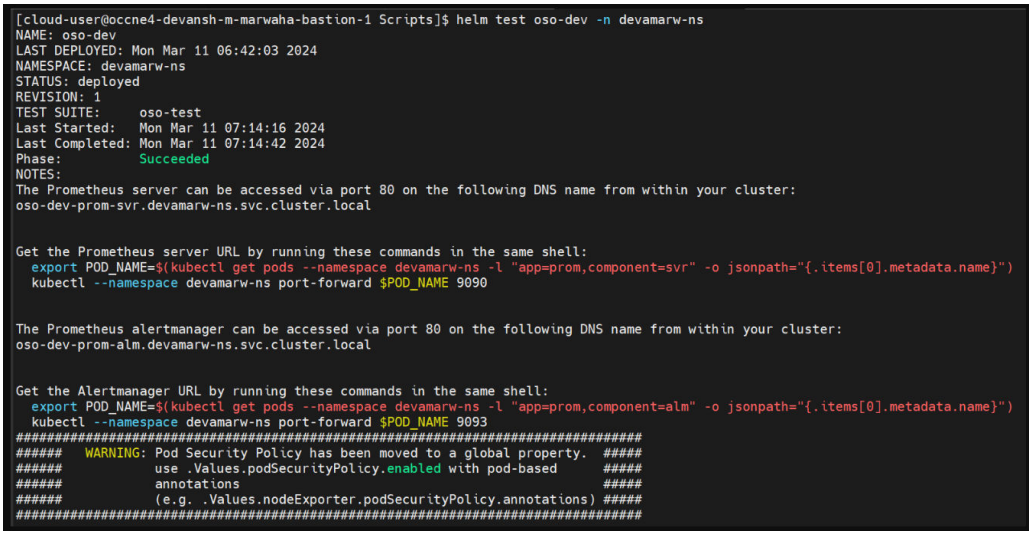
vii. Open Prometheus GUI and verify if the Target is up or not under **Status→Targets**.

d. Run the following command to perform a Helm test. Populate the values in the `ocoso_csar_24_3_1_0_0_prom_custom_values.yaml` and `ocoso_csar_24_3_1_0_0_alm_custom_values.yaml` files to run the Helm test.

```
$ helm test <release-name> -n <namespace>
```

> **Note**
>
> Helm Test can be run for the first time smoothly, but if some issue occurs, and there's a need to re-run the helm test, first you will have to delete the existing test job and repeat the "helm test" command as shown above.
>
> ```
> $ kubectl get jobs.batch -n <namespace>
>
> $ kubectl delete jobs.batch oso-test -n <namespace>
> ```

Figure 2-1 Helm Test



2.4 Postinstallation Tasks

This section explains the postinstallation tasks for OSO.

2.4.1 Verifying Installation

To verify if OSO is installed:

1. Run the following command to verify the OSO version.

   ```
   helm ls -n <oso-namespace>
   ```

   For example:

   ```
   helm ls -n oso-name
   ```

   Sample output:

   ```
   NAME    NAMESPACE       REVISION        UPDATED
   STATUS          CHART                APP VERSION
   oso     oso          3               2024-03-27 18:06:42.575679972 +0000 UTC
   deployed        prometheus-15.16.1      24.1.0
   ```

2. Run the following commands to verify if pods are up and running:

   ```
   $ kubectl get pods --namespace <deployment-namespace-name>
   ```

   For example:

   ```
   $ kubectl get pods -n occne-infra
   ```

   Sample output:

   ```
   NAME                            READY   STATUS    RESTARTS   AGE
   oso-prom-alm-0                  2/2     Running   0          14h
   oso-prom-alm-1                  2/2     Running   0          14h
   oso-prom-svr-84c8c7d488-qsnvx   2/2     Running   0          14h
   ```

3. Run the following commands to verify if services are up and running and are assigned an EXTERNAL-IP (if LoadBalancer is being used):

   ```
   $ kubectl get service --namespace <deployment-namespace-name>
   ```

   For example:

   ```
   $ kubectl get service -n occne-infra
   ```

   Sample output:

   ```
   NAME                                TYPE           CLUSTER-IP
   EXTERNAL-IP     PORT(S)           AGE
   oso-prom-alm                        ClusterIP      10.233.16.83
   <none>          80/TCP      14h
   ```

```
oso-prom-alm-headless                ClusterIP      None
<none>          80/TCP,6783/TCP   14h
oso-prom-svr                         ClusterIP      10.233.46.136
<none>          80/TCP      14h
```

4. Verify that all the GUIs are accessible.

> **Note**
>
> Prometheus and Alertmanager GUIs can be accessed only using the CNC Console. For more information about accessing Prometheus and Alertmanager GUIs using CNC Console, see *Oracle Communications Cloud Native Configuration Console User Guide*.

If the service is of type LoadBalancer, use EXTERNAL-IP to open the Prometheus GUI. Refer to Step 2 to get the services and their EXTERNAL-IPs.

Example to access service IP address with output:

```
# kubectl get nodes -o wide
NAME     STATUS   ROLES    AGE   VERSION   INTERNAL-IP     EXTERNAL-IP
OS-IMAGE                  KERNEL-VERSION                  CONTAINER-RUNTIME
master   Ready    master   87d   v1.17.1   10.75.226.13
&lt;none&gt;        Oracle Linux Server 7.5
4.1.12-112.16.4.el7uek.x86_64   docker://19.3.11
slave1   Ready    &lt;none&gt;   87d   v1.17.1   10.75.225.177
&lt;none&gt;        Oracle Linux Server 7.5
4.1.12-112.16.4.el7uek.x86_64   docker://19.3.11
slave2   Ready    &lt;none&gt;   87d   v1.17.1   10.75.225.47
&lt;none&gt;        Oracle Linux Server 7.5
4.1.12-112.16.4.el7uek.x86_64   docker://19.3.11
```

```
# kubectl get service -n ocnrf
NAME                                TYPE           CLUSTER-IP
EXTERNAL-IP     PORT(S)           AGE
oso-prom-alm                        ClusterIP      10.103.63.10
&lt;none&gt;         80/TCP            35m
oso-prom-alm-headless               ClusterIP      None
&lt;none&gt;         80/TCP            35m
oso-prom-svr                        ClusterIP      10.101.91.81
&lt;none&gt;         80/TCP            35m
```

> **Note**
>
> In case Dualstack with IPv6 is enabled, change the service type to LoadBalancer in the custom values file. You can see both IPs when calling the `kubectl get service` command, and the output is as follows:

```
[root@master Scripts]# kubectl get service -n ocnrf
NAME                          TYPE           CLUSTER-IP
EXTERNAL-IP                           PORT(S)           AGE
oso-prom-alm                  LoadBalancer   10.103.63.10
10.75.202.205,2606:b400:605:b809::2  80/TCP            35m
oso-prom-alm-headless         ClusterIP      None
<none>                                80/TCP            35m
oso-prom-svr                  LoadBalancer   10.101.91.81
10.75.202.204,2606:b400:605:b809::1  80/TCP            35m
```

Figure 2-2 Prometheus GUI

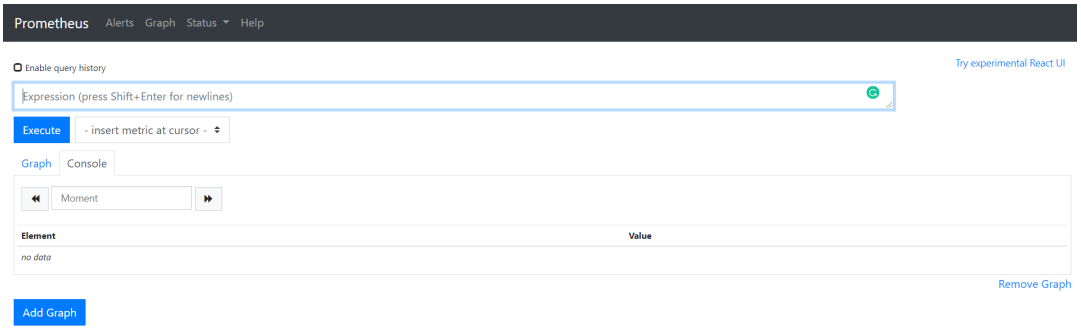
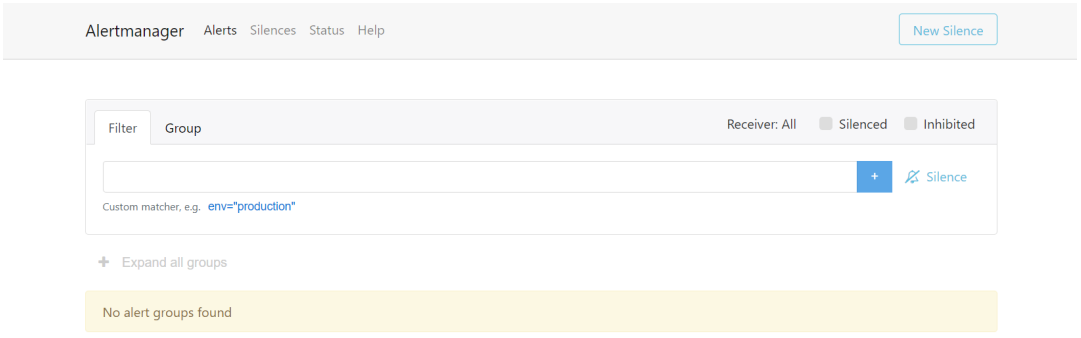


Figure 2-3 Alert Manager GUI

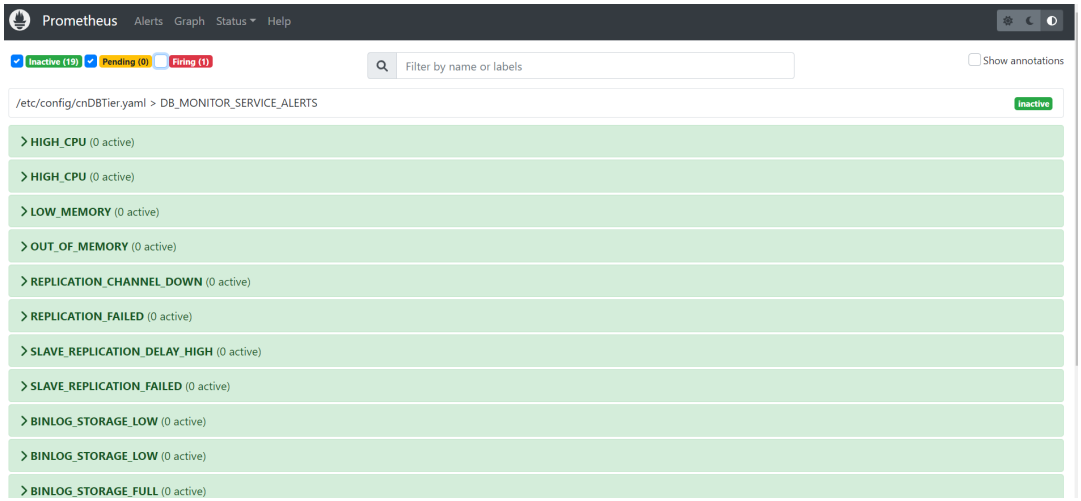


5. On the Prometheus GUI, click **Alerts** to verify that all the alerts (NF Alerts) are visible.

> **Note**
>
> OSO Prometheus doesn't have any alerts of its own, therefore the GUI must appear empty initially. You can patch any NF alert rules in this section.

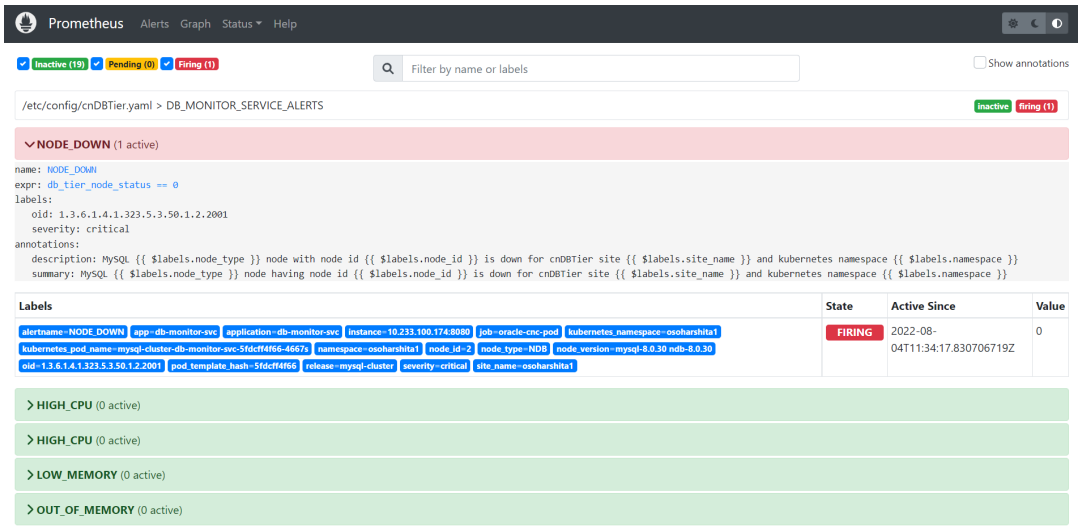
The following image displays the alerts for cnDBTier:

Figure 2-4 Prometheus GUI



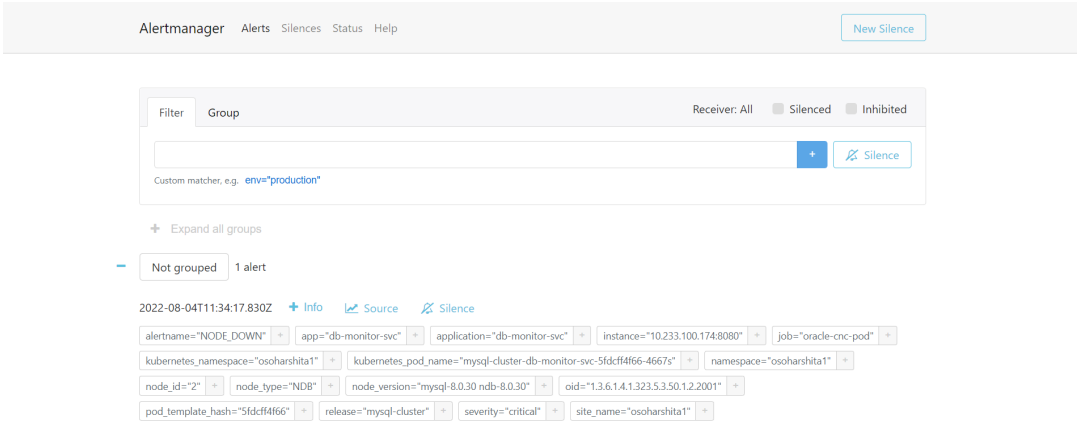
After alerts are raised, the GUI must display the triggered alerts as shown in the following image:

Figure 2-5 Prometheus GUI - Alerts



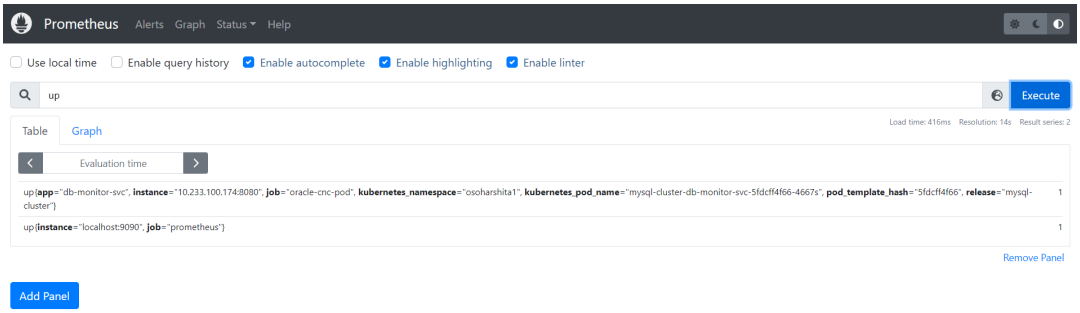
6. Select **Alerts** tab on the Alertmanager GUI to view the triggered alerts as shown in the following image:

Figure 2-6 Alertmanager - Alerts



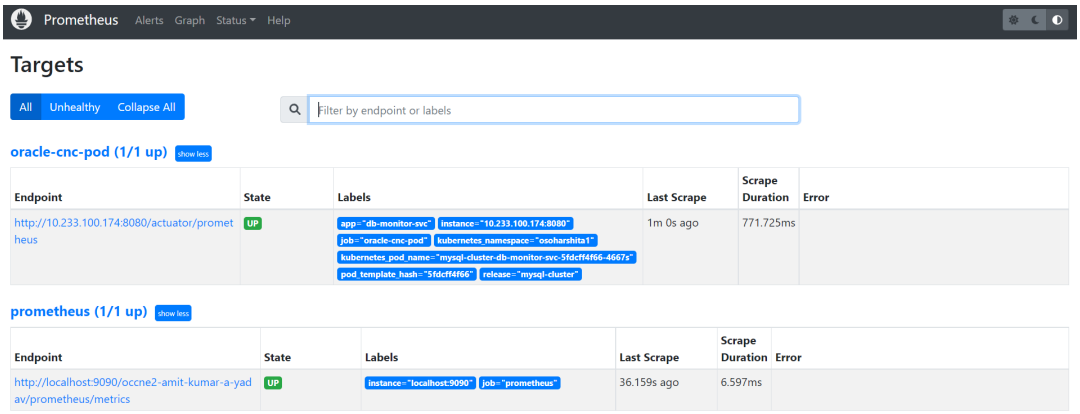
7. On the Prometheus GUI, click **Graph** to verify if the expected metrics (example, NF Metrics) are displayed. The following image displays a sample Prometheus Graph with metrics:

Figure 2-7 Prometheus Graph



8. On the Prometheus GUI, navigate to **Status** and then click **Target** to verify if the configured targets are displayed.
The following image shows Prometheus targets that are being extracted:

Figure 2-8 Prometheus Target



> **Note**
>
> Post installation, Prometheus can be vertically scaled to add more resources to an existing Prometheus server for load balancing. For more information about, Prometheus vertical scaling, see Prometheus Vertical Scaling.

3
Customizing OSO

This chapter provides information about customizing Oracle Communications Operations Services Overlay (OSO) deployment in a cloud native environment.

The OSO deployment is customized by overriding the default values of various configurable parameters in the `<custom-value.yaml>` files.

3.1 Naming Conventions and Limits

This section provides the naming conventions and limits for OSO containers and labels.

Naming Conventions and Limits for Containers

OSO supports customization of container names using custom prefix and suffix. Currently, the name field supports up to 63 characters (including the custom prefix, custom suffix, and default fixed name).

> **Note**
>
> - The maximum length of default fixed names is 17.
> - To avoid name truncation, limit the number of characters in the custom prefix and suffix to match the defined limit.

The following table displays the default fixed names of containers without prefix and suffix:

Table 3-1 Default Fixed Names of Containers

POD Name	Fixed Container Name
Prometheus-server(prom-svr)	• prom-svr-cm-r • prom-svr
Alertmanager(prom-alm)	• prom-alm • prom-alm-cm-r

You can use the following values in the `ocoso_csar_24_3_1_0_0_prom_custom_values.yaml` and `ocoso_csar_24_3_1_0_0_alm_custom_values.yaml` files to add the custom prefix and suffix to container names:

```
global:
# provide the prefix and suffix to be added to k8Resources
  k8Resource:
    container:
      prefix:
      suffix:
```

Naming Conventions and Limits for Labels

1. OSO supports custom labels for Kubernetes resources. It provides three types of label definitions that must be configured in the `ocoso_csar_24_3_1_0_0_prom_custom_values.yaml` and `ocoso_csar_24_3_1_0_0_alm_custom_values.yaml` files:

   a. **Global Labels(allResources)**: These labels are attached to all the Kubernetes resources. This is depicted as TYPE1 in the following sample `ocoso_csar_24_3_1_0_0_prom_custom_values.yaml` file.

   b. **LB and NON-LB TYPE label**: These labels are attached to LoadBalancer and non-LoadBalancer type deployments, statefulsets, and services. This is depicted as TYPE2 in the following sample `ocoso_csar_24_3_1_0_0_prom_custom_values.yaml` file.

   c. **Service specific label**: These labels are attached to each service in their specific service-label metadata sections and can be used to uniquely label services, regardless of them being LoadBalancer or non-LoadBalancer.

   > **Note**
   >
   > Prometheus and Alertmanager have their own set of service-specific labels.

   You can use the following values in the `ocoso_csar_24_3_1_0_0_prom_custom_values.yaml` file to provide custom labels for Kubernetes resources:

```
customExtension:
# TYPE1 Label
    allResources:
      labels: {}
# TYPE2 Labels
    lbServices:
      labels: {}

    nonlbServices:
      labels: {}

    lbDeployments:
      labels: {}

    nonlbDeployments:
      labels: {}

    lbStatefulSets:
      labels: {}
..
```

> **Note**
>
> You must be well aware of the naming guidelines while labeling the resources. For more information on Helm naming guidelines, see Helm Labels and Annotations.

3.2 OSO Configuration Parameters

Overview

This section provides the configuration parameters details that are used to install the OSO CSAR packages manually.

OSO CSAR is packaged with the following open source images:

- prometheus
- alertmanager
- configmapreload
- 24_3_common_oso

Prerequisites

- Docker or Podman must be installed.
- Helm repository must be set up and accessible.

3.2.1 Prometheus Configuration Parameters

This section provides the Prometheus configuration parameters details that are used to install the OSO CSAR packages manually.

Configure `ocoso_csar_24_3_1_0_0_prom_custom_values.yaml` with the specific values.

Table 3-2 OSO Configuration Parameter

Parameter	Description	Details
`OSO-RELEASE-NAME`	This is a conditional parameter. Indicates the Helm release name for the OSO release. Example: promsvcname: oso-prom-svralmsvcname: oso-prom-alm	**Data Type:** String **Default Value:** NA **Range:** NA
`USEASM`	This is a conditional parameter. Indicates if the Aspen Service Mesh (ASM) environment is used or not. Example: useasm: true	**Data Type:** Boolean **Default Value:** false **Range:** true, false
`NAMESPACE`	This is a mandatory parameter. Indicates the namespace where OSO will be installed. This value is required in multiple places in the custom values.yaml file. Example: namespace: ocnf	**Data Type:** String **Default Value:** NA **Range:** NA
`CLUSTER-NAME`	This is a mandatory parameter. Indicates the name of the cluster where OSO is deployed. This value is required in multiple places in the custom values.yaml file. Example: clustername: oso-cluster	**Data Type:** String **Default Value:** NA **Range:** NA

Table 3-2 (Cont.) OSO Configuration Parameter

Parameter	Description	Details

USE-DEFAULT-SVCACCOUNT-PROM	This is a mandatory parameter.
	If default svcaccount is required at cluster scope, then set this parameter to true, else set it to false and create your own svcaccount and configure OSO.
	Example:
	`serviceAccounts:`
	`  prometheus:`
	`  create: false`
		Data Type: Boolean
		Default Value: false
		Range: true, false

serviceAccountNamePromSvr	This is a mandatory parameter.
	Indicates the ServiceAccount that is to be used by the Prometheus resource.
	Users must create ServiceAccount with the required namespace access defined along with the Role or RoleBinding for the Prometheus service.
	Example:
	`# Service account for Prometheus`
	`  serviceAccountNamePromSvr: "oso-prom"`
	If you do not have a ServiceAccount, then set the USE-DEFAULT-SVCACCOUNT-PROM parameter to true.
		Data Type: String
		Default Value:""
		Range: NA

PROMETHEUS-REPO-ADDRESS	This is a mandatory parameter. Indicates the docker image repo URL for Prometheus location. Example: Bastion:5000/occne.io/oso/prometheus	**Data Type:** String **Default Value:** NA **Range:** NA
HELM-TEST-IMAGE-REPO-ADDRESS	This is a mandatory parameter. Indicates the repo URL of the Helm test image location. Example: `Bastion:5000/occne.io/occne/24_2_common_pod`	**Data Type:** String **Default Value:** NA **Range:** NA

Table 3-2 (Cont.) OSO Configuration Parameter

Parameter	Description	Details
CONFIGMAP-RELOAD-REPO-ADDRESS	This is a mandatory parameter. Indicates the docker image repository URL for configmap-reload location. Example: `Bastion:5000/occne.io/oso/configmapreload`	**Data Type:** String **Default Value:** NA **Range:** NA

server:
.....
##
namespaces to monitor(instead of monitoring all - clusterwide).
 	
##Needed if you want to run without Cluster-admin privileges.
 	
namespaces:
- NAMESPACE
	This is a mandatory parameter.
	If clusterrole is used, comment the namespaces section. If role is used, then give the namespace where OSO is deployed.
	Example:
	
	`server:`
	`.....`
	`## namespaces to monitor(instead of monitoring all - clusterwide).`
	`## Needed if you want to run without Cluster-admin privileges.`
	`## namespaces:`
	`## - NAMESPACE`
		Data Type: String
		Default Value: NA
		Range: NA

Table 3-2 (Cont.) OSO Configuration Parameter

Parameter	Description	Details

BEARER_TOKEN	This is a conditional parameter.
	Set the Authorization header on every remote write request with the configured bearer token. In this case, it is Cortex Server.
	Example:
	
	`remote_write`
	`# bearer_token for cortex server to be configured`
	`bearer_token: eyJhbGciOiJIUzUxMiIsInR5cCI6IkpXVCJ9.`
		Data Type: String
		Default Value: NA
		Range: NA

STORAGE_RETENTION_SIZE_PROM	This is a mandatory parameter.
	Indicates the retention size for Prometheus persistence volume.
	It should be atleast 85% of the PV size (6.8GB).
	Example:
	
	`extraArgs:`
	
	`storage.tsdb.retention.size: 6.8GB`
		Data Type: String
		Default Value: NA
		Range: NA

PROM-PV-SIZE	This is a mandatory parameter.
	Indicates the size for the persistence volume of Prometheus.
	Example:
	
	`persistentVolume:`
	`  enabled: true`
	`  size: 8Gi`
		Data Type: String
		Default Value: NA
		Range: NA

PROM-STORAGE-CLASS	This is a mandatory parameter.
	Indicates the storage class for Prometheus.
	Example:
	
	`persistentVolume:`
	`  enabled: true`
	`  ......`
	`  storageClass: standard`
		Data Type: String
		Default Value:standard
		Range: NA

Table 3-2 (Cont.) OSO Configuration Parameter

Parameter	Description	Details
PROM-DATA-RETENTION-DAYS	This is a mandatory parameter. Indicates the retention days for prometheus persistence volume. Example: `retention: 7d`	**Data Type:** String **Default Value:** NA **Range:** NA
OSO_CORTEX_URL	This is a conditional parameter. Indicates the URL where Prometheus writes metrics data. Example: `http://<some-ip>:<some-port>/api/prom/push`	**Data Type:** String **Default Value:** NA **Range:** NA
OSO_REMOTE_WRITE_TIMEOUT	This is a conditional parameter. Indicates the maximum amount of time Prometheus waits for a response to the remote write request. Example: 30s	**Data Type:** String **Default Value:** NA **Range:** NA

namespaces	This is a conditional parameter.
	If you are using external Service Account with namespace specific Role or RoleBinding Access, the namespaces must be configured in Prometheus for extracting the given allowed namespaces.
	Uncomment the following lines in the custom values.yaml file and replace ns1 and ns2 with namespaces you want to configure for extracting, in each of the Prometheus jobs.
	`- job_name: 'oracle-cnc-pod'`
	`kubernetes_sd_configs:`
	`- role: pod`
	`#namespaces:`
	`# names:`
	`# - ns1`
	`# - ns2`
	You can add more namespaces in the list as per your requirement.
		Data Type: String
		Default Value: NA
		Range: NA

Table 3-2 (Cont.) OSO Configuration Parameter

Parameter	Description	Details

`Prefix`
`Suffix`
	This is a conditional parameter.
	Indicates the prefix and suffix that will be added to K8s Resource names.
	Example:
	
	`global:`
	`  k8Resource:`
	`  container:`
	`  prefix: oracle1`
	`  suffix: oracle2`
	
	Note: Currently, only Containers are configured with prefix and suffix.
		Data Type: String
		Default Value: NA
		Range: NA

`allResources:`
`labels:`
`{}`
	This is a conditional parameter.
	Indicates the Global Labels (allResources) that will be attached to all Kubernetes resources.
	Example:
	
	`allResources:`
	`  labels:`
	
	`  env : production`
		Data Type: String
		Default Value: NA
		Range: NA

Table 3-2 (Cont.) OSO Configuration Parameter

Parameter	Description	Details
`lbServices: labels: {} nonlbServices: labels: {} lbDeployments: labels: {} nonlbDeployments: labels: {} lbStatefulSets: labels: {}`	This is a conditional parameter. Indicates the LB and non-LB TYPE labels that will be attached to the load balancer and non-load balancer type deployments, statefulsets, and services. Example: `lbServices: labels: key1 : value1 nonlbServices: labels: key2 : value2 lbDeployments: labels: key3 : value3 nonlbDeployments: labels: key4 : value4 lbStatefulSets: labels: key5 : value5`	**Data Type:** String **Default Value:** NA **Range:** NA
`annotations: {}`	This is a conditional parameter. Specifies the custom annotations to be added to OSO deployments that are associated to disable the sidecar injections into OSO pods. Example: `annotations: - sidecar.istio.io/inject: "false"`	**Data Type:** Boolean **Default Value:** false **Range:** true, false

Table 3-2 (Cont.) OSO Configuration Parameter

Parameter	Description	Details
alertmanagers: - kubernetes_sd_configs: - role: pod # namespace to be configured namespaces: names: {} relabel_configs: - source_labels: [__meta_kubernetes_namespace] # namespace to be configured regex: {}	This is a mandatory parameter. Indicates if this parameter is used to add namespaces for Alertmanager. Example: alertmanagers: - kubernetes_sd_configs: - role: pod # namespace to be configured namespaces: names: - ns1 relabel_configs: - source_labels: [__meta_kubernetes_namespace] regex: ns1	**Data Type:** String **Default Value:** NA **Range:** NA

3.2.2 Alertmanager Configuration Parameters

This section provides the alertmanager configuration parameters details that are used to install the OSO CSAR packages manually.

Configure the `ocoso_csar_24_3_1_0_0_alm_custom_values.yaml` with the defined values.

> **Note**
>
> Configure the following parameters using Table 3-2 table:
>
> - HELM-TEST-IMAGE-REPO-ADDRESS
> - USEASM
> - NAMESPACE
> - CONFIGMAP-RELOAD-REPO-ADDRESS
> - prefix
> - suffix
> - sidecar annotations
> - labels

Table 3-3 OSO Alertmanager Configuration Parameters

Parameter	Description	Details
OSO-ALM-RELEASE-NAME-alm	This is a mandatory parameter. Indicates the Helm release name for installing the Alertmanager in OSO release. Example: oso-a	**Data Type:** String **Default Value:** NA **Range:** NA
ALERTMANAGER-REPO-ADDRESS	This is a mandatory parameter. Indicates the repo URL for alertmanager location. Example: Bastion:5000/occne.io/oso/alertmanager	**Data Type:** String **Default Value:** NA **Range:** NA
RETENTION-TIME-ALM	This is a mandatory parameter. Indicates the define retention time for alertmanager persistence. Example: 120h	**Data Type:** String **Default Value:** NA **Range:** NA
USE-DEFAULT-SVCACCOUNT-ALM	This is a mandatory parameter. Indicates if default svcaccount or own svcaccount is required for configuring OSO. If this value is true, default svcaccount at cluster scope is required. If this value is false, create your own svcaccount and configure OSO	**Data Type:** Boolean **Default Value:** true **Range:** true, false
ENABLE_DUAL_STACK	This is a conditional parameter. Indicates if dual stack support is available for alertmanager. If this value is true, dual stack support (IPV4 and IPV6) is enabled for alertmanager. If this value is false, dual stack support is not enabled for alertmanger.	**Data Type:** Boolean **Default Value:** true **Range:** true, false

Table 3-3 (Cont.) OSO Alertmanager Configuration Parameters

Parameter	Description	Details
ALM-STORAGE-CLASS	This is a conditional parameter. Indicates the alertmanager storage class configuration.	**Data Type:** String **Default Value:** standard **Range:** NA
ALM-PV-SIZE	This is a conditional parameter. Indicates the size for the alertmanager persistence volume.	**Data Type:** String **Default Value:** 2Gi **Range:** NA

annotations: {}	This is a conditional parameter.
	Specifies the custom annotations to be added to OSO deployments that are associated to disable the sidecar injections into OSO pods.
	Note: This parameter is mandatory if ASM is enabled.
	Example:
	`annotations:`
	`  - sidecar.istio.io/inject: "false"`
		Data Type: Boolean
		Default Value: false
		Range: true, false

4

Using OSO with IPv6

This section details the steps to access OSO services like Prometheus and Alertmanager using IPv6.

> **Note**
>
> You can use the Installing OSO using CSAR procedure to install OSO in an IPv6 cluster.

- To access Prometheus GUI using IPv6, use the following URL format and open the GUI using any IPv6 supported browser:

  Load Balancer:

  ```
  LoadBalancerIP-Prometheus]
  ```

  Example:

  ```
  http://[2606:b400:605:b819:4631:92ff:fe73:9d2c]
  ```

  Node Port:

  ```
  [AnyWorkerNodeIP]:NodePort-Prometheus
  ```

  Example:

  ```
  http://[2606:b400:605:b819:4631:92ff:fe73:9d2c]:30090
  ```

- Currently, Alertmanager has an issue in accessing Alertmanager GUI with IPv6 (see, Parsing URLs with IPv6 host issue). To overcome this issue, perform the following steps to use the DNS names concept as Alertmanager does not support IPv6 address format.

  1. Open the `hosts` file from the following path: `C:\Windows\System32\drivers\etc\hosts`.
  2. Add the following entry to map the worker-node IP of IPv6 cluster with a DNS name:

     ```
     # DNS Mapping for alertmanager
     2606:b400:605:b819:4631:92ff:fe73:9d2c alertmanager
     ```

  3. Open GUI using NodePort and DNS name. For example, `http://alertmanager:30854/alertmanager`

5
Upgrading OSO

This chapter provides information about upgrading Oracle Communications Operations Services Overlay (OSO) deployment to the latest release. It is recommended to perform OSO upgrade in a specific order. For more information about the upgrade order, see *Oracle Communications Cloud Native Core, Solution Upgrade Guide*.

> **Note**
>
> This is an installation only release. OSO does not support upgrade from 24.2.x and 24.1.x to 24.3.x.

5.1 Supported Upgrade Paths

The following table lists the supported upgrade paths for OSO:

Table 5-1 Supported Upgrade Path

Source Release	Target Release
24.2.0	24.3.1
24.1.x	24.3.1

5.2 Prerequisites

Following are the prerequisites for upgrading OSO:

- Ensure that the system has OSO 24.2.0 or 23.4.x installed and the OSO is working properly.
- Verify if all the pods and services are up and running.
- Ensure that the NFs are upgraded after the OSO upgrade.

5.3 Upgrading OSO using CSAR

This section describes the procedure to upgrade an existing OSO using CSAR.

1. Prepare the `ocoso_csar_24_3_1_0_0_custom_values.yaml` file by performing the following steps. These .yaml file is used as the input during the upgrade process.
   a. Download OSO CSAR zip file from My Oracle Support (MOS).
   b. Unzip the CSAR zip file:

      ```
      $ unzip <OSO CSAR package>
      ```

c. Upload all the artifacts present in the `Artifacts/Images` folder to the configured repository. For more information on the artifacts, see the Installation Prerequisite section.

d. Update the Prometheus and Alertmanager `ocoso_csar_24_3_1_0_0_custom_values.yaml` file present in the `Artifacts/Scripts` directory with the required values as mentioned in OSO Configuration Parameters.

e. Untar the Helm charts tgz file available in the `Artifacts/Scripts` directory.

```
cd Artifacts/Scripts
$ tar -xvzf <helm charts.tgz file>
```

2. Run the following command to upgrade the existing OSO release:

```
$ helm upgrade <release-name> <path to the OSO helm chart> -n <namespace> -
f <path to ocoso_csar_24.3.1_custom_values.yaml file>
```

3. Post the Helm upgrade a new Prometheus pod spuns up, which is in container creating state. To make it into running state, detach the PVC from old pod in K8s and let it attach to this new upgraded pod.

```
$ kubectl get deployment -n <oso-namespace>
# Find Prometheus deployment name and replace it in below commands, Both
the commands below needs to be run at the same time
$ kubectl -n <oso-namespace> scale deploy <oso-deployment-name> --
replicas=0
$ kubectl -n <oso-namespace> scale deploy <oso-deployment-name> --
replicas=1
```

4. Verify if all the pods are up and running with the latest versions and changes.
5. Verify if the upgraded OSO is working as expected by performing the Verifying Installation procedure.

6
Rolling Back OSO

This chapter provides information about rolling back Oracle Communications Operations Services Overlay (OSO) deployment to previous releases.

> **Note**
>
> OSO does not support roll back from 24.3.x to previous releases.

6.1 Supported Rollback Paths

The following table lists the supported rollback paths for OSO:

Table 6-1 Supported Rollback Path

Source Release	Target Release
24.3.1	24.2.0
24.3.1	24.1.x

6.2 Prerequisites

Following are the prerequisites to rollback OSO:

- Ensure that the system has OSO 24.3.1 installed and the OSO is working properly.
- Verify if all the pods and services are up and running.

6.3 Rollback Tasks

To roll back from OSO 24.3.1 to previous releases 23.4.x or 24.2.0:

1. Check the revision you want to roll back your release to.

   ```
   $ helm -n <namespace> history <oso-release-name>
   ```

   For example:

   ```
   $ helm -n dbtier history oso
   ```

   Sample output

   ```
   REVISION        UPDATED                         STATUS
   CHART                   APP VERSION             DESCRIPTION
   1               Mon Mar 25 17:46:51 2024        superseded
   prometheus-15.16.1      24.3.1-62-g28668fe Install complete
   ```

```
2               Mon Mar 25 17:48:01 2024        deployed
prometheus-15.16.1      24.3.1-62-g28668fe Upgrade complete
```

2. Pick the revision number to which you wish to roll back your release into. In above example, it will be 1. Run the following command to roll back:

   ```
   $ helm  -n <namespace> rollback <oso-release-name> <oso revision number>
   ```

   For example:

   ```
   $ helm -n dbtier rollback oso 1
   ```
   Sample output

   ```
   Rollback was a success! Happy Helming!
   ```

3. Verify if the rollback was successful to a previous version, perform the following command:

   ```
   $ helm -n <namespace> history <oso-release-name>
   ```

   For example:

   ```
   $ helm -n dbtier history oso
   ```
   Sample output

   ```
   REVISION        UPDATED                         STATUS
   CHART                   APP VERSION             DESCRIPTION
   1               Mon Mar 25 17:46:51 2024        superseded
   prometheus-15.16.1      24.3.1-62-g28668fe Install complete
   2               Mon Mar 25 17:48:01 2024        superseded
   prometheus-15.16.1      24.3.1-62-g28668fe Upgrade complete
   3               Wed Mar 27 18:37:02 2024        deployed
   prometheus-15.16.1      24.3.1-62-g28668fe Rollback to 1
   ```

4. Post Helm rollback a new Prometheus pod is created, which is in pending state. To make it into running state, we have to detach the PVC from old pod and attach it to the new pod using below steps:

   ```
   $ kubectl get deployment -n <oso-namespace>
   # Find Prometheus deployment name and replace it in below commands, Both
   the commands below needs to be run at the same time
   $ kubectl -n <oso-namespace> scale deploy <oso-deployment-name> --
   replicas=0
   $ kubectl -n <oso-namespace> scale deploy <oso-deployment-name> --
   replicas=1
   ```

   For example:

   ```
   $ kubectl get deployment -n OSO
   ```

   ```
    $ kubectl -n OSO scale deployment oso-prom-svr --replicas=0
    $ kubectl -n OSO scale deployment oso-prom-svr --replicas=1
   ```

Sample Output:

```
$ kubectl -n testoso get deployment
NAME           READY   UP-TO-DATE   AVAILABLE   AGE
oso-prom-svr   1/1     1            1           23h
$ kubectl -n testoso scale deploy oso-prom-svr --replicas=0
deployment.apps/oso-prom-svr scaled
$ kubectl -n testoso scale deploy oso-prom-svr --replicas=1
deployment.apps/oso-prom-svr scaled
```

A
Key Information About OSO

This section provides important information that the user must know about OSO.

The following are some of the important points about OSO installation that you must be aware of:

1. All the services having GUI are converted to ClusterIP as Cloud Native Configuration Console (CNC Console) hyperlinks for OSO GUIs are currently not enabled. Therefore, you can access the GUIs through CNC Console only.
   Run the following command to get the list of services and check their service type.

   ```
   $ kubectl get service -n <namespace>
   ```

   Sample output:

   ```
   NAME                              TYPE          CLUSTER-IP      EXTERNAL-
   IP        PORT(S)         AGE
   oso-prom-alm                    ClusterIP       10.233.48.28
   <none>        80:31717/TCP        4d13h
   oso-prom-alm-headless           ClusterIP       None
   <none>        80/TCP,6783/TCP     4d13h
   oso-prom-svr                    ClusterIP       10.233.22.117
   <none>        80:30266/TCP        4d13h
   ```

2. Since Alertmanager is a StatefulSets, the Persistent Volume (PV) or Persistent Volume Claim (PVC) do not get deleted automatically when the following helm uninstallation command is run for the respective OSO helm-release:

   ```
   $ helm uninstall oso-release-name -n <namespace>
   ```

   This uninstallation command removes all resources related to the OSO release. However it does not remove the PVs or PVCs for Alertmanager. Therefore, perform the following steps to manually remove PVs and PVCs in Alertmanager:

   a. Run the following command to retrieve the list of PVCs:

      ```
      $ kubectl get pvc -n <namespace>
      ```

      Sample output:

      ```
      NAME             STATUS    VOLUME
      CAPACITY   ACCESS MODES   STORAGECLASS   AGE
      oso-prom-svr     BOUND     pvc-a639c8cb-eee1-4fa3-be1b-d4c4e5d66b9a
      8Gi        RWO            standard       159m
      oso-prom-alm-0   BOUND     pvc-a639c8cb-eee1-4fa3-be1b-d4c4e5d6689J
      2Gi        RWO            standard       159m
      ```

```
oso-prom-alm-1   BOUND     pvc-a639c8cb-eee1-4fa3-be1b-d4c4e5d6689k
2Gi        RWO            standard       159m
```

b. Run the following command to delete the PVCs:

```
$ kubectl delete pvc oso-prom-svr oso-prom-alm-0 oso-prom-alm-1 -n
<namespace>
```

Sample output:

```
persistentvolumeclaim "oso-prom-svr" deleted
persistentvolumeclaim "oso-prom-alm-0" deleted
persistentvolumeclaim "oso-prom-alm-1" deleted
```

c. Run the following command to retrieve the PV details:

```
$ kubectl get pv -n <namespace>
```

Sample output:

```
NAME                                       CAPACITY  ACCESS MODES
RECLAIM POLICY   STATUS
CLAIM
STORAGECLASS       REASON   AGE
pvc-104a1c0c-72f7-4ae0-b368-71c4e22082ce   2Gi        RWO
Delete           Bound    osoxyz2/storage-volume-oso-xyz2-prom-
alm-1           standard                   4d13h
pvc-1e75d93e-0731-45f2-9ef7-46cf8cbbb3f9   2Gi        RWO
Delete           Bound    osoxyz1/storage-volume-oso-xyz-prom-
alm-0            standard                   4d13h
pvc-30400fe7-61dd-4e2f-8a50-149e4e14be90   2Gi        RWO
Delete           Bound    osoxyz2/oso-xyz2-prom-
svr                           standard                  4d12h
```

d. Run the following command to delete the PVs:

```
$ kubectl delete pv pvc-104a1c0c-72f7-4ae0-b368-71c4e22082ce
pvc-1e75d93e-0731-45f2-9ef7-46cf8cbbb3f9
pvc-30400fe7-61dd-4e2f-8a50-149e4e14be90 -n osoxyz1
```

3. Alertmanager has persistent storage enabled in this release (24.2.0) to store notifications and silences.
4. OSO 24.3.0 is install only release since we have segregated Alertmanager and prometheus helm charts. These charts will be installed separately from 24.3.0 onwards.

B
Prometheus Vertical Scaling

This section describes the procedure for vertical scaling of Prometheus.

To scale Prometheus deployments, follow these steps:

1. Get the list of deployments and identify the OSO Prometheus deployment with the suffix `prom-svr`:

   ```
   # To list all the deployments in the OSO namespace
   $ kubectl -n <OSO_namespace> get deployments
   # To filter the deployment name by its suffix
   $ kubectl -n <OSO_namespace> get deployments | grep prom-svr
   ```

2. Edit the OSO deployment using the following command:

   > **Note**
   >
   > This will open a *vi* editor with the deployment's yaml definition.

   ```
   $ kubectl -n <OSO_namespace> edit deployment <oso_deployment_name>-prom-svr
   ```

3. Find the `resources` section for the `prom-svr` container in the edit mode of deployment, and edit the amount of resources as per the requirements.

   ```
   name: prom-svr
   ports:
   ... # ports definitions
   readinessProbe:
   ... # readiness probe definition
   resources:
     limits:
       cpu: "2"
       memory: 4Gi
     requests:
       cpu: "2"
       memory: 4Gi
   ```

4. Save and quit from the editor after making the required changes in the yaml file for the CPU and memory. In case of any errors while editing, the editor opens again and error message appears at the top of the yaml file as a comment.

   > **Note**
   >
   > If any of these objects have two containers each, you will find two resources sections. For more information about how to assign resources, see https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/.