Oracle® Communications

Cloud Native Core, Operations Services Overlay User Guide

Release 26.1.200
G55294-01
July 2026

Oracle Communications Cloud Native Core, Operations Services Overlay User Guide, Release 26.1.200

G55294-01

Contents

Preface

- Documentation Accessibility
- Diversity and Inclusion
- Conventions

Documentation Accessibility

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

Access to Oracle Support

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

Diversity and Inclusion

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

Conventions

The following text conventions are used in this document:

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

My Oracle Support

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

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

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

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

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

Acronyms

The following table provides information about the acronyms used in the document.

Table Acronyms

Acronym	Description
CNE	Cloud Native Environment
CNC	Cloud Native Core
CRD	Custom Resource Definition
CSAR	Cloud Service Archive
NF	Network Function
OCCNE	Oracle Communications Cloud Native Environment
ONAP	Open Network Automation Platform
OSO	Operations Services Overlay
OSDC	Oracle Software Download Center
TSDB	Time Series Database

What's New in This Guide

This section lists the documentation updates for release 26.1.2xx.

Release 26.1.200- G55294-01, July 2026

- Added the metrics and alerts in the Alert Forwarding to Kafka section.
- Added a reference for automated configuration of Alert rules in the Updating Alert Rules for an NF section.
- Added Table 3-1.
- Added the following sections to include the list of metrics and alerts supported for APM service:
  - Metrics
  - OSO Alerts
- Added the Update Kafka Certificates in Kubernetes secret section with the procedure for updating Kafka CA and TLS certificates stored in Kubernetes secrets.
- Added the Important Performance Considerations section, which provides guidance for querying high-volume metrics in the Prometheus UI.

1 Overview

The Oracle Communications Operations Services Overlay (OSO) 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.

OSO is an independent deliverable distinct from Oracle Communications Cloud Native Core, Cloud Native Environment (CNE). OSO assists efficient integration of network measurement mechanisms, resource allocation, and dynamic decision-making based on a common set of measurements.

This document guides the OSO users to operate and maintain OSO services.

1.1 References

Following are the reference documents:

- *Oracle Communications Cloud Native Core, Operations Services Overlay Installation and Upgrade Guide*
- *Oracle Communications Cloud Native Core, Operations Services Overlay Network Impact Report*
- *Oracle Communications Cloud Native Core Release Notes*
- *Oracle Communications Cloud Native Core Licensing Information User Guide*
- *Oracle Communications Cloud Native Core Solution Installation, Upgrade, and Fault Recovery Guide*
- *Oracle Communications Cloud Native Core Security Guide*

2 Installing and Configuring OSO

Oracle Communications Operations Services Overlay (OSO) can be installed on any supported version of Kubernetes. For more information about how to install and configure OSO, see *Oracle Communications Cloud Native Core, Operations Services Overlay Installation and Upgrade Guide*.

3
OSO Services

Oracle Communications Operations Services Overlay (OSO) provides the following observability services to help users observe the behavior of applications running in their cloud native environment.

OSO Features List with Configuration Status

The following table lists the OSO features along with their configuration status.

Table 3-1 OSO Features List with Configuration Status

Feature Name	Enabled or Disabled By Default	Configuration Type
Support for Leapfrog Upgrade and Rollback This feature is applicable only to release **26.1.200**. This capability is provided specifically for this release and will not be supported in future releases.	Enabled	Helm
Support for Alerts in Alert Forwarding to Kafka Alert Forwarding to Kafka	Enabled	Helm
Secure Kafka Integration for Alert Forwarding Alert Forwarding to Kafka	No	Helm
Alert Forwarding to Kafka Alert Forwarding to Kafka	No	Helm
Alert Automation OSO Alerts	No	Helm
Support for Time Series Database (TSDB) Snapshot Support for Time Series Database (TSDB) Snapshot	No	Helm

3.1 Metrics

This section describes the observability metrics provided by the OSO services.

3.1.1 Metrics Collection

OSO uses Prometheus to collect metrics from all the Cloud Native Core (CNC) applications deployed in the user's CNE. Prometheus collects metrics from the following:

- OSO services: Prometheus collects and stores metrics generated by the OSO services, such as Prometheus and AlertManager.

- CNC Network Functions: OSO Prometheus receives customer metrics generated and delivered by the CNC NFs.

3.1.2 Storing Metrics

OSO Prometheus stores all the metrics in an internal Time Series Database (TSDB).

3.1.3 Support for Time Series Database (TSDB) Snapshot

Prometheus uses Time Series Database (TSDB) to store the metrics. Along with metric storage, OSO captures a snapshot at a specific point of time with the available data in the Prometheus data store. OSO allows you to capture these snapshots without shutting down or disrupting the Prometheus instance.

This feature can be used for the following:

- Backups
- Recovery
- Debugging

For more information about the procedure for capturing TSDB snapshots, see the "Creating Backup of Prometheus Time Series Database (TSDB) Using Snapshot Utility" section in *Oracle Communications Cloud Native Core, Operations Services Overlay Installation and Upgrade Guide*.

3.2 OSO Alerts

OSO uses AlertManager to raise alerts. These alerts inform the user about the aspect of OSO that requires attention.

The applications deployed on OSO define their alerts to inform the user about the problems specific to each application. For more information on how applications can load alerting rules, see the Updating Alert Rules for an NF section.

Automated Configuration of NF Alerts

From release 25.1.200 onwards, OSO offers an automated mechanism to configure NF alerts after the successful deployment of all respective NFs and OSO components. This feature streamlines the management of OSO alert rules by dynamically updating the Prometheus ConfigMap during the Helm chart installation or upgrade process. As a result, manual intervention is minimized, ensuring consistent and efficient alert rule deployment across environments.

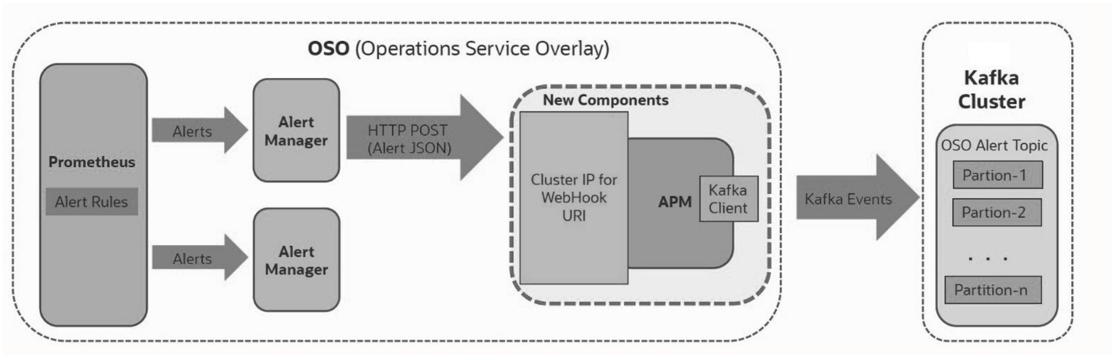
For more information about automation of alert configuration in OSO, see the "Automated Configuration of NF Alerts" section in *Oracle Communications Cloud Native Core, Operations Services Overlay Installation and Upgrade Guide*.

3.2.1 Alert Forwarding to Kafka

With this feature, OSO introduces a Alert Processing Microservice (APM) that consumes HTTP-based alerts from OSO (Alert manager), converts them into Kafka-compatible JSON messages, and publishes them to a designated Kafka cluster. This feature can be deployed independently using a dedicated Helm chart, offering flexibility to add or remove it without affecting the core OSO platform components.

From release 25.2.201 onwards, APM supports Kafka connections over TLS. This allows alert forwarding to Kafka using an encrypted connection.

Figure 3-1 Alert Forwarding to Kafka



The image illustrates the alert processing flow enabled by the APM within OSO:

1. Prometheus continuously monitors the system metrics and applies alert rules to detect issues or threshold breaches.
2. When an alert condition is met, Prometheus sends the alert to the Alert Manager within OSO. The Alertmanager aggregates the incoming alerts for further processing based on the configuration.
3. The Alert Manager sends the processed alert as a JSON payload via an HTTP POST request to the Cluster IP.
4. The APM receives the HTTP POSTed alert.
5. APM's Kafka client converts the alert into a Kafka-compatible JSON message. It manages retries, backoff, and delivery timeouts for reliable delivery.
6. APM publishes the alert to the Kafka Cluster.
7. The Kafka Cluster stores the alerts.

Managing the Alert Forwarding to Kafka

The APM feature is **optional** in OSO. It is not enabled by default and must be explicitly installed and configured if alert forwarding to Kafka is needed. If omitted, core OSO functionality is unaffected.

This section lists the Helm configuration details for this feature.

This feature can be enabled or disabled at the time of OSO deployment using the following Helm parameters:

Perform the following configuration to enable this feature using the Helm:

1. Open the `ocoso_csar_26_1_200_0_0_apm_custom_values.yaml` file.
2. Set the value of `apmEnabled` to *true* to enable the alert forwarding to Kafka cluster using APM.
3. Configure the values for the following parameters:
   - `env.kafkaIp`: Indicates the external bootstrap service IP, which maps to the Kafka brokers.

- `env.kafkaPort`:Indicates the external bootstrap service port which maps to the Kafka brokers.
- `env.kafkaTopic`:Indicates the Kafka topic to which APM will publish alert messages.
- `env.kafkaGroupId`:Indicates a name that the user provides for the Kafka group id.
- `env.severityList`: Indicates the list of alert severities to be used in the environment configuration.

> **Note**
>
> Configure the following parameters in the `ocoso_csar_26_1_200_0_0_alm_custom_values.yaml` file.
>
> ```
> config:
>    receivers:
>     - name: default-receiver
>        webhook_configs:
>          - url: 'http://APM-SVC-NAME.NAMESPACE.svc:APM-SVC-PORT/
> webhook'
>            send_resolved: true
> ```

For more information about the parameters, see *Oracle Communications Cloud Native Core, Operations Services Overlay Installation and Upgrade Guide*.

4. [Optional] Check the value of `apmTLS.enable` parameter is *true* to enable Kafka connections over TLS.
   Configure `apmTLS.caSecret` with the name of the Kubernetes secret containing ca.crt.

   `apmTLS.serverSecret` with the name of the Kubernetes secret for establishing mTLS connections.

   For more information about the parameters, see *Oracle Communications Cloud Native Core, Operations Services Overlay Installation and Upgrade Guide*.
5. Save the file.
6. Install OSO. For more information about the installation procedure, see *Oracle Communications Cloud Native Core, Operations Services Overlay Installation and Upgrade Guide*.

Observability

Metrics

The following metrics are added for this feature:

- `kafka_producer_tx_messages`
- `kafka_producer_tx_bytes`
- `kafka_producer_msg_queue_count`
- `kafka_producer_msg_queue_bytes`
- `kafka_producer_outgoing_buffer_count`
- `kafka_producer_requests`

- `kafka_producer_errors`
- `kafka_producer_latency_ms_[suffix]`
- `kafka_producer_msgs_received`
- `kafka_producer_msgs_acked`

For more information about the metrics, see Alert Processing Microservice Metrics .

Alerts

The following alerts are added for this feature:

- KafkaLostAck
- KafkaProducerMsgsErrors
- KafkaProducerQueueCritical

For more information about the alerts, see OSO Alerts.

KPIs

There are no new KPIs related to this feature.

3.3 Important Performance Considerations

When querying a metric in the Prometheus UI, a query that returns a very large number of samples might fail with an `Invalid JSON response` error.

To reduce the number of samples returned, apply an appropriate label filter to the metric query. For example, instead of querying:

```
some_test_metric
```

query the metric with a label filter that is applicable to that metric:

```
some_test_metric{label="value"}
```

4
OSO Metrics and Alerts

4.1 OSO Metrics

This section includes information about metrics for Oracle Communications Cloud Native Core, Operations Services Overlay

The name of the metrics may contain suffix such as total, seconds, max and so on. It gets added by the micrometer registry if it is not present in the metrics name. The metric name has the following format for suffix: `<Basename of the metric>_<Suffix>`

Table 4-1 Metrics type and Suffix

Metric Type	Suffix	Description
Counter	_total	Represents the total number of occurrences of an event or traffic, such as measuring the total amount of traffic received and transmitted by OSO, and so on.
Gauge	NA	Represents a single numerical value that changes randomly. This metric type is used to measure various parameters, such as OSO load values, memory usage, and so on.
Histogram	_max, _bucket, _count, or _sum	Represents the sample observations like request durations or response sizes and counts them in configurable buckets.

Latency Metrics Format for OSO

The following metrics `*_latency_seconds_[suffix]` continue to be supported.

- `*_latency_seconds_max`
- `*_latency_seconds_bucket`
- `*_latency_seconds_count`
- `*_latency_seconds_sum`

Dimensions Legend for the Metrics

The following table includes the details about the metrics dimensions:

Table 4-2 Dimensions Legend

Dimension	Description
`client_id`	Indicates the Id of APM provided in custom values (kafkaGroupId). **Sample Values**: my-python-producer

Table 4-2 (Cont.) Dimensions Legend

Dimension	Description
`error_code`	Indicates the error code received for the request sent. **Sample Values**: 408/503/500
`resp_code`	Indicates the response code received with successful message. **Sample Values**: 200
`severity`	Indicates the severity.

4.1.1 Alert Processing Microservice Metrics

Table 4-3 kafka_producer_tx_messages

Field	Details
Description	Indicates the cumulative count of Kafka messages successfully produced by this client.
Type	Counter
Dimensions	client_id

Table 4-4 kafka_producer_tx_bytes

Field	Details
Description	Indicates the cumulative number of payload bytes successfully produced to Kafka.
Type	Counter
Dimensions	client_id

Table 4-5 kafka_producer_msg_queue_count

Field	Details
Description	Indicates the current number of messages waiting in the producer’s internal queue.
Type	Gauge
Dimensions	client_id

Table 4-6 kafka_producer_msg_queue_bytes

Field	Details
Description	Indicates the current total size (in bytes) of messages waiting in the producer queue.
Type	Gauge
Dimensions	client_id

Table 4-7 kafka_producer_outgoing_buffer_count

Field	Details
Description	Indicates the current number of queued messages ready to be transmitted to Kafka brokers.
Type	Gauge
Dimensions	client_id

Table 4-8 kafka_producer_requests

Field	Details
Description	Indicates the cumulative count of Kafka protocol requests sent by the producer.
Type	Counter
Dimensions	client_id

Table 4-9 kafka_producer_errors

Field	Details
Description	Indicates the cumulative count of producer-side errors, broken down by error_code.
Type	Counter
Dimensions	client_id, error_code

Table 4-10 kafka_producer_latency_ms_[suffix]

Field	Details
Description	Indicates the end-to-end produce latency distribution in milliseconds (tracked by histogram buckets), labeled by response code. **Note**: • `*_latency_ms_[suffix]` can represent any of the supported formats: – `*_latency_ms_max` – `*_latency_ms_bucket` – `*_latency_ms_count` – `*_latency_ms_sum`
Type	Histogram
Dimensions	resp_code

Table 4-11 kafka_producer_msgs_received

Field	Details
Description	Indicates the cumulative count of messages received by the APM service, labeled by severity.
Type	Counter
Dimensions	severity

Table 4-12 kafka_producer_msgs_acked

Field	Details
Description	Indicates the cumulative count of messages acknowledged by the APM service, labeled by response code.
Type	Counter
Dimensions	resp_code

4.2 OSO Alerts

This section includes information about the OSO alerts.

The following table describes the various alert levels generated by OSO:

Table 4-13 Alerts Levels or Severity Types

Alerts Levels/Severity Types	Definition
Critical	Indicates a severe issue that poses a significant risk to safety, security, or operational integrity. It requires immediate response to address the situation and prevent serious consequences. Raised for conditions may affect the service of OSO.
Major	Indicates a more significant issue that has an impact on operations or poses a moderate risk. It requires prompt attention and action to mitigate potential escalation. Raised for conditions may affect the service of OSO.
Minor	Indicates a situation that is low in severity and does not pose an immediate risk to safety, security, or operations. It requires attention but does not demand urgent action. Raised for conditions may affect the service of OSO.
Info or Warn (Informational)	Provides general information or updates that are not related to immediate risks or actions. These alerts are for awareness and do not typically require any specific response. WARN and INFO alerts may not impact the service of OSO.

4.2.1 KafkaLostAck

Table 4-14 KafkaLostAck

Field	Details
Description	Ack loss ratio exceeded threshold for sustained traffic indicating message acknowledgment failures.
Summary	APM Kafka producer ack failures detected.
Severity	Critical
Condition	Ack loss ratio has stayed above 2% for 10m while traffic is sustained (>200 messages in 5m)

Table 4-14 (Cont.) KafkaLostAck

Field	Details
Metric Used	`kafka_producer_msgs_received_total`, `kafka_producer_msgs_acked_total` **Note**: Used to calculate acknowledgment loss ratio
Recommended Actions	The alert is cleared automatically when acknowledgment rates return to normal. **Steps**: 1. Check APM-Kafka configuration (topic, Kafka IP, port, TLS/mTLS). 2. Verify Kafka availability (pods, services, deployments). 3. Check APM service pod logs for high traffic or large message saturation. 4. Investigate network issues (latency, packet loss). 5. Correlate with other alerts such as producer errors or queue issues. If the issue persists, contact *My Oracle Support*.

4.2.2 KafkaProducerMsgsErrors

Table 4-15 KafkaProducerMsgsErrors

Field	Details
Description	Kafka producer is experiencing sustained message send failures.
Summary	APM Kafka producer message failure detected.
Severity	Critical
Condition	Producer send errors exceeded 5 in 2m and persisted for 5m, indicating a sustained failure condition.
Metric Used	`kafka_producer_errors` **Note**: Tracks producer-side errors categorized by error_code.
Recommended Actions	The alert is cleared automatically when error rate drops below threshold. **Steps:** 1. Check APM-Kafka configuration (topic, Kafka IP, port, TLS/mTLS). 2. Inspect APM service pod logs for specific error codes. 3. Validate TLS/mTLS certificates used by APM. 4. Verify topic existence and partition availability. 5. Check for broker overload or request rejection issues.

4.2.3 KafkaProducerQueueCritical

Table 4-16 KafkaProducerQueueCritical

Field	Details
Description	Kafka producer queue depth is critically high indicating backlog in message delivery.
Summary	APM Kafka producer queue critically high.
Severity	Critical
Condition	Producer queue depth exceeded 1000 within the last 5m and remained elevated for 5m.
Metric Used	`kafka_producer_msg_queue_count` **Note**: Represents number of messages pending in producer queue.
Recommended Actions	The alert is cleared automatically when queue depth returns to normal levels. **Steps:**: 1. Check if related alerts (KafkaLostAck, KafkaProducerMsgsErrors) are also triggered. 2. Compare producer send rate vs acknowledgment rate using Prometheus. 3. Run queries: `kafka_producer_msgs_received_total` and `kafka_producer_msgs_acked_total` 4. Check APM service pod logs for retries and reconnects. 5. If traffic is high, reduce producer rate or increase partitions. 6. If Kafka is slow, investigate broker, network, or storage bottlenecks.

5
Maintenance Procedures

This section provides details about the OSO maintenance procedures.

5.1 Postinstallation Configuration

5.1.1 Changing Metrics Storage Allocation

The following procedure describes how to increase or decrease the amount of persistent storage allocated to Prometheus for metrics storage.

Prerequisites

The user must calculate the revised amount of persistent storage required by the metrics.

Procedure

1. A Prometheus resource is used to configure all Prometheus instances running in OSO. Run the following command to identify the Prometheus resource:

   ```
   kubectl get prometheus -n <namespace>
   ```

2. Run the following command to resize the Prometheus metric allocation size by setting the value of `allowVolumeExpansion` to true.

   ```
   $ kubectl -n <namespace> get sc
   $ '{"allowVolumeExpansion": true}'
   ```

3. Run the following command to scale the Prometheus pod down.

   ```
   $ kubectl -n <namespace> scale deploy oso-prom-svr --replicas 0
   ```

4. Run the following command to change the pvc size of Prometheus pods:

   ```
   $ kubectl -n <namespace> edit pvc oso-prom-svr
   ```

> **Note**
>
> You will be placed in a *vi* editor session that contains all of the configurations for the OSO Prometheus pvc. Scroll down to the line that contains the "spec.Capacity" key, then update the value to the <desired increased pv size> as configured in the above step. The file must look similar to the following example:
>
> ```
> spec.Capacity: 10Gi
> ```
>
> Type "`:wq`" to exit the editor session and save the changes.

5. Run the following command to verify that the pvc size change was applied:

```
$ kubectl get pv | grep oso-prom-svr
```

> **Note**
>
> Wait until the new desired size gets reflected "10Gi".

6. Once both the pv sizes are updated to the new desired size, run the following command to scale up the Prometheus pods:

```
$ kubectl -n <namespace> scale deploy oso-prom-svr --replicas 1
```

> **Note**
>
> You will be placed in a *vi* editor session that contains all of the configurations for the OSO Prometheus instances. Scroll down to the line that contains the "replicas" key, then change the value back to 2. This scale backs up both the pods. The file must look similar to the following example:

7. Run the following command to verify that the Prometheus pods are up and running:

```
kubectl get pods -n <namespace> | grep oso-prometheus
```

5.2 Managing 5G NFs

This section describes procedures to manage 5G NFs in CNE OSO.

5.2.1 Updating Alert Rules for an NF

This section describes the procedure to add or update the alerting rules for any Cloud Native Core (CNC) 5G Network Functions (NF) in OSO Prometheus GUI.

From release 26.1.200 onwards, use the OSO-ALR-CONFIG feature for configuring the NF alerts on OSO as the following manual procedure will get deprecated and not supported in future. For more details, see "Automated Configuration of NF Alerts" section in *Oracle Communications Cloud Native Core, Operations Services Overlay Installation and Upgrade Guide*.

Prerequisites

- All NFs are required to create a separate `Alert-rules`.
- For OSO Prometheus: A valid OSO release must be installed and an alert file describing all NF alert rules according to old format is required.

Add or Update Alert Rules

Perform the following steps to add alert rules in OSO Prometheus GUI:

1. Take the backup of current configuration map of OSO Prometheus.

   ```
   $ kubectl get configmaps <OSO-prometheus-configmap-name> -o yaml -n
   <namespace> /tmp/tempPrometheusConfig.yaml
   ```

   Where,

   - <OSO-prometheus-configmap-name> is the name of the OSO Prometheus configuration map.
   - <namespace> is the OSO namespace.

2. Check and add the NF Alert file name inside the Prometheus configuration map. <nf-alertsname> varies from NF to NF, and can be retrieved from each individual NF alert rules file.

   For example, in the following screenshot, "alertscndbtier" is the nf-alertsname for cnDBTier.

   Figure 5-1 OSO Alert file

   ```
   [cloud-user@occne2-cgbu-cne-dbtier-bastion-1 Scripts]$ cat dbalert.yaml
   apiVersion: v1
   data:
     alertscndbtier: |
       groups:
       - name: DB_MONITOR_SERVICE_ALERTS
         rules:
         - alert: NODE_DOWN
           expr: db_tier_node_status == 0
   ```

   After retrieving the nf-alertsname run the following steps:

   ```
   $ sed -i '/etc\/config\/<nf-alertsname>/d' /tmp/tempPrometheusConfig.yaml
   $ sed -i '/rule_files:/a\    \- /etc/config/<nf-alertsname>' /tmp/
   tempPrometheusConfig.yaml
   ```

3. Update the configuration map with the updated file.

   ```
   $ kubectl -n <namespace> replace configmap <OSO-prometheus-configmap-name>
   -f
       /tmp/tempPrometheusConfig.yaml
   ```

4. Patch the NF alert rules in OSO Prometheus configuration map by mentioning the Alert-rule file path.

   ```
   $ kubectl patch configmap <OSO-prometheus-configmap-name> -n <namespace> --
   type merge --patch "$(cat ./NF_altertrules.yaml)"
   ```

6 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/.

7 Update Kafka Certificates in Kubernetes secret

This section describes how to update the Kafka CA certificate, TLS certificate, or private key stored in Kubernetes Secrets during a maintenance window.

Update the Kafka certificate or key using the following commands:

> **Note**
>
> - In APM custom-values file (`ocoso_csar_26_1_200_0_0_apm_custom_values.yaml`), we assume that `apmTLS.autoReloadTLS.enable` is *true*, if this value is *false*, follow the section below and run rollout restart of APM deployment.
> - Updates to Kafka certificates happen only during the maintenance window.

1. Encode the new certificates *ca.crt* or *tls.key* or *tls.crt* with base64:

```
CACRT=$(base64 -w0 ca.crt)
CRT=$(base64 -w0 tls.crt)
KEY=$(base64 -w0 tls.key)
```

2. Update the contents using the Kubectl patch commands:

```
kubectl patch secret kafka-ca-secret -n <oso-namespace> -p "{\"data\":
{\"ca.crt\":\"$CACRT\"}}"
kubectl patch secret my-tls-secret -n <oso-namespace> -p "{\"data\":
{\"tls.crt\":\"$CRT\",\"tls.key\":\"$KEY\"}}"
```

3. Run the following command to perform a rolling restart of the APM deployment:

```
kubectl rollout restart deployment <apm-deployment-name>
```