Oracle® Communications

Cloud Native Core, Cloud Native Environment User Guide

Release 23.4.6
F89908-13
March 2025

Oracle Communications Cloud Native Core, Cloud Native Environment User Guide, Release 23.4.6

F89908-13

Contents

A References

My Oracle Support

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

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

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

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

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

Acronyms

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

Table Acronyms and Terminologies

Acronyms	Definition
5G NF	3GPP 5G Network Function
Bastion Host	The Bastion Host is a node that hosts general orchestration support for the site. The Bastion Node runs as a virtual machine on a Kubernetes Master Host. It is used to host the automation environment and run the install automation. The install automation provisions and configures all other hosts, nodes, and switches within the frame. After the installation process is completed, the Bastion Host continues to serve as the customer gateway to cluster operations and control.
BIOS	Basic Input Output System
CCV	Custom Configurable Volumes
CLI	Command Line Interface
Cluster	A collection of hosts and nodes dedicated to providing a cloud native runtime environment for containerized services and applications. The cluster is comprised of the collection of Master and Worker Nodes and is managed by Kubernetes.
CNE	Oracle Communications Cloud Native Core, Cloud Native Environment
CNC Console	Oracle Communications Cloud Native Configuration Console
Computer or Machine	A computer is a machine or device that performs processes, calculations, and operations based on instructions provided by a software or hardware program. Machine and computer are equivalent and interchangeable. Therefore, a virtual machine or physical machine is equivalent to a virtual computer or physical computer.
Container	An encapsulated software service. All 5G applications and Operations, Administration, Maintenance (OAM) functions are delivered as containerized software. The purpose of the OCCNE is to host containerized software providing 5G Network Functions (NFs) and services.
CSI	Container Storage Interface
DB	Database
DBMS	Database Management System
DHCP(D)	Dynamic Host Configuration Protocol
DNS	Domain Name Server
FQDN	Fully Qualified Domain name
GUI	Graphical User Interface
HDD	Hard Disk Drive
Host	A **host** is a computer that is loaded with an Operating System (OS) and is accessible over a network. This includes physical or virtual machines. Though **host** and **node** have different meanings, they can often be confused. Therefore, in this document, the **host** refers to the role of the physical computer in the CNE solution, and **Host OS** refers to the operating system running on the physical computer.
HP	Hewlett Packard
HPE	Hewlett Packard Enterprise
HTTP	HyperText Transfer Protocol

Table (Cont.) Acronyms and Terminologies

Acronyms	Definition
Installer Bootstrap Host	As an early step in the site installation process, one of the hosts is minimally provisioned to act as an Installer Bootstrap Host. The Installer Bootstrap Host has a very short lifetime since its job is to provision a Bastion Host. The Bastion Host performs the remainder of the OCCNE installation. The Installer Bootstrap Host is also known as the Bootstrap Host. Later in the installation process, this Bootstrap Host (RMS1) is reprovisioned as a Kubernetes Master Host.
iLO	HPE Integrated Lights-Out Management System
IP	Internet Protocol, can be used as shorthand to refer to an IP layer 3 address.
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
ISO	International Organization for Standardization; generally used as shorthand to refer to an ISO 9660 optical disk file system image
KVM	Keyboard, Video, Mouse. This abbreviation is a homograph of another meaning for the same abbreviation. Where these are used within the document, the document will provide context to understand which one is being referenced.
KVM	Kernel Virtual Machine. This abbreviation is a homograph of another meaning for the same abbreviation. Where these are used within the document, the document will provide context to understand which one is being referenced.
K8s	Shorthand alias for Kubernetes
K8s Controller Node	Some nodes in the system such as RMS 1, 2, and 3 are dedicated to providing container management. These nodes are responsible for managing all of the containerized workloads that run on the Kubernetes Worker Nodes.
K8s Worker Node	Some nodes in the system such as the blade servers within the enclosure, are dedicated to hosting containerized software and providing the 5G application services.
LB	Load Balancer
LBVM	Load Balancer Virtual Machine
MAC	Media Access Control address
Management Host	The Management Host is a physical machine in the RMS frame that contains a special configuration to support hardware installation and configuration of other components within a frame. For CNE, there is one machine, RMS1, with dedicated connectivity to out of band (OOB) interfaces on the Top of Rack switches. The OOB interfaces provide the connectivity required to initialize ToR switches. When referring to a machine as a **Management Host**, the context is related to its OOB connections that are unique to the RMS1 configuration.
MBE	Minimal Bootstrapping Environment
NFS	Network File System
Node	In general use, a "**node**" is any system or device connected to a network. A "**node**" is usually a networking endpoint. In this document, a "**node**" refers to a service within the CNE, such as "**Kubernetes Controller Node**", which provides the Kubernetes etcd data and serves as a management entity for the cluster.
NTP	Network Time Protocol
OAM	Operations, Administration, Maintenance

Table (Cont.) Acronyms and Terminologies

Acronyms	Definition
OS	Operating System
OSD	Object Storage Device
OSDC	Oracle Software Download Center
PAP	Peer Address Pool
PKI	Public Key Infrastructure
POAP	PowerOn Auto Provisioning
PSP	Pod Security Policies
PVC	Persistent Volume Claim
PXE	Pre-Boot Execution Environment
RAID	Redundant Array of Independent Disks
RAM	Random Access Memory
RBSU	ROM Based Setup Utility
Rack Mount Server (RMS)	Rack Mount Server denotes a profile for a specific computer designed to support a server function. An RMS is mounted directly into a frame and is individually powered and networked.
RPM	Red Hat Package Manager
SAS	Serial Attached SCSI
Server	A server is a physical computer. Servers are rack-mounted servers (RMS).
SSD	Solid State Drive
TAR	Short for Tape Archive, and sometimes referred to as tarball, a file that has the TAR file extension is a file in the Consolidated Unix Archive format.
TCP	Transmission Control Protocol
TGZ	Tar and GNUzip
TLA	Three Letter Acronym
TLD	Top Level Domain
ToR	Top of Rack - Colloquial term for the pair of Cisco 93180YC-EX switches
UEFI	Unified Extensible Firmware Interface
ULN	Unbreakable Linux Network
URL	Uniform Resource Locator
VM	Virtual Machine
VCD	vCloud Director
vCNE	Virtualized CNE. A Virtualized CNE is a cloud native environment that is deployed on VMs instead of Bare Metal servers.
VDC	Virtual Data Center
VSP	Virtual Serial Port
YUM	Yellowdog Updator, Modified (a Linux Package Manager)

What's New in This Guide

This section introduces the documentation updates for release 23.4.x.

Release 23.4.6-F89908-13, March 2025

- Updated the procedure to add a Kubernetes worker node in the Adding a Kubernetes Worker Node section.
- Updated the procedure to renew Kubernetes certificates in the Renewing Kubernetes Certificates section.

Release 23.4.6-F89908-12, November 2024

- Added the Premaintenance Check for VMware Deployments section to provide details about verifying the content of the `compute/main.tf` and `compute-lbvm/main.tf` files before performing any maintenance procedure in a VMware deployment.
- Updated the procedure to configure SNMP trap receivers in the Configuring SNMP Trap Destinations section.

Release 23.4.6-F89908-08, September 2024

Updated the procedure to update Openstack credentials in the Updating OpenStack Credentials section.

Release 23.4.6-F89908-07, July 2024

Removed the following API server alerts from the Kubernetes Alerts section as they are not applicable to this release:

- APISERVER_CERTIFICATE_EXPIRATION_90D
- APISERVER_CERTIFICATE_EXPIRATION_30D
- APISERVER_CERTIFICATE_EXPIRATION_7D

Release 23.4.4-F89908-06, May 2024

Added the `a_record` request parameter in the following sections to add or delete an SRV record:

- Adding an SRV Record
- Deleting an SRV Record

Release 23.4.4-F89908-04, April 2024

Updated the steps to update OpenStack credentials in the Updating OpenStack Credentials section.

Release 23.4.1-F89908-03, February 2024

No updates are made to this document in this release.

Release 23.4.0-F89908-02, February 2024

Updated the common service versions in the Frequently Used Common Services section.

Release 23.4.0-F89908-01, December 2023

- Updated the release number to 23.4.x in the entire document.
- Added details about Kubernetes secret encryption in the Security section.
- Updated the OpenSearch data node count to 5 and added additional information about configuring the number of data nodes for OpenSearch in the Redundancy section.
- Made the following updates in the Resource Utilization section:
  - Added a note to state that the resource allocation limits provided for the observability tools can vary depending on various scenarios and that users must consult with workload teams to come up with suitable allocation limits.
  - Updated the resource utilization values of the following common services:
    - Prometheus
    - OpenSearch Master
    - OpenSearch Data
    - OpenSearch Client
    - Fluentd OpenSearch
- Made the following updates in the Common Services Alerts section:
  - Added the `OPENSEARCH_DATA_PVC_NEARLY_FULL` alert.
  - Added additional recommended actions for the `OPENSEARCH_TOO_FEW_DATA_NODES_RUNNING` alert.
- Updated the CNE cluster version requirement details in the following sections:
  - Activating Local DNS
  - Prerequisites for Activating Local DNS
  - Prerequisites for Adding SRV Records
- Added details about default Grafana dashboards in the Metrics section.
- Updated the Accessing the Bastion Host section to add additional information about accessing, managing, and troubleshooting Bastion Hosts.
- Updated the procedures to update MTU value in the Changing Network MTU section.
- Added a new OpenSearch component and updated the PCV values in the PV size table in the Changing OpenSearch Storage Allocation section.
- Removed kubelet certificate details and updated the comments in the Renewing Kubernetes Certificates section.
- Updated the steps to add a worker node for BareMetal or VMware deployment in the Adding a Kubernetes Worker Node section.
- Updated the file `hosts.ini` name in the Removing a Kubernetes Worker Node section.
- Updated the steps to update OpenStack credentials in the Updating OpenStack Credentials section.

1 Introduction

This document provides information to operate and maintain Oracle Communications Cloud Native Core, Cloud Native Environment (CNE) clusters.

1.1 Overview

CNE is an essential infrastructure code that provisions, configures, and manages cloud native environments.

1.2 References

Refer to the following documents for more information about CNE:

- *Oracle Communications Cloud Native Core, Cloud Native Environment Installation, Upgrade, and Fault Recovery Guide*
- *Oracle Communications Cloud Native Core, Cloud Native Environment Network Impact Report*

1.3 Key Terms

The following table lists the terminologies used in this document.

Table 1-1 Key Terms

Term	Definition
Host	A **host** is a computer that is loaded with an Operating System (OS) and is accessible over a network. This includes physical or virtual machines. Though **host** and **node** have different meanings, they can often be confused. Therefore, in this document, the **host** refers to the role of the physical computer in the CNE solution, and **Host OS** refers to the operating system running on the physical computer.
Node	In general use, a "**node**" is any system or device connected to a network. A "**node**" is usually a networking endpoint. In this document, a "**node**" refers to a service within the CNE, such as "**Kubernetes Controller Node**", which provides the Kubernetes etcd data and serves as a management entity for the cluster.
Computer or Machine	A computer is a machine or device that performs processes, calculations, and operations based on instructions provided by a software or hardware program. Machine and computer are equivalent and interchangeable. Therefore, a virtual machine or physical machine is equivalent to a virtual computer or physical computer.
Server	A server is a physical computer. Servers are rack-mounted servers (RMS).
Rack Mount Server (RMS)	RMS denotes a profile for a specific computer designed to support a server function. An RMS is mounted directly into a frame and is individually powered and networked.

Table 1-1 (Cont.) Key Terms

Term	Definition
Management Host	The Management Host is a physical machine in the RMS frame that contains a special configuration to support hardware installation and configuration of other components within a frame. For CNE, there is one machine, RMS1, with dedicated connectivity to out of band (OOB) interfaces on the Top of Rack switches. The OOB interfaces provide the connectivity required to initialize ToR switches. When referring to a machine as a **Management Host**, the context is related to its OOB connections that are unique to the RMS1 configuration.
Bastion Host	The Bastion Host is a node that hosts general orchestration support for the site. The Bastion Node runs as a virtual machine on a Kubernetes Master Host. It is used to host the automation environment and run the install automation. The install automation provisions and configures all other hosts, nodes, and switches within the frame. After the installation process is completed, the Bastion Host continues to serve as the customer gateway to cluster operations and control.
Installer Bootstrap Host	As an early step in the site installation process, one of the hosts is minimally provisioned to act as an Installer Bootstrap Host. The Installer Bootstrap Host has a very short lifetime since its job is to provision a Bastion Host. The Bastion Host performs the remainder of the OCCNE installation. The Installer Bootstrap Host is also known as the Bootstrap Host. Later in the installation process, this Bootstrap Host (RMS1) is reprovisioned as a Kubernetes Master Host.
K8s Controller Node	Some nodes in the system such as RMS 1, 2, and 3 are dedicated to providing container management. These nodes are responsible for managing all of the containerized workloads that run on the Kubernetes Worker Nodes.
K8s Worker Node	Some nodes in the system such as the blade servers within the enclosure, are dedicated to hosting containerized software and providing the 5G application services.
Container	An encapsulated software service. All 5G applications and Operations, Administration, Maintenance (OAM) functions are delivered as containerized software. The purpose of the OCCNE is to host containerized software providing 5G Network Functions (NFs) and services.
Cluster	A collection of hosts and nodes dedicated to providing a cloud native runtime environment for containerized services and applications. The cluster is comprised of the collection of Master and Worker Nodes and is managed by Kubernetes.
Virtualized CNE	A Virtualized CNE is a cloud native environment that is deployed on VMs instead of Bare Metal servers.

2
Deployment Models

Oracle Communications Cloud Native Core, Cloud Native Environment (CNE) can be installed either on bare metal servers or virtual machines. For more information about how to install and configure CNE for each deployment model, see *Oracle Communications Cloud Native Core, Cloud Native Environment Installation, Upgrade, and Fault Recovery Guide*. The following sections explain different CNE deployments.

2.1 Bare Metal Deployment

CNE is installed on HP Gen 8, Gen 9, and Gen 10 servers. CNE can also be installed on the Netra X8-2 and X9-2 servers. CNE requires the following servers for bare metal deployments:

- 3 Kubernetes controller servers
- A minimum of 6 Kubernetes worker servers

There are no additional hardware required for installing or booting the Bootstrap Host and the Bastion Hosts since they are created as virtual machines on the Kubernetes controller nodes. For more information on installing and configuring Bootstrap Host and Bastion Hosts, see the *Oracle Communications Cloud Native Core, Cloud Native Environment Installation, Upgrade, and Fault Recovery Guide*.

2.2 Virtualized Deployment

CNE can be installed on OpenStack or VMware virtualized infrastructures. CNE uses Terraform to acquire virtual resources such as virtual machines, networking, security rules, and so on, from the virtual infrastructures. CNE is installed onto these virtual resources. The following virtual machines are required for a virtualized deployment of CNE:

- 1 Bootstrap Host VM (can be uninstalled after installation is complete)
- 2 Bastion Host VMs
- 3 Kubernetes controller VMs
- A minimum of 6 Kubernetes worker VMs

For more information on installing and configuring Bootstrap Host and Bastion Hosts, see *Oracle Communications Cloud Native Core, Cloud Native Environment Installation, Upgrade, and Fault Recovery Guide*.

3 CNE Services

CNE provides the following common services for all the installed applications:

- Runtime Environment - CNE provides a runtime environment in which you can run all the cloud native applications.
- Automation - CNE provides automation solutions to deploy, upgrade, and maintain cloud native applications.
- Security - CNE provides multi-level security measures to protect against malicious attacks.
- Redundancy - CNE provides redundancy or high availability through the specification of anti-affinity rules at the infrastructure level.
- Observability - CNE provides services to capture metrics, logs, and traces for both itself (CNE) and the cloud native applications.
- Maintenance Access - CNE provides a Bastion Host to access the Kubernetes cluster for maintenance purposes.

3.1 Runtime Environment

The primary function of CNE is to provide a runtime environment by using Kubernetes for cloud native applications.

> **Note**
>
> CNE 23.4.x supports Kubernetes version 1.27.x.

3.1.1 Kubernetes Cluster

The Kubernetes cluster in CNE contains three controller nodes. Kubernetes uses these controller nodes to coordinate the execution of application workloads across many worker nodes.

The Kubernetes cluster can:

- sustain the loss of one controller node with no impact to the CNE service.
- tolerate the loss of two controller nodes without losing the cluster configuration data. Kubernetes is placed in the *read-only* mode.

In the case where two controller nodes are lost, Kubernetes is in the *read-only* mode. The application workload continues to run on the worker node. However, no changes can be made to the cluster configuration data. Due to this situation, it becomes impossible to install new applications and make changes to the application-specific custom resources that are used to change an application's configuration.

The CNE Kubernetes can contain many worker nodes and the minimum recommendation is six. Kubernetes distributes workloads across all the available worker nodes. The Kubernetes placement algorithm attempts to balance workloads across all available worker nodes so that

no worker node is overloaded. Kubernetes also attempts to honor the affinity or anti-affinity rules specified by applications.

Additional Kubernetes cluster details are as follows:

- Cluster configuration data, as well as application-specific custom resources, are stored in etcd.
- Workloads in the Kubernetes cluster are run by the containerd runtime.
- NTP time synchronization is maintained by chrony.

3.1.2 Networking

In-cluster Networking

CNE includes the Calico plug-in for networking within the Kubernetes cluster. Calico provides network security solutions for containers. Calico is known for its performance, flexibility, and power.

External Networking

You must configure at least one external network to allow the applications that run in CNE to communicate with applications that are outside of the CNE platform.

You can also configure multiple external networks to CNE to allow specific applications to communicate on networks that are dedicated for specific purposes. For more information on configuring external networks, see *Oracle Communications Cloud Native Core, Cloud Native Environment Installation, Upgrade, and Fault Recovery Guide*.

3.1.3 Storage

Bare Metal Deployment Storage

When CNE is deployed on Bare Metal servers, it constructs a Ceph storage cluster from the disk drives attached to the servers designated as Kubernetes worker nodes. The Ceph cluster is then integrated with Kubernetes so that CNE common services and applications that run on CNE can use the Ceph cluster for persistent data storage.

Virtual Deployment Storage

When CNE is deployed on a virtual infrastructure, it delivers a *cloud provider* that corresponds to the Kubernetes cloud provider interface. For more information, see cloud provider interface. The cloud provider interacts with the Kubernetes cluster and the storage manager to provide storage when the applications request it. For example, OpenStack supports *Cinder* storage manager and VMware supports *vSphere* storage manager.

After Kubernetes has access to persistent storage, CNE then creates several Kubernetes `StorageClass` classes for the common services and applications to use.

- CNE creates a storage class named `Standard`, which allows applications to store application-specific data. `Standard` is the default storage class and is used when the applications create a Persistent Volume Claim (PVC) with no StorageClass reference.
- CNE creates one storage class to allow Prometheus to store metrics data.
- CNE creates two storage classes to allow OpenSearch to store log and trace data.

3.1.4 Ingress Traffic Delivery

Each application must create a Kubernetes Service of the type **LoadBalancer** to allow the clients from an external network to connect with the applications within the CNE. Each **LoadBalancer** service is assigned to a service IP from an external network. CNE provides load balancers to ensure that ingress traffic from external clients is distributed evenly across Kubernetes worker nodes that host the application pods.

For Bare Metal deployments, the top of rack (ToR) switches perform the load balancing function. For virtualized deployments, dedicated virtual machines perform the load balancing function.

For more information on selecting an external network for an NF application, see the installation instructions of the NF.

> **Note**
>
> CNE does not support **NodePort** services because they are considered insecure in a production environment.

Ingress traffic distribution

CNE Load Balancers distribute ingress traffic evenly across all Kubernetes worker nodes. Each ingress message is distributed in a round-robin fashion to all in-service Kubernetes worker nodes, regardless of the sender, or destination IP or port.

> **Note**
>
> - CNE Load Balancers distribute ingress traffic to Kubernetes worker nodes and not pods. Once an ingress message is delivered to a Kubernetes worker node, Kubernetes selects a pod belonging to the target service and delivers the ingress message to the pod. For more information on how Kubernetes delivers ingress messages to pods in the target service, see Kubernetes Network Model.
> - CNE Load Balancers support only Cluster value for the spec.externalTrafficPolicy field in the Kubernetes service specification and do not support Local value.

3.1.5 Egress Traffic Delivery

CNE uses the Load Balancers used for ingress traffic distribution to deliver egress requests to servers outside the CNE cluster. CNE Load Balancers also perform Egress Network Address Translation (NAT) on the egress requests to ensure that the source IP field of all egress requests contains an IP address that belongs to the external network to which the egress request is delivered. This is done to ensure that the egress requests are not rejected by security rules on the external networks.

For more information on selecting an external network for an NF application, see to the installation instructions of the NF.

3.1.6 DNS Query Services

CoreDNS is used to resolve Domain Name System (DNS) queries for services within the Kubernetes cluster. DNS queries for services outside the Kubernetes cluster are routed to DNS nameservers running on the Bastion Hosts, which in turn use customer-specified DNS servers outside the CNE to resolve these DNS queries.

3.1.6.1 Local DNS

Local DNS feature is a reconfiguration of core DNS (CoreDNS) to support external hostname resolution. Local DNS allows the pods and services running inside the CNE cluster to connect with the ones running outside the CNE cluster using core DNS. That is, when Local DNS is enabled, CNE routes the connection to external hosts through core DNS rather than the nameservers on the Bastion Hosts. For information about activating this feature, see Activating Local DNS.

Local DNS provides Local DNS API, that runs on each Bastion server, to define the external hostnames as custom records. These records are used to identify and locate hosts, services, or NFs outside the CNE cluster. Local DNS supports adding and removing the following records:

- A Records: allows to define the hostname and the IP address to locate the service
- SRV Records: allows to define the location (hostname and port number) of a specific service and how your domain handles the service

For information about adding and removing these records, see Adding and Removing DNS Records.

For additional details about this feature, see Activating and Configuring Local DNS.

3.2 Automation

CNE provides considerable automation throughout the phases of an application's deployment, upgrade, and maintenance cycles.

3.2.1 Deployment

CNE provides the ***Helm*** application to automate the deployment of applications into the Kubernetes cluster. The applications must provide a ***Helm chart*** to perform automated deployment. For more information about deploying an application in CNE, see Maintenance Procedures section.

You can deploy CNE automatically through custom scripts provided with the CNE platform. For more information about installing CNE, see *Oracle Communications Cloud Native Core, Cloud Native Environment Installation, Upgrade, and Fault Recovery Guide*.

3.2.2 Upgrade

CNE provides ***Helm*** to automate the upgrade of applications in the Kubernetes cluster. The applications must provide a ***Helm chart*** to perform an automated software upgrade. For more information about an application upgrade in CNE, see Maintenance Procedures section.

You can upgrade the CNE platform automatically through the execution of a Continuous Delivery (CD) pipeline. For more information on CNE upgrade instructions, see *Oracle*

Communications Cloud Native Core, Cloud Native Environment Installation, Upgrade, and Fault Recovery Guide.

3.2.3 Maintenance

CNE provides a combination of automated and manual procedures for performing maintenance operations on the Kubernetes cluster and common services. For more instructions about performing maintenance operations, see Maintenance Procedures section.

3.3 Security

CNE provides multi-level security measures to protect against malicious attacks.

- All Kubernetes nodes and Bastion Hosts are security-hardened to prevent unauthorized access.
- Maintenance operations on the Kubernetes cluster can only be performed from the Bastion Hosts to prevent Denial of Service (DOS) attacks against the Kubernetes cluster.
- Only the Kubernetes worker nodes can access the Kubernetes controller nodes to protect sensitive cluster data.
- Kyverno provides the policies to ensure that malicious applications don't corrupt the Kubernetes controller, worker nodes, and cluster data.
- Bastion Host container registry is secured with TLS and can be accessed from CNE only.
- Kubernetes secrets are encrypted before they are stored using secretbox. This ensures that the Kubernetes secrets are secure and accessible to authorized users only.

For more information on security hardening, see *Oracle Communications Cloud Native Core, Security Guide*.

3.4 Redundancy

This section provides detail about maintaining redundancy in Kubernetes, ingress load balancing, common services, and Bastion Host.

Infrastructure

In virtualized infrastructures, CNE requires the specification of antiaffinity rules to distribute Kubernetes controller and worker nodes across several physical servers. In Bare Metal infrastructures, each Kubernetes controller and worker node are placed on its own physical server.

Kubernetes

For the Kubernetes cluster, CNE runs three controller nodes with an etcd node on each controller node.

The etcd node allows the cluster to:

- sustain the loss of one controller node with no impact on the service.
- sustain the loss of two controller nodes without losing the cluster data in the etcd database. However, in cases where two controller nodes are lost, Kubernetes is placed in *read-only* mode. In the *read-only* mode, the creation of new pods and the deployment of new services are not possible.

- restore etcd data from the backup when all the three controller nodes fail. When all the three controller nodes fail, all the cluster data in etcd is lost, and it is essential to restore data from the backup to run the operations.

CNE uses internal load balancers to distribute API requests from applications running in worker nodes evenly across all controller nodes.

Common Services

- OpenSearch uses three master nodes, three ingress nodes, and five data nodes by default. The number of data nodes provisioned can be configured as per the requirement. For more details, see the "Common Installation Configuration" section in *Oracle Communications Cloud Native Core, Cloud Native Environment Installation, Upgrade, and Fault Recovery Guide*. The system can withstand failure of one node in each of these groups with no impact on the service.
- Fluentd OpenSearch runs as a `DaemonSet`, where one Fluentd OpenSearch pod runs on each Kubernetes worker node. When the worker node is up and running, Fluentd OpenSearch sends the logs for that node to OpenSearch.
- Two Prometheus instances are run simultaneously to provide redundancy. Each Prometheus instance independently collects and stores metrics from all CNE services and all applications running in the CNE Kubernetes cluster. Automatic deduplication removes the repetitive metric information in Prometheus.

Kubernetes distributes common service pods across Kubernetes worker nodes such that even the loss of one worker node has no impact on the service. Kubernetes automatically reschedules the failed pods to maintain redundancy.

Ingress and Egress Traffic Delivery

- For virtual installations, the Load Balancer VMs (LBVM) perform both ingress and egress traffic delivery.
- Two Load Balancer VMs are run for each configured external network in virtual installations. These LB VMs run in an active or standby configuration. Only one LB VM (the active LB VM) processes the ingress traffic for a given external network.
- The active LB VM is continuously monitored. If the active LB VM fails, the standby LB VM is automatically promoted to be the active LB VM.
- If the virtual infrastructure is able to recover the failed LB VM, The recovered LB VM automatically becomes the standby LB VM, and load balancing redundancy is restored. If the infrastructure is NOT able to recover the failed LB VM, you must perform the *Restoring a Failed Load Balancer* fault recovery procedure described in *Oracle Communications Cloud Native Core, Cloud Native Environment Installation, Upgrade, and Fault Recovery Guide* to restore load balancing redundancy.

Bastion Hosts

- There are two Bastion Hosts per CNE.
- Bastion Hosts run in an active or standby configuration.
- Bastion Host health is monitored from an application that runs in the Kubernetes cluster, and instructs standby Bastion Host to take over when active Bastion Host fails.
- A DaemonSet running on each Kubernetes worker node ensures that the worker node always retrieves container images and Helm charts from the active Bastion Host.
- Container images, Helm charts, and other files essential to CNE internal operations are synchronized from the active Bastion Host to the standby Bastion Host periodically. This

way, when an active Bastion Host fails, the standby appears as an identical replacement upon switchover.

3.5 Observability

CNE provides services to capture metrics, logs, and traces for both itself and the cloud native applications. You can use the observability data for problem detection, troubleshooting, and preventive maintenance. CNE also includes tools to filter, view, and analyze the observability data.

3.5.1 Metrics

Metrics collection

Prometheus collects the application metrics and CNE metrics from the following services:

- The servers (Bare Metal) or VMs (virtualized) that host the Kubernetes cluster. The ***node-exporter*** service collects hardware and Operating system (OS) metrics exposed by the Linux OS.
- The ***kube-state-metrics*** service collects information about the state of all Kubernetes objects. Since Kubernetes uses objects to store information about all nodes, deployments, and pods in the Kubernetes cluster, ***kube-state-metrics*** effectively captures metrics on all aspects of the Kubernetes cluster.
- All of the CNE services generate metrics that Prometheus collects and stores.

Metrics storage

Prometheus stores the application metrics and CNE metrics in an internal time-series database.

Metrics filtering and viewing

Grafana allows you to view and filter metrics from Prometheus. CNE offers some default Grafana dashboards which can be cloned and customized as per your requirement. For more information about these default dashboards, see CNE Grafana Dashboards.

For more information about CNE Metrics, see Metrics.

3.5.2 Alerts

CNE uses ***AlertManager*** to raise alerts. These alerts notify the user when any of its common services are in abnormal conditions. CNE delivers alerts and Simple Network Management Protocol (SNMP) traps to an external SNMP trap manager. For more information about the detailed CNE definition and description of each alert, see Alerts section.

Applications deployed on CNE can define their alerts to inform the user of problems specific to each application. For instructions about applications loading the alerting rules, see Maintenance Procedures.

3.5.3 Logs

Logs collection

Fluentd OpenSearch collects logs for applications installed in CNE and CNE services logs.

Logs storage

CNE stores its logs in Oracle OpenSearch. Each log message is written to an OpenSearch index. The source of the log message determines which index the record is written to is as follows:

- If the applications in the CNE generates the logs, these logs are stored in the current day's application index named *logstash-YYYY.MM.DD*.
- If the CNE services generate the logs, these logs are stored in the current day's CNE index named *occne-logstash-YYYY.MM.DD*.

CNE creates new log indices each day. Each day's indices are uniquely named by appending the current date to the index name as a suffix. For example, all application logs generated on November 13, 2021, are written to the *logstash-2021.11.13* index. By default, indices are retained for seven days.

Only current-day indices are writable and they are configured for efficient storage of new log messages. Current-day indices are called "hot" indices and stored on "hot" data nodes. At the end of each day, new current-day indices are created. At the same time, the previous day's indices are marked as read-only, so no new log messages are written to them, and compressed, to allow for more efficient storage. These previous-day indices are considered "warm" indices, and are stored on "warm" data nodes.

Log filtering and viewing

Oracle OpenSearch Dashboard filters and views logs that are available in OpenSearch.

3.5.4 Traces

Trace collection

Jaeger collects traces from applications deployed in CNE.

> **Note**
>
> Jaeger captures only a small percentage of all application message flows. The default capture rate is .01% (1 in 10,000 message flows).

Traces are not collected from CNE services.

Trace storage

CNE stores its traces in Oracle OpenSearch. The applications deployed in CNE generate traces and stores them in the *jaeger-trace-YYYY.MM.DD* index. CNE creates a new Jaeger index each day. For example, all traces generated by applications on November 13, 2021, are written to the *jaeger-trace-2021.11.13* index.

Trace filtering and viewing

Oracle OpenSearch Dashboard filters and views the traces that are available in Oracle OpenSearch.

3.6 Maintenance Access

To access the Kubernetes cluster for maintenance purposes, CNE provides 2 Bastion Hosts. Each Bastion Host provides command-line access to several troubleshooting and maintenance tools such as kubectl, Helm, and Jenkins.

3.7 Frequently Used Common Services

This section provides some of the frequently used common services and their version supported by CNE 23.4.x. You can find the complete list of third-party services in the dependencies TGZ file provided as part of the software delivery package.

Table 3-1 Frequently Used Common Services

Common Service	Supported Version
AlertManager	0.25.0
Grafana	9.5.3
Prometheus	2.44.0
Calico	3.25.2
cert-manager	1.12.4
Containerd	1.7.5
Fluentd - OpenSearch	1.16.2
HAProxy	2.6.7
Helm	3.12.3
Istio	1.18.2
Jaeger	1.45.0
Kubernetes	1.27.5
Kyverno	1.9.0
MetalLB	0.13.11
Oracle OpenSearch	2.3.0
Oracle OpenSearch Dashboard	2.3.0
Prometheus Operator	0.65.1

4 Resource Utilization

CNE constrains resources such as CPU and RAM to each common service. Resource constraints help to ensure that the services don't consume excess resources.

During initial CNE deployment, each service is provided an initial CPU and RAM allocation. Each service is allowed to consume each resource (CPU and RAM) to a specified upper limit while it continues to run.

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 initial allocation limit and the upper limit are set to the same value. The resource requests and limits are provided in the following table:

> **Note**
>
> Observability tools such as Prometheus,OpenSearch, Fleuntd, and Jaeger perform resource intensive operations. Even a small change of events can increase the resource consumption exponentially. For example:
>
> - Changing the log severity level from `WARN` to `INFO` increases the CPU and memory usage dramatically.
> - Adding new metric labels or enabling new monitors have big impact on the performance of metric ingestion and can cause resource starvation.
> - Adding more traces to Jaeger spikes the ingestion rate in OpenSearch and causes the data nodes to starve.
>
> Therefore, for observability tools, the limits provided in the following table must not be considered as production ready values as they vary greatly depending on the workload on CNE. It is strongly recommended to consult with the workload teams to get indicators for observability resource allocation.

Table 4-1 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	4000	16384	16384	2
Prometheus Node Exporter	800	800	512	512	1 per node
Prometheus Operator	100	200	100	200	1
Prometheus AlertManager	20	20	64	64	2
Prometheus Kube State Metrics	20	20	32	100	1
Promxy	100	100	512	512	1
OpenSearch Master	1000	1000	100	2048	3
OpenSearch Data	1000	4000	16384	101904 (100Gi)	7
OpenSearch Client	1000	3000	100	32609(32Gi)	3

Table 4-1 (Cont.) CPU and RAM Resource Requests and Limits

Service	CPU Initial Request (m)	CPU Limit (m)	RAM Initial Request (Mi)	RAM Limit (Mi)	Instances
OpenSearch Dashboard	100	100	512	512	1
occne-metrics-server	100	100	200	200	1
occne-alertmanager-snmp-notifier	100	100	128	128	1
Fluentd OpenSearch	100	500	128	12228(12Gi)	1 per worker node
Jaeger Collector	500	1250	512	1024	1
Jaeger query	256	500	128	512	1
MetalLB Controller	100	100	100	100	1
MetalLB Speaker	100	100	100	100	1 per worker node
LB Controller (vCNE only)	10	500	128	1024	1
Egress Controller	100	1000	200	500	1 per worker node
Bastion Controller	10	200	128	256	1
Kyverno	100	200	256	512	3

The overall common services resource usage varies on each worker node. The common services listed in Table 4-1 are evenly distributed across all worker nodes in the Kubernetes cluster.

5 Metrics

CNE uses Prometheus to collect metrics from CNE services. These metrics are used by the alert rules defined by CNE to detect abnormal conditions. This section provides details about CNE metrics.

5.1 Load Balancer Metrics

This section provides details about Load Balancer metrics.

Table 5-1 vcne_lb_status

Field	Details
Description	The status of the Load Balancer VM. The value of this metric is: • "0" if the LB VM is UP • "1" if the LB VM is DOWN
Type	Gauge
Dimensions	`external_network`: The name of the IP pool for which the LB is performing load balancing. `role`: The role of the LB VM. The value is set to ACTIVE or STANDBY. `ip_address`: The IP of the LB VM `name`: The name of the LB VM

5.2 Bastion Host Metrics

This section provides details about Bastion Host metrics.

Table 5-2 bastion_host_status

Field	Details
Description	The status of the Bastion Host. The value of this metric is: • "2" if the Bastion Host is ACTIVE • "1" if the Bastion Host is STANDBY • "0" if the Bastion Host FAILED
Type	Gauge
Dimensions	`ip_address`: The IP of the Bastion Host `name`: The name of the Bastion Host

6 Alerts

Alerts are used to detect abnormal conditions in CNE and notify the user when any of the common services are not operating normally.

Each alert rule uses the values of one or more metrics stored in Prometheus to identify the abnormal conditions. Prometheus periodically evaluates each rule to ensure that CNE is operating normally. When rule evaluation indicates an abnormal condition, Prometheus sends an alert to the AlertManager. The resulting alert contains information about what part of the CNE cluster is affected for troubleshooting. Each alert is assigned with a severity level to inform the user of the seriousness of the alerting condition. This section provides details about CNE alerts.

6.1 Kubernetes Alerts

This section provides details about Kubernetes alerts.

Table 6-1 DISK_SPACE_LOW

Field	Details
Description	Cluster-name : {{ $externalLabels.cluster }} Disk space is almost RUNNING OUT for kubernetes node {{ $labels.kubernetes_node }} for partition {{ $labels.mountpoint }}. Available space is {{$value }}%(< 20% left). Instance = {{ $labels.instance }}
Summary	Cluster-name : {{ $externalLabels.cluster }} Disk space is RUNNING OUT on node {{ $labels.kubernetes_node }} for partition {{ $labels.mountpoint }}
Cause	Disk space is running out on node. More than 80% of the allocated resources is consumed on the node.
Recommended Actions	• In case of vCNE, the flavour of the worker nodes can be increased to a larger flavor with more storage. • Additional space can also be reclaimed by running "`podman system prune -fa`" to remove any unreferenced image layers. • Verify how much space is being consumed by `/var/log` partition. If it is consuming a lot of space, logs can be rotated or shrinked to reclaim some space.

Table 6-2 CPU_LOAD_HIGH

Field	Details
Description	CPU load is high on host <*node name*>CPU load {{ $value }} %Instance : {{ $labels.instance }}
Summary	CPU load is high on host {{ $labels.kubernetes_node }}
Cause	CPU load is more than 80% of the allocated resources on the node.

Table 6-2 (Cont.) CPU_LOAD_HIGH

Field	Details
Recommended Actions	• In case of vCNE, the flavour of the worker nodes can be increased to a larger flavour with more number of VCPUs • Manually evict unnecessary pods from the node with high CPU load to reduce load on the node. • Draining and uncordon node also help in rebalancing CPU load on worker nodes

Table 6-3 LOW_MEMORY

Field	Details
Description	Node {{ $labels.kubernetes_node }} available memory at {{ $value | humanize }} percent.
Summary	Node {{ $labels.kubernetes_node }} running out of memory
Cause	The available memory of a node is consumed more than 80% of the allocated memory.
Recommended Actions	• In case of vCNE, flavour of the worker nodes can be increased having larger RAM size. • Manually evict unnecessary pods from the node with high memory load to reduce load on the node. • Draining and uncordon node help in rebalancing the CPU load on worker nodes.

Table 6-4 OUT_OF_MEMORY

Field	Details
Description	Node {{ $labels.kubernetes_node }} out of memory
Summary	Node {{ $labels.kubernetes_node }} out of memory
Cause	Node available memory is consumed more than 90% of the allocated memory.
Recommended Actions	• In case of vCNE, flavour of the worker nodes can be increased having larger RAM size. • Manually evict unnecessary pods from the node with high memory load to reduce load on that node.

Table 6-5 NTP_SANITY_CHECK_FAILED

Field	Details
Description	NTP service sanity check failed on node {{ $labels.kubernetes_node }}
Summary	Clock is not synchronized on node {{ $labels.kubernetes_node }}
Cause	Clock is not synchronized on the node.

Table 6-5 (Cont.) NTP_SANITY_CHECK_FAILED

Field	Details
Recommended Actions	Steps to synchronize chronyd on node: 1. log in to the node on which you want to synchronize clock. 2. Run the following command: `sudo su;` 3. Run the following command: `systemctl restart chronyd;` 4. Watch chronyc tracking. 5. Run the following command: `sudo reboot` • If the issue is not resolved, Contact Oracle support.

Table 6-6 NETWORK_INTERFACE_FAILED

Field	Details
Description	Network interface {{ $labels.device }} on node {{ $labels.kubernetes_node }} is unavailable.
Summary	Network interface {{ $labels.device }} on node {{ $labels.kubernetes_node }} is unavailable.
Cause	Network interface is unavailable on the node.
Recommended Actions	Contact Oracle support.

Table 6-7 PVC_NEARLY_FULL

Field	Details
Description	Persistent volume claim {{ $labels.persistentvolumeclaim }} has {{ $value }}% of allocated space remaining.
Summary	Persistent volume claim {{ $labels.persistentvolumeclaim }} is nearly full.
Cause	PVC storage is filled to 80% of allocated space.

Table 6-7 (Cont.) PVC_NEARLY_FULL

Field	Details

Recommended Actions	1. Manually clean up PVC data for Prometheus:
	
	`$ kubectl get pods -o wide`
	`$ kubectl get pvc -n occne-infra`
	`$ Login to the nodes where Prometheus is deployed`
	`$ sudo su; lsblk`
	`$ Above command will give the path of the PVC(fetched in 2nd command), cd into it`
	`$ cd prometheus-db/`
	`$ rm -rf *`
	
	For Oracle OpenSearch:
	
	`$ kubectl get pods -o wide`
	`$ kubectl get pvc -n occne-infra`
	`$ Login to the nodes where Oracle OpenSearch-Data/Master nodes are deployed`
	`$ sudo su; lsblk`
	`$ Above command will give the path of the PVC(fetched in 2nd command), cd into it`
	`$ cd nodes/0`
	`$ rm -rf *`
	
	2. Use the following procedures to increase the size of PVC.
	• For Prometheus, see Changing Metrics Storage Allocation.
	• For Oracle OpenSearch, see Changing Oracle OpenSearch Storage Allocation.

Table 6-8 PVC_FULL

Field	Details
Description	Persistent volume claim {{ $labels.persistentvolumeclaim }} has {{ $value }}% of allocated space remaining.
Summary	Persistent volume claim {{ $labels.persistentvolumeclaim }} is full.
Cause	PVC storage is filled to 90% of the allocated space.
Recommended Actions	NA

Table 6-9 NODE_UNAVAILABLE

Field	Details
Description	Kubernetes node {{ $labels.kubernetes_node }} is not in Ready state.

Table 6-9 (Cont.) NODE_UNAVAILABLE

Field	Details
Summary	Kubernetes node {{ $labels.kubernetes_node }} is unavailable.
Cause	Node is not in ready state.
Recommended Actions	First, check if the given node is in running or shutoff state. If the node is in shutoff state, try restarting it from Openstack or iLo If the node is in running state, then perform the following steps: 1. Log in to the node. 2. Check the kubelet status and try to reboot it.

Table 6-10 ETCD_NODE_DOWN

Field	Details
Description	Etcd is not running or is unavailable.
Summary	Etcd is down.
Cause	Etcd is not running.
Recommended Actions	Refer to the following document to restore the failed etcd: https://kubernetes.io/docs/tasks/administer-cluster/configure-upgrade-etcd/#restoring-an-etcd-cluster

Table 6-11 CEPH_OSD_NEARLY_FULL

Field	Details
Description	Utilization of storage device {{ $labels.ceph_daemon }} has crossed 75% on host {{ $labels.hostname }}.
Summary	OSD storage device is nearly full.
Cause	OSD storage device is 75% full.
Recommended Actions	Contact Oracle support.

Table 6-12 CEPH_OSD_FULL

Field	Details
Description	Utilization of storage device {{ $labels.ceph_daemon }} has crossed 80% on host {{ $labels.hostname }}.
Summary	OSD storage device is critically full.
Cause	OSD storage device is 80% full.
Recommended Actions	Contact Oracle support.

Table 6-13 CEPH_OSD_DOWN

Field	Details
Description	Storage node {{ $labels.ceph_daemon }} is down.

Table 6-13 (Cont.) CEPH_OSD_DOWN

Field	Details
Summary	Storage node {{ $labels.ceph_daemon }} is down.
Cause	Ceph OSD is down.
Recommended Actions	Contact Oracle support.

Table 6-14 VSPHERE_CSI_CONTROLLER_FAILED

Field	Details
Description	The vSphere CSI controller process failed.
Summary	The vSphere CSI controller process failed
Cause	`Vsphere_csi_controller` is down.
Recommended Actions	Contact Oracle support.

6.2 Common Services Alerts

This section provides details about common services alerts.

Table 6-15 OPENSEARCH_CLUSTER_HEALTH_RED

Field	Details
Description	Cluster Name : {{ $externalLabels.cluster }} All the primary and replica shards are not allocated in Oracle OpenSearch cluster {{ $labels.cluster }} for instance {{ $labels.instance }}
Summary	Cluster Name : {{ $externalLabels.cluster }} Both primary and replica shards are not available.
Cause	Some or all of the shards (primary) are not ready.

Table 6-15 (Cont.) OPENSEARCH_CLUSTER_HEALTH_RED

Field	Details

Recommended Actions	Check the index for which the primary and replica shard is not able to be created and check for the indices that are in yellow or red state. Remove them by using the following procedure:
	1. Run the following command on Bastion Host to check if any indices are in yellow or red state:
	`kubectl -n occne-infra exec -it occne-opensearch-client-0 -- curl localhost:9200/_cat/indices`
	2. Run the following command to delete the indices in yellow or red state:
	`kubectl -n occne-infra exec -it occne-opensearch-client-0 -- curl localhost:9200/_cat/indices | grep 'yellow|red' | awk '{ print $3 }' | xargs -I{} kubectl -n occne-infra exec -it opensearch-client-0 -- curl -XDELETE localhost:9200/{}`
	3. Run the following command to verify if the indices with yellow or red state are deleted:
	`kubectl -n occne-infra exec -it opensearch-client-0 -- curl localhost:9200/_cat/indices`
	4. Restart the OpenSearch cluster in the following sequence: Master → Data → Client.
	If this procedure did not resolve the issue, then clean up all the indexes to restore OpenSearch in GREEN state.

Table 6-16 OPENSEARCH_CLUSTER_HEALTH_YELLOW

Field	Details
Description	Cluster Name : {{ $externalLabels.cluster }} The primary shard has been allocated in {{ $labels.cluster }} for Instance {{ $labels.instance }} but replicas for the shard cloud not be allocated.
Summary	Cluster Name : {{ $externalLabels.cluster }} The primary shard is allocated but replicas are not.
Cause	Indicates that OpenSearch has allocated all of the primary shards, but some or all of the replicas have not been allocated. This issue is observed in some cases after a node restart or shutdown.

Table 6-16 (Cont.) OPENSEARCH_CLUSTER_HEALTH_YELLOW

Field	Details

Recommended Actions	The yellow alarms are observed often after a shutdown or restart of a node. Most of the times, Oracle OpenSearch recovers on its own. If not, perform the following procedure to remove the replicas from the problematic index whose replica is not able to be allocated.
	
	`PUT /logstash-2021.08.21/_settings`
	`{`
	`"index" : {`
	`"number_of_replicas":0`
	`}`
	`}`

Table 6-17 OPENSEARCH_TOO_FEW_DATA_NODES_RUNNING

Field	Details
Description	Cluster Name : {{ $externalLabels.cluster }} There are only {{$value}} OpenSearch data nodes running in {{ $labels.cluster }} cluster.
Summary	Cluster Name : {{ $externalLabels.cluster }} {{ $labels.cluster }} cluster running on less than total number of data nodes.
Cause	1. Data nodes are either crashed or are in 0/1 state due to insufficient space in PVC. 2. PVC is full.
Recommended Actions	• Check the OpenSearch data or master pods' running status • If any of the OpenSearch pods are in the "not ready" state but running, check whether its associated PVC is full or not. • If PVC is not full, then contact Oracle support. • Ensure that the count of the data nodes is equal to the value of the `opensearch_data_replicas_count + 3` variable. When more data nodes are added, the alert must be corrected accordingly.

Table 6-18 PROMETHEUS_NODE_EXPORTER_NOT_RUNNING

Field	Details
Description	Prometheus Node Exporter is NOT running on host {{ $labels.kubernetes_node }}.
Summary	Prometheus Node Exporter is NOT running.
Cause	Pod is repeatedly crashing and is in "CrashLoopBackOff" or 0/1 or "ImagePullBackOff" state due to insufficient memory/ CPU, or issues with image used by pod.

Table 6-18 (Cont.) PROMETHEUS_NODE_EXPORTER_NOT_RUNNING

Field	Details
Recommended Actions	1. If resource utilization is the issue then Increase the Resources by editing the Node-Exporter Daemonset and search for resources section in it and increase the CPU or RAM accordingly. `$ kubectl edit ds occne-kube-prom-stack-prometheus-node-exporter -n occne-infra` 2. If Resource utilization is not the issue, then Contact Oracle support.

Table 6-19 FLUENTD_OPENSEARCH_NOT_AVAILABLE

Field	Details
Description	Fluentd-OpenSearch is not running or is otherwise unavailable.
Summary	Fluentd-OpenSearch is down.
Cause	Pod is repeatedly crashing and is in "CrashLoopBackOff" or 0/1 or "ImagePullBackOff" state due to insufficient memory/ CPU, or issues with image used by pod.
Recommended Actions	1. If resource utilization is the issue, then increase the resources by editing the Fluentd-OpenSearch daemonset and search for resources section in it and increase the CPU or RAM accordingly. `$ kubectl edit ds occne-fluentd-opensearch -n occne-infra` 2. If resource utilization is not the issue, then contact Oracle support.

Table 6-20 OPENSEARCH_DOWN

Field	Details
Description	OpenSearch is not running or is otherwise unavailable.
Summary	OpenSearch is down.
Cause	1. Pod is repeatedly crashing and is in "CrashLoopBackOff" or 0/1 or "ImagePullBackOff"state due to insufficient memory or CPU, or issues with image used by pod. 2. OpenSearch cluster is unavailable.

Table 6-20 (Cont.) OPENSEARCH_DOWN

Field	Details
Recommended Actions	1. If resource utilization is the issue, then increase the resources by editing the "occne-opensearch-cluster" statefulset. Search for resources section in the statefulset and increase the CPU or RAM accordingly. 2. If resource utilization is not the issue, then contact Oracle support.

Table 6-21 PROMETHEUS_DOWN

Field	Details
Description	All Prometheus instances are down. No metrics will be collected until at least one Prometheus instance is restored.
Summary	Metrics collection is down.
Cause	1. Pod is repeatedly crashing and is in "CrashLoopBackOff" or 0/1 or "ImagePullBackOff" state due to insufficient memory/CPU, or issues with image used by pod. 2. PVC is full.

Recommended Actions	1. If resource utilization is the issue, then increase the resources by editing the Prometheus CRD. Search for resources section in the CRD and increase the CPU/RAM accordingly.
	`kubectl edit prometheus occne-kube-prom-stack-kube-prometheus -n occne-infra`
	2. If resource utilization is not the issue, then contact Oracle support.
	3. Increase the PVC Size by referring to the Changing Metrics Storage Allocation section.
	4. Run the following commands to manually clean up the PVC data:
	`$ kubectl get pods -o wide`
	`$ kubectl get pvc -n occne-infra`
	`$ Login to the nodes where Prometheus is deployed`
	`$ sudo su; lsblk`
	`$ Above command will give the path of the PVC(fetched in 2nd command), cd into it`
	`$ cd prometheus-db/`
	`$ rm -rf *`

Table 6-22 ALERT_MANAGER_DOWN

Field	Details
Description	All alert manager instances are down. No alerts will be received until at least one alert manager instance is restored.
Summary	Alert notification is down.
Cause	Pod is repeatedly crashing and is in "CrashLoopBackOff" or 0/1 or "ImagePullBackOff"state due to insufficient memory/ CPU, or issues with image used by pod.
Recommended Actions	1. If resource utilization is the issue, then increase the resources by editing the Alertmanager CRD. Search for resources section in the Alertmanager CRD and increase the CPU or RAM accordingly. `kubectl edit alertmanager occne-kube-prom-stack-kube-alertmanager -n occne-infra` 2. If resource utilization is not the issue, then contact Oracle support.

Table 6-23 SNMP_NOTIFIER_DOWN

Field	Details
Description	SNMP Notifier is not running or is unavailable.
Summary	SNMP Notifier is down.
Cause	Pod is repeatedly crashing and is in "CrashLoopBackOff" or 0/1 or "ImagePullBackOff"state due to insufficient memory/ CPU, or issues with image used by pod.
Recommended Actions	1. If resource utilization is the issue, then increase the resources by editing the "occne-snmp-notifier" deployment. Search for the resources section in the deployment and increase the CPU or RAM accordingly. 2. If resource utilization is not the issue, then contact Oracle support.

Table 6-24 JAEGER_DOWN

Field	Details
Description	Jaeger collector is not running or is unavailable.
Summary	Jaeger is down.
Cause	1. Pod is repeatedly crashing and is in "CrashLoopBackOff" or 0/1 or "ImagePullBackOff" state due to insufficient memory/CPU, or issues with image used by pod. 2. OpenSerach is not available.

Table 6-24 (Cont.) JAEGER_DOWN

Field	Details
Recommended Actions	1. If resource utilization is the issue then Increase the Resources by editing the "occne-tracer-jaeger-collector" Deployment and search for resources section in it and increase the CPU or RAM accordingly. 2. If resource utilization is not the issue, then contact Oracle support. 3. Bring OpenSearch cluster back into healthy state by following the resolution mentioned in the OPENSEARCH_CLUSTER_HEALTH_RED alert. All Master, Client and Data pods must be up and running.

Table 6-25 METALLB_SPEAKER_DOWN

Field	Details
Description	The MetalLB speaker on worker node {{ $labels.instance }} is down.
Summary	A MetalLB speaker is down.
Cause	Pod is repeatedly crashing and is in "CrashLoopBackOff" or 0/1 or "ImagePullBackOff" state due to insufficient memory/ CPU, or issues with image used by pod.
Recommended Actions	1. If resource utilization is the issue, then increase the resources by editing the Metallb-speaker daemonset. Search for resources section in the daemonset and increase the CPU or RAM accordingly. `$ kubectl edit ds occne-metallb-speaker -n occne-infra` 2. If resource utilization is not the issue, then contact Oracle support.

Table 6-26 METALLB_CONTROLLER_DOWN

Field	Details
Description	The MetalLB controller is not running or is unavailable.
Summary	The MetalLB controller is down.
Cause	1. Pod is repeatedly crashing and is in "CrashLoopBackOff" or 0/1 or "ImagePullBackOff" state due to insufficient memory or CPU, or issues with the image used by the pod.

Table 6-26 (Cont.) METALLB_CONTROLLER_DOWN

Field	Details
Recommended Actions	1. If resource utilization is the issue, then increase the resources by editing the "occne-metallb-controller" deployment. Search for the resources section in the deployment and increase the CPU or RAM accordingly. 2. If resource utilization is not the issue, then contact Oracle support.

Table 6-27 GRAFANA_DOWN

Field	Details
Description	Grafana is not running or is unavailable.
Summary	Grafana is down.
Cause	1. Pod is repeatedly crashing and is in "CrashLoopBackOff" or 0/1 or "ImagePullBackOff" state due to insufficient memory/CPU, or issues with image used by pod. 2. Prometheus is not available.
Recommended Actions	1. If resource utilization is the issue, then increase the resources by editing the "occne-kube-prom-stack-grafana" deployment. Search for the resources section in the deployment and increase the CPU or RAM accordingly. 2. If resource utilization is not the issue, then contact Oracle support.

Table 6-28 LOAD_BALANCER_NO_HA

Field	Details
Description	A single load balancer serving the {{ $labels.external_network }} network has failed. Load balancing will continue to operate in simplex mode.
Summary	A load balancer for the {{ $labels.external_network }} network is down.
Cause	One of the LBVM is down.
Recommended Actions	Replace the failed LBVM: See #unique_69 section to replace the failed LBVM.

Table 6-29 LOAD_BALANCER_NO_SERVICE

Field	Details
Description	All Load Balancers serving the {{ $labels.external_network }} network have failed. External access for all services on this network is unavailable.

Table 6-29 (Cont.) LOAD_BALANCER_NO_SERVICE

Field	Details
Summary	Load balancing for the {{ $labels.external_network }} network is unavailable.
Cause	Both LBVMs are down as a result, the external network is down.
Recommended Actions	Replace one LBVM, wait for lb_monitor to convert it from STANDBY to ACTIVE state (run lb_monitor.py manually if needed) and then replace another LBVM. See #unique_69 section to replace failed LBVM.

Table 6-30 LOAD_BALANCER_FAILED

Field	Details
Description	Load balancer {{ $labels.name }} at IP {{ $labels.ip_address }} on the {{ $labels.external_network }} network has failed. Perform Load Balancer recovery procedure to restore.
Summary	A load balancer failed.
Cause	One of the LBVMs or both the LBVMs are down.
Recommended Actions	Although this alert is not always service affecting, the Load Balancer must be restored to restore high availability for load balancing. Replace one of the LBVMs or both the LBVMs. See #unique_69 section to replace failed LBVM.

Table 6-31 PROMETHEUS_NO_HA

Field	Details
Description	A Prometheus instance has failed. Metrics collection will continue to operate in simplex mode.
Summary	A Prometheus instance is down.
Cause	Pod is repeatedly crashing and is in "CrashLoopBackOff" or 0/1 or "ImagePullBackOff" state due to insufficient memory/ CPU, or issues with image used by pod.

Table 6-31 (Cont.) PROMETHEUS_NO_HA

Field	Details

Recommended Actions	1. If resource utilization is the issue, then increase the resources by editing the Prometheus CRD. Search for the resources section the Prometheus CRD and increase the CPU or RAM accordingly.
	`kubectl edit prometheus occne-kube-prom-stack-kube-prometheus -n occne-infra`
	2. If resource utilization is not the issue then contact Oracle support.
	3. Increase PVC size by referring to the Changing Metrics Storage Allocation section.
	4. Manually clean-up PVC data:
	`$ kubectl get pods -o wide`
	`$ kubectl get pvc -n occne-infra`
	`$ Login to the nodes where Prometheus is deployed`
	`$ sudo su; lsblk`
	`$ Above command will give the path of the PVC(fetched in 2nd command), cd into it`
	`$ cd prometheus-db/`
	`$ rm -rf *`

Table 6-32 ALERT_MANAGER_NO_HA

Field	Details
Description	An AlertManager instance has failed. Alert management will continue to operate in simplex mode.
Summary	An AlertManager instance is down.
Cause	Pod is repeatedly crashing and is in "CrashLoopBackOff" or 0/1 or "ImagePullBackOff" state due to insufficient memory/CPU ir issues with the image used by pod.
Recommended Actions	1. If resource utilization is the issue, then increase the resources by editing the Alertmanager CRD. Search for the resources section in the Alertmanager CRD and increase the CPU or RAM accordingly. `kubectl edit alertmanager occne-kube-prom-stack-kube-alertmanager -n occne-infra` 2. If resource utilization is not the issue, then contact Oracle support.

Table 6-33 PROMXY_METRICS_AGGREGATOR_DOWN

Field	Details
Description	Promxy failed. Metrics will be retrieved from a single Prometheus instance only.
Summary	Promxy is down.
Cause	Pod is repeatedly crashing and is in "CrashLoopBackOff" or 0/1 or "ImagePullBackOff" state due to insufficient memory/ CPU, or issues with image used by pod.
Recommended Actions	As metrics are retrieved from a single Prometheus instance, there may be gaps in the retrieved data. Promxy must be restarted to restore the full data retrieval capabilities. 1. If resource utilization is the issue, then increase the resources by editing the "occne-promxy" deployment. Search for the resources section in the deployment and increase the CPU or RAM accordingly. 2. If resource utilization is not the issue, then contact Oracle support.

Table 6-34 VCNE_LB_CONTROLLER_FAILED

Field	Details
Description	The vCNE LB Controller process failed.
Summary	The vCNE LB Controller process failed.
Cause	Pod is repeatedly crashing and is in the "CrashLoopBackOff", 0/1, or "ImagePullBackOff" state due to insufficient memory/CPU, or issues with the image used by the pod.
Recommended Actions	1. If resource utilization is the cause, then increase the resource by editing the "occne-lb-controller-server" deployment. Search for the resources section in the deployment and increase the CPU or RAM accordingly. 2. If resource utilization is not the cause, then contact Oracle support.

Table 6-35 VMWARE_CSI_CONTROLLER_FAILED

Field	Details
Description	The VmWare CSI Controller process failed.
Summary	The VmWare CSI Controller process failed.
Cause	The CSI Controller process failed. **Note:** This alert is raised only when CNE is installed on a VMware infrastructure.
Recommended Actions	Contact Oracle support.

Table 6-36 EGRESS_CONTROLLER_NOT_AVAILABLE

Field	Details
Description	Egress controller is not running or is unavailable.
Summary	Egress controller is down
Cause	Pod is repeatedly crashing and is in "CrashLoopBackOff" or 0/1 or "ImagePullBackOff" state due to insufficient memory/ CPU, or issues with image used by pod.
Recommended Actions	1. If resource utilization is the cause, then increase the resource by editing the "occne-egress-controller" daemonset. Search for the resources section in the daemonset and increase the CPU or RAM accordingly. 2. If resource utilization is not the cause, then contact Oracle support.

Table 6-37 OPENSEARCH_DATA_PVC_NEARLY_FULL

Field	Details
Description	OpenSearch Data Volume {{ $persistentvolumeclaim }} has {{ $value }}% of allocated space remaining. Once full, this will cause OpenSearch cluster to start throwing index_block_exceptions, either increase Opensearch data PVC or remove unnecessary indices.
Summary	OpenSearch Data Volume is nearly full.
Cause	OpenSearch data PVCs are nearly full.
Recommended Actions	Perform one of the following recommendations: • Increase the PVC size of OpenSearch cluster data for which the alert is raised. • Delete the old indices from OpenSearch Dashboards > dev tools > DELETE <index_name_to_be_deleted>.

6.3 Bastion Host Alerts

This section provides details about Bastion Host alerts.

Table 6-38 BASTION_HOST_FAILED

Field	Details
Description	Bastion Host {{ $labels.name }} at IP address {{ $labels.ip_address }} is unavailable.
Summary	Bastion Host {{ $labels.name }} is unavailable.
Cause	One of the Bastion Hosts fail to respond to liveness tests.
Recommended Actions	Contact Oracle support.

Table 6-39 ALL_BASTION_HOSTS_FAILED

Field	Details
Description	All Bastion Hosts are unavailable.

Table 6-39 (Cont.) ALL_BASTION_HOSTS_FAILED

Field	Details
Summary	All Bastion Hosts are unavailable.
Cause	All Bastion Hosts fail to respond to liveness tests.
Recommended Actions	Contact Oracle support.

7 Maintenance Procedures

This chapter provides detailed instructions about how to maintain the CNE platform.

7.1 Premaintenance Check for VMware Deployments

This section provides details about the checks that must be run on VMware deployments before performing any maintenance procedures.

1. Verify the content of the `compute/main.tf` and `compute-lbvm/main.tf` files:
   a. Run the following command to verify the content of the `compute/main.tf` file:

      ```
      $ cat /var/occne/cluster/${OCCNE_CLUSTER}/modules/compute/main.tf |
      grep 'ignore_changes\|override_template_disk' -C 2
      ```

      Ensure that the content of the file exactly matches the following content:

      ```
       }

        override_template_disk {
          bus_type    = "paravirtual"
          size_in_mb  = var.disk
      --

        lifecycle {
          ignore_changes = [
            vapp_template_id,
            template_name,
            catalog_name,
            override_template_disk
           ]
        }
      --
        }

        override_template_disk {
          bus_type    = "paravirtual"
          size_in_mb  = var.disk
      --

        lifecycle {
          ignore_changes = [
            vapp_template_id,
            template_name,
            catalog_name,
            override_template_disk
           ]
        }
      ```

b. Run the following command to verify the content of the `compute-lbvm/main.tf` file:

```
$ cat /var/occne/cluster/${OCCNE_CLUSTER}/modules/compute-lbvm/main.tf
| grep 'ignore_changes\|override_template_disk' -C 2
```

Ensure that the content of the file exactly matches the following content:

```
}

  override_template_disk {
    bus_type    = "paravirtual"
    size_in_mb  = var.disk
--

  lifecycle {
    ignore_changes = [
      vapp_template_id,
      template_name,
      catalog_name,
      override_template_disk
     ]
  }
--
  }

  override_template_disk {
    bus_type    = "paravirtual"
    size_in_mb  = var.disk
--

  lifecycle {
    ignore_changes = [
      vapp_template_id,
      template_name,
      catalog_name,
      override_template_disk
     ]
  }
```

2. If the files don't contain the `ignore_changes` argument, then edit the files and add the argument to each of the "`vcd_vapp_vm`" resources:

   a. Run the following command to edit the `compute/main.tf` file:

   ```
   $ vi /var/occne/cluster/${OCCNE_CLUSTER}/modules/compute/main.tf
   ```

   b. Add the following content between each `override_template_disk` code block and `metadata = var.metadata` line for each "`vcd_vapp_vm`" resource:

   ```
   lifecycle {
       ignore_changes = [
         vapp_template_id,
         template_name,
         catalog_name,
         override_template_disk
   ```

```
    ]
}
```

c. Save the `compute/main.tf` file.

d. Run the following command to edit the `compute-lbvm/main.tf` file:

```
$ vi /var/occne/cluster/${OCCNE_CLUSTER}/modules/compute/main.tf
```

e. Add the following content between each `override_template_disk` code block and `metadata = var.metadata` line for each "vcd_vapp_vm" resource:

```
lifecycle {
   ignore_changes = [
     vapp_template_id,
     template_name,
     catalog_name,
     override_template_disk
    ]
}
```

f. Save the `compute-lbvm/main.tf` file.

g. Repeat step 1 to ensure that the content of the files matches the ones provided in the step.

7.2 Accessing the CNE

This section describes the procedures to access an CNE for maintenance purposes.

7.2.1 Accessing the Bastion Host

This section provides information about how to access a CNE Bastion Host.

Prerequisites

- SSH private key must be available on the server or VM that is used to access the Bastion Host.
- The SSH private keys generated or provided during the installation must match the authorized key (public) present in the Bastion Hosts. For more information about the keys, see the installation prerequisites in *Oracle Communications Cloud Native Core, Cloud Native Environment Installation, Upgrade, and Fault Recovery Guide*.

Procedure

All commands must be run from a server or VM that has network access to the CNE Bastion Hosts. To access the Bastion Host, perform the following tasks.

7.2.1.1 Logging in to the Bastion Host

This section describes the procedure to log in to the Bastion Host.

1. Determine the Bastion Host IP address.
   Contact your system administrator to obtain the IP addresses of the CNE Bastion Hosts. The system administrator can obtain the IP addresses from the OpenStack Dashboard, VMware Cloud Director, or by other means such as from the BareMetal Hosts.

2. To log in to the Bastion Host, run the following command:

> **Note**
>
> The default value for `<user_name>` is `cloud-user` (for vCNE) or `admusr` (for Baremetal).

```
$ ssh -i /<ssh_key_dir>/<ssh_key_name>.key
<user_name>@<bastion_host_ip_address>
```

7.2.1.2 Copying Files to the Bastion Host

This section describes the procedure to copy the files to the Bastion Host.

1. Determine the Bastion Host IP address.
   Contact your system administrator to obtain the IP addresses of the CNE Bastion Hosts. The system administrator can obtain the IP addresses from the Openstack Dashboard, VMware Cloud Director, or by other means such as from the BareMetal Hosts.
2. To copy files to the Bastion Host, run the following command:

```
$ scp -i /<ssh_key_dir>/<ssh_key_name>.key <source_file>
<user_name>@<bastion_host_ip_address>:/<path>/<dest_file>
```

7.2.1.3 Managing Bastion Host

The Bastion Host comes with the following built-in scripts to manage the Bastion Hosts:

- is_active_bastion
- get_active_bastion
- get_other_bastions
- update_active_bastion.sh

These scripts are used to get details about Bastion Hosts, such as checking if the current Bastion Host is the active one and getting the list of other Bastions. This section provides the procedures to manage Bastion Hosts using these scripts.

These scripts are located in the `/var/occne/cluster/$OCCNE_CLUSTER/artifacts/` directory. You don't have to change the directory to run these scripts. You can run these scripts from anywhere within a Bastion Host like a system command as the directory containing the scripts is a part of `$PATH`.

All the scripts interact with the bastion-controller pod and its database directly by querying it or updating it through Kubectl. The commands fails to run in the following conditions:

- If the lb-controller pod is not running.
- If the kubectl admin configuration is not set properly.

For more information about the possible errors and troubleshooting, seeTroubleshooting Bastion Host.

7.2.1.3.1 Verifying if the Current Bastion Host is the Active One

This section describes the procedure to verify if the current Bastion Host is the active one using the `is_active_bastion` script.

- Run the following command to check if the current Bastion Host is the active Bastion Host:

  ```
  $ is_active_bastion
  ```

  On running the command, the system runs the `is_active_bastion` script to compare the current Bastion IP against the active Bastion IP retrieved from the database of the bastion-controller pod. If the IP stored at the bastion-controller database is equal to the IP of the Bastion where the script is run from, then the system displays the following response:

  ```
  IS active-bastion
  ```

  If the IP addresses don't match, the system displays the following response indicating, the current Bastion Host is not the active one:

  ```
  NOT active-bastion
  ```

7.2.1.3.2 Getting the Host IP or Hostname of the Current Bastion Host

This section provides details about getting the Host IP or Hostname of the current Bastion Host using the `get_active_bastion` script.

- Run the following command to get the Host IP of the current Bastion Host:

  ```
  $ get_active_bastion
  ```

  On running the command, the system runs the `get_active_bastion` script to get the IP address of the Bastion from the `bastion-controller` DB:

  Sample output:

  ```
  192.168.200.10
  ```

- Run the following command to get the Hostname of the current Bastion Host.

  ```
  $ DBFIELD=name get_active_bastion
  ```

  The `DBFIELD=name` parameter in the command is the additional parameter that is passed to get the Hostname from the `bastion-controller` DB.

  Sample output:

  ```
  occne1-rainbow-bastion-1
  ```

7.2.1.3.3 Getting the List of Other Bastion Hosts

This section provides details about getting the list of other Bastion Hosts in the cluster using the `get_other_bastions` script.

- Run the following command to get the Hostnames of other Bastion Hosts in the cluster:

  ```
  $ get_other_bastions
  ```

  Sample output:

  ```
  occne1-rainbow-bastion-2
  ```

- Run the following command to get the Host IPs of the other Bastion Hosts in the cluster.

  ```
  $ DBFIELD=ipaddr get_other_bastions
  ```

  The `DBFIELD=ipaddr` parameter in the command is the additional parameter to get the Host IPs from the `bastion-controller` DB.

  Sample output:

  ```
  192.168.200.11
  ```

- You can provide additional parameters to filter the list of other Bastion Hosts in the cluster.

  For example, you can use the `CRITERIA="state == 'HEALTHY'"` or `CRITERIA="state != 'FAILED'` filter criteria to fetch the list of other Bastion Hosts in the cluster that are active:

  ```
  $ CRITERIA="state == 'HEALTHY'" get_other_bastions
  ```

  Sample output:

  ```
  occne1-rainbow-bastion-2
  ```

7.2.1.3.4 Changing the Bastion Host to Active

This section provides the steps to make a standby Bastion Host as active using the `update_active_bastion.sh` script.

- Run the following command to make the current Bastion Host as the active Bastion Host:

  ```
  $ update_active_bastion.sh
  ```

  On running the command, the system runs the `get_active_bastion` script to compare the current Bastion IP with the active Bastion IP retrieved from the `bastion-controller` pod DB. If the IP stored in the `bastion-controller` DB is equal to the IP of the Bastion where the script is run from, then the system takes no action as the current Bastion is already the active Bastion. Otherwise, the system updates the DB with the current IP of the Bastion Host, making it the new active Bastion.

  Sample output showing the response when the current Bastion is already the active Bastion:

  ```
  2023-11-24:17-17-34: Setting 192.168.200.10 as bastion
  Bastion already set to 192.168.200.10
  ```

Sample output showing the response when the current Bastion is updated as the active Bastion:

```
2023-11-24:17-29-09: Setting 192.168.200.11 as bastion
```

7.2.1.4 Troubleshooting Bastion Host

This section describes the issues that you may encounter while using Bastion Host and their troubleshooting guidelines.

Permission Denied Error While Running Kubernetes Command

Problem:
Users may encounter "Permission Denied" error while running Kubernetes commands if there is no proper access.

Error Message:

```
error: error loading config file "/var/occne/cluster/occne1-rainbow/artifacts/
admin.conf": open /var/occne/cluster/occne1-rainbow/artifacts/admin.conf:
permission denied
```

Resolution:
Verify permission access to `admin.conf`. The user running the command must be able to run basic `kubectl` commands to use the Bastion scripts.

Commands Take Too Long to Respond and Fail to Return Output

Problem:
A command may take too long to display any output. For example, running the `is_active_bastion` command may take too long to respond leading to the timed out error.

Error Message:

```
error: timed out waiting for the condition
```

Resolution:

- Verify the status of the bastion-controller. This error can occur if the pods are not running or in a crash state due to various reasons such as lack of resources at the cluster.
- Print the bastion controller logs to check the issue. For example, print the logs and check if a loop crash error is caused due to lack of resources.

  ```
  $ kubectl logs -n ${OCCNE_NAMESPACE} deploy/occne-bastion-controller
  ```

  Sample output:

  ```
  Error from server (BadRequest): container "bastion-controller" in pod
  "occne-bastion-controller-797db5f845-hqlm6" is waiting to start:
  ContainerCreating
  ```

Command Not Found Error

Problem:

User may encounter `command not found` error while running a script.

Error Message:

```
-bash: is_active_bastion: command not found
```

Resolution:
Run the following command and verify that the `$PATH` variable is set properly and contains the artifacts directory.

> **Note**
>
> By default, CNE sets up the path automatically during the installation.

```
$ echo $PATH
```

Sample output showing the correct `$PATH`:

```
/home/cloud-user/.local/bin:/home/cloud-user/bin:/usr/local/bin:/usr/bin:/usr/
local/sbin:/usr/sbin:/var/occne/cluster/occne1-rainbow/artifacts/
istio-1.18.2/bin/:/var/occne/cluster/occne1-rainbow/artifacts
```

7.3 General Configuration

This section describes the general configuration tasks for CNE.

7.3.1 Configuring SNMP Trap Destinations

This section describes the procedure to set up SNMP notifiers within CNE, such that the AlertManager can send alerts as SNMP traps to one or more SNMP receivers.

1. Perform the following steps to verify the cluster condition before setting up multiple trap receivers:
   a. Run the following command and verify that the `alertmanager` and `snmp-notifier` services are running:

      ```
      $ kubectl get services --all-namespaces | grep -E 'snmp-notifier|
      alertmanager'
      ```

      Sample output:

      ```
      NAMESPACE     NAME
      TYPE           CLUSTER-IP      EXTERNAL-IP
      PORT(S)                                  AGE
      occne-infra    occne-kube-prom-stack-kube-alertmanager
      LoadBalancer   10.233.16.156   10.75.151.178
      80:31100/TCP                        11m
      occne-infra    occne-alertmanager-snmp-
      notifier                              ClusterIP      10.233.41.30
      <none>          9464/TC                              11m
      ```

b. Run the following command and verify that the `alertmanager` and `snmp-notifier` pods are running:

```
$ kubectl get pods --all-namespaces | grep -E 'snmp-notifier|
alertmanager'
```

Sample output:

```
occne-infra     alertmanager-occne-kube-prom-stack-kube-
alertmanager-0          2/2     Running   0          18m
occne-infra     alertmanager-occne-kube-prom-stack-kube-
alertmanager-1          2/2     Running   0          18m
occne-infra     occne-alertmanager-snmp-notifier-744b755f96-
m8vbx                           1/1     Running   0          18m
```

2. Perform the following steps to edit the default `snmp-destination` and add a new `snmp-destination`:

a. Run the following command from Bastion Host to get the current `snmp-notifier` resources:

```
$ kubectl get all -n occne-infra | grep snmp
```

Sample output:

```
pod/occne-alertmanager-snmp-notifier-75656cf4b7-gw55w 1/1 Running 0 37m
service/occne-alertmanager-snmp-notifier ClusterIP 10.233.29.86 <none>
9464/TCP 10h
deployment.apps/occne-alertmanager-snmp-notifier 1/1 1 1 10h
replicaset.apps/occne-alertmanager-snmp-notifier-75656cf4b7 1 1 1 37m
```

b. The `snmp-destination` is the interface IP address of the trap receiver to get the traps. Edit the deployment to modify `snmp-destination` and add a new `snmp-destination` when needed:

i. Run the following command to edit the deployment:

```
$ kubectl edit -n occne-infra deployment occne-alertmanager-snmp-
notifier
```

ii. From the vi editor, move down to the `snmp-destination` section. The default configuration is as follows:

```
- --snmp.destination=127.0.0.1:162
```

iii. Add a new destination to receive the traps.
For example:

```
 - --snmp.destination=192.168.200.236:162
```

iv. If want to add multiple trap receivers, add them in multiple new lines.

For example:

```
- --snmp.destination=192.168.200.236:162
- --snmp.destination=10.75.135.11:162
- --snmp.destination=10.33.64.50:162
```

v. After editing, use the `:x` or `:wq` command to save the exit.
Sample output:

```
deployment.apps/occne-alertmanager-snmp-notifier edited
```

c. Perform the following steps to verify the new replicaset and delete the old replicaset:

i. Run the following command to get the resource and check the restart time to verify that the pod and replicaset are regenerated:

```
$  kubectl get all -n occne-infra | grep snmp
```

Sample output:

```
pod/occne-alertmanager-snmp-notifier-88976f7cc-xs8mv
1/1     Running   0             90s
service/occne-alertmanager-snmp-notifier
ClusterIP      10.233.29.86    <none>
9464/TCP                                          10h
deployment.apps/occne-alertmanager-snmp-notifier           1/1
1            1           10h
replicaset.apps/occne-alertmanager-snmp-
notifier-75656cf4b7           0         0         0       65m
replicaset.apps/occne-alertmanager-snmp-
notifier-88976f7cc            1         1         1       90s
```

ii. Identify the old replicaset from the previous step and delete it.
For example, the restart time of the `replicaset.apps/occne-alertmanager-snmp-notifier-75656cf4b7` in the previous step output is 65m. This indicates that it is the old replica set. Use the following command to delete the old replicaset:

```
$ kubectl delete -n occne-infra replicaset.apps/occne-alertmanager-
snmp-notifier-75656cf4b7
```

d. Port 162 of the server must be open and have some application to catch the traps to test if the new trap receiver receives the SNMP traps. This step may vary depending on the type of server. The following codeblock provides an example for Linux server:

```
$ sudo iptables -A INPUT -p udp -m udp --dport 162 -j ACCEPT
$ sudo dnf install -y tcpdump
$ sudo tcpdump -n -i <interface of the ip address set in snmp-
destination> port 162
```

7.3.2 Changing Network MTU

This section describes the procedure to modify the Maximum Transmission Unit (MTU) of the Kubernetes internal network after the initial CNE installation.

Changing MTU on Internal Interface (eth0) for vCNE (OpenStack or VMware)

1. Run the following commands from the Bastion host to launch the provision container:

   ```
   $ podman run -it --rm --rmi --network host --name DEPLOY_${OCCNE_CLUSTER} -
   v /var/occne/cluster/${OCCNE_CLUSTER}:/host -v /var/occne:/var/occne:rw -e
   OCCNEINV=/host/hosts -v /var/www/html/occne:/var/www/html/occne -e
   'OCCNEARGS=--extra-vars=occne_hostname=${OCCNE_CLUSTER}-bastion-1 -i /host/
   occne.ini' ${CENTRAL_REPO}:5000/occne/provision:${OCCNE_VERSION} bash
   ```

2. Run the following command from the provision container Bash shell session to change the MTU for an OpenStack deployment:

   ```
   $ ansible -i /host/hosts k8s-cluster -m shell -a 'sudo  nmcli con mod
   "System eth0" 802-3-ethernet.mtu <MTU value>;  sudo  nmcli con up  "System
   eth0"'$ exit
   ```

3. Run the following command from the provision container Bash shell session to change the MTU for a VMware deployment:

   > **Note**
   >
   > If the interface connection name isn't `ens192`, then run the following commands to find the connection name:
   >
   > a. Run the "`ip address`" command to get the IP address.
   >
   > b. Run the "`nmcli con show`" command to get the connection name.

   ```
   $ ansible -i /host/hosts k8s-cluster -m shell -a 'sudo  nmcli con mod
   ens192 802-3-ethernet.mtu <MTU value>;  sudo  nmcli con up ens192'$ exit
   ```

Changing MTU on All Interfaces of BareMetal VM Host and VM Guest

> **Note**
>
> - The MTU value on the VM host depends on the ToR switch configuration:
>   - *cisco Nexus9000 93180YC-EX* has "`system jumbomtu`" up to *9216*.
>   - If you're using `port-channel/vlan-interface/uplnk-interface-to-customer-switch`, then run the "`system jumbomtu <mtu>`" command and configure "`mtu <value>`" up to the value obtained from the command.
>   - If you're using other types of ToR switches, you can configure the MTU value of VM host up to the maximum MTU value of the switch. Therefore, check the switches for the maximum MTU value and configure the MTU value accordingly.
> - The following steps are for a standard setup with bastion-1 or master-1 on host-1, bastion-2 or master-2 on host-2, and master-3 on host-3. If you have a different setup, then modify the commands accordingly. Each step in this procedure is performed to change MTU for the VM host and the Bastion on the VM host.

1. SSH to k8s-host-2 from bastion-1:

   ```
   $ ssh k8s-host-2
   ```

2. Run the following command to show all the connections:

   ```
   $ nmcli con show
   ```

3. Run the following commands to modify the MTU value on all the connections:

   > **Note**
   >
   > Modify the connection names in the following commands according to the connection names obtained from step 2.

   ```
   $ sudo nmcli con mod bond0 802-3-ethernet.mtu <MTU value>
   $ sudo nmcli con mod bondbr0 802-3-ethernet.mtu <MTU value>
   $ sudo nmcli con mod "vlan<mgmt vlan id>-br" 802-3-ethernet.mtu <MTU value>
   $ sudo nmcli con up bond0
   $ sudo nmcli con up bondbr0
   $ sudo nmcli con up "vlan<mgmt vlan id>-br"
   ```

4. Run the following commands if there is `vlan<ilo_vlan_id>-br` on this host:

   ```
   $ sudo nmcli con mod "vlan<ilo vlan id>-br" 802-3-ethernet.mtu <MTU value>
   $ sudo nmcli con up "vlan<ilo vlan id>-br"
   ```

5. After the values are updated on VM host, run the following commands to shut down all the VM guests:

   ```
   $ sudo virsh list --all
   $ sudo virsh shutdown <VM guest>
   ```

where, <VM guest> is the VM guest name obtained from the $ sudo virsh list --all command.

6. Run the virsh list command until the status of the VM guest is changed to "shut off":

```
$ sudo virsh list --all
```

7. Run the following command to start the VM guest:

```
$ sudo virsh  start <VM guest>
```

where, <VM guest> is the name of the VM guest.

8. Wait until bastion-2 is reachable and run the following command to SSH to bastion-2:

```
$ ssh bastion-2
```

9. Run the following command to list all connections in bastion-2:

```
$ nmcli con show
```

10. Run the following commands to modify the MTU value on all the connections in bastion-2:

> **Note**
>
> Modify the connection names in the following commands according to the connection names obtained in step 8.

```
$ sudo nmcli con mod "System enp1s0" 802-3-ethernet.mtu <MTU value>
$ sudo nmcli con mod "System enp2s0" 802-3-ethernet.mtu <MTU value>
$ sudo nmcli con mod "System enp3s0" 802-3-ethernet.mtu <MTU value>
$ sudo nmcli con up "System enp1s0"
$ sudo nmcli con up "System enp2s0"
$ sudo nmcli con up "System enp3s0"
```

11. Wait until bastion-2 is reachable and run the following command to SSH to bastion-2:

```
$ ssh bastion-2
```

12. Repeat steps 9 and 10 to change the MTU value on k8s-host-1 and bastion-1.
13. Repeat steps 1 to 10 to change the MTU values on k8s-host-3 and restart all VM guests on it. You can use bastion-1 or bastion-2 for performing this step.

> **Note**
>
> For the VM guests that are controller nodes, perform only the virsh shutdown and virsh start commands to restart the VM guests. The MTU values of these controller nodes are updated in the following section.

Changing MTU on enp1s0 or bond0 Interface for BareMetal Controller or Worker Nodes

1. Run the following command to launch the provision container:

   ```
   $ podman run -it --rm --network host -v /var/occne/cluster/$
   {OCCNE_CLUSTER}:/host winterfell:5000/occne/provision:<release> /bin/bash
   ```

   Where, <release> is the currently installed release.

   This creates a Bash shell session running within the provision container.

2. Run the following commands to change enp1s0 interfaces for controller nodes and validate MTU value of the interface:

   a. Change enp1s0 interfaces for controller nodes:
   Replace <MTU value> in the command with a real integer value.

   ```
   $ ansible -i /host/hosts.ini kube-master -m shell -a 'sudo  nmcli con
   mod "System enp1s0" 802-3-ethernet.mtu <MTU value>;  sudo  nmcli con up
   "System enp1s0"'
   ```

   b. Validate the MTU value of the interface:

   ```
   $ ansible -i /host/hosts.ini kube-master -m shell -a 'ip link show
   enp1s0'
   ```

3. Run the following commands to change bond0 interfaces for worker nodes and validate the MTU value of the interface:

   a. Change bond0 interfaces for controller nodes:
   Replace <MTU value> in the command with a real integer value.

   ```
   $ ansible -i /host/hosts.ini kube-node -m shell -a 'sudo  nmcli con mod
   bond0 802-3-ethernet.mtu <MTU value>;  sudo  nmcli con up bond0'
   ```

   b. Validate the MTU value of the interface:

   ```
   $ ansible -i /host/hosts.ini kube-node -m shell -a 'ip link show
   bond0'$ exit
   ```

Changing MTU on vxlan Interface (vxlan.calico) for BareMetal and vCNE

1. Log in to the Bastion host and run the following command:

   ```
   $ kubectl edit daemonset calico-node -n kube-system
   ```

2. Locate the line with FELIX_VXLANMTU and replace the current `<MTU value>` with the new integer value:

   > **Note**
   >
   > `vlan.calico` has an extra header in the packet. The modified MTU value must be at least 50 lower than the MTU set in previous steps to work.

   ```
   - name: FELIX_VXLANMTU
           value: "<MTU value>"
   ```

3. Use `:x` to save and exit the vi editor and run the following command:

   ```
   $ kubectl rollout restart daemonset calico-node -n kube-system
   ```

4. Run the following command to provision container:

   ```
   $ podman run -it --rm --network host -v /var/occne/cluster/$
   {OCCNE_CLUSTER}:/host winterfell:5000/occne/provision:$
   {OCCNE_VERSION} /bin/bash
   ```

5. Validate the MTU value of the interface on the controller nodes and worker nodes:
   - For BareMetal, run the following command to validate the MTU value:

     ```
     $ ansible -i /host/hosts.ini k8s-cluster -m shell -a 'ip link show
     vxlan.calico'
     ```

   - For vCNE (OpenStack or VMware), run the following command to validate the MTU value:

     ```
     $ ansible -i /host/hosts k8s-cluster -m shell -a 'ip link show
     vxlan.calico'
     ```

> **Note**
>
> It takes some time for all the nodes to change to the new MTU. If the MTU value isn't updated, run the command several times to see the changes in the values.

Changing MTU on Calico Interfaces (cali*) for vCNE or BareMetal

1. Log in to Bastion host and launch the provision container for vCNE or BareMetal using commands from Step 1 of *Change MTU on eth0 interface for vCNE* and *Change MTU on enp1s0 or bond0 interface for BareMetal*.
2. Run the ansible command for all worker nodes from the provision container:

> **Note**
>
> Run this command for worker nodes only and not for controller nodes.

   - Run the following command for a BareMetal deployment:

> **Note**
>
> Replace <MTU value> in the command with an integer value without quote.

   ```
   bash-4.4# ansible -i /host/hosts.ini kube-node -m shell -a 'sudo sed -i
   '/\\\"mtu\\\"/d' /etc/cni/net.d/calico.conflist.template; sudo sed -i
   "/\\\"type\\\": \\\"calico\\\"/a \ \ \ \ \ \ \\\"mtu\\\": <MTU
   ```

```
value>," /etc/cni/net.d/calico.conflist.template'
bash-4.4# exit
```

- Run the following command for a vCNE deployment:

  > **Note**
  >
  > Replace <MTU value> in the command with an integer value without quote.

```
bash-4.4# ansible -i /host/hosts kube-node -m shell -a 'sudo sed -i '/\\
\"mtu\\\"/d' /etc/cni/net.d/calico.conflist.template; sudo sed -i "/\\
\"type\\\": \\\"calico\\\"/a \ \ \ \ \ \ \\\"mtu\\\": <MTU
value>," /etc/cni/net.d/calico.conflist.template'
bash-4.4# exit
```

3. Log in to the Bastion Host and run the following command to restart the deamonset:

   ```
   $ kubectl rollout restart daemonset calico-node -n kube-system
   ```

4. Run the following commands to delete deployment and reapply with YAML file. The calico interface MTU change takes effect while starting a new pod on the node.
   a. Verify that the deployment is READY 1/1 before delete and reapply:

      ```
      $ kubectl get deployment occne-kube-prom-stack-grafana -n occne-infra
      ```

      Sample output:

      ```
      NAME                            READY   UP-TO-DATE   AVAILABLE   AGE
      occne-kube-prom-stack-grafana   1/1     1            1           10h
      ```

   b. Run the following commands to delete the deployment and reapply with YAML file:

      ```
      $ kubectl get deployment occne-kube-prom-stack-grafana -n occne-infra -
      o yaml  > dp-occne-kube-prom-stack-grafana.yaml
      $ kubectl delete deployment occne-kube-prom-stack-grafana -n occne-
      infra
      $ kubectl apply -f dp-occne-kube-prom-stack-grafana.yaml
      ```

5. Run the following commands to verify the MTU change on worker nodes:
   - Verify which node has the new pod:

     ```
     $ kubectl get pod -A -o wide | grep occne-kube-prom-stack-grafana
     ```

     Sample output:

     ```
      occne-infra     occne-kube-prom-stack-grafana-79f9b5b488-
     cl76b                    3/3     Running     0               60s
     10.233.120.22   k8s-node-2.littlefinger.lab.us.oracle.com     <none>
     <none>
     ```

- Use SSH to log in to the node and check the calico interface change. Change only the last interface MTU due to the new pod for the services. Other calico interfaces' MTU will be changed when other services are changed.

  ```
  $ ssh k8s-node-2.littlefinger.lab.us.oracle.com

  [admusr@k8s-node-2 ~] $ ip link
  ```

  Sample output:

  ```
  ...
  35: calia44682149a1@if4: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1480 qdisc
  noqueue state UP mode DEFAULT group default
  link/ether ee:ee:ee:ee:ee:ee brd ff:ff:ff:ff:ff:ff link-netns cni-7f1a8116-5acf-
  b7df-5d6a-eb4f56330cf1
  115: calif0adcd64a1c@if4: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu <MTU value>
  qdisc noqueue state UP mode DEFAULT group default
  link/ether ee:ee:ee:ee:ee:ee brd ff:ff:ff:ff:ff:ff link-netns
  cni-7b99dc36-3b3b-75c6-e27c-9045eeb8242d
  ```

7.3.3 Changing Metrics Storage Allocation

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

Prerequisites

The revised amount of persistent storage required by metrics must be calculated. Rerun the metrics storage calculations as provided in the "Preinstallation Tasks" section of *Oracle Communications Cloud Native Core, Cloud Native Environment Installation, Upgrade, and Fault Recovery Guide*, and record the calculated *total_metrics_storage* value.

> **Note**
>
> When you increase the storage size for Prometheus, the retention size must also be increased to maintain the purging cycle of Prometheus. The default retention is set to 6.8 GB. If the storage is increased to a higher value and retention remains at 6.8 GB, the amount of data that is stored inside the storage is still 6.8 GB. Therefore, follow the Changing Retention Size of Prometheus procedure to calculate the retention size and update the retention size in Prometheus. These steps are applied while performing Step 3.

Procedure

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

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

   Sample output:

   ```
    NAME                                    VERSION   DESIRED   READY
   RECONCILED   AVAILABLE   AGE
   ```

```
occne-kube-prom-stack-kube-prometheus   v2.44.0   2         2
True         True        20h
```

2. Run the following command to edit the Prometheus resource:

```
$ kubectl edit prometheus occne-kube-prom-stack-kube-prometheus -n occne-
infra
```

> **Note**
>
> You are placed in a *vi* editor session that contains all of the configuration for the CNE Prometheus instances.
> Scroll down to the line that contains the "storage" key(line-91), then change the value to <desired pv size>. Also scroll down to the line that contains the "replicas" key(line-54), then change the value to 0. This scales down both the pods. The file must look similar to the following example.

Sample output:

```
53       release: occne-kube-prom-stack
    54   replicas: 2
    55   resources:
    56     limits:
    57       cpu: 2000m
    58       memory: 4Gi
    59     requests:
    60       cpu: 2000m
    61       memory: 4Gi
    62   retention: 14d
    63   retentionSize: 6.8GB
    64   routePrefix: /occne4-utpalkant-kumar/prometheus
    65   ruleNamespaceSelector: {}
    66   ruleSelector:
    67     matchLabels:
    68       role: cnc-alerting-rules
    69   scrapeInterval: 1m
    70   scrapeTimeout: 30s
    71   secrets:
    72   - etcd-occne4-utpalkant-kumar-k8s-ctrl-1
    73   - etcd-occne4-utpalkant-kumar-k8s-ctrl-2
    74   - etcd-occne4-utpalkant-kumar-k8s-ctrl-3
    75   securityContext:
    76     fsGroup: 2000
    77     runAsGroup: 2000
    78     runAsNonRoot: true
    79     runAsUser: 1000
    80   serviceAccountName: occne-kube-prom-stack-kube-prometheus
    81   serviceMonitorNamespaceSelector: {}
    82   serviceMonitorSelector: {}
    83   shards: 1
    84   storage:
    85     volumeClaimTemplate:
    86       spec:
    87         accessModes:
    88         - ReadWriteOnce
    89         resources:
    90           requests:
```

```
91            storage: 10Gi
92        storageClassName: occne-metrics-sc
```

> **Note**
>
> Type ":wq" to exit the editor session and save the changes. Verify that Prometheus instances are scaled down.

3. To change the pvc size of Prometheus pods, run the following command:

   ```
   $ kubectl edit pvc prometheus-occne-kube-prom-stack-kube-prometheus-db-
   prometheus-occne-kube-prom-stack-kube-prometheus-0 -n occne-infra
   ```

   > **Note**
   >
   > You will be placed in a *vi* editor session that contains all of the configuration for the CNE Prometheus pvc. Scroll down to the line that contains the "spec.resources.requests.storage" key, then update the value to the <desired pv size>. The file must look similar to the following example:
   >
   > ```
   > spec:
   >   accessModes:
   >   - ReadWriteOnce
   >   resources:
   >     requests:
   >       storage: 10Gi
   >   storageClassName: occne-metrics-sc
   >   volumeMode: Filesystem
   > ```
   >
   > Type ":wq" to exit the editor session and save the changes.

4. To verify if the pvc size change is applied, run the following command:

   ```
   $ kubectl get pv |grep kube-prom-stack-kube-prometheus-0
   ```

   Sample output:

   ```
   pvc-3c595b70-4265-42e8-a0ca-623b28ce4221  10Gi  RWO  Delete  Bound  occne-
   infra/prometheus-occne-kube-prom-stack-kube-prometheus-db-prometheus-occne-
   kube-prom-stack-kube-prometheus-0   occne-metrics-sc
   ```

   > **Note**
   >
   > Wait until the new desired size "10Gi" gets reflected. Repeat step 3 and step 4 for "kube-prom-stack-kube-prometheus-1" pvc.

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

   ```
   $ kubectl edit prometheus occne-kube-prom-stack-kube-prometheus -n occne-infra
   ```

   > **Note**
   >
   > You will be placed in a *vi* editor session that contains all of the configuration for the CNE 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:
   >
   > ```
   > 53       release: occne-kube-prom-stack
   > 54   replicas: 2
   > 55   resources:
   > 56     limits:
   > 57       cpu: 2000m
   > 58       memory: 4Gi
   > 59     requests:
   > 60       cpu: 2000m
   > 61       memory: 4Gi
   > 62   retention: 14d
   > ```

6. To verify that the Prometheus pods are up and running, run the following command:

   ```
   $ kubectl get pods -n occne-infra | grep kube-prom-stack-kube-prometheus
   ```

   Example output:

   ```
   prometheus-occne-kube-prom-stack-kube-prometheus-0 2/2 Running 1 40s
   prometheus-occne-kube-prom-stack-kube-prometheus-1 2/2 Running 1 29s
   ```

7.3.4 Changing OpenSearch Storage Allocation

This section describes the procedure to increase the amount of persistent storage allocated to OpenSearch for data storage.

Prerequisites

- Calculate the revised amount of persistent storage required by OpenSearch. Rerun the OpenSearch storage calculations as provided in the "Preinstallation Taks" section of *Oracle Communications Cloud Native Core, Cloud Native Environment Installation, Upgrade, and Fault Recovery Guide*, and record the calculated `log_trace_active_storage` and `log_trace_inactive_storage` values.

Procedure

This procedure uses the value of `log_trace_active_storage` for opensearch-data PV size and `log_trace_inactive_storage` for opensearch-master PV size. The following table displays the sample PV sizes considered in this procedure:

OpenSearch Component	Current PV Size	Desired PV Size
occne-opensearch-master	500Mi	500Mi
occne-opensearch-data	10Gi	200Gi (log_trace_active_storage)
opensearch-data-replicas-count	5	7

Expanding PV size for opensearch-master nodes

1. Store the output of the current configuration values for the `os-master-helm-values.yaml` file.

   ```
   $ helm -n occne-infra get values occne-opensearch-master > os-master-helm-values.yaml
   ```

2. Update the PVC size block in the `os-master-helm-values.yaml` file. The PVC size must be updated to the newly required PVC size (in this case, 50Gi as per the sample value considered). The `os-master-helm-values.yaml` file is required in Step 8 to recreate `occne-opensearch-master` Statefulset.

   ```
   $ vi os-master-helm-values.yaml
   persistence:
     enabled: true
     image: occne-repo-host:5000/docker.io/busybox
     imageTag: 1.31.0
     size: <desired size>Gi
     storageClass: occne-esmaster-sc
   ```

3. Delete the statefulset of `occne-opensearch-cluster-master` by running the following command:

   ```
   $ kubectl -n occne-infra delete sts --cascade=orphan occne-opensearch-cluster-master
   ```

4. Delete the `occne-opensearch-cluster-master-2` pod by running the following command:

   ```
   $ kubectl -n occne-infra delete pod occne-opensearch-cluster-master-2
   ```

5. Update the PVC storage size in the PVC of `occne-opensearch-cluster-master-2` by running the following command:

   ```
   $ kubectl -n occne-infra patch -p '{ "spec": { "resources": { "requests": { "storage": "40Gi" }}}}' pvc occne-opensearch-cluster-master-occne-opensearch-cluster-master-2
   ```

6. Get the PV volume ID from the PVC of `opensearch-master-2`:

   ```
   $ kubectl get pvc -n occne-infra | grep master-2
   ```

   Sample output:

   ```
   occne-opensearch-cluster-master-occne-opensearch-cluster-master-2   Bound
   pvc-9d9897c1-b7b9-43a3-bf78-f03b91ea4d72   30Gi       RWO            occne-esmaster-sc   17h
   ```

In this case, the PV volume ID in the sample output is *pvc-9d9897c1-b7b9-43a3-bf78-f03b91ea4d72*.

7. Hold on to the PV attached to `occne-opensearch-cluster-master-2` PVC using the volume ID until the newly updated size gets reflected. Verify the updated PVC value by running the following command:

   ```
   $ kubectl get pv -w | grep pvc-9d9897c1-b7b9-43a3-bf78-f03b91ea4d72
   ```

   Sample output:

   ```
   pvc-9d9897c1-b7b9-43a3-bf78-f03b91ea4d72   30Gi       RWO
   Delete           Bound    occne-infra/occne-opensearch-cluster-master-
   occne-opensearch-cluster-master-2    occne-esmaster-sc            17h
   pvc-9d9897c1-b7b9-43a3-bf78-f03b91ea4d72   40Gi       RWO
   Delete           Bound    occne-infra/occne-opensearch-cluster-master-
   occne-opensearch-cluster-master-2    occne-esmaster-sc            17h
   ```

8. Run Helm upgrade to recreate the `occne-opensearch-master` statefulset:

   ```
   $ helm upgrade -f os-master-helm-values.yaml occne-opensearch-master
   opensearch-project/opensearch -n occne-infra
   ```

9. Once the deleted pod (master-2) and its statefulset are up and running, check the pod's PVC status and verify if it reflects the updated size.

   ```
   $ kubectl get pvc -n occne-infra | grep master-2
   ```

   Sample output:

   ```
   occne-opensearch-cluster-master-occne-opensearch-cluster-master-2
   Bound    pvc-9d9897c1-b7b9-43a3-bf78-f03b91ea4d72  40Gi
   RWO            occne-esmaster-sc   17h
   e.g id: pvc-9d9897c1-b7b9-43a3-bf78-f03b91ea4d72
   ```

10. Repeat steps 3 through 9 for each of the remaining pods, one after the other (in order master-1, master-0).

Expanding PV size for opensearch-data nodes

1. Store the output of the current configuration values for `os-master-helm-values.yaml` file.

   ```
   $ helm -n occne-infra get values occne-opensearch-data > os-data-helm-
   values.yaml
   ```

2. Update the PVC size block in the `os-master-helm-values.yaml` file. The PVC size must be updated to the newly required PVC size (in this case, 200Gi as per the sample value considered). The `os-master-helm-values.yaml` file is required in Step 8 of this procedure to recreate the `occne-opensearch-data` statefulset.

   ```
   $ vi os-data-helm-values.yaml
   ```

   Sample output:

```
persistence:
  enabled: true
  image: occne-repo-host:5000/docker.io/busybox
  imageTag: 1.31.0
  size: <desired size>Gi
  storageClass: occne-esdata-sc
```

3. Delete the statefulset of `occne-opensearch-opensearch-data` by the running the following command:

   ```
   $ kubectl -n occne-infra delete sts --cascade=orphan occne-opensearch-
   cluster-data
   ```

4. Delete the `occne-opensearch-cluster-data-2`.

   ```
   $ kubectl -n occne-infra delete pod occne-opensearch-cluster-data-2
   ```

5. Update the PVC storage size in the PVC of `occne-opensearch-cluster-data-2`.

   ```
   $ kubectl -n occne-infra patch -p '{ "spec": { "resources": { "requests":
   { "storage": "20Gi" }}}}' pvc occne-opensearch-cluster-data-occne-
   opensearch-cluster-data-2
   ```

6. Get the PV volume ID from the PVC of `opensearch-data-2`.

   ```
   $ kubectl get pvc -n occne-infra | grep data-2
   ```

   Sample output:

   ```
   occne-opensearch-cluster-data-occne-opensearch-cluster-data-2   Bound
   pvc-80a56d73-d7b7-417f-a7a7-c8484bc8171d   10Gi       RWO            occne-
   esdata-sc   17h
   ```

7. Hold on to the PV attached to opensearch-data-2 PVC using the volume ID until the newly updated size gets reflected. Verify the updated PVC value by running the following command:

   ```
   $ kubectl get pv -w | grep pvc-80a56d73-d7b7-417f-a7a7-c8484bc8171d
   ```

   Sample output:

   ```
   pvc-80a56d73-d7b7-417f-a7a7-c8484bc8171d   10Gi       RWO
   Delete           Bound    occne-infra/occne-opensearch-cluster-data-occne-
   opensearch-cluster-data-2    occne-esdata-sc            17h
   pvc-80a56d73-d7b7-417f-a7a7-c8484bc8171d   20Gi       RWO
   Delete           Bound    occne-infra/occne-opensearch-cluster-data-occne-
   opensearch-cluster-data-2    occne-esdata-sc            17h
   ```

8. Run helm upgrade to recreate the `occne-opensearch-data` statefulset

   ```
   $ helm upgrade -f os-data-helm-values.yaml occne-opensearch-data
   opensearch-project/opensearch -n occne-infra
   ```

9. Once the deleted pod (data-2) and its statefulset are up and running, check the pod's PVC status and verify if it reflects the updated size.

   ```
   $ kubectl get pvc -n occne-infra | grep data-2
   ```

   Sample output:

   ```
   occne-opensearch-cluster-data-occne-opensearch-cluster-data-2   Bound
   pvc-80a56d73-d7b7-417f-a7a7-c8484bc8171d  20Gi       RWO            occne-
   esdata-sc   17h
   ```

10. Repeat steps 3 through 9 for each of the remaining pods, one after the other (in the order, data-1, data-0,..).

7.3.5 Changing the RAM and CPU Resources for Common Services

This section describes the procedure to change the RAM and CPU resources for CNE common services.

Prerequisites

Before changing the RAM, CPU, or both the resources for CNE common services, make sure that the following prerequisites are met:

- The cluster must be in a healthy state. This can verified by checking if all the common services are up and running.

> **Note**
>
> – When changing the CPU and RAM resources for any component, the limit value must always be greater than or equal to the requested value.
>
> – Run all the commands in this section from the Bastion Host.

7.3.5.1 Changing the Resources for Prometheus

This section describes the procedure to change the RAM or CPU resources for Prometheus.

Procedure

1. Run the following command to edit the Prometheus resource:

   ```
   kubectl edit prometheus occne-kube-prom-stack-kube-prometheus -n occne-
   infra
   ```

   The system opens a `vi` editor session that contains all the configuration for the CNE Prometheus instances.

2. Scroll to the resources section and change the CPU and Memory resources to the desired values. This updates the resources for both the prometheus pods.
   For example:

   ```
   resources:
       limits:
   ```

```
      cpu: 2000m
      memory: 4Gi
    requests:
      cpu: 2000m
      memory: 4Gi
```

3. Type `:wq` to exit the editor session and save the changes.
4. Verify if both the Prometheus pods are restarted:

   ```
   kubectl get pods -n occne-infra |grep kube-prom-stack-kube-prometheus
   ```

   Sample output:

   ```
   prometheus-occne-kube-prom-stack-kube-prometheus-0             2/2     Running
   0             85s
   prometheus-occne-kube-prom-stack-kube-prometheus-1             2/2     Running
   0             104s
   ```

7.3.5.2 Changing the Resources for Alertmanager

This section describes the procedure to change the RAM or CPU resources for Alertmanager.

Procedure

1. Run the following command to edit the Alertmanager resource:

   ```
   kubectl edit alertmanager occne-kube-prom-stack-kube-alertmanager -n occne-
   infra
   ```

   The system opens a `vi` editor session that contains all the configuration for the CNE Alertmanager instances.
2. Scroll to the resources section and change the CPU and Memory resources to the desired values. This updates the resources for the Alertmanager pods.
   For example:

   ```
   resources:
       limits:
         cpu: 20m
         memory: 64Mi
       requests:
         cpu: 20m
         memory: 64Mi
   ```

3. Type `:wq` to exit the editor session and save the changes.
4. Verify if the Alertmanager pods are restarted:

   ```
   kubectl get pods -n occne-infra |grep alertmanager
   ```

   Sample output:

   ```
   alertmanager-occne-kube-prom-stack-kube-alertmanager-0         2/2     Running
   0             16s
   alertmanager-occne-kube-prom-stack-kube-alertmanager-1         2/2     Running
   0             35s
   ```

7.3.5.3 Changing the Resources for Grafana

This section describes the procedure to change the RAM or CPU resources for Grafana.

Procedure

1. Run the following command to edit the Grafana resource:

   ```
   kubectl edit deploy occne-kube-prom-stack-grafana -n occne-infra
   ```

   The system opens a `vi` editor session that contains all the configuration for the CNE Grafana instances.
2. Scroll to the resources section and change the CPU and Memory resources to the desired values. This updates the resources for the Grafana pod.
   For example:

   ```
   resources:
       limits:
         cpu: 100m
         memory: 128Mi
       requests:
         cpu: 100m
         memory: 128Mi
   ```

3. Type `:wq` to exit the editor session and save the changes.
4. Verify if the Grafana pod is restarted:

   ```
   kubectl get pods -n occne-infra |grep grafana
   ```

   Sample output:

   ```
   occne-kube-prom-stack-grafana-84898d89b4-nzkr4                3/3     Running
   0             54s
   ```

7.3.5.4 Changing the Resources for Kube State Metrics

This section describes the procedure to change the RAM or CPU resources for kube-state-metrics.

Procedure

1. Run the following command to edit the kube-state-metrics resource:

   ```
   kubectl edit deploy occne-kube-prom-stack-kube-state-metrics -n occne-infra
   ```

   The system opens a `vi` editor session that contains all the configuration for the CNE kube-state-metrics instances.
2. Scroll to the resources section and change the CPU and Memory resources to the desired values. This updates the resources for the kube-state-metrics pod.

For example:

```
resources:
    limits:
      cpu: 20m
      memory: 100Mi
    requests:
      cpu: 20m
      memory: 32Mi
```

3. Type `:wq` to exit the editor session and save the changes.
4. Verify if the kube-state-metrics pod is restarted:

   ```
   kubectl get pods -n occne-infra |grep kube-state-metrics
   ```

   Sample output:

   ```
   occne-kube-prom-stack-kube-state-metrics-cff54c76c-t5k7p        1/1     Running
   0             20s
   ```

7.3.5.5 Changing the Resources for OpenSearch

This section describes the procedure to change the RAM or CPU resources for OpenSearch.

Procedure

1. Run the following command to edit the opensearch-master resource:

   ```
   kubectl edit sts occne-opensearch-cluster-master -n occne-infra
   ```

   The system opens a `vi` editor session that contains all the configuration for the CNE opensearch-master instances.
2. Scroll to the resources section and change the CPU and Memory resources to the desired values. This updates the resources for the opensearch-master pod.
   For example:

   ```
   resources:
       limits:
         cpu: "1"
         memory: 2Gi
       requests:
         cpu: "1"
         memory: 2Gi
   ```

3. Type `:wq` to exit the editor session and save the changes.
4. Verify if the opensearch-master pods are restarted:

   ```
   kubectl get pods -n occne-infra |grep opensearch-cluster-master
   ```

   Sample output:

   ```
   occne-opensearch-cluster-master-0                          1/1     Running
   0             3m34s
   occne-opensearch-cluster-master-1                          1/1     Running
   ```

```
0             4m8s
occne-opensearch-cluster-master-2                          1/1     Running
0             4m19s
```

> **Note**
>
> Repeat this procedure for opensearch-data and opensearch-client pods if required.

7.3.5.6 Changing the Resources for OpenSearch Dashboard

This section describes the procedure to change the RAM or CPU resources for OpenSearch Dashboard.

Procedure

1. Run the following command to edit the opensearch-dashboard resource:

   ```
   kubectl edit deploy occne-opensearch-dashboards -n occne-infra
   ```

   The system opens a `vi` editor session that contains all the configuration for the CNE opensearch-dashboard instances.
2. Scroll to the resources section and change the CPU and Memory resources to the desired values. This updates the resources for the opensearch-dashboard pod.
   For example:

   ```
   resources:
       limits:
         cpu: 100m
         memory: 512Mi
       requests:
         cpu: 100m
         memory: 512Mi
   ```

3. Type `:wq` to exit the editor session and save the changes.
4. Verify if the opensearch-dashboard pod is restarted:

   ```
   kubectl get pods -n occne-infra |grep dashboard
   ```

   Sample output:

   ```
   occne-opensearch-dashboards-7b7749c5f7-jcs7d               1/1     Running
   0             20s
   ```

7.3.5.7 Changing the Resources for Fluentd OpenSearch

This section describes the procedure to change the RAM or CPU resources for Fluentd OpenSearch.

Procedure

1. Run the following command to edit the `occne-fluentd-opensearch` resource:

   ```
   kubectl edit ds occne-fluentd-opensearch -n occne-infra
   ```

   The system opens a `vi` editor session that contains all the configuration for the CNE Fluentd OpenSearch instances.
2. Scroll to the resources section and change the CPU and memory resources to the desired values. This updates the resources for the Fluentd OpenSearch pods.
   For example:

   ```
   resources:
       limits:
         cpu: 100m
         memory: 128Mi
       requests:
         cpu: 100m
         memory: 128Mi
   ```

3. Type `:wq` to exit the editor session and save the changes.
4. Verify if the Fluentd OpenSearch pods are restarted:

   ```
   kubectl get pods -n occne-infra |grep fluentd-opensearch
   ```

   Sample output:

   ```
   occne-fluentd-opensearch-kcx87                                    1/1
   Running   0             19s
   occne-fluentd-opensearch-m9zhz                                    1/1
   Running   0             9s
   occne-fluentd-opensearch-pbbrw                                    1/1
   Running   0             14s
   occne-fluentd-opensearch-rstqf                                    1/1
   Running   0             4s
   ```

7.3.5.8 Changing the Resources for Jaeger Agent

This section describes the procedure to change the RAM or CPU resources for Jaeger Agent.

Procedure

1. Run the following command to edit the jaeger-agent resource:

   ```
   kubectl edit ds occne-tracer-jaeger-agent -n occne-infra
   ```

   The system opens a `vi` editor session that contains all the configuration for the CNE jaeger-agent instances.

2. Scroll to the resources section and change the CPU and Memory resources to the desired values. This updates the resources for the jaeger-agent pods.
   For example:

   ```
   resources:
       limits:
         cpu: 500m
         memory: 512Mi
       requests:
         cpu: 256m
         memory: 128Mi
   ```

3. Type `:wq` to exit the editor session and save the changes.
4. Verify if the jaeger-agent pods are restarted:

   ```
   kubectl get pods -n occne-infra |grep jaeger-agent
   ```

   Sample output:

   ```
   occne-tracer-jaeger-agent-dpn4v                                  1/1     Running
   0             58s
   occne-tracer-jaeger-agent-dvpnv                                  1/1     Running
   0             62s
   occne-tracer-jaeger-agent-h4t67                                  1/1     Running
   0             55s
   occne-tracer-jaeger-agent-q92ld                                  1/1     Running
   0             51s
   ```

7.3.5.9 Changing the Resources for Jaeger Query

This section describes the procedure to change the RAM or CPU resources for Jaeger Query.

Procedure

1. Run the following command to edit the jaeger-query resource:

   ```
   kubectl edit deploy occne-tracer-jaeger-query -n occne-infra
   ```

   The system opens a `vi` editor session that contains all the configuration for the CNE jaeger-query instances.
2. Scroll to the resources section and change the CPU and Memory resources to the desired values. This updates the resources for the jaeger-query pod.
   For example:

   ```
   resources:
       limits:
         cpu: 500m
         memory: 512Mi
       requests:
         cpu: 256m
         memory: 128Mi
   ```

3. Type `:wq` to exit the editor session and save the changes.

4. Verify if the jaeger-query pod is restarted:

   ```
   kubectl get pods -n occne-infra |grep jaeger-query
   ```

   Sample output:

   ```
   occne-tracer-jaeger-query-67bdd85fcb-hw67q                    2/2     Running
   0             19s
   ```

   Note

   Repeat this procedure for the jaeger-collector pod if required.

7.3.6 Activating and Configuring Local DNS

This section provides information about activating and configuring local DNS.

7.3.6.1 Activating Local DNS

Local DNS allows CNE clusters to perform domain name lookups within Kubernetes clusters. This section provides the procedure to activate the local DNS feature on a CNE cluster.

Note

Before activating Local DNS, ensure that you are aware about the following conditions:

- Local DNS does not handle backups of any added record.
- You must run this procedure to activate local DNS only after installing or upgrading to release 23.4.x.

7.3.6.1.1 Prerequisites

Before activating local DNS, ensure that the following prerequisites are met:

- Ensure that the cluster is running in a healthy state.
- Ensure that the CNE cluster is running with version 23.4.x. You can validate the CNE version by echoing the `OCCNE_VERSION` environment variable on Bastion Host:

  ```
  echo $OCCNE_VERSION
  ```

- Ensure that the cluster is running with the Bastion DNS configuration.

7.3.6.1.2 Preactivation Checks

This section provides information about the checks that are performed before activating local DNS.

Determining the Active Bastion Host

1. Log in to one of the Bastion Hosts (for example, Bastion 1) and determine if that Bastion Host is active or not by running the following command:

   ```
   $ is_active_bastion
   ```

   The system displays the following output if the Bastion Host is active :

   ```
   IS active-bastion
   ```

2. If the current Bastion is not active, then log in to the mate Bastion Host and verify if it is active:

   ```
   $ is_active_bastion
   ```

   The system displays the following output if the Bastion Host is active :

   ```
   IS active-bastion
   ```

Verifying if Local DNS is Already Activated

1. Navigate to the cluster directory:

   ```
   $ cd /var/occne/cluster/${OCCNE_CLUSTER}
   ```

2. Open the `occne.ini` file (for vCNE) or `hosts.ini` file (for Bare Metal) and verify if the `local_dns_enabled` variable under the `occne:vars` header is set to *False*.
   Example for vCNE:

   ```
   $ cat occne.ini
   ```

   Sample output:

   ```
   [occne:vars]
   .
   local_dns_enabled=False
   .
   ```

   Example for Bare Metal:

   ```
   $ cat hosts.ini
   ```

   Sample output:

   ```
   [occne:vars]
   .
   local_dns_enabled=False
   .
   ```

   If `local_dns_enabled` is set to *True*, then it indicates that local DNS feature is already enabled in the CNE cluster.

> **Note**
>
> Ensure that the first character of the variable value (*True* or *False*) is capitalized and there is no space before and after the equal to sign.

7.3.6.1.3 Enabling Local DNS

This section provides the procedure to enable Local DNS in a CNE cluster.

1. Log in to the active Bastion Host and run the following command to navigate to the cluster directory:

   ```
   $ cd /var/occne/cluster/${OCCNE_CLUSTER}
   ```

2. Open the `occne.ini` file (for vCNE) or `hosts.ini` file (for Bare metal) in edit mode:
   Example for vCNE:

   ```
   $ vi occne.ini
   ```

   Example for Bare Metal:

   ```
   $ vi hosts.ini
   ```

3. Set the `local_dns_enabled` variable under the `occne:vars` header to *True*. If the `local_dns_enabled` variable is not present under the `occne:vars` header, then add the variable.

   > **Note**
   >
   > Ensure that the first character of the variable value (*True* or *False*) is capitalized and there is no space before and after the equal to sign.

   For example,

   ```
   [occne:vars]
   .
   local_dns_enabled=True
   .
   ```

4. For vCNE (OpenStack or VMware) deployments, additionally add the `provider_domain_name` and `provider_ip_address` variables under the `occne:vars` section of the `occne.ini` file. You can obtain the provider domain name and IP address from the provider administrator and set the variable values accordingly.
   The following block shows the sample `occne.ini` file with the additional variables:

   ```
   [occne:vars]
   .
   local_dns_enabled=True
   provider_domain_name=<cloud provider domain name>
   provider_ip_address=<cloud provider IP address>
   .
   ```

5. Update the cluster with the new settings in the ini file:

   ```
   $ OCCNE_CONTAINERS=(K8S) OCCNE_STAGES=(DEPLOY) OCCNE_ARGS='--tags=coredns'
   pipeline.sh
   ```

7.3.6.1.4 Validating Local DNS

This section provides the steps to validate if you have successfully enabled local DNS.

Local DNS provides the `validateLocalDns.py` script to validate if you have successfully enabled Local DNS. The `validateLocalDns.py` script is located in the `/var/occne/cluster/${OCCNE_CLUSTER}/artifacts/maintenance/validateLocalDns.py` folder. This automated script validates Local DNS by performing the following actions:

1. Creating a test record
2. Reloading local DNS
3. Querying the test record from within a pod
4. Getting the response (Success status)
5. Deleting the test record

To run the `validateLocalDns.py` script:

1. Log in to the active Bastion Host and navigate to the cluster directory:

   ```
   $ cd /var/occne/cluster/${OCCNE_CLUSTER}
   ```

2. Run the `validateLocalDns.py` script:

   ```
   $ ./artifacts/maintenance/validateLocalDns.py
   ```

   Sample output:

   ```
   Beginning local DNS validation
    - Validating local DNS configuration in occne.ini
    - Adding DNS A record.
    - Adding DNS SRV record.
    - Reloading local coredns.
    - Verifying local DNS A record.
      - DNS A entry has not been propagated, retrying in 10 seconds (retry 1/5)
    - Verifying local DNS SRV record.
    - Deleting DNS SRV record.
    - Deleting DNS A record.
    - Reloading local coredns.
   Validation successful
   ```

   > **Note**
   >
   > If the script encounters an error, it returns an error message indicating which part of the process failed. For more information about troubleshooting local DNS errors, see Troubleshooting Local DNS.

3. Once you successfully enable Local DNS, add the external hostname records using the Local DNS API to resolve external domain names using CoreDNS. For more information, see Adding and Removing DNS Records.

7.3.6.2 Adding and Removing DNS Records

This section provides the procedures to add and remove DNS records ("A" records and SRV records) using Local DNS API to the core DNS configuration.

Each Bastion Host runs a version of the Local DNS API as a service on port 8000. The system doesn't require any authentication from inside a Bastion Host and runs the API requests locally.

7.3.6.2.1 Prerequisites

Before adding or removing DNS records, ensure that the following prerequisites are met:

- The Local DNS feature must be enabled on the cluster. For more information about enabling Local DNS, see Activating Local DNS.
- The CNE cluster version must be 23.2.x or above.

7.3.6.2.2 Adding an A Record

This section provides information on how to use the Local DNS API to create or add an A record in the CNE cluster.

The system creates zones to group records with similar domain names together. When an API call is made to the Local DNS API, the request payload (request body) is used to create the zones automatically. The system creates a new zone when it identifies a domain name for the first time. Henceforth, it stores each request coming with the same domain name within the same zone.

> **Note**
>
> - You cannot create and maintain identical A records.
> - You cannot create two A records with the same name.
> - You cannot create two A records with the same IP address within the same zone.

The following table provides details on how to use the Local DNS API to add an "A" record:

Table 7-1 Adding an A Record

Request URL	HTTP Method	Content Type	Request Body	Response Code	Sample Response

`http://localhost:8000/occne/dns/a`	POST	application/json	`{ "name": "string", "ttl": "integer", "ip-address": "string" }`
			
			Note: Define each field in the request body within double quotes (" ").
			Sample request:
			
			`curl -X POST http://localhost:8000/occne/dns/a \ -H 'Content-Type: application/json' \ -d '{"name":"occne.lab.oracle.com","ttl":"3600","ip-address":"175.80.200.20"}'`
				• 200: Record Added Successfully.
				• 400: Already exists, request payload/ parameters incomplete or not valid.
				• 503: Could not be added, internal error.
					`200: DNS A record added in coredns file for occne.lab.oracle.com 175.80.200.20 3600, msg SUCCESS: Zone info and A record updated for domain name`

The following table provides details about the request body parameters:

Table 7-2 Request Body Parameters

Parameter	Required or Optional	Type	Description
name	Required	string	Fully-Qualified Domain Name (FQDN) to be include in the core DNS. This parameter can contain multiple subdomains where each subdomain can range between 1 and 63 characters and contain the following characters: `[a-zA-Z0-9_-]`. This parameter cannot start or end with `-` or `_`. The last segment of this parameter is taken as the domain name to create zones. For example, `sample.oracle.com`
ip-address	Required	string	The IP address to locate a service. For example, `xxx.xxx.xxx.xxx`. The API supports IPv4 protocol only.
ttl	Required	integer	The Time To Live (TTL) in seconds. This is the amount of time the record is allowed to be cached by a resolver. The minimum and the maximum value that can be set are 300 and 3600 respectively.

7.3.6.2.3 Deleting an A Record

This section provides information on how to use the Local DNS API to delete an A record in the CNE cluster.

> **Note**
>
> - When the last A record in a zone is deleted, the system deletes the zone as well.
> - You cannot delete an A record that is linked to an existing SRV record. You much first delete the linked SRV record to delete the A record.

The following table provides details on how to use the Local DNS API to delete an "A" record:

Table 7-3 Deleting an A Record

Request URL	HTTP Method	Content Type	Request Body	Response Code	Sample Response

`http://localhost:8000/occne/dns/a`	DELETE	application/json	`{ "name": "string", "ip-address": "string" }`
			
			Note: Define each field in the request body within double quotes (" ").
			Sample request:
			
			`curl -X DELETE http://localhost:8000/occne/dns/a \ -H 'Content-Type: application/json' \ -d '{"name":"occne.lab.oracle.com","ip-address":"175.80.200.20"}'`
				• 200: Record Deleted Successfully.
				• 400: Record does not exist, or request payload/ parameters are incomplete or not valid.
				• 503: Could not be deleted, internal error.
					200: `DNS A record added in coredns file for occne.lab.oracle.com 175.80.200.203600, msg SUCCESS: Zone info and A record updated for domain name`

The following table provides details about the request body parameters:

Table 7-4 Request Body Parameters

Parameter	Required or Optional	Type	Description
name	Required	string	Fully-Qualified Domain Name (FQDN). This parameter can contain multiple subdomains where each subdomain can range between 1 and 63 characters and contain the following characters: `[a-zA-Z0-9_-]`. This parameter cannot start or end with `-` or `_`. For example, `sample.oracle.com`
ip-address	Required	string	The IP address to locate a service. For example, `xxx.xxx.xxx.xxx`.

7.3.6.2.4 Adding an SRV Record

This section provides information on how to use the Local DNS API to create or add an SRV record in the CNE cluster.

SRV records are linked to A records. To add a new SRV record, you must already have an A record in the system. Adding an SRV record creates a dependency between the A record and the SRV record. Therefore, to delete an A record, you must first delete the SRV record pointing to the A record. When you are adding an SRV record, the system adds the SRV record to the designated zone, matching the domain name and the related A record.

> **Note**
>
> - You cannot create and maintain identical SRV records. However, you can have a different protocol for the same combo service and target A record.
> - Currently, there is no provision to edit an existing SRV record. If you want to edit an SRV record, then delete the existing SRV record and then re-add the record with the updated parameters (weight, priority, or TTL).

The following table provides details on how to use the Local DNS API to create an SRV record:

Table 7-5 Adding an SRV Record

Request URL	HTTP Method	Content Type	Request Body	Response Code	Sample Response

`https://localhost:8000/occne/dns/srv`	POST	application/json	`{ "service": "string", "protocol": "string", "dn": "string", "ttl": "integer" "priority": "integer", "weight": "integer", "port": "integer", "server": "string", "a_record": "string" }`
			
			Note: Define each field in the request body within double quotes (" ").
			Sample request:
			
			`curl -X POST http://localhost:8000/occne/dns/srv \ -H 'Content-Type: application/json' \ -d '{"service":"sip","protocol":"tcp","dn":"lab.oracle.com","ttl":"3600","weight":"100","port":"35061","server":"occne","priority":"10","a_`
				• 200: Record Added Successfully.
				• 400: Request payload/ parameters incomplete or not valid.
				• 409: Already exists.
				• 503: Could not be added, internal error.
					200: `SUCCESS: SRV record successfully added to config map coredns.`

Table 7-5 (Cont.) Adding an SRV Record

Request URL	HTTP Method	Content Type	Request Body	Response Code	Sample Response
			`record":"occne.lab.oracle.com"}'`		

The following table provides details about the request body parameters:

Table 7-6 Request Body Parameters

Parameter	Required or Optional	Type	Description
service	Required	string	The symbolic name for the service, such as "sip", and "my_sql". The value of this parameter can range between 1 and 63 characters and contain the following characters: [a-zA-Z0-9_-]. The parameter cannot start or end with - or _.
protocol	Required	string	The protocol supported by the service. The allowed values are: • tcp • tcp

Table 7-6 (Cont.) Request Body Parameters

Parameter	Required or Optional	Type	Description
dn	Required	string	The domain name that the SRV record is applicable to. This parameter can contain multiple subdomains where each subdomain can range between 1 and 63 characters and contain the following characters: `[a-zA-Z0-9_-]`. For example: `lab.oracle.com.` If the SRV record is applicable to the entire domain, then provide only the domain name without subdomains. For example, `oracle.com` The length of the Top Level Domains (TLD) must be between 1 and 6 characters and must only contain the following characters: `[a-z]`. For example: .com, .net, and .uk.
ttl	Required	integer	The Time To Live (TTL) in seconds. This is the amount of time the record is allowed to be cached by a resolver. This value can range between 300 and 3600.
priority	Required	integer	The priority of the current SRV record in comparison to the other SRV records. The values can range from 0 to n.
weight	Required	integer	The weight of the current SRV record in comparison to the other SRV records with the same priority. The values can range from 0 to n.
port	Required	integer	The port on which the target service is found. The values can range from 1 to 65535.

Table 7-6 (Cont.) Request Body Parameters

Parameter	Required or Optional	Type	Description
server	Required	string	The name of the machine providing the service without including the domain name (value provided in the `dn` field). The value can range between 1 and 63 characters and contain the following characters: `[a-zA-Z0-9_-]`. The parameter cannot start or end with `-` or `_`. For example: `occne-node1`.
a_record	Required	string	The "A" record name to which the SRV is added. The "A" record mentioned here must be already added. Otherwise the request fails.

7.3.6.2.5 Deleting an SRV Record

This section provides information on how to use the Local DNS API to delete an SRV record in the CNE cluster.

> **Note**
>
> To delete an SRV record, the details in the request payload must exactly match the details, such as weight, priority, and ttl, of an existing SRV record.

The following table provides details on how to use the Local DNS API to delete an SRV record:

Table 7-7 Deleting an SRV Record

Request URL	HTTP Method	Content Type	Request Body	Response Code	Sample Response

`https://localhost:8000/occne/dns/srv`	DELETE	application/json	`{ "service": "string", "protocol": "string", "dn": "string", "ttl": "integer" "priority": "integer", "weight": "integer", "port": "integer", "server": "string" "a_record": "string" }`
			
			Note: Define each field in the request body within double quotes (" ").
			Sample request:
			
			`curl -X DELETE http://localhost:8000/occne/dns/srv \ -H 'Content-Type: application/json' \ -d '{"service":"sip","protocol":"tcp","dn":"lab.oracle.com","ttl":"3600","weight":"100","port":"35061","server":"occne","priority":"10","a_`
				• 200: Record Deleted Successfully.
				• 400: Record does not exist, or request payload/ parameters incomplete or not valid.
				• 503: Could not be deleted, internal error.
					200: `SUCCESS: SRV record successfully deleted from config map coredns`

Table 7-7 (Cont.) Deleting an SRV Record

Request URL	HTTP Method	Content Type	Request Body	Response Code	Sample Response
			`record":"occne.lab.oracle.com"}'`		

The following table provides details about the request body parameters:

Table 7-8 Request Body Parameters

Parameter	Required or Optional	Type	Description
service	Required	string	The symbolic name for the service, such as "sip", and "my_sql". The value of this parameter can range between 1 and 63 characters and contain the following characters: [a-zA-Z0-9_-]. The parameter cannot start or end with `-` or _.
protocol	Required	string	The protocol supported by the service. The allowed values are: • tcp • tcp
dn	Required	string	The domain name that the SRV record is applicable to. This parameter can contain multiple subdomains where each subdomain can range between 1 and 63 characters and contain the following characters: `[a-zA-Z0-9_-]`. The length of the Top Level Domains (TLD) must be between 1 and 6 characters and must only contain the following characters: `[a-z]`. For example: .com, .net, and .uk.

Table 7-8 (Cont.) Request Body Parameters

Parameter	Required or Optional	Type	Description
ttl	Required	integer	The Time To Live (TTL) in seconds. This is the amount of time the record is allowed to be cached by a resolver. This value can range between 300 and 3600.
priority	Required	integer	The priority of the current SRV record in comparison to the other SRV records. The values can range from 0 to n.
weight	Required	integer	The weight of the current SRV record in comparison to the other SRV records with the same priority. The values can range from 0 to n.
port	Required	integer	The port on which the target service is found. The values can range from 1 to 65535.
server	Required	string	The name of the machine providing the service minus the domain name (the value in the dn field). The value can range from 1 and 63 characters and contain the following characters: `[a-zA-Z0-9_-]`. The parameter cannot start or end with `-` or _.
a_record	Required	string	The "A" record name from which the SRV is deleted. The "A" record mentioned here must be already added. Otherwise the request fails.

7.3.6.3 Reloading Local or Core DNS Configurations

This section provides information about reloading core DNS configuration using the `reload` endpoint provided by Local DNS API.

> **Note**
>
> You must reload the core DNS configuration to commit the last configuration update, whenever you:
>
> - add or remove multiple records in the same zone
> - update a single or multiple DNS records

The following table provides details on how to use the Local DNS API endpoint to reload the core DNS configuration:

Table 7-9 Reloading Local or Core DNS Configurations

Request URL	HTTP Method	Content Type	Request Body	Response Code	Sample Response

`http://localhost:8000/occne/coredns/reload`	POST	application/json	`{ "deployment-name": "string", "namespace": "string" }`
			
			Note:
			• Define each field in the request body within double quotes (" ").
			• Currently, the system always uses the default values for the `"deployment-name"` and `"namespace"` parameters and doesn't consider any modification done to them. This will be address in the future releases.
			
			Sample request to reload the core DNS without payload (using the default values):
			
			`curl -X POST http://localhost:8000/occne/coredns/reload`
			
			Sample request to reload the core DNS using the payload:
			
			`curl -X POST http://localhost:8000/occne/coredns/reload \ -H 'Content-`
				• 200: Local DNS or Core DNS Reloaded Successfully.
				• 400: Request payload/ parameters incomplete or not valid.
				• 503: Could not be reloaded, internal error.
					200: `Deployment reloaded, msg SUCCESS: Reloaded coredns deployment in ns kube-system`

Table 7-9 (Cont.) Reloading Local or Core DNS Configurations

Request URL	HTTP Method	Content Type	Request Body	Response Code	Sample Response
			`Type: application/ json' \ -d '{"deployment-name":"coredns","namespace":"kube-system"}'`		

The following table provides details about the request body parameters:

Table 7-10 Request Body Parameters

Parameter	Required or Optional	Type	Description
deployment-name	Required	string	The deployment Name to be reloaded. The value must be a valid Kubernetes deployment name. The default value is *coredns*.
namespace	Required	string	The namespace where the deployment exists. The value must be a valid Kubernetes namespace name. The default value is *kube-system*.

7.3.6.4 Other Local DNS API Endpoints

This section provides information about the additional endpoints provided by Local DNS API.

Get Data

The Local DNS API provides an endpoint to get the current configuration, zones and records of local DNS or core DNS.

The following table provides details on how to use the Local DNS API endpoint to get the Local DNS or core DNS configuration details:

Table 7-11 Get Local DNS or Core DNS Configurations

Request URL	HTTP Method	Content Type	Request Body	Response Code	Sample Response
`http://localhost:8000/occne/dns/data`	GET	NA	Sample request: `curl -X GET http://localhost:8000/occne/dns/data`	• 200: Returns core DNS configmap data, including zones and records. • 503: Could not get data, internal error.	200: `[True, {'api_version': 'v1', 'binary_data': None, 'data': {'Corefile': '.:53 {\n' ... # Output Omitted ... 'db.oracle.com': ';oracle.com db file\n' 'oracle.com. 300 ' 'IN SOA ns1.oracle.com andrei.oracle.com ' '201307231 3600 10800 86400 3600\n' 'occne1.us.oracle.com. ' '3600 IN A ' '10.65.200.182\n' '_sip._tcp.lab.o`

Table 7-11 (Cont.) Get Local DNS or Core DNS Configurations

Request URL	HTTP Method	Content Type	Request Body	Response Code	Sample Response
					racle.com 30 IN SRV 10 102 32061 ' 'occne.lab.oracle.com.\n' 'occne.lab.oracle.com. ' '3600 IN A ' '175.80.200.20\n', ... # Output Omitted ...

7.3.6.5 Troubleshooting Local DNS

This section describes the issues that you may encounter while configuring Local DNS and their troubleshooting guidelines.

By design, the Local DNS functionality is built on top of the core DNS (CoreDNS). Therefore, all the troubleshooting, logging, and configuration management are performed directly on the core DNS. Each cluster runs a CoreDNS deployment (2 pods), with the rolling update strategy. Therefore, any change in the configuration is applied to both the pods one by one. This process can take some time (approximately, 30 to 60 seconds to reload both pods).

A NodeLocalDNS daemonset is a cache implementation of core DNS. The NodeLocalDNS runs as a pod on each node and is used for quick DNS resolution. When a pod requires a certain domain name resolution, it first checks its NodeLocalDNS pod, the one running in the same node, for resolution. If the pod doesn't get the required resolution, then it forwards the request to the core DNS.

All local DNS records are stored inside the core DNS configmap grouped by zones.

> **Note**
>
> Use the active Bastion to run all the troubleshooting procedures in this section.

7.3.6.5.1 Troubleshooting Local DNS API

This section provides the troubleshooting guidelines for the common scenarios that you may encounter while using Local DNS API.

Validating Local DNS API

Local DNS API, that is used to add or remove DNS records, runs as a service ("bastion_http_server") in all Bastion servers. To validate if the API is up and running, run the following command from a Bastion server:

```
$ systemctl status bastion_http_server
```

Sample output:

```
● bastion_http_server.service - Bastion http server
Loaded: loaded (/etc/systemd/system/bastion_http_server.service; enabled; vendor preset:
disabled)
Active: active (running) since Wed 2023-04-12 00:12:51 UTC; 1 day 19h ago
Main PID: 283470 (gunicorn)
Tasks: 4 (limit: 23553)
Memory: 102.6M
CGroup: /system.slice/bastion_http_server.service
├──283470 /usr/bin/python3.6 /usr/local/bin/gunicorn --workers=3 --bind 0.0.0.0:8000 --
chdir /bin/bastion_http_setup wsgi:app --max-requests 0 --timeout 5 --keep>
├──283474 /usr/bin/python3.6 /usr/local/bin/gunicorn --workers=3 --bind 0.0.0.0:8000 --
chdir /bin/bastion_http_setup wsgi:app --max-requests 0 --timeout 5 --keep>
├──283476 /usr/bin/python3.6 /usr/local/bin/gunicorn --workers=3 --bind 0.0.0.0:8000 --
chdir /bin/bastion_http_setup wsgi:app --max-requests 0 --timeout 5 --keep>
└──641094 /usr/bin/python3.6 /usr/local/bin/gunicorn --workers=3 --bind 0.0.0.0:8000 --
chdir /bin/bastion_http_setup wsgi:app --max-requests 0 --timeout 5 --keep>
```

The sample output shows the status of the `Bastion http server` service as *active (running)* and *enabled*. All Bastion servers have their own independent version of this service. Therefore, it is recommended to check the status of all Bastion servers.

Starting or Restarting Local DNS API

If Local DNS API is not running, run the following command to start or restart it:

Command to start Local DNS API:

```
$ sudo systemctl start bastion_http_server
```

Command to restart Local DNS API:

```
$ sudo systemctl restart bastion_http_server
```

The start and restart commands don't display any output on completion. To check the status of Local DNS API, perform the Validating Local DNS API procedure.

If bastion_http_server doesn't run even after starting or restarting it, refer to the following section to check its log.

Generating and Checking Local DNS Logs

This section provides details about generating and checking Local DNS logs.

Generating Local DNS Logs

You can use `journalctl` to get the logs of Local DNS API that runs as a service (`bastion_http_server`) on each bastion server.

Run the following command to get the logs of Local DNS API:

```
$ journalctl -u bastion_http_server
```

Run the following command to print only the latest 20 logs of Local DNS API:

```
journalctl -u bastion_http_server --no-pager -n 20
```

> **Note**
>
> In the interactive mode, you can use the keyboard shortcuts to scroll through the logs. The system displays the latest logs at the end.

Sample output:

```
-- Logs begin at Tue 2023-04-11 22:36:02 UTC. --
Apr 12 16:33:27 test-bastion-1.novalocal gunicorn[283474]: 2023-04-12 16:33:27,357
BHHTTP:INFO: Request payload: Record name occne.lab.oracle.com record ip 175.80.200.20
[/bin/bastion_http_setup/bastionApp.py:125]
Apr 12 16:33:27 test-bastion-1.novalocal gunicorn[283474]: 2023-04-12 16:33:27,357
BHHTTP:INFO: Domain name oracle.com db name db.oracle.com for record entry [/bin/
bastion_http_setup/coreDnsData.py:362]
Apr 12 16:33:27 test-bastion-1.novalocal gunicorn[283474]: 2023-04-12 16:33:27,369
BHHTTP:INFO: SUCCESS:  Validate coredns common config msg data oracle.com [/bin/
bastion_http_setup/commons.py:36]
Apr 12 16:33:27 test-bastion-1.novalocal gunicorn[283474]: 2023-04-12 16:33:27,380
BHHTTP:INFO: SUCCESS:  A Record deleted msg data occne.lab.oracle.com [/bin/
bastion_http_setup/commons.py:36]
Apr 12 16:33:27 test-bastion-1.novalocal gunicorn[283474]: 2023-04-12 16:33:27,380
BHHTTP:INFO: SUCCESS:  A Record deleted msg data occne.lab.oracle.com [/bin/
bastion_http_setup/commons.py:36]
Apr 12 16:33:27 test-bastion-1.novalocal gunicorn[283474]: 2023-04-12 16:33:27,380
BHHTTP:INFO: Domain name oracle.com db name db.oracle.com for record entry [/bin/
bastion_http_setup/coreDnsData.py:362]
Apr 12 16:33:27 test-bastion-1.novalocal gunicorn[283474]: 2023-04-12 16:33:27,388
BHHTTP:INFO: SUCCESS:  Validate coredns common config msg data oracle.com [/bin/
bastion_http_setup/commons.py:36]
Apr 12 16:33:27 test-bastion-1.novalocal gunicorn[283474]: 2023-04-12 16:33:27,388
BHHTTP:INFO: DNS A record deleted in coredns file for occne.lab.oracle.com
175.80.200.20,  msg SUCCESS:  SUCCESS:  A Record deleted [/bin/bastion_http_setup/
commons.py:47]
Apr 12 16:34:13 test-bastion-1.novalocal gunicorn[283474]: 2023-04-12 16:34:13,487
BHHTTP:INFO: Deployment reloaded,  msg SUCCESS:  Reloaded coredns deployment in ns kube-
system [/bin/bastion_http_setup/commons.py:47]
```

Checking Local DNS Logs

Local DNS logs contain information, errors, and debug messages about all the activities in Local DNS. The following table lists some of the sample messages and their description:

Table 7-12 Local DNS Log Messages

Message	Type/ Level	Description
Deployment reloaded, msg SUCCESS: Reloaded coredns deployment in ns kube-system	INFO	Success message indicating that the core DNS deployment reloaded successfully.
Validate coredns common config msg data oracle.com	INFO	Indicates that the module was able to process core DNS configuration data for a specific domain name.
Request payload incomplete. Request requires name and ip-address, error missing param 'ip-address'	ERROR	Indicates an invalid payload. The API sends this type of messages when the payload used for a given record is not valid or not complete.
FAILED: A record occne.lab.oracle.com does not exists in Zone db.oracle.com	ERROR	This message is used by an API module to trigger a creation of a new zone. This error message does not require any intervention.
Already exists: DNS A record in coredns file for occne.lab.oracle.com 175.80.200.20 3600, msg SUCCESS: A record occne.lab.oracle.com already exists in Zone db.oracle.com, msg: Record occne.lab.oracle.com cannot be duplicated.	ERROR	Same domain name error. Records in the same zone cannot be duplicated, have the same name, or share the same IP address. This message is displayed if either of these conditions is true.
DNS A record deleted in coredns file for occne.lab.oracle.com 175.80.200.20, msg SUCCESS: A Record deleted	INFO	Success message indicating that an A record was deleted successfully.
DNS A record added in coredns file for occne.lab.oracle.com 175.80.200.20 3600, msg SUCCESS: Zone info and A record updated for domain name	INFO	Success message indicating that the API has successfully added a new A record and updated the zone information.
ERROR in app: Exception on /occne/dns/a [POST] ... Traceback Omitted	ERROR	Fatal error indicating that an exception has occurred while processing a request. You can get more information by performing a traceback. This type of error is not common and must be reported as a bug.
Zone already present with domain name oracle.com	DEBUG	This type of debug messages are not enabled by default. They are usually used to print a high amount of information while troubleshooting.
FAILED: Unable to add SRV record: _sip._tcp.lab.oracle.com. 3600 IN SRV 10 100 35061 occne.lab.oracle.com. - record already exists - data: ... Data Omitted	ERROR	Error message indicating that the record already exists and cannot be duplicated.

7.3.6.5.2 Troubleshooting Core DNS

This section provides information about troubleshooting Core DNS using the core DNS logs.

Local DNS records are added to CoreDNS configuration. Therefore, the logs are generated and reported by the core DNS pods. As per the default configuration, CoreDNS reports information logs only on start up (for example, after a reload) and on running into an error.

- Run the following command to print all logs from both core DNS pods to the terminal, separated by name:

  ```
  $ for pod in $(kubectl -n kube-system get pods | grep coredns | awk
  '{print $1}'); do echo "----- $pod -----"; kubectl -n kube-system
  logs $pod; done
  ```

  Sample output:

  ```
  ----- coredns-8ddb9dc5d-5nvrv -----
  [INFO] plugin/ready: Still waiting on: "kubernetes"
  [INFO] plugin/auto: Inserting zone `occne.lab.oracle.com.' from: /etc/
  coredns/..2023_04_12_16_34_13.510777403/db.occne.lab.oracle.com
  .:53
  [INFO] plugin/reload: Running configuration SHA512 =
  2bc9e13e66182e6e829fe1a954359de92746468f433b8748589dfe16e1afd0e790e1ff75415ad40ad1771
  1abfc7a8348fdda2770af99962db01247526afbe24a
  CoreDNS-1.9.3
  linux/amd64, go1.18.2, 45b0a11
  ----- coredns-8ddb9dc5d-6lf5s -----
  [INFO] plugin/auto: Inserting zone `occne.lab.oracle.com.' from: /etc/
  coredns/..2023_04_12_16_34_15.930764941/db.occne.lab.oracle.com
  .:53
  [INFO] plugin/reload: Running configuration SHA512 =
  2bc9e13e66182e6e829fe1a954359de92746468f433b8748589dfe16e1afd0e790e1ff75415ad40ad1771
  1abfc7a8348fdda2770af99962db01247526afbe24a
  CoreDNS-1.9.3
  linux/amd64, go1.18.2, 45b0a11
  ```

- Additionally, you can pipe the above command to a file for better readability and sharing:

  ```
  $ for pod in $(kubectl -n kube-system get pods | grep coredns | awk
  '{print $1}'); do echo "----- $pod -----"; kubectl -n kube-system
  logs $pod; done > coredns.logs
  $ vi coredns.logs
  ```

- Run the following command to get the latest logs from any of the CoreDNS pods:

  ```
  $ kubectl -n kube-system --tail 20 logs $(kubectl -n kube-system get pods
  | grep coredns | awk  '{print $1 }' | head -n 1)
  ```

  This command prints the latest 20 log entries. You can modify the `--tail` value as per your requirement.

  Sample output:

  ```
  [INFO] plugin/auto: Inserting zone `occne.lab.oracle.com.' from: /etc/
  coredns/..2023_04_13_19_29_29.1646737834/db.occne.lab.oracle.com
  .:53
  [INFO] plugin/reload: Running configuration SHA512 =
  ```

```
2bc9e13e66182e6e829fe1a954359de92746468f433b8748589dfe16e1afd0e790e1ff75415ad40ad1771
1abfc7a8348fdda2770af99962db01247526afbe24a
CoreDNS-1.9.3
linux/amd64, go1.18.2, 45b0a11
```

7.3.6.5.3 Troubleshooting DNS Records

This section provides information about validating, and querying internal and external records.

> **Note**
>
> Use the internal cluster network to resolve the records added to core DNS through local DNS API. The system does not respond if you query for a DNS record from outside the cluster (for example, querying from a Bastion server).

Validating Records

You can use any pod to access and query a DNS record in core DNS. However, most of the pods do not have the network utilities to directly query a record. In such cases, you can include the network utilities, such as bind-utils, bundled with the pods to allow them to access and query records.

You can also use the MetalLB controller pod, which is already bundled with bind-utils, to query the DNS records:

- Run the following command from a Bastion server to query an A record:

  ```
  $ kubectl -n occne-infra exec -i -t $(kubectl -n occne-infra get pod |
  grep metallb-cont | awk '{print $1}') -- nslookup occne.lab.oracle.com
  ```

  Sample output:

  ```
  .oracle.com
  Server: 169.254.25.10
  Address: 169.254.25.10:53


  Name: occne.lab.oracle.com
  Address: 175.80.200.20
  ```

- Run the following command from a Bastion server to query an SRV record:

  ```
  $ kubectl -n occne-infra exec -i -t $(kubectl -n occne-infra get pod |
  grep metallb-cont | awk '{print $1}') -- nslookup -type=srv
  _sip._tcp.lab.oracle.com
  ```

  Sample output:

  ```
  Server:         169.254.25.10
  Address:        169.254.25.10:53

  _sip._tcp.lab.oracle.com        service = 10 100 35061 occne.lab.oracle.com
  ```

> **Note**
>
> Reload the core DNS configuration after adding multiple records to ensure that your changes are applied.

The following code block provides the command to query a DNS record using a pod that is created on a different namespace. By default, this pod is not bundled with the CNE cluster:

> **Note**
>
> This example considers that an A record is already loaded to occne1.us.oracle.com using the API.

```
$ kubectl -n occne-demo exec -it test-app -- nslookup occne1.us.oracle.com
```

Sample output:

```
.oracle.com
Server: 169.254.25.10
Address: 169.254.25.10:53


Name: occne1.us.oracle.com
Address: 10.65.200.182
```

Querying Non Existing or External Records

You cannot access or query an external record or a record that is not added using the API. The system terminates such queries with an error code.

For example:

- the following codeblock shows a case where a non existing A record is queried:

  ```
  $ kubectl -n occne-infra exec -i -t $(kubectl -n occne-infra get pod |
  grep metallb-cont | awk '{print $1}') -- nslookup not-in.oracle.com
  ```

  Sample output:

  ```
  Server:         169.254.25.10
  Address:        169.254.25.10:53

  ** server can't find not-in.oracle.com: NXDOMAIN

  ** server can't find not-in.oracle.com: NXDOMAIN

  command terminated with exit code 1
  ```

- the following codeblock shows a case where a non existing SRV record is queried:

  ```
  $ kubectl -n occne-infra exec -i -t $(kubectl -n occne-infra get pod |
  grep metallb-cont | awk '{print $1}') -- nslookup not-in.oracle.com
  ```

Sample output:

```
Server:         169.254.25.10
Address:        169.254.25.10:53

** server can't find not-in.oracle.com: NXDOMAIN

** server can't find not-in.oracle.com: NXDOMAIN

command terminated with exit code 1
```

Querying Internal Services

Core DNS is configured to resolve internal services by default. Therefore, you can query any internal Kubernetes services as usual.

For example,

- the following codeblock shows a case where an A record is queried from an internal Kubernetes service:

  ```
  $ kubectl -n occne-infra exec -i -t $(kubectl -n occne-infra get pod |
  grep metallb-cont | awk '{print $1}') -- nslookup kubernetes
  ```

  Sample output:

  ```
  Server:         169.254.25.10
  Address:        169.254.25.10:53


  Name:   kubernetes.default.svc.test
  Address: 10.233.0.1

  ** server can't find kubernetes.svc.test: NXDOMAIN
  ** server can't find kubernetes.svc.test: NXDOMAIN
  ** server can't find kubernetes.test: NXDOMAIN
  ** server can't find kubernetes.test: NXDOMAIN
  ** server can't find kubernetes.occne-infra.svc.test: NXDOMAIN
  ** server can't find kubernetes.occne-infra.svc.test: NXDOMAIN
  ```

  The sample output displays the response from default.svc.test as "Kubernetes", as a service, exists only in the default namespace.

- the following codeblock shows a case where an SRV record is queried from an internal Kubernetes service:

  ```
  $ kubectl -n occne-infra exec -i -t $(kubectl -n occne-infra get pod |
  grep metallb-cont | awk '{print $1}') -- nslookup -type=srv
  kubernetes.default.svc.test
  ```

  Sample output:

  ```
  Server:         169.254.25.10
  Address:        169.254.25.10:53

  kubernetes.default.svc.occne3-toby-edwards      service = 0 100 443
  kubernetes.default.svc.test

  ** server can't find kubernetes.svc.test: NXDOMAIN
  ** server can't find kubernetes.occne-infra.svc.test: NXDOMAIN
  ** server can't find kubernetes.test: NXDOMAIN
  ```

The sample output displays the response from default.svc.test as "Kubernetes", as a service, exists only in the default namespace.

7.3.6.5.4 Accessing Configuration Files

This section provides information about accessing configuration files for troubleshooting.

Local DNS does not have any configuration file of its own; it relies entirely on CoreDNS config files. CoreDNS and nodelocaldns (CoreDNS cache daemon) have their configuration files as configmaps on the kube-system namespace.

> **Note**
>
> Local DNS API takes care of configurations and modifications by default. Therefore, it is not recommended to access or update the configmaps as manual intervention to these files can potentially break the entire CoreDNS functionality.

If there is absolute necessity to access configmap for troubleshooting, then use the data endpoint to access records of all zones along with the CoreDNS configuration.

Sample configuration:

```
# The following line, starting with "db.DOMAIN-NAME" represents a Zone file
'db.oracle.com': ';oracle.com db file\n'
                 'oracle.com.                  300                  ' # All zone files
contain a default SOA entry auto generated
                 'IN      SOA     ns1.oracle.com  andrei.oracle.com '
                 '201307231 3600 10800 86400 3600\n'
                 'occne.lab.oracle.com.                   ' # User added A record
                 '3600                  IN      A       175.80.200.20\n'
                 '_sip._tcp.lab.oracle.com 30 IN SRV 10 102 32061 ' # User added SRV record
                 'occne.lab.oracle.com.\n'
                 'occne1.us.oracle.com.                   ' # User added A record
                 '3600                  IN      A       '
                 '10.65.200.182\n'},
```

7.3.6.5.5 Troubleshooting Validation Script Errors

The local DNS feature provides the `validateLocalDns.py` script to validate if the Local DNS feature is activated successfully. This section provides information about troubleshooting some of the common issues that occur while using the `validateLocalDns.py` script.

Local DNS variable is not set properly

You can encounter the following or a similar error message while running the validation script if the local DNS variable is not set properly:

```
Beginning local DNS validation
 - Getting the occne-metallb-controller pod's name.
 - Validating occne.ini.
Unable to continue - err: Cannot continue - local_dns_enabled variable is set to False,
which is not valid to continue..
```

In such cases, ensure that:

- the `local_dns_enabled` variable is set to *True*: `local_dns_enabled=True`
- there are no black spaces before and after the "=" sign

- the variable is typed correctly as it is case sensitive

> **Note**
>
> To successfully enable Local DNS, you must follow the entire activation procedure. Otherwise, the system doesn't enable the feature successfully even after you set the `ocal_dns_enabled` variable to the correct value.

Unable to access the test pod

The validation script uses the `occne-metallb-controller` pod to validate the test record. This is because the DNS records can be accessed from inside the cluster only, and the MetalLB pod contains the necessary utility tools to access the records by default. You can encounter the following error while running the validation script if the MetalLB pod is not accessible:

```
Beginning local DNS validation
 - Getting the occne-metallb-controller pod's name.
  - Error while trying to get occne-metallb-controller pod's name, error: ...
```

In such cases, ensure that the `occne-metallb-controller` is accessible.

Unable to add a test record

While performing a validation, the validation script creates and removes a test record using the API to check if the Local DNS feature is working properly. You can encounter the following error when the script is unable to create a test record:

```
Beginning local DNS validation
 - Getting the occne-metallb-controller pod's name.
 - Validating occne.ini.
 - Adding DNS A record.
Unable to continue - err: Failed to add DNS entry.
```

The following table lists some of the common issues why the script can fail to add a test record, along with the resolutions:

Table 7-13 Validation Script Errors and Resolutions

Issue	Error Message	Resolution
The script is previously run and interrupted before it finished. The script possibly created a test record the previous time it was run unsuccessfully. When the script is run again, it tries to create a duplicate test record and fails.	Cannot add a duplicate record. Test record: *name:occne.dns.local.com, ip-address: 10.0.0.3*	Delete the existing test record from the system and rerun the validation script.
A record similar to the test record is added manually.	Cannot add a duplicate record. Test record: *name:occne.dns.local.com, ip-address: 10.0.0.3*	Delete the existing test record from the system and rerun the validation script.
Local DNS API is not available.	The Local DNS API is not running or is in an error state	Validate if the Local DNS feature is enabled properly. For more information, see Troubleshooting Local DNS API.

Table 7-13 (Cont.) Validation Script Errors and Resolutions

Issue	Error Message	Resolution
Local DNS API returns 50X status code.	Kubernetes Admin Configmap missing or misconfigured	Check if Kubernetes admin.conf is properly set to allow the API to interact with Kubernetes.

> **Note**
>
> The name and ip-address of the test record are managed by the script. Use these details for validation purpose only.

Unable to reload configuration

You can encounter the following error if the validation script fails to reload the core DNS configuration:

```
Beginning local DNS validation
 - Getting the occne-metallb-controller pod's name.
 - Validating occne.ini.
 - Adding DNS A record.
 - Adding DNS SRV record.
 - Reloading local coredns.
  - Error while trying to reload the local coredns, error: .... # Reason Omitted
```

In such cases, analyze the cause of the issue using the Local DNS logs. For more information, see Troubleshooting Local DNS API.

Other miscellaneous errors

If you are encountering other miscellaneous errors (such as, "unable to remove record"), follow the steps in the Troubleshooting Local DNS API section to generate logs and analyze the issue.

7.4 Managing the Kubernetes Cluster

This section provides instructions on how to manage the Kubernetes Cluster.

7.4.1 Creating CNE Cluster Backup

This section describes the procedure to create a backup of CNE cluster data using the `createClusterBackup.py` script.

Critical CNE data can be damaged or lost during a fault recovery scenario. Therefore, it is advised to take a backup of your CNE cluster data regularly. These backups can be used to restore your CNE cluster when the cluster data is lost or damaged.

Backing up a CNE cluster data involves the following steps:

1. Backing up Bastion Host data
2. Backing up Kubernetes data using Velero

The `createClusterBackup.py` script is used to backup both the bastion host data and Kubernetes data.

Prerequisites

Before creating CNE cluster backup, ensure that the following prerequisites are met:

- Velero must be activated successfully. For Velero installation procedure, see *Oracle Communications Cloud Native Core, Cloud Native Environment Installation, Upgrade, and Fault Recovery Guide*.
- Velero v1.10.0 server must be installed and running.
- Velero CLI for v1.10.0 must be installed and running.
- boto3 python module must be installed. For more information, see the "Configuring PIP Repository" section in the *Oracle Communications Cloud Native Core, Cloud Native Environment Installation, Upgrade, and Fault Recovery Guide*.
- The S3 Compatible Object Storage Provider must be configured and ready to be used.
- The following S3 related credentials must be available:
  - Endpoint Url
  - Access Key Id
  - Secret Access Key
  - Region Name
  - Bucket Name
- An external S3 compatible data store to store backup data must have been configured while installing CNE.
- Cluster must be in a good state, that is all content included in the following namespaces must be up and running:
  - occne-infra
  - cert-manager
  - kube-system
  - rook-ceph (for bare metal)
  - istio-system
- All bastion-controller and lb-controller PVCs must be in "Bound" status.

> **Note**
>
> - This procedure creates only a CNE cluster backup that contains bastion host data, including Kubernetes.
> - For Kubernetes, this procedure creates the backup content included in the following namespaces only:
>   - occne-infra
>   - cert-manager
>   - kube-system
>   - rook-ceph (for bare metal)
>   - istio-system
> - You must take the bastion backup in the ACTIVE bastion only.

7.4.1.1 Creating a Backup of Bastion Host and Kubernetes Data

This section describes the procedure to back up the Bastion Host and Kubernetes data using the `createClusterBackup.py` script.

Procedure

1. Run the following command to verify if you are currently on an active Bastion. If you are not, log in to an active Bastion and continue this procedure.

   ```
   $ is_active_bastion
   ```

   Sample output:

   ```
   IS active-bastion
   ```

2. Use the following commands to run the `createClusterBackup.py` script:

   ```
   $ cd /var/occne/cluster/${OCCNE_CLUSTER}/artifacts
   $ ./backup/createClusterBackup.py
   ```

   Sample output:

   ```
   Initializing cluster backup occne-cluster-20230717-183615

   No /var/occne/cluster/occne-cluster/artifacts/backup/cluster_backups_log.json log
   file, creating new one
   Creating bastion backup: 'occne-cluster-20230717-183615'

   Successfully created bastion backup

   GENERATED LOG FILE AT: /var/occne/cluster/occne-cluster/
   createBastionBackup-20230717-183615.log

   Creating velero backup: 'occne-cluster-20230717-183615'

   Successfully created velero backup

   Successfully created cluster backup
   ```

```
GENERATED LOG FILE AT: /var/occne/cluster/occne-cluster/
createClusterBackup.py-20230717-183615.log
```

3. If the `createClusterBackup.py` script fails due to a missing boto3 library, then perform the following steps to add your proxy and download boto3. Else, move to Step 3.
   a. Run the following commands to install boto3 library:

      ```
      export http_proxy=YOUR_PROXY
      export https_proxy=$http_proxy
      export HTTP_PROXY=$http_proxy
      export HTTPS_PROXY=$http_proxy


      pip3 install boto3
      ```

      While installing boto3 library, you may see a warning regarding the versions of dependencies. You can ignore the warning as the boto3 library can work without these dependencies.
   b. Once you install boto3 library, run the following commands to unset the proxy:

      ```
      unset HTTP_PROXY
      unset https_proxy
      unset http_proxy
      unset HTTPS_PROXY
      ```

4. Navigate to the `/home/cloud-user` directory and verify if the backup tar file is generated.
5. Log in to your S3 cloud storage and verify if the Bastion Host data is uploaded successfully.

7.4.1.2 Verifying Backup in S3 Bucket

This section describes the procedure to verify the CNE cluster data backup in S3 bucket.

S3 bucket contains two folders:

- bastion-data-backups: for storing Bastion backup
- velero-backup: for storing Velero backup

1. Verify if the Bastion Host data is stored as a `.tar` file in the `{BUCKET_NAME}/bastion-data-backups/{CLUSTER-NAME}/{BACKUP_NAME}` folder. Where, `{CLUSTER-NAME}` is the name of the cluster and`{BACKUP_NAME}` is the name of the backup.
2. Verify if the Velero Kubernetes backup is stored in the `{BUCKET_NAME}/velero-backup/{BACKUP_NAME}/` folder. Where, `{BACKUP_NAME}` is the name of the backup.

> **⚠ Caution**
>
> The `velero-backup` folder must not be modified manually as this folder is managed by Velero. Modifying the folder can corrupt the structure or files.

For information about restoring CNE cluster from a backup, see "Restoring CNE from Backup" in *Oracle Communications Cloud Native Core, Cloud Native Environment Installation, Upgrade, and Fault Recovery Guide*.

7.4.2 Renewing Kubernetes Certificates

Some of the Kubernetes certificates in your cluster are valid for a period of one year. These certificates include various important files that secure the communication within your cluster, such as the API server certificate, the etcd certificate, and the controller manager certificate. To maintain the security and operation of your CNE Kubernetes cluster, it is important to keep these certificates updated. The certificates are renewed automatically during the CNE upgrade. If you have not performed an CNE upgrade in the last year, you must run this procedure to renew your certificates for the continued operation of the CNE Kubernetes cluster.

Introduction

Kubernetes uses many different TLS certificates to secure access to internal services. There certificates are automatically renewed during upgrade. However, if upgrade is not performed regularly, these certificates may expire and cause the Kubernetes cluster to fail. To avoid this situation follow the procedure below to renew all certificates used by Kubernetes. This procedure can also be used to renew expired certificates and restore access to the Kubernetes cluster.

List of K8s internal certificates

The following table lists the Kubernetes (K8s) internal certificates and their validity.

Table 7-14 Kubernetes Internal Certificates and Validity Period

Node Type	Componet Name	.crt File Path	Validity (in years)	.pem File Path	Validity (in years)
Kubernetes Controller	etcd	/etc/pki/ca-trust/source/anchors/etcd-ca.crt	100	/etc/ssl/etcd/ssl/admin-<node_name>.pem	100
Kubernetes Controller	etcd	NA	NA	/etc/ssl/etcd/ssl/ca.pem	100
Kubernetes Controller	etcd	NA	NA	/etc/ssl/etcd/ssl/member-<node_name>.pem	100
Kubernetes Controller	etcd	NA	NA	/etc/ssl/etcd/ssl/member-<node_name>.pem	100
Kubernetes Controller	Kubernetes	/etc/kubernetes/ssl/ca.crt	10	NA	NA
Kubernetes Controller	Kubernetes	/etc/kubernetes/ssl/apiserver.crt	1	NA	NA
Kubernetes Controller	Kubernetes	/etc/kubernetes/ssl/apiserver-kubelet-client.crt	1	NA	NA
Kubernetes Controller	Kubernetes	/etc/kubernetes/ssl/front-proxy-ca.crt	10	NA	NA

Table 7-14 (Cont.) Kubernetes Internal Certificates and Validity Period

Node Type	Componet Name	.crt File Path	Validity (in years)	.pem File Path	Validity (in years)
Kubernetes Controller	Kubernetes	/etc/kubernetes/ssl/front-proxy-client.crt	1	NA	NA
Kubernetes Node	Kubernetes	/etc/kubernetes/ssl/ca.crt	10	NA	NA

You can use the above table to keep a record of the Kubernetes certificates and their validity details. Fill the Validity column in the table with your certificates' validity.

> ⚠ **Caution**
>
> Run this procedure on each controller node and verify that the certificates are renewed successfully to avoid cluster failures. The controller nodes are the orchestrator and maintainers of the metadata of all objects and components of the cluster. If you do not run this procedure on all the controller nodes and the certificates expire, the integrity of the cluster and the applications that are deployed on the cluster are staged at risk. This causes the communication within the internal components to be lost resulting in a total cluster failure. In such a case, you must recover each controller node or in the worst case scenario, recover the complete cluster.

Prerequisites

Checking Certificate Expiry

Log in to any Kubernetes controller node and run the following commands to verify the expiration date for the Kubernetes certificates:

```
$ sudo su
# export PATH=$PATH:/usr/local/bin
# kubeadm certs check-expiration
```

Sample output:

```
[check-expiration] Reading configuration from the cluster...
[check-expiration] FYI: You can look at this config file with 'kubectl -n kube-system
get cm kubeadm-config -o yaml'
W0214 13:39:25.870724   84036 utils.go:69] The recommended value for "clusterDNS" in
"KubeletConfiguration" is: [10.233.0.10]; the provided value is: [169.254.25.10]

CERTIFICATE                EXPIRES                  RESIDUAL TIME   CERTIFICATE
AUTHORITY   EXTERNALLY MANAGED
admin.conf                 Feb 14, 2026 17:42 UTC   364d
ca                      no
apiserver                  Feb 14, 2026 17:42 UTC   364d
ca                      no
apiserver-kubelet-client   Feb 14, 2026 17:42 UTC   364d
ca                      no
controller-manager.conf    Feb 14, 2026 17:42 UTC   364d
ca                      no
front-proxy-client         Feb 14, 2026 17:42 UTC   364d            front-proxy-
```

```
ca          no
scheduler.conf             Feb 14, 2026 17:42 UTC   364d
ca                      no
super-admin.conf           Feb 14, 2026 17:42 UTC   364d
ca                      no

CERTIFICATE AUTHORITY   EXPIRES                  RESIDUAL TIME   EXTERNALLY MANAGED
ca                      Feb 12, 2035 17:42 UTC   9y              no
front-proxy-ca          Feb 12, 2035 17:42 UTC   9y              no
```

Procedure

1. Use SSH to log in to the active Bastion Host.
2. Run the following command to verify if the Bastion Host is the active Bastion Host:

   ```
   $ is_active_bastion
   ```

   The system displays the following output if the Bastion Host is the active Bastion Host:

   ```
   IS active-bastion
   ```

   If the Bastion Host is not the active Bastion Host, try a different Bastion Host.

   > **Note**
   >
   > If the certificates are expired, the `is_active_bastion` command doesn't work as it depends on `kubectl`. In this case, skip this step and move to the next step.

3. Perform the following steps to log in to a controller node as a root user and back up the SSL directory:
   a. Use SSH to log in to Kubernetes controller node as a root user:

      ```
      $ ssh <k8s-ctrl-node>
      $ sudo su
      # export PATH=$PATH:/usr/local/bin
      ```

   b. Take a backup of the `ssl` directory:

      ```
      # cp -r /etc/kubernetes/ssl /etc/kubernetes/ssl_backup
      ```

4. Renew all `kubeadm` certificates:

   ```
   # kubeadm certs renew all
   ```

   Sample output:

   ```
   [renew] Reading configuration from the cluster...
   [renew] FYI: You can look at this config file with 'kubectl -n kube-system get cm
   kubeadm-config -o yaml'
   W0212 18:04:43.840444 3620859 utils.go:69] The recommended value for "clusterDNS" in
   "KubeletConfiguration" is: [10.233.0.10]; the provided value is: [169.254.25.10]

   certificate embedded in the kubeconfig file for the admin to use and for kubeadm
   itself renewed
   ```

```
certificate for serving the Kubernetes API renewed
certificate for the API server to connect to kubelet renewed
certificate embedded in the kubeconfig file for the controller manager to use renewed
certificate for the front proxy client renewed
certificate embedded in the kubeconfig file for the scheduler manager to use renewed
certificate embedded in the kubeconfig file for the super-admin renewed

Done renewing certificates. You must restart the kube-apiserver, kube-controller-
manager, kube-scheduler and etcd, so that they can use the new certificates.
```

5. Perform the following steps to remove the manifest files in the `/etc/kubernetes/manifests/` directory and restart the static pods:

> **Note**
>
> This step requires removing (moving the file to `tmp` folder) the manifest files in the `/etc/kubernetes/manifests/` directory and copying back the file to the same directory to restart the `kube-apiserver` pod. Each time you remove and copy the manifest files, the system waits for a period configured in `fileCheckFrequency`. `fileCheckFrequency` is a Kubelet configuration and the default value is *20* seconds.

   a. Perform the following steps to restart the API server pod:

      i. Remove the `kube-apiserver` pod:

      ```
      # mv /etc/kubernetes/manifests/kube-apiserver.yaml /tmp
      ```

      ii. Run the watch command until the `kube-apiserver` pod is removed. When the pod is removed, use `Ctrl+C` to exit the watch command:

      ```
      # watch -n 1 "sudo /usr/local/bin/crictl -r unix:///run/containerd/
      containerd.sock ps | grep -e api -e kube-controller-manager -e
      scheduler"
      ```

      Sample output:

      ```
      Every 1.0s: sudo /usr/local/bin/crictl -r unix:///run/containerd/
      contain...  occne-example-k8s-ctrl-1: Fri Feb 14 13:52:26 2025

      ff79b19fdffd7       9aa1fad941575       27 seconds ago
      Running             kube-scheduler                    2
        ab0da7c51b413       kube-scheduler-occne-example-k8s-ctrl-1
      64059f7efadc5       175ffd71cce3d       27 seconds ago
      Running             kube-controller-manager           3
        9591cd755dae4       kube-controller-manager-occne-example-k8s-
      ctrl-1
      ```

      iii. Restore the `kube-apiserver` pod to the `/etc/kubernetes/manifests/` directory:

      ```
      # mv /tmp/kube-apiserver.yaml /etc/kubernetes/manifests
      ```

iv. Run the watch command until the `kube-apiserver` pod appears in the output. When the pod appears, use `Ctrl+C` to exit the watch command:

```
# watch -n 1 "sudo /usr/local/bin/crictl -r unix:///run/containerd/
containerd.sock ps | grep -e api -e kube-controller-manager -e
scheduler"
```

Sample output:

```
Every 1.0s: sudo /usr/local/bin/crictl -r unix:///run/containerd/
contain...  occne-example-k8s-ctrl-1: Fri Feb 14 13:53:28 2025

67c8d5c42645f       6bab7719df100       10 seconds ago
Running             kube-apiserver                     0
   3bb9f31dad8c6       kube-apiserver-occne-example-k8s-ctrl-1
ff79b19fdffd7       9aa1fad941575       About a minute ago
Running             kube-scheduler                     2
   ab0da7c51b413       kube-scheduler-occne-example-k8s-ctrl-1
64059f7efadc5       175ffd71cce3d       About a minute ago
Running             kube-controller-manager            3
   9591cd755dae4       kube-controller-manager-occne-example-k8s-
ctrl-1
```

b. Perform the following steps to restart the controller manager pod:

i. Remove the `kube-controller-manager` pod:

```
# mv /etc/kubernetes/manifests/kube-controller-manager.yaml /tmp
```

ii. Run the watch command until the `kube-controller-manager` pod is removed. When the pod is removed, use `Ctrl+C` to exit the watch command:

```
# watch -n 1 "sudo /usr/local/bin/crictl -r unix:///run/containerd/
containerd.sock ps | grep -e api -e kube-controller-manager -e
scheduler"
```

Sample output:

```
Every 1.0s: sudo /usr/local/bin/crictl -r unix:///run/containerd/
contain...  occne-example-k8s-ctrl-1: Fri Feb 14 13:55:48 2025

67c8d5c42645f       6bab7719df100       2 minutes ago
Running             kube-apiserver                     0
  3bb9f31dad8c6       kube-apiserver-occne-example-k8s-ctrl-1
ff79b19fdffd7       9aa1fad941575       3 minutes ago
Running             kube-scheduler                     2
  ab0da7c51b413       kube-scheduler-occne-example-k8s-ctrl-1
```

iii. Restore the `kube-controller-manager` pod to the `/etc/kubernetes/manifests/` directory:

```
# mv /tmp/kube-controller-manager.yaml /etc/kubernetes/manifests
```

iv. Run the watch command until the `kube-controller-manager` pod appears in the output. When the pod appears, use `Ctrl+C` to exit the watch command:

```
# watch -n 1 "sudo /usr/local/bin/crictl -r unix:///run/containerd/
containerd.sock ps | grep -e api -e kube-controller-manager -e
scheduler"
```

Sample output:

```
Every 1.0s: sudo /usr/local/bin/crictl -r unix:///run/containerd/
contain...  occne-example-k8s-ctrl-1: Fri Feb 14 13:57:11 2025

fa16530da2e04       175ffd71cce3d       15 seconds ago
Runningeconds ago   kube-controller-manager            0
  9b6c69c940bfa       kube-controller-manager-occne-example-k8s-
ctrl-1
67c8d5c42645f       6bab7719df100       3 minutes ago
Running             kube-apiserver                     0
  3bb9f31dad8c6       kube-apiserver-occne-example-k8s-ctrl-1
ff79b19fdffd7       9aa1fad941575       5 minutes ago
Running             kube-scheduler                     2
  ab0da7c51b413       kube-scheduler-occne-example-k8s-ctrl-1
```

c. Perform the following steps to restart the scheduler pod:

i. Remove the `kube-scheduler` pod:

```
# mv /etc/kubernetes/manifests/kube-scheduler.yaml /tmp
```

ii. Run the watch command until the `kube-scheduler` pod is removed. When the pod is removed, use `Ctrl+C` to exit the watch command:

```
# watch -n 1 "sudo /usr/local/bin/crictl -r unix:///run/containerd/
containerd.sock ps | grep -e api -e kube-scheduler -e scheduler"
```

Sample output:

```
Every 1.0s: sudo /usr/local/bin/crictl -r unix:///run/containerd/
contain...  occne-example-k8s-ctrl-1: Thu Feb 13 13:16:06 2025

fa16530da2e04       175ffd71cce3d       19 minutes ago
Running             kube-controller-manager            0
  9b6c69c940bfa       kube-controller-manager-occne-example-k8s-
ctrl-1
67c8d5c42645f       6bab7719df100       23 minutes ago
Running             kube-apiserver                     0
  3bb9f31dad8c6       kube-apiserver-occne-example-k8s-ctrl-1
```

iii. Restore the `kube-scheduler` pod to the `/etc/kubernetes/manifests/` directory:

```
# mv /tmp/kube-scheduler.yaml /etc/kubernetes/manifests
```

iv. Run the watch command until the `kube-scheduler` pod appears in the output. When the pod appears, use `Ctrl+C` to exit the watch command:

```
# watch -n 1 "sudo /usr/local/bin/crictl -r unix:///run/containerd/
containerd.sock ps | grep -e api -e kube-scheduler -e scheduler"
```

Sample output:

```
Every 1.0s: sudo /usr/local/bin/crictl -r unix:///run/containerd/
contain...  occne-example-k8s-ctrl-1: Fri Feb 14 14:16:35 2025

8c4500f3d61d7       9aa1fad941575       16 seconds ago
Running             kube-scheduler                     0
  7c175d8106f0c       kube-scheduler-occne-example-k8s-ctrl-1
fa16530da2e04       175ffd71cce3d       19 minutes ago
Running             kube-controller-manager            0
  9b6c69c940bfa       kube-controller-manager-occne-example-k8s-
ctrl-1
67c8d5c42645f       6bab7719df100       23 minutes ago
Running             kube-apiserver                     0
  3bb9f31dad8c6       kube-apiserver-occne-example-k8s-ctrl-1
```

6. Renew the `admin.conf` file and update the contents of `$HOME/.kube/config`. Type *yes* when prompted.

```
# cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
cp: overwrite '/root/.kube/config'? yes

# chown $(id -u):$(id -g) $HOME/.kube/config
```

7. Run the following command to validate if the certificates are renewed:

```
# kubeadm certs check-expiration
```

Sample output:

```
[check-expiration] Reading configuration from the cluster...
[check-expiration] FYI: You can look at this config file with 'kubectl -n
kube-system get cm kubeadm-config -o yaml'
W0214 14:21:49.907835  143445 utils.go:69] The recommended value for
"clusterDNS" in "KubeletConfiguration" is: [10.233.0.10]; the provided
value is: [169.254.25.10]

CERTIFICATE                EXPIRES                  RESIDUAL TIME
CERTIFICATE AUTHORITY   EXTERNALLY MANAGED
admin.conf                 Feb 14, 2026 18:51 UTC   364d
ca                      no
apiserver                  Feb 14, 2026 18:51 UTC   364d
ca                      no
apiserver-kubelet-client   Feb 14, 2026 18:51 UTC   364d
ca                      no
controller-manager.conf    Feb 14, 2026 18:51 UTC   364d
ca                      no
front-proxy-client         Feb 14, 2026 18:51 UTC   364d            front-
```

```
proxy-ca          no
scheduler.conf              Feb 14, 2026 18:51 UTC   364d
ca                        no
super-admin.conf            Feb 14, 2026 18:51 UTC   364d
ca                        no

CERTIFICATE AUTHORITY   EXPIRES                  RESIDUAL TIME
EXTERNALLY MANAGED
ca                      Feb 12, 2035 17:42 UTC   9y              no
front-proxy-ca          Feb 12, 2035 17:42 UTC   9y              no
```

8. Perform steps 3 through 7 on the remaining controller nodes.
9. Exit from the root user privilege:

   ```
   # exit
   ```

10. Copy the `/etc/kubernetes/admin.conf` file from the controller node to the artifacts directory of the active Bastion.

    > **Note**
    > - Replace `<OCCNE_ACTIVE_BASTION>` and `<OCCNE_CLUSTER>` with the values corresponding to your system. Refer to Step 2 for the value of `<OCCNE_ACTIVE_BASTION>` (For example, `occne-example-bastion-1`).
    > - Type *yes* and enter your password if prompted.

    ```
    $ sudo scp /etc/kubernetes/admin.conf ${USER}@<OCCNE_ACTIVE_BASTION>:/var/
    occne/cluster/<OCCNE_CLUSTER>/artifacts
    ```

11. Log in to the active Bastion Host and update the server address in the `admin.conf` file to https://lb-apiserver.kubernetes.local:6443:

    ```
    $ ssh <active-bastion>
    $ sed -i 's#https://127.0.0.1:6443#https://lb-
    apiserver.kubernetes.local:6443#' /var/occne/cluster/${OCCNE_CLUSTER}/
    artifacts/admin.conf
    ```

12. If you are using a Load Balancer VM (LBVM), perform the following steps to delete the existing `lb-controller-admin` secret and create a new one:
    a. Run the following command to delete the existing `lb-controller-admin` secret:

       ```
       $ kubectl -n occne-infra delete secret lb-controller-admin-config
       ```

    b. Run the following command to create a new `lb-controller-admin` secret from the updated `admin.conf` file:

       ```
       $ kubectl -n occne-infra create secret generic lb-controller-admin-
       config --from-file=/var/occne/cluster/${OCCNE_CLUSTER}/artifacts/
       admin.conf
       ```

13. If you are using a Load Balancer VM (LBVM), perform the following steps to patch the `lb-controller-admin-config` secret and restart the `lb-controller-server` pod:
    a. Patch the `lb-controller-admin-config` secret:

    ```
    $ echo -n "$(kubectl get secret lb-controller-admin-config -n occne-infra -o jsonpath='{.data.admin\.conf}' | base64 -d | sed 's#https://lb-apiserver.kubernetes.local:6443#https://kubernetes.default:443#g')" | base64 -w0 | xargs -I{} kubectl -n occne-infra patch secret lb-controller-admin-config --patch '{"data":{"admin.conf":"{}"}}'
    ```

    b. Remove the `lb-controller-server` pod:

    ```
    $ kubectl scale deployment/occne-lb-controller-server -n occne-infra --replicas=0
    ```

    c. Run the watch command until the `occne-lb-controller-server` pod is removed. When the pod is removed, use `Ctrl+C` to exit the watch command:

    ```
    $ watch -n 1 "kubectl -n occne-infra get pods | grep lb-controller"
    ```

    d. Restore the `lb-controller-server` pod:

    ```
    $ kubectl scale deployment/occne-lb-controller-server -n occne-infra --replicas=1
    ```

    e. Run the watch command until the `occne-lb-controller-server` pod appears in the output. When the pod appears, use `Ctrl+C` to exit the watch command:

    ```
    $ watch -n 1 "kubectl -n occne-infra get pods | grep lb-controller"
    ```

14. Renew the Kyverno certificates by deleting the secrets from the `kyverno` namespace:

    > **Note**
    >
    > You must perform this step to renew the Kyverno certificates manually as the current verion of Kyverno doesn't support automatic renewal of certificates.

    ```
    $ kubectl delete secret occne-kyverno-svc.kyverno.svc.kyverno-tls-ca -n kyverno
    ```

    Sample output:

    ```
    secret "occne-kyverno-svc.kyverno.svc.kyverno-tls-ca" deleted
    ```

    ```
    $ kubectl delete secret occne-kyverno-svc.kyverno.svc.kyverno-tls-pair -n kyverno
    ```

Sample output:

```
secret "occne-kyverno-svc.kyverno.svc.kyverno-tls-pair" deleted
```

15. Perform the following steps to verify if the secrets are recreated and the certificates are renewed:
    a. Run the following command to verify the Kyverno secrets:

    ```
    $ kubectl get secrets -n kyverno
    ```

    Sample output:

    ```
    NAME                                                TYPE
    DATA   AGE
    occne-kyverno-svc.kyverno.svc.kyverno-tls-ca     kubernetes.io/tls
    2      21s
    occne-kyverno-svc.kyverno.svc.kyverno-tls-pair   kubernetes.io/tls
    2      11s
    sh.helm.release.v1.occne-kyverno-policies.v1     helm.sh/release.v1
    1      26h
    sh.helm.release.v1.occne-kyverno.v1              helm.sh/release.v1
    1      26h
    ```

    b. Run the following commands to review the expiry dates of Kyverno certificates:

    ```
    $ for secret in $(kubectl -n kyverno get secrets --no-headers | grep
    kubernetes.io/tls | awk {'print $1'}); do currdate=$(date +'%s');
    echo $secret; expires=$(kubectl -n kyverno get secrets $secret -o
    jsonpath="{.data['tls\.crt']}" | base64 -d | openssl x509 -enddate -
    noout | awk -F"=" {'print $2'} | xargs -d '\n' -I {} date -d '{}'
    +'%s'); if [ $expires -le $currdate ]; then echo "Certificate invalid,
    expired: $(date -d @${expires})"; echo "Need to renew certificate
    using:"; echo "kubectl -n kyverno delete secret $secret"; else echo
    "Certificate valid, expires: $(date -d @${expires})"; fi done
    ```

    Sample output:

    ```
    occne-kyverno-svc.kyverno.svc.kyverno-tls-ca
    Certificate valid, expires: Wed Feb 25 05:35:03 PM EST 2026
    occne-kyverno-svc.kyverno.svc.kyverno-tls-pair
    Certificate valid, expires: Fri Jul 25 06:35:12 PM EDT 2025
    ```

Renewing Kubelet Server Certificate

This section provides the procedure to renew Kubelet server certificate using the `renew-kubelet-server-cert.sh` script.

The certificate rotation configuration of the Kubelet server renews the Kubelet client certificates automatically, as this configuration is enabled by default. The `renew-kubelet-server-cert.sh` script sets the `--rotate-server-certificates` flag to *true*, which enables the `serverTLSBootstrap` variable in the Kubelet configuration.

1. Use SSH to log in to the active Bastion Host.

2. Run the following command to verify if the Bastion Host is the active Bastion Host:

   ```
   $ is_active_bastion
   ```

   The system displays the following output if the Bastion Host the active Bastion Host:

   ```
   IS active-bastion
   ```

   If the Bastion Host is not the active Bastion Host, try a different Bastion Host.

   > **Note**
   >
   > If the certificates are expired, the `is_active_bastion` command doesn't work as it depends on `kubectl`. In this case, skip this step and move to the next step.

3. Navigate to the `/var/occne/cluster/${OCCNE_CLUSTER}/artifacts/` directory:

   ```
   $ cd /var/occne/cluster/${OCCNE_CLUSTER}/artifacts/
   ```

4. Run the `renew-kubelet-server-cert.sh` script:

   ```
   $ ./renew-kubelet-server-cert.sh
   ```

   Sample output:

   ```
   ============ Checking if all nodes are accessible via ssh ============
   occne3-k8s-ctrl-1
   occne3-k8s-ctrl-2
   occne3-k8s-ctrl-3
   occne3-k8s-node-1
   occne3-k8s-node-2
   occne3-k8s-node-3
   occne3-k8s-node-4
   All nodes are healthy and accessible using ssh, Starting kubelet server
   certificate renewal procedure now...
   --------------------------------------------------------------------------
   ------------------
   Starting renewal of K8s kubelet server certificate for occne3-k8s-ctrl-1.
   Adding the line --rotate-server-certificates=true --tls-cipher-
   suites=TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 to kubelet environment file.
   Restarting Kubelet to trigger Certificate signing request...
   Kubelet is successfully restarted!
   A signing request has been raised, Verifying it now....
   A Certificate signing request csr-lfsq9 has been found, Approving it now!
   certificatesigningrequest.certificates.k8s.io/csr-lfsq9 approved
   The CSR has been approved for the node occne3-k8s-ctrl-1.
   Checking if the new K8s kubelet server certificate has been generated...
   New K8s kubelet server certificate has been successfully generated for the
   node occne3-k8s-ctrl-1 as shown below.
   lrwxrwxrwx. 1 root root 59 Jul 24 08:05 kubelet-server-current.pem -
   > /var/lib/kubelet/pki/kubelet-server-2024-07-24-08-05-40.pem
   Marked occne3-k8s-ctrl-1 as RENEWED.
   ```

```
Kubelet server certificate creation was successful for the node occne3-k8s-
ctrl-1.
```

7.4.3 Renewing the Kubernetes Secrets Encryption Key

This section describes the procedure to renew the key that is used to encrypt the Kubernetes Secrets stored in the CNE Kubernetes cluster.

Procedure
The key that is used to encrypt Kubernetes Secrets does not expire. However, it is recommended to change the encryption key periodically to ensure the security of your Kubernetes Secrets. If you think that your key is compromised, you must change the encryption key immediately.

To renew a Kubernetes Secrets encryption key, perform the following steps:

1. From bastion host, run the following commands:

```
$ NEW_KEY=$(head -c 32 /dev/urandom | base64)
$ KEY_NAME=$(cat /dev/random | tr -dc '[:alnum:]' | head -c 10)
$ kubectl get nodes | awk '/control-plane/ {print $1}' | xargs -I{} ssh {}
" sudo sed -i '/keys:$/a\        - name: key_$KEY_NAME\n\
secret: $NEW_KEY' /etc/kubernetes/ssl/secrets_encryption.yaml; sudo
cat /etc/kubernetes/ssl/secrets_encryption.yaml"
```

   This creates a random encryption key with a random key name, and adds it to the `/etc/kubernetes/ssl/secrets_encryption.yaml` file within each controller node. The output shows the new encryption key, the key name, and the contents of the `/etc/kubernetes/ssl/secrets_encryption.yaml` file.

   Sample Output:

```
This site is for the exclusive use of Oracle and its authorized customers
and partners. Use of this site by customers and partners is subject to the
Terms of Use and Privacy Policy for this site, as well as your contract
with Oracle. Use of this site by Oracle employees is subject to company
policies, including the Code of Conduct. Unauthorized access or breach of
these terms may result in termination of your authorization to use this
site and/or civil and criminal penalties.
kind: EncryptionConfig
apiVersion: v1
resources:
  - resources:
    - secrets

    providers:
    - secretbox:
        keys:
        - name: key_ZOJ1Hf5OCx
          secret: l+CaDTmMkC85LwJRiWJ0LQPYVtOyZ0TdtNZ2ij+kuGA=
        - name: key
          secret: ZXJ1Ulk2U0xSbWkwejdreTlJWkFrZmpJZjhBRzg4U00=
    - identity: {}
```

2. Restart the API server by running the following command. This ensures that all the secrets get encrypted with the new key while encrypting the secrets in the next step:

   ```
   kubectl get nodes | awk '/control-plane/ {print $1}' | xargs -I{} ssh {} "
   sudo mv /etc/kubernetes/manifests/kube-apiserver.yaml ~; sleep 2; sudo mv
   ~/kube-apiserver.yaml /etc/kubernetes/manifests"
   ```

3. To encrypt all the existing secrets with a new key, run the following command:

   ```
   kubectl get secrets --all-namespaces -o json | kubectl replace -f
   ```

   Sample output:

   ```
   -secret/occne-cert-manager-webhook-ca replaced
   secret/sh.helm.release.v1.occne-cert-manager.v1 replaced
   secret/istio-ca-secret replaced
   secret/cloud-config replaced
   secret/external-openstack-cloud-config replaced
   secret/occne-kyverno-svc.kyverno.svc.kyverno-tls-ca replaced
   secret/occne-kyverno-svc.kyverno.svc.kyverno-tls-pair replaced
   secret/sh.helm.release.v1.occne-kyverno-policies.v1 replaced
   secret/sh.helm.release.v1.occne-kyverno.v1 replaced
   secret/alertmanager-occne-kube-prom-stack-kube-alertmanager replaced
   secret/etcd-occne6-j-jorge-l-lopez-k8s-ctrl-1 replaced
   secret/etcd-occne6-j-jorge-l-lopez-k8s-ctrl-2 replaced
   secret/etcd-occne6-j-jorge-l-lopez-k8s-ctrl-3 replaced
   secret/lb-controller-user replaced
   secret/occne-alertmanager-snmp-notifier replaced
   secret/occne-kube-prom-stack-grafana replaced
   secret/occne-kube-prom-stack-kube-admission replaced
   secret/occne-kube-prom-stack-kube-prometheus-scrape-confg replaced
   secret/occne-metallb-memberlist replaced
   secret/occne-tracer-jaeger-elasticsearch replaced
   secret/prometheus-occne-kube-prom-stack-kube-prometheus replaced
   secret/prometheus-occne-kube-prom-stack-kube-prometheus-tls-assets-0
   replaced
   secret/prometheus-occne-kube-prom-stack-kube-prometheus-web-config replaced
   secret/sh.helm.release.v1.occne-alertmanager-snmp-notifier.v1 replaced
   secret/sh.helm.release.v1.occne-bastion-controller.v1 replaced
   secret/sh.helm.release.v1.occne-fluentd-opensearch.v1 replaced
   secret/sh.helm.release.v1.occne-kube-prom-stack.v1 replaced
   secret/sh.helm.release.v1.occne-lb-controller.v1 replaced
   secret/sh.helm.release.v1.occne-metallb.v1 replaced
   secret/sh.helm.release.v1.occne-metrics-server.v1 replaced
   secret/sh.helm.release.v1.occne-opensearch-client.v1 replaced
   secret/sh.helm.release.v1.occne-opensearch-dashboards.v1 replaced
   secret/sh.helm.release.v1.occne-opensearch-data.v1 replaced
   secret/sh.helm.release.v1.occne-opensearch-master.v1 replaced
   secret/sh.helm.release.v1.occne-promxy.v1 replaced
   secret/sh.helm.release.v1.occne-tracer.v1 replaced
   secret/webhook-server-cert replaced
   Error from server (Conflict): error when replacing "STDIN": Operation
   cannot be fulfilled on secrets "alertmanager-occne-kube-prom-stack-kube-
   alertmanager-generated": the object has been modified; please apply your
   changes to the latest version and try again
   ```

```
Error from server (Conflict): error when replacing "STDIN": Operation
cannot be fulfilled on secrets "alertmanager-occne-kube-prom-stack-kube-
alertmanager-tls-assets-0": the object has been modified; please apply
your changes to the latest version and try again
Error from server (Conflict): error when replacing "STDIN": Operation
cannot be fulfilled on secrets "alertmanager-occne-kube-prom-stack-kube-
alertmanager-web-config": the object has been modified; please apply your
changes to the latest version and try again
```

> **Note**
>
> You may see some errors on the output depending on how the secret is created. You can ignore these errors and verify the encrypted secret using the following step.

4. To verify if the new key is used for encrypting the existing secrets, run the following command from a controller node. Replace **<cert pem file>**, **<key pem file>** and **<secret>** in the following command with the corresponding values.

   ```
   sudo ETCDCTL_API=3 /usr/local/bin/etcdctl --cert /etc/ssl/etcd/ssl/<cert
   pem file> --key /etc/ssl/etcd/ssl/<key pem file> get /registry/secrets/
   default/<secret> -w fields | grep Value
   ```

   Example:

   ```
   [cloud-user@occne3-user-k8s-ctrl-3 ~]$ sudo ETCDCTL_API=3 /usr/local/bin/
   etcdctl --cert /etc/ssl/etcd/ssl/node-occne3-user-k8s-ctrl-1.pem --
   key /etc/ssl/etcd/ssl/node-occne3-user-k8s-ctrl-1-key.pem get /registry/
   secrets/default/secret1 -w fields | grep Value
   "Value" :
   "k8s:enc:secretbox:v1:key_ZOJ1Hf5OCx:&9\x90\u007f'*6\x0e\xf8]\x98\xd7t1\xa9
   |\x90\x93\x88\xebc\xa9\xfe\x82<\xeb\xaa\x17$
   \xa4\x14%m\xb7<\x1d\xf7N\b\xa7\xbaZ\xb0\xd4#\xbev)
   \x1bv9\x19\xdel\xab\x89@\xe7\xaf$L\xb8)\xc9\x1bl\x13\xc1V\x1b\xf7\bX\x88\xe
   7\ue131\x1dG\xe2_\x04\xa2\xf1n\xf5\x1dP\
   \4\xe7)^\x81go\x99\x98b\xbb\x0e\xc0R;>
   j\xeeV54\xac\x06\t\x1b9\xd5N\xa77\xd9\x03 ☒\x05\xfb%\xa1\x81\xd5\x0e
   \xcax\xc4\x1cz6\xf3\xd8\xf9?Щ\x9a%
   \x9b\xe5\xa7й\xcd!,\xb8\x8b\xc2\xcf\xe2\xf2|
   \x8f\x90\xa9\x05y\xc5\xfc\xf7\x87\xf9\x13\x0e4[i\x12\xcc\xfaR\xdf3]\xa2V\x1
   b\xbb\xeba6\x1c\xba\v\xb0p}\xa5;\x16\xab\x8e\xd5Ol\xb7\x87BW\tY;寄
   \xca\x87Y;\n;/
   \xf2\x89\xa1\xcc\xc3\xc9\xe3\xc5\v\x1b\x88\x84
   \xc6\x00\xb4\xed\xa5\xe2\xfa\xa9\xff \xd9k\xf2\x04\x8f\x81,l"
   ```

   This example shows a new key, **key_ZOJ1Hf5OCx**, being used to encrypt **secret1**.

7.4.4 Removing a Kubernetes Controller Node

This section describes the procedure to remove a controller node from the CNE Kubernetes cluster in a vCNE deployment.

> **Note**
>
> - A controller node must be removed from the cluster only when it is required for maintenance.
> - This procedure is applicable for vCNE (OpenStack and VMWare) deployments only.
> - This procedure is applicable for removing a single controller node only.

7.4.4.1 Removing a Controller Node in OpenStack Deployment

This section describes the procedure to remove a single controller node from the CNE Kubernetes cluster in an OpenStack deployment.

Procedure

1. Locate the controller node internal IP address by running the following command from the Bastion Host:

   ```
   $ kubectl get nodes -o wide | egrep control |  awk '{ print $1, $2, $6}'
   ```

   For example:

   ```
   $ [cloud-user@occne7-test-bastion-1 ~]$ kubectl get node -o wide | egrep
   control |  awk '{ print $1, $2, $6}'
   ```

   Sample output:

   ```
   occne7-test-k8s-ctrl-1 NotReady 192.168.201.158
   occne7-test-k8s-ctrl-2 Ready 192.168.203.194
   occne7-test-k8s-ctrl-3 Ready 192.168.200.115
   ```

   Note that the status of controller node 1 is `NotReady` in the sample output.

2. Run the following commands to backup the `terraform.tfstate` file:

   ```
   $ cd /var/occne/cluster/${OCCNE_CLUSTER}
   $ cp terraform.tfstate ${OCCNE_CLUSTER}/terraform.tfstate.backup
   ```

3. From the Bastion Host, use SSH to log in to a working controller node and run the following commands to list the etcd members:

   ```
   $ ssh <working control node hostname>
   # sudo su
   # source /etc/etcd.env
   ```

```
# /usr/local/bin/etcdctl --endpoints https://<working control node IP
address>:2379 --cacert=${ETCD_PEER_TRUSTED_CA_FILE} --cert=$
{ETCD_CERT_FILE} --key=${ETCD_KEY_FILE} member list
```

For example:

```
$ ssh occne7-test-k8s-ctrl-2

[cloud-user@occne7-test-k8s-ctrl-2]$ sudo su

[root@occne7-test-k8s-ctrl-2 cloud-user]# source /etc/etcd.env

[root@occne7-test-k8s-ctrl-2 cloud-user]# /usr/local/bin/etcdctl --endpoints https://
192.168.203.194:2379 --cacert=${ETCD_PEER_TRUSTED_CA_FILE} --cert=${ETCD_CERT_FILE}
--key=${ETCD_KEY_FILE} member list
52513ddd2aa49770, started, etcd1, https://192.168.201.158:2380, https://
192.168.201.158:2379, false
80845fb2b5120458, started, etcd3, https://192.168.200.115:2380, https://
192.168.200.115:2379, false
f1200d9975868073, started, etcd2, https://192.168.203.194:2380, https://
192.168.203.194:2379, false
```

a. From the output, identify the etcd (etcd1, etcd2, or etcd3) to which the failed controller node belongs.

b. Copy the controller node ID that is displayed in the first column of the output to be used later in the procedure.

4. If the failed controller node is reachable, use SSH to log in to the failed controller node from the Bastion Host and stop etcd service by running the following commands:

```
$ ssh <failed control node hostname>

$ sudo systemctl stop etcd
```

Example:

```
$ ssh occne7-test-k8s-ctrl-1

$ sudo systemctl stop etcd
```

5. From the Bastion Host, use SSH to log in to a working controller node and remove the failed controller node from the etcd member list:

```
$ ssh <working control node hostname>
$ sudo su
$ source /etc/etcd.env
$ /usr/local/bin/etcdctl --endpoints https://<working control node IP
address>:2379 --cacert=${ETCD_PEER_TRUSTED_CA_FILE} --cert=$
{ETCD_CERT_FILE} --key=${ETCD_KEY_FILE} member remove <failed control node
ID>
```

Example:

```
[root@occne7-test-k8s-ctrl-2 cloud-user]# /usr/local/bin/etcdctl --endpoints https://
192.168.203.194:2379 --cacert=${ETCD_PEER_TRUSTED_CA_FILE} --cert=${ETCD_CERT_FILE}
--key=${ETCD_KEY_FILE} member remove 52513ddd2aa49770
```

Sample output:

```
Member 52513ddd2aa49770 removed from cluster f347ab69786ba4f7
```

6. Validate if the failed node is removed from the etcd member list:

   ```
   $ /usr/local/bin/etcdctl --endpoints https://<working control node IP
   address>:2379 --cacert=${ETCD_PEER_TRUSTED_CA_FILE} --cert=$
   {ETCD_CERT_FILE} --key=${ETCD_KEY_FILE} member list
   ```

   For example:

   ```
   [root@occne7-test-k8s-ctrl-2 cloud-user]# /usr/local/bin/etcdctl --
   endpoints https://192.168.203.194:2379 --cacert=$
   {ETCD_PEER_TRUSTED_CA_FILE} --cert=${ETCD_CERT_FILE} --key=$
   {ETCD_KEY_FILE} member list
   80845fb2b5120458, started, etcd3, https://192.168.200.115:2380, https://
   192.168.200.115:2379, false
   f1200d9975868073, started, etcd2, https://192.168.203.194:2380, https://
   192.168.203.194:2379, false
   ```

7. From the Bastion Host, switch the controller nodes in `terraform.tfstate` by running the following commands:

   > **Note**
   >
   > Perform this step only if the failed controller node is a **etcd1** member.

   ```
   $ cd /var/occne/cluster/$OCCNE_CLUSTER
   $ cp terraform.tfstate terraform.tfstate.original
   $ python3 scripts/switchTfstate.py
   ```

   For example:

   ```
   [cloud-user@occne7-test-bastion-1]$ python3 scripts/switchTfstate.py
   ```

   Sample output:

   ```
   Beginning tfstate switch order k8s control nodes

           terraform.tfstate.lastversion created as backup

   Controller Nodes order before rotation:
   occne7-test-k8s-ctrl-1
   occne7-test-k8s-ctrl-2
   occne7-test-k8s-ctrl-3

   Controller Nodes order after rotation:
   occne7-test-k8s-ctrl-2
   occne7-test-k8s-ctrl-3
   occne7-test-k8s-ctrl-1

   Success: terraform.tfstate rotated for cluster occne7-test
   ```

8. Remove the failed controller node from the cluster by performing one the following steps in the Bastion Host depending on whether the failed controller node is reachable or not:

- If the failed controller node is reachable, run the following commands to remove the controller node from the cluster:

  ```
  $ kubectl cordon <failed control node hostname>
  $ kubectl drain <failed control node hostname> --force --ignore-
  daemonsets  --delete-emptydir-data
  $ kubectl delete node <failed control node hostname>
  ```

  Example:

  ```
  $ [cloud-user@occne7-test-bastion-1]$ kubectl cordon occne7-test-k8s-
  ctrl-1
  $ [cloud-user@occne7-test-bastion-1]$ kubectl drain occne7-test-k8s-
  ctrl-1 --force --ignore-daemonsets  --delete-emptydir-data
  $ [cloud-user@occne7-test-bastion-1]$ kubectl delete node occne7-test-
  k8s-ctrl-1
  ```

- If the failed controller node is not reachable, run the following commands to remove the controller node from the cluster:

  ```
  $ kubectl cordon <failed control node hostname>
  $ kubectl delete node <failed control node hostname>
  ```

  Example:

  ```
  $ [cloud-user@occne7-test-bastion-1]$ kubectl cordon occne7-test-k8s-
  ctrl-1
  $ [cloud-user@occne7-test-bastion-1]$ kubectl delete node occne7-test-
  k8s-ctrl-1
  ```

9. Verify if the failed controller node is deleted from cluster.

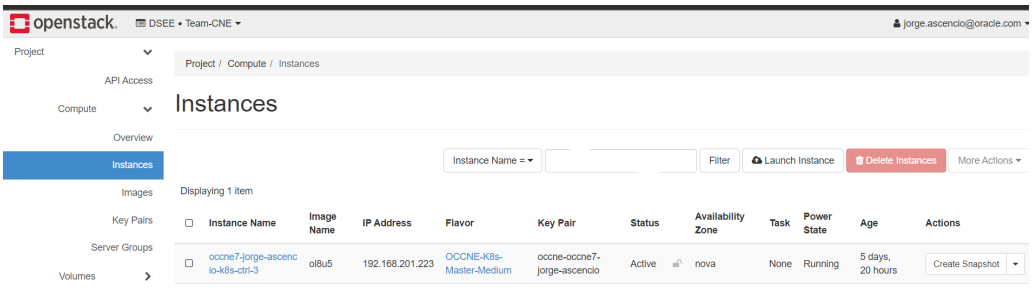
   ```
   $ kubectl get node
   ```

   Sample output:

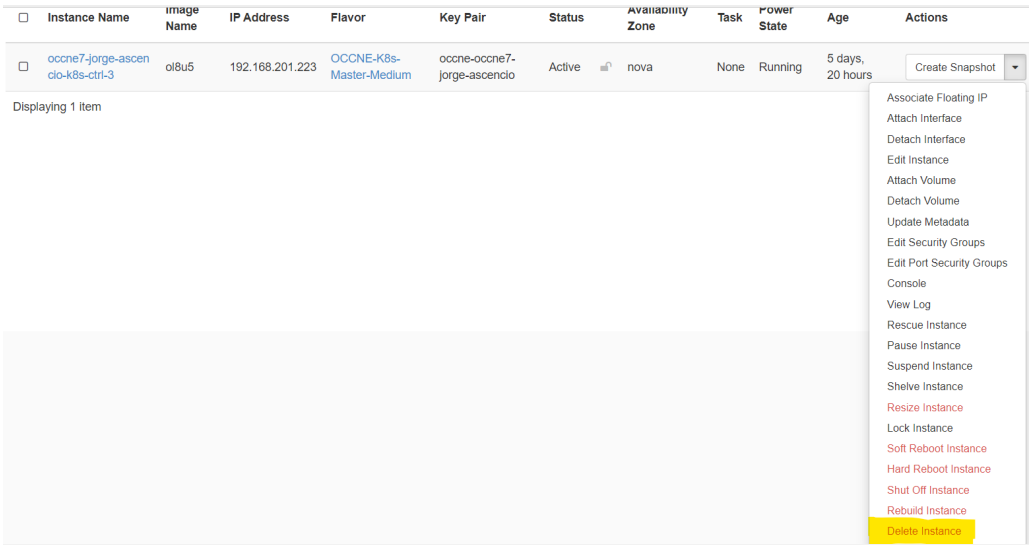
   ```
   [cloud-user@occne7-test-bastion-1]$ kubectl get node
   NAME                   STATUS ROLES                AGE VERSION
   occne7-test-k8s-ctrl-2 Ready  control-plane,master 82m v1.23.7
   occne7-test-k8s-ctrl-3 Ready  control-plane,master 82m v1.23.7
   occne7-test-k8s-node-1 Ready  <none>               81m v1.23.7
   occne7-test-k8s-node-2 Ready  <none>               81m v1.23.7
   occne7-test-k8s-node-3 Ready  <none>               81m v1.23.7
   occne7-test-k8s-node-4 Ready  <none>               81m v1.23.7
   ```

> **Note**
>
> If you are not able to run `kubectl` commands from the Bastion Host, update the `/var/occne/cluster/$OCCNE_CLUSTER/artifacts/admin.conf` file with the new working node IP address:
>
> ```
> vi /var/occne/cluster/occne7-test/artifacts/admin.conf
>  server: https://192.168.203.194:6443
> ```

10. Delete the failed controller node's instance using the Openstack GUI:
    a. Log in to OpenStack cloud using your credentials.
    b. From the **Compute** menu, select **Instances**, and locate the failed controller node's instance that you want to delete, as shown in the following image:



    c. On the instance record, click the drop-down option in the **Actions** column, select **Delete Instance** to delete the failed controller node's instance, as shown in the following image:



7.4.4.2 Removing a Controller Node in VMware Deployment

This section describes the procedure to remove a single controller node from the CNE Kubernetes cluster in a VMware deployment.

Procedure

1. Locate the controller node internal IP address by running the following command from the Bastion Host:

   ```
   $ kubectl get node -o wide | egrep ctrl |  awk '{ print $1, $2, $6}'
   ```

   Sample output:

   ```
   occne7-test-k8s-ctrl-1 192.168.201.158
   occne7-test-k8s-ctrl-2 192.168.203.194
   occne7-test-k8s-ctrl-3 192.168.200.115
   ```

   For example:

   ```
   $ [cloud-user@occne7-test-bastion-1 ~]$ kubectl get node -o wide | egrep
   control |  awk '{ print $1, $2, $6}'
   ```

   Sample output:

   ```
   occne7-test-k8s-ctrl-1 NotReady 192.168.201.158
   occne7-test-k8s-ctrl-2 Ready 192.168.203.194
   occne7-test-k8s-ctrl-3 Ready 192.168.200.115
   ```

   Note that the status of control node 1 is `NotReady` in the sample output.

2. Backup the **terraform.tfstate** file by running the following commands:

   ```
   $ cd /var/occne/cluster/${OCCNE_CLUSTER}
   $ cp terraform.tfstate ${OCCNE_CLUSTER}/terraform.tfstate.backup
   ```

3. On the Bastion Host, use SSH to log in to a working controller node and run the following commands to list the etcd members:

   ```
   $ ssh <working control node hostname>
   # sudo su
   # source /etc/etcd.env
   # /usr/local/bin/etcdctl --endpoints https://<working control node IP
   address>:2379 --cacert=${ETCD_PEER_TRUSTED_CA_FILE} --cert=$
   {ETCD_CERT_FILE} --key=${ETCD_KEY_FILE} member list
   ```

   For example:

   ```
   $ ssh occne7-test-k8s-ctrl-2

   [cloud-user@occne7-test-k8s-ctrl-2]$ sudo su
   ```

```
[root@occne7-test-k8s-ctrl-2 cloud-user]# source /etc/etcd.env

[root@occne7-test-k8s-ctrl-2 cloud-user]# /usr/local/bin/etcdctl --endpoints https://
192.168.203.194:2379 --cacert=${ETCD_PEER_TRUSTED_CA_FILE} --cert=${ETCD_CERT_FILE}
--key=${ETCD_KEY_FILE} member list
```

Sample output:

```
52513ddd2aa49770, started, etcd1, https://192.168.201.158:2380, https://
192.168.201.158:2379, false
80845fb2b5120458, started, etcd3, https://192.168.200.115:2380, https://
192.168.200.115:2379, false
f1200d9975868073, started, etcd2, https://192.168.203.194:2380, https://
192.168.203.194:2379, false
```

a. From the output, identify the etcd (etcd1, etcd2, or etcd3) to which the failed controller node belongs.

b. Copy the controller node ID that is displayed in the first column of the output to be used later in the procedure.

4. If the failed controller node is reachable, use SSH to log in to the failed controller node from the Bastion Host and stop etcd service by running the following commands:

```
$ ssh <failed control node hostname>

$ sudo systemctl stop etcd
```

For example:

```
$ ssh occne7-test-k8s-ctrl-1

$ sudo systemctl stop etcd
```

5. From the Bastion Host, use SSH to log in to a working controller node and remove the failed controller node from the etcd member list:

```
$ ssh <working control node hostname>
$ sudo su
$ source /etc/etcd.env
$ /usr/local/bin/etcdctl --endpoints https://<working control node IP
address>:2379 --cacert=${ETCD_PEER_TRUSTED_CA_FILE} --cert=$
{ETCD_CERT_FILE} --key=${ETCD_KEY_FILE} member remove <failed control node
ID>
```

For example:

```
[root@occne7-test-k8s-ctrl-2 cloud-user]# /usr/local/bin/etcdctl --
endpoints https://192.168.203.194:2379 --cacert=$
{ETCD_PEER_TRUSTED_CA_FILE} --cert=${ETCD_CERT_FILE} --key=$
{ETCD_KEY_FILE} member remove 52513ddd2aa49770
```

Sample output:

```
Member 52513ddd2aa49770 removed from cluster f347ab69786ba4f7
```

6. Validate if the failed node is removed from the etcd member list:

   ```
   $ /usr/local/bin/etcdctl --endpoints https://<working control node IP
   address>:2379 --cacert=${ETCD_PEER_TRUSTED_CA_FILE} --cert=$
   {ETCD_CERT_FILE} --key=${ETCD_KEY_FILE} member list
   ```

   For example:

   ```
   [root@occne7-test-k8s-ctrl-2 cloud-user]# /usr/local/bin/etcdctl --endpoints https://
   192.168.203.194:2379 --cacert=${ETCD_PEER_TRUSTED_CA_FILE} --cert=${ETCD_CERT_FILE}
   --key=${ETCD_KEY_FILE} member list
   ```

   Sample output:

   ```
   80845fb2b5120458, started, etcd3, https://192.168.200.115:2380, https://
   192.168.200.115:2379, false
   f1200d9975868073, started, etcd2, https://192.168.203.194:2380, https://
   192.168.203.194:2379, false
   ```

7. From the Bastion Host, switch the controller nodes in `terraform.tfstate` by running the following commands:

   > **Note**
   >
   > Perform this step only if the failed controller node is a **etcd1** member.

   ```
   $ cd /var/occne/cluster/${OCCNE_CLUSTER}
   $ cp terraform.tfstate terraform.tfstate.original
   $ python3 scripts/switchTfstate.py
   ```

   For example:

   ```
   [cloud-user@occne7-test-bastion-1]$ python3 scripts/switchTfstate.py
   ```

   Sample output:

   ```
   Beginning tfstate switch order k8s control nodes

           terraform.tfstate.lastversion created as backup

   Controller Nodes order before rotation:
   occne7-test-k8s-ctrl-1
   occne7-test-k8s-ctrl-2
   occne7-test-k8s-ctrl-3

   Controller Nodes order after rotation:
   occne7-test-k8s-ctrl-2
   occne7-test-k8s-ctrl-3
   occne7-test-k8s-ctrl-1

   Success: terraform.tfstate rotated for cluster occne7-test
   ```

8. Remove the failed controller node from the cluster by performing one the following steps in the Bastion Host depending on whether the failed controller node is reachable or not:

- If the failed controller node is reachable, run the following commands to remove the controller node from the cluster:

  ```
  $ kubectl cordon <failed control node hostname>
  $ kubectl drain <failed control node hostname> --force --ignore-
  daemonsets  --delete-emptydir-data
  $ kubectl delete node <failed control node hostname>
  ```

  For example:

  ```
  $ [cloud-user@occne7-test-bastion-1]$ kubectl cordon occne7-test-k8s-ctrl-1
  $ [cloud-user@occne7-test-bastion-1]$ kubectl drain occne7-test-k8s-ctrl-1 --
  force --ignore-daemonsets  --delete-emptydir-data
  $ [cloud-user@occne7-test-bastion-1]$ kubectl delete node occne7-test-k8s-ctrl-1
  ```

- If the failed controller node is not reachable, run the following commands to remove the controller node from the cluster:

  ```
  $ kubectl cordon <failed control node hostname>
  $ kubectl delete node <failed control node hostname>
  ```

  Example:

  ```
  $ [cloud-user@occne7-test-bastion-1]$ kubectl cordon occne7-test-k8s-
  ctrl-1
  $ [cloud-user@occne7-test-bastion-1]$ kubectl delete node occne7-test-
  k8s-ctrl-1
  ```

9. Verify if the failed controller node is deleted from cluster.

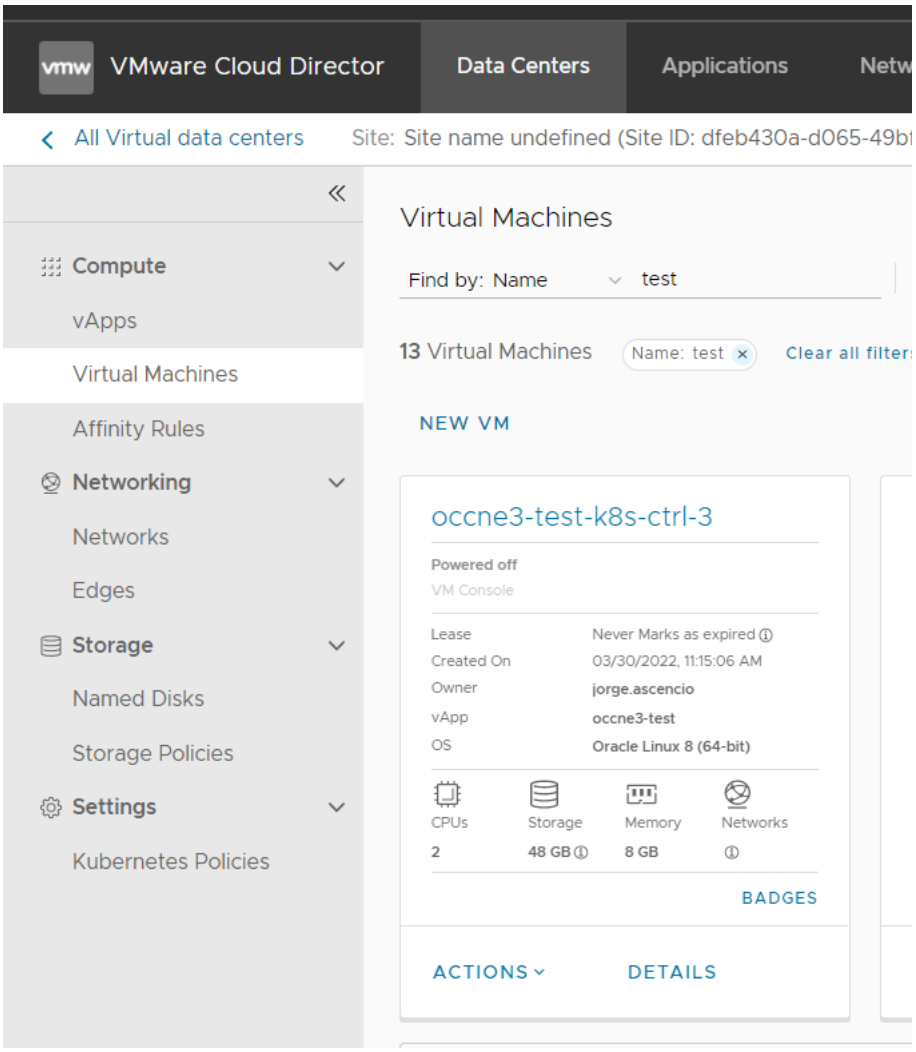
   ```
   $ kubectl get node
   ```

   Sample output:

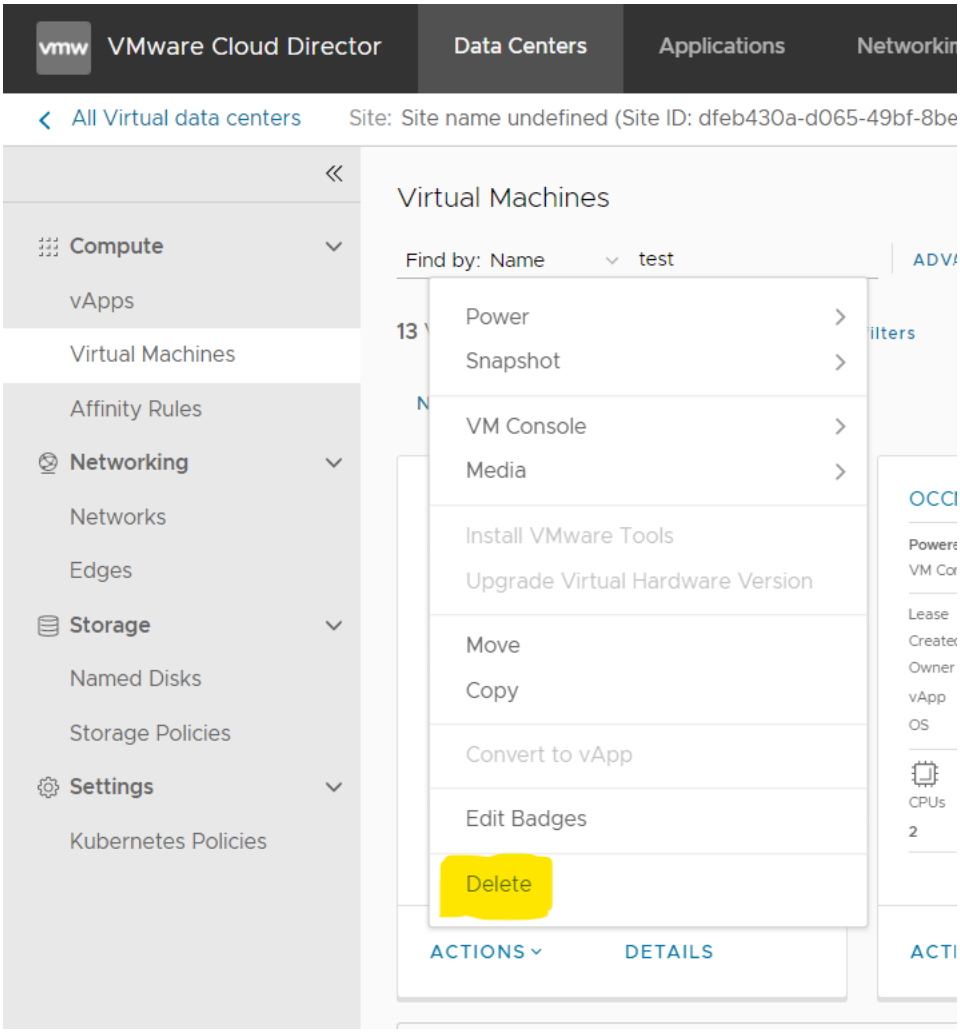
   ```
   NAME                   STATUS ROLES                AGE VERSION
   occne7-test-k8s-ctrl-2 Ready  control-plane,master 82m v1.23.7
   occne7-test-k8s-ctrl-3 Ready  control-plane,master 82m v1.23.7
   occne7-test-k8s-node-1 Ready  <none>               81m v1.23.7
   occne7-test-k8s-node-2 Ready  <none>               81m v1.23.7
   occne7-test-k8s-node-3 Ready  <none>               81m v1.23.7
   occne7-test-k8s-node-4 Ready  <none>               81m v1.23.7
   ```

> **Note**
>
> If you are not able to run `kubectl` commands from the Bastion Host, update the `/var/occne/cluster/$OCCNE_CLUSTER/artifacts/admin.conf` file with the new working node IP address:
>
> ```
> vi /var/occne/cluster/occne7-test/artifacts/admin.conf
>
> server: https://192.168.203.194:6443
> ```

10. Delete the failed controller node's VM using the VMWare GUI:
    a. Log in to VMware cloud using your credentials.
    b. From the **Compute** menu, select **Virtual Machines**, and locate the failed controller node's VM to delete, as shown in the following image:



    c. From the **Actions** menu, select **Delete** to delete the failed controller node's VM, as shown in the following image:



7.4.5 Adding a Kubernetes Worker Node

This section provides the procedure to add additional worker nodes to a previously installed CNE Kubernetes cluster.

Note

- For a BareMetal installation, ensure that you are familiar with the inventory file preparation procedure. For more information about this procedure, see "*Inventory File Preparation*" section in *Oracle Communications Cloud Native Core, Cloud Native Environment Installation, Upgrade, and Fault Recovery Guide*.
- Run this procedure from the active Bastion Host only.
- You can add only one node at a time using this procedure.

Adding a Kubernetes Worker Node on BareMetal

> **Note**
>
> For any failure or successful run, the system maintains all Terraform and pipeline output in the `/var/occne/cluster/${OCCNE_CLUSTER}/addBmWkrNodeCapture-<mmddyyyy_hhmmss>.log` file.

1. Log in to Bastion Host and verify if it's an active Bastion Host. If the Bastion Host isn't an active Bastion Host, then log in to another.
   Use the following command to check if the Bastion Host is an active Bastion Host:

   ```
   $ is_active_bastion
   ```

   The system displays the following output if the Bastion Host is an active Bastion Host:
   `IS active-bastion`

   The system displays the following output if the Bastion Host isn't an active Bastion Host:

   `NOT active-bastion`
2. Run the following command to navigate to the cluster directory:

   ```
   $ cd /var/occne/cluster/${OCCNE_CLUSTER}/
   ```
3. Run the following command to open the `host.ini` file in edit mode:

   ```
   $ vi host.ini
   ```
4. Perform the following steps to edit the `hosts.ini` file and add the node details:
   a. Run the following command to open the `hosts.ini` file in edit mode:

      ```
      $ vi hosts.ini
      ```

   b. Add the node details under the `[host_hp_gen_X]` or `[host_netra_X]` hardware header, depending on your hardware type:

      ```
      [host_hp_gen_10]/[host_netra_X]
      k8s-node.example.oracle.com ansible_host=<ipv4> hp_ilo=<ipv4> mac=<mac-address> pxe_config_ks_nic=<nic0>
      pxe_config_nic_list=<nic0>,<nic1>,<nic2> pxe_uefi=False
      ```

      where, `<NODE_FULL_NAME>` is the full name of the node that is added.

> **Note**
>
> - <NODE_FULL_NAME>, ansible_host, hp_ilo or netra_ilom, and mac are the required parameters and their values must be unique in the host.ini file.
> - <mac-address> must be a string of six two-digit hexadecimal numbers separated by a dash. For example, a2-27-3d-d3-b4-00.
> - All IP addresses must be in proper IPV4 format.
> - pxe_config_ks_nic, pxe_config_nic_list, and pxe_uefi are the optional parameters. The node details can also contain other optional parameters that are not listed in the example.
> - All the required and optional parameters must be in the <KEY>=<VALUE> format without any space between the equal to sign.
> - All defined parameters must have a valid value.
> - Comments must be added in a separate line using # and must not be added at the end of the line.

For example, the following code block displays the node details of a worker node (k8s-node-5.test.us.oracle.com) added under the [host_hp_gen_10] hardware header:

```
...
[host_hp_gen_10]
k8s-host-1.test.us.oracle.com ansible_host=179.1.5.2 hp_ilo=172.16.9.44
mac=a2-27-3d-d3-b4-00 oam_host=10.75.216.13
k8s-host-2.test.us.oracle.com ansible_host=179.1.5.3 hp_ilo=172.16.9.45
mac=4d-d9-1a-e2-7e-e8 oam_host=10.75.216.14
k8s-host-3.test.us.oracle.com ansible_host=179.1.5.4 hp_ilo=172.16.9.46
mac=e1-15-b4-1d-32-10

k8s-node-1.test.us.oracle.com ansible_host=179.1.5.5 hp_ilo=172.16.9.47
mac=3b-d2-2d-f6-1e-20
k8s-node-2.test.us.oracle.com ansible_host=179.1.5.6 hp_ilo=172.16.9.48
mac=a8-1a-37-b1-c0-dc
k8s-node-3.test.us.oracle.com ansible_host=179.1.5.7 hp_ilo=172.16.9.49
mac=a4-be-2d-3f-21-f0
k8s-node-4.test.us.oracle.com ansible_host=179.1.5.8 hp_ilo=172.16.9.35
mac=3a-d9-2c-e6-35-18
# New node
k8s-node-5.test.us.oracle.com ansible_host=179.1.5.9 hp_ilo=172.16.9.46
mac=2a-e1-c3-d4-12-a9
...
```

c. Add the full name of the node under the [kube-node] header.

```
[kube-node]
<NODE_FULL_NAME>
```

where, <NODE_FULL_NAME> is the full name of the node that is added.

For example, the following code block shows the full name of the worker node (`k8s-node-5.test.us.oracle.com`) added under the `[kube-node]` header:

```
...
[kube-node]
k8s-node-1.test.us.oracle.com
k8s-node-2.test.us.oracle.com
k8s-node-3.test.us.oracle.com
k8s-node-4.test.us.oracle.com
# New node
k8s-node-5.test.us.oracle.com
...
```

d. Save the `host.ini` file and exit.

5. Navigate to the `maintenance` directory:

```
$ cd /var/occne/cluster/${OCCNE_CLUSTER}/artifacts/maintenance
```

6. The `addBmWorkerNode.py` script in the `maintenance` directory is used to add Kubernetes worker node on BareMetal. Run the following command to add one worker node at a time:

```
$ ./addBmWorkerNode.py -nn <NODE_FULL_NAME>
```

where, `<NODE_FULL_NAME>` is the full name of the node that you added to the `host.ini` file in the previous steps.

For example:

```
$ ./addBmWorkerNode.py -nn k8s-5.test.us.oracle.com
```

Sample output:

```
Beginning add worker node: k8s-5.test.us.oracle.com
 - Backing up configuration files
 - Verify hosts.ini values
 - Updating /etc/hosts on all nodes with new node
    - Successfully updated file: /etc/hosts on all servers - check /var/
occne/cluster/test/addBmWkrNodeCapture-05312024_224446.log for details.
 - Set maintenance banner
    - Successfully set maintenance banner - check /var/occne/cluster/test/
addBmWkrNodeCapture-05312024_224446.log for details.
 - Create toolbox
 - Checking if the rook-ceph toolbox deployment already exists.
    - rook-ceph toolbox deployment already exists, skipping creation.
 - Wait for Toolbox pod
 - Waiting for Toolbox pod to be in Running state.
    - ToolBox pod in namespace rook-ceph is now in Running state.
 - Updating OS on new node
    - Successfully run Provisioning pipeline - check /var/occne/cluster/
test/addBmWkrNodeCapture-05312024_224446.log for details.
 - Scaling new node into cluster
    - Successfully run k8_install scale playbook - check /var/occne/
cluster/test/addBmWkrNodeCapture-05312024_224446.log for details.
 - Running verification
```

```
    - Node k8s-5.test.us.oracle.com verification passed.
  - Restarting rook-ceph operator
    - rook-ceph pods ready!
  - Restoring default banner
     - Successfully run POST stage on PROV container - check /var/occne/
cluster/test/addBmWkrNodeCapture-05312024_224446.log for details.
Worker node: k8s-5.test.us.oracle.com added successfully
```

7. Run the following commands to verify if the node is added successfully:
   a. Run the following command and verify if the new node is in the Ready state:

      ```
      $ kubectl get nodes
      ```

      Sample output:

      ```
      NAME                                STATUS   ROLES           AGE
      VERSION
      k8s-master-1.test.us.oracle.com   Ready    control-plane   7d15h
      v1.29.1
      k8s-master-2.test.us.oracle.com   Ready    control-plane   7d15h
      v1.29.1
      k8s-master-3.test.us.oracle.com   Ready    control-plane   7d15h
      v1.29.1
      k8s-node-1.test.us.oracle.com     Ready    <none>          7d15h
      v1.29.1
      k8s-node-2.test.us.oracle.com     Ready    <none>          7d15h
      v1.29.1
      k8s-node-4.test.us.oracle.com     Ready    <none>          7d15h
      v1.29.1
      k8s-node-5.test.us.oracle.com     Ready    <none>          14m
      v1.29.1
      ```

   b. Run the following command and verify if all pods are in the Running or Completed state:

      ```
      $ kubectl get pod -A
      ```

   c. Run the following command and verify if the services are running and the service GUIs are reachable:

      ```
      $ kubectl get svc -A
      ```

Adding a Kubernetes Worker Node on vCNE (OpenStack and VMware)

> **Note**
>
> For any failure or successful run, the system maintains all Terraform and pipeline output in the `/var/occne/cluster/${OCCNE_CLUSTER}/addWrkNodeCapture-<mmddyyyy_hhmmss>.log` file.

1. Log in to a Bastion Host and ensure if all the pods are in the Running or Completed state:

   ```
   $ kubectl get pod -A
   ```

2. Verify if the services are reachable and if the common services GUIs are accessible using the LoadBalancer EXTERNAL-IPs:

   ```
   $ kubectl get svc -A | grep LoadBalancer
   $ curl <svc_external_ip>
   ```

3. Navigate to the cluster directory:

   ```
   $ cd /var/occne/cluster/$OCCNE_CLUSTER/
   ```

4. Run the following command to open the `$OCCNE_CLUSTER/cluster.tfvars` file. Search for the `number_of_k8s_nodes` parameter in the file and increment the value of the parameter by one.

   ```
   $ vi $OCCNE_CLUSTER/cluster.tfvars
   ```

   The following example shows the current value of `number_of_k8s_nodes` set to *5*:

   ```
   ...
   # k8s nodes
   #
   number_of_k8s_nodes = 5
   ...
   ```

   The following example shows the value of `number_of_k8s_nodes` incremented by one to *6*.

   ```
   ...
   # k8s nodes
   #
   number_of_k8s_nodes = 6
   ...
   ```

5. For OpenStack, perform this step to source the `openrc.sh` file. The `openrc.sh` file sets the necessary environment variables for OpenStack. For VMware, skip this step and move to the next step.

   a. Source the `openrc.sh` file.

      ```
      $ source openrc.sh
      ```

   b. Enter the OpenStack username and password when prompted.
      The following block shows the username and password prompt displayed by the system:

      ```
      Please enter your OpenStack Username:
       Please enter your OpenStack Password as <username>:
      ```

6. Run the following command to ensure that the `openstack-cacert.pem` file exists in the same folder and the file is populated with appropriate certificates if TLS is supported:

   ```
   $ ls /var/occne/cluster/$OCCNE_CLUSTER
   ```

Sample output:

```
...
openstack-cacert.pem
...
```

7. Run the `addWorkerNode.py` script to add a worker node:

> **Note**
>
> The system backs up number of files such as `lbvm/lbCtrlData.json`, `cluster.tfvars`, `hosts.ini`, `terraform.tfstate` (renamed to terraform.tfstate.ORIG), and `/etc/hosts` into the `/var/occne/cluster/${OCCNE_CLUSTER}/backUpConfig` directory. These files are backed up only once to take a backup of the original files.

```
$ ./scripts/addWorkerNode.py
```

Sample output for OpenStack:

```
Starting addWorkerNode instance for the last worker node.

 - Backing up configuration files...

 - Checking if cluster.tfvars matches with the terraform state...
   Succesfully checked the number_of_k8s_nodes parameter in the
cluster.tfvars file.

 - Running terraform apply to update its state...
   Successfully applied Openstack terraform apply - check /var/occne/
cluster/occne-test/addWkrNodeCapture-11262024_220914.log for details

 - Get name for the new worker node...
   Successfully retrieved the name of the new worker node.

 - Update /etc/hosts files on all previous servers...
   Successfully updated file: /etc/hosts on all servers - check /var/occne/
cluster/occne-test/addWkrNodeCapture-11262024_220914.log for details.

 - Setting maintenance banner...
   Successfully set maintenance banner - check /var/occne/cluster/occne-
test/addWkrNodeCapture-11262024_220914.log for details.

 - Running pipeline.sh for provision - can take considerable time to
complete...
   Successfully run Provisioning pipeline - check /var/occne/cluster/occne-
test/addWkrNodeCapture-11262024_220914.log for details.

 - Running pipeline.sh for k8s_install - can take considerable time to
complete...
   Successfully run K8s pipeline - check /var/occne/cluster/occne-test/
addWkrNodeCapture-11262024_220914.log for details.
```

```
 - Get IP address for the new worker node...
   Successfully retrieved IP address of the new worker node occne-test-k8s-
node-5.

 - Update lbCtrlData.json file...
   Successfully updated file: /var/occne/cluster/occne-test/lbvm/
lbCtrlData.json.

 - Update lb-controller-ctrl-data and lb-controller-master-ip configmap...
   Successfully created configmap lb-controller-ctrl-data.
   Successfully created configmap lb-controller-master-ip.

 - Restarting LB Controller POD to bind in configmaps...
   Successfully restarted deployment occne-lb-controller-server.
   Waiting for occne-lb-controller-server deployment to return to Running
status.
   Deployment "occne-lb-controller-server" successfully rolled out

 - Update servers from new occne-lb-controller pod...
   Successfully updated server list for each service in haproxy.cfg on
LBVMs with new node: occne-test-k8s-node-5.

 - Restoring default banner...
   Successfully restored default banner - check /var/occne/cluster/occne-
test/addWkrNodeCapture-11262024_220914.log for details.

Worker node successfully added to cluster: occne-test
```

Sample output for VMware:

```
Starting addWorkerNode instance for the last worker node.

 - Backing up configuration files...

 - Checking if cluster.tfvars matches with the terraform state...
   Succesfully checked the number_of_k8s_nodes parameter in the
cluster.tfvars file.

 - Running terraform apply to update its state...
   VmWare terraform apply -refresh-only successful - check /var/occne/
cluster/occne5-chandrasekhar-musti/addWkrNodeCapture-11282023_115313.log
for details.
   VmWare terraform apply successful - node - check /var/occne/cluster/
occne5-chandrasekhar-musti/addWkrNodeCapture-11282023_115313.log for
details.

 - Get name for the new worker node...
   Successfully retrieved the name of the new worker node.

 - Running pipeline.sh for provision - can take considerable time to
complete...
   Successfully run Provisioning pipeline - check /var/occne/cluster/
occne5-chandrasekhar-musti/addWkrNodeCapture-11282023_115313.log for
details.
```

```
 - Running pipeline.sh for k8s_install - can take considerable time to
complete...
   Successfully run K8s pipeline - check /var/occne/cluster/occne5-
chandrasekhar-musti/addWkrNodeCapture-11282023_115313.log for details.

 - Get IP address for the new worker node...
   Successfully retrieved IP address of the new worker node occne5-
chandrasekhar-musti-k8s-node-4.

 - Update /etc/hosts files on all previous servers...
   Successfully updated file: /etc/hosts on all servers - check /var/occne/
cluster/occne5-chandrasekhar-musti/addWkrNodeCapture-11282023_115313.log
for details.

 - Update lbCtrlData.json file...
   Successfully updated file: /var/occne/cluster/occne5-chandrasekhar-
musti/lbvm/lbCtrlData.json.

 - Update lb-controller-ctrl-data and lb-controller-master-ip configmap...
   Successfully created configmap lb-controller-ctrl-data.
   Successfully created configmap lb-controller-master-ip.

 - Deleting LB Controller POD: occne-lb-controller-server-5d8cd867b7-s5gb2
to bind in configmaps...
   Successfully restarted deployment occne-lb-controller-server.
   Waiting for occne-lb-controller-server deployment to return to Running
status.
   Deployment "occne-lb-controller-server" successfully rolled out

 - Update servers from new occne-lb-controller pod...
   Successfully updated server list for each service in haproxy.cfg on
LBVMs with new node: occne5-chandrasekhar-musti-k8s-node-4.

Worker node successfully added to cluster: occne5-chandrasekhar-musti
```

8. If there's a failure in the previous step, perform the following steps to rerun the script:
   a. Copy backup files to the original files:

      ```
      $ cp /var/occne/cluster/${OCCNE_CLUSTER}/cluster.tfvars $
      {OCCNE_CLUSTER}/cluster.tfvars
      $ cp /var/occne/cluster/${OCCNE_CLUSTER}/backupConfig/lbCtrlData.json
      lbvm/lbCtrlData.json
      # sudo cp /var/occne/cluster/${OCCNE_CLUSTER}/backupConfig/hosts /etc/
      hosts
      ```

   b. If you ran Podman commands before the failure, then drain the new pods before rerunning the script:

      ```
      $ kubectl drain --ignore-daemonsets <worker_node_hostname>
      ```

      For example:

      ```
      $ kubectl drain --ignore-daemonsets ${OCCNE_CLUSTER}-k8s-node-5
      ```

c. Rerun the `addWorkerNode.py` script:

```
$ scripts/addWorkerNode.py
```

d. After rerunning the script, uncordon the nodes:

```
$ kubectl uncordon <new node>
```

For example:

```
$ kubectl uncordon ${OCCNE_CLUSTER}-k8s-node-5
```

9. Verify the nodes, pods, and services:
   a. Verify if the new nodes are in Ready state by running the following command:

```
$ kubectl get nodes
```

   b. Verify if all pods are in the Running or Completed state by running the following command:

```
$ kubectl get pod -A -o wide
```

   c. The services are running and the services GUIs are reachable:

```
$ kubectl get svc -A
```

7.4.6 Removing a Kubernetes Worker Node

This section describes the procedure to remove a worker node from the CNE Kubernetes cluster after the original CNE installation. This procedure is used to remove a worker node that is unreachable (crashed or powered off), or that is up and running.

Procedure

> **Note**
>
> - This procedure is used to remove only one node at a time. If you want to remove multiple nodes, then perform this procedure on each node.
> - Removing multiple worker nodes can cause unwanted side effects such as increasing the overall load of your cluster. Therefore, before removing multiple nodes, make sure that there is enough capacity left in the cluster.
> - CNE requires a minimum of three worker nodes to properly run some of the common services such as, Opensearch, Bare Metal Rook Ceph cluster, and any daemonsets that require three or more replicas.
> - For a vCNE deployment, this procedure is used to remove only the last worker node in the Kubernetes. Therefore, refrain from using this procedure to remove any other worker node.

Remove a Kubernetes Worker Node in Openstack and VMware Deployment

> **Note**
>
> For any failure or successful run, the system maintains all terraform and pipeline output in the `/var/occne/cluster/${OCCNE_CLUSTER}/removeWrkNodeCapture-<mmddyyyy_hhmmss>.log` file.

1. Log in to a Bastion Host and verify the following:
   a. Run the following command to verify if all pods are the Running or Completed:

      ```
      $ kubectl get pod -A
      ```

      Sample output:

      ```
      NAMESPACE       NAME
      READY   STATUS                  RESTARTS         AGE
      cert-manager    occne-cert-manager-6dcffd5b9-jpzmt
      1/1     Running                 1 (3h17m ago)    4h56m
      cert-manager    occne-cert-manager-cainjector-5d6bccc77d-f4v56
      1/1     Running                 2 (3h15m ago)    3h48m
      cert-manager    occne-cert-manager-webhook-b7f4b7bdc-rg58k
      0/1     Completed               0                3h39m
      cert-manager    occne-cert-manager-webhook-b7f4b7bdc-tx7gz
      1/1     Running                 0                3h17m
      ...
      ```

   b. Run the following command to verify if the service LoadBalancer IPs are reachable and common service GUIs are running:

      ```
      $ kubectl get svc -A | grep LoadBalancer
      ```

      Sample output:

      ```
      occne-infra    occne-kibana                                    LoadBalancer
      10.233.36.151   10.75.180.113   80:31659/TCP
      4h57m
      occne-infra    occne-kube-prom-stack-grafana                   LoadBalancer
      10.233.63.254   10.75.180.136   80:32727/TCP
      4h56m
      occne-infra    occne-kube-prom-stack-kube-alertmanager         LoadBalancer
      10.233.32.135   10.75.180.204   80:30155/TCP
      4h56m
      occne-infra    occne-kube-prom-stack-kube-prometheus           LoadBalancer
      10.233.3.37     10.75.180.126   80:31964/TCP
      4h56m
      occne-infra    occne-promxy-apigw-nginx                        LoadBalancer
      10.233.42.250   10.75.180.4     80:30100/TCP
      4h56m
      occne-infra    occne-tracer-jaeger-query                       LoadBalancer
      10.233.4.43     10.75.180.69    80:32265/TCP,16687:30218/TCP
      4h56m
      ```

2. Navigate to the `/var/occne/cluster/${OCCNE_CLUSTER}/` directory:

   ```
   $ cd /var/occne/cluster/${OCCNE_CLUSTER}/
   ```

3. Open the `$OCCNE_CLUSTER/cluster.tfvars` file and decrement the value of the `number_of_k8s_nodes` field by 1:

   ```
   $ vi $OCCNE_CLUSTER/cluster.tfvars
   ```

   The following example shows the current value of `number_of_k8s_nodes` set to 6:

   ```
   ...
   # k8s nodes
   #
   number_of_k8s_nodes = 6
   ...
   ```

   The following example shows the value of `number_of_k8s_nodes` decremented by 1 to 5:

   ```
   ...
   # k8s nodes
   #
   number_of_k8s_nodes = 5
   ...
   ```

4. For OpenStack, perform this step to establish a connection between Bastion Host and OpenStack cloud. For VMware, skip this step and move to the next step.
   Source the `openrc.sh` file. Enter the Openstack username and password when prompted. The `openrc.sh` file sets the necessary environment variables for OpenStack. Once you source the file, ensure that the `openstack-cacert.pem` file exists in the same folder and the file is populated for TLS support:

   ```
   $ source openrc.sh
   ```

   The following block shows the username and password prompt displayed by the system:

   ```
   Please enter your OpenStack Username:
    Please enter your OpenStack Password as <username>:
    Please enter your OpenStack Domain:
   ```

5. Run the following command to get the list of nodes:

   ```
   $ kubectl get nodes -o wide | grep -v control-plane
   ```

   Sample output:

   ```
   NAME     STATUS   ROLES           AGE     VERSION   INTERNAL-IP       EXTERNAL-IP
   OS-IMAGE                 KERNEL-VERSION                   CONTAINER-RUNTIME
   occne6-my-cluster-k8s-node-1   Ready    <none>          6d23h   v1.25.6
   192.168.201.183   <none>        Oracle Linux Server 8.7
   5.4.17-2136.316.7.el8uek.x86_64   containerd://1.6.15
   occne6-my-cluster-k8s-node-2   Ready    <none>          6d23h   v1.25.6
   192.168.201.136   <none>        Oracle Linux Server 8.7
   5.4.17-2136.316.7.el8uek.x86_64   containerd://1.6.15
   occne6-my-cluster-k8s-node-3   Ready    <none>          6d23h   v1.25.6
   192.168.201.131   <none>        Oracle Linux Server 8.7
   5.4.17-2136.316.7.el8uek.x86_64   containerd://1.6.15
   occne6-my-cluster-k8s-node-4   Ready    <none>          6d23h   v1.25.6
   192.168.200.100   <none>        Oracle Linux Server 8.7
   5.4.17-2136.316.7.el8uek.x86_64   containerd://1.6.15
   ```

6. Run the following command to obtain the worker node IPs and verify if the worker node IPs match with the list obtained in Step 4:

   ```
   $ kubectl exec -it $(kubectl -n occne-infra get pods | grep occne-lb-
   controller-server) -n occne-infra -- /bin/bash -c "sqlite3 /data/sqlite/db/
   lbCtrlData.db 'SELECT * FROM nodeIps;'"
   ```

   Sample output:

   ```
   192.168.201.183
   192.168.201.136
   192.168.201.131
   192.168.200.100
   ```

7. Run the `removeWorkerNode.py` script.

   > **Note**
   >
   > The system backs up the `lbvm/lbCtrlData.json`, `cluster.tfvars`, `hosts.ini`, `terraform.tfstate`, and `/etc/hosts` files into the `/var/occne/cluster/${OCCNE_CLUSTER}/backUpConfig` directory. These files are backed up only once to back up the original files.

   ```
   $ ./scripts/removeWorkerNode.py
   ```

   Example for OpenStack deployment:

   ```
   $ ./scripts/removeWorkerNode.py
   ```

   Sample output:

   ```
   Starting removeWorkerNode instance for the last worker node.

    - Backing up configuration files...

    - Checking if cluster.tfvars matches with the terraform state...
      Succesfully checked the number_of_k8s_nodes parameter in the cluster.tfvars file.

    - Getting the IP address for the worker node to be deleted...
      Successfully gathered occne7-devansh-m-marwaha-k8s-node-4's ip: 192.168.200.105.

    - Draining  node - can take considerable time to complete...
      Successfully drained occne7-devansh-m-marwaha-k8s-node-4 node.

    - Removing  node from the cluster...
      Successfully removed occne7-devansh-m-marwaha-k8s-node-4 from the cluster.

    - Running terraform apply to update its state...
      Successfully applied Openstack terraform apply - check /var/occne/cluster/occne7-
   devansh-m-marwaha/removeWkrNodeCapture-11282023_090320.log for details

    - Updating /etc/hosts on all servers...
      Successfully updated file: /etc/hosts on all servers - check /var/occne/cluster/
   occne7-devansh-m-marwaha/removeWkrNodeCapture-11282023_090320.log for details.

    - Updating lbCtrlData.json file...
   ```

```
   Successfully updated file: /var/occne/cluster/occne7-devansh-m-marwaha/lbvm/
lbCtrlData.json.

 - Updating lb-controller-ctrl-data and lb-controller-master-ip configmap...
   Successfully created configmap lb-controller-ctrl-data.
   Successfully created configmap lb-controller-master-ip.

 - Deleting LB Controller POD: occne-lb-controller-server-fc869755-lm4hd to bind in
configmaps...
   Successfully restarted deployment occne-lb-controller-server.
   Waiting for occne-lb-controller-server deployment to return to Running status.
   Deployment "occne-lb-controller-server" successfully rolled out

 - Update servers from new occne-lb-controller pod...
   Successfully removed the node: occne7-devansh-m-marwaha-k8s-node-4 from server
list for each service in haproxy.cfg on LBVMs.

Worker node successfully removed from cluster: occne7-devansh-m-marwaha
```

Example for VMware deployment:

```
./scripts/removeWorkerNode.py
```

Sample output:

```
Starting removeWorkerNode instance for the last worker node.
   Successfully obtained index 3 from node occne5-chandrasekhar-musti-k8s-node-4.

 - Backing up configuration files...

 - Checking if cluster.tfvars matches with the terraform state...
   Succesfully checked the number_of_k8s_nodes parameter in the cluster.tfvars file.

 - Getting the IP address for the worker node to be deleted...
   Successfully gathered occne5-chandrasekhar-musti-k8s-node-4's ip: 192.168.1.15.

 - Draining  node - can take considerable time to complete...
   Successfully drained occne5-chandrasekhar-musti-k8s-node-4 node.

 - Removing  node from the cluster...
   Successfully removed occne5-chandrasekhar-musti-k8s-node-4 from the cluster.

 - Running terraform apply to update its state...
   Successfully applied VmWare terraform apply - check /var/occne/cluster/occne5-
chandrasekhar-musti/removeWkrNodeCapture-11282023_105101.log fodetails.

 - Updating /etc/hosts on all servers...
   Successfully updated file: /etc/hosts on all servers - check /var/occne/cluster/
occne5-chandrasekhar-musti/removeWkrNodeCapture-11282023_1051.log for details.

 - Updating lbCtrlData.json file...
   Successfully updated file: /var/occne/cluster/occne5-chandrasekhar-musti/lbvm/
lbCtrlData.json.

 - Updating lb-controller-ctrl-data and lb-controller-master-ip configmap...
   Successfully created configmap lb-controller-ctrl-data.
   Successfully created configmap lb-controller-master-ip.

 - Deleting LB Controller POD: occne-lb-controller-server-7b894fb6b5-5cr8g to bind
in configmaps...
   Successfully restarted deployment occne-lb-controller-server.
```

```
   Waiting for occne-lb-controller-server deployment to return to Running status.
   Deployment "occne-lb-controller-server" successfully rolled out

 - Update servers from new occne-lb-controller pod...
   Successfully removed the node: occne5-chandrasekhar-musti-k8s-node-4 from server
list for each service in haproxy.cfg on LBVMs.

Worker node successfully removed from cluster: occne5-chandrasekhar-musti
```

8. Verify if the specified node is removed:
   a. Run the following command to list the worker nodes:

      ```
      $ kubectl get nodes -o wide
      ```

      Sample output:

      ```
      NAME                              STATUS   ROLES                  AGE   VERSION
      INTERNAL-IP       EXTERNAL-IP   OS-IMAGE                KERNEL-
      VERSION                      CONTAINER-RUNTIME
      occne6-my-cluster-k8s-ctrl-1   Ready    control-plane,master   6d23h v1.25.6
      192.168.203.106   <none>        Oracle Linux Server 8.7
      5.4.17-2136.316.7.el8uek.x86_64   containerd://1.6.15
      occne6-my-cluster-k8s-ctrl-2   Ready    control-plane,master   6d23h v1.25.6
      192.168.202.122   <none>        Oracle Linux Server 8.7
      5.4.17-2136.316.7.el8uek.x86_64   containerd://1.6.15
      occne6-my-cluster-k8s-ctrl-3   Ready    control-plane,master   6d23h v1.25.6
      192.168.202.248   <none>        Oracle Linux Server 8.7
      5.4.17-2136.316.7.el8uek.x86_64   containerd://1.6.15
      occne6-my-cluster-k8s-node-1   Ready    <none>                 6d23h v1.25.6
      192.168.201.183   <none>        Oracle Linux Server 8.7
      5.4.17-2136.316.7.el8uek.x86_64   containerd://1.6.15
      occne6-my-cluster-k8s-node-2   Ready    <none>                 6d23h v1.25.6
      192.168.201.136   <none>        Oracle Linux Server 8.7
      5.4.17-2136.316.7.el8uek.x86_64   containerd://1.6.15
      occne6-my-cluster-k8s-node-3   Ready    <none>                 6d23h v1.25.6
      192.168.201.131   <none>        Oracle Linux Server 8.7
      5.4.17-2136.316.7.el8uek.x86_64   containerd://1.6.15
      ```

   b. Run the following command and check if the targeted worker node is removed:

      ```
      $ kubectl exec -it $(kubectl -n occne-infra get pods | grep occne-lb-
      controller-server) -n occne-infra -- /bin/bash -c "sqlite3 /data/
      sqlite/db/lbCtrlData.db 'SELECT * FROM nodeIps;'"
      ```

      Sample output:

      ```
      192.168.201.183
      192.168.201.136
      192.168.201.131
      ```

Remove a Kubernetes Worker Node in Bare Metal Deployment

> **Note**
>
> For any failure or successful run, the system maintains all pipeline outputs in the `/var/occne/cluster/${OCCNE_CLUSTER}/removeWrkNodeCapture-<mmddyyyy_hhmmss>.log` file. The system displays other outputs, messages, or errors directly on the terminal during the runtime of the script.

1. Log in to Bastion Host and verify if it's an active Bastion Host. If the Bastion Host isn't an active Bastion Host, then log in to another.

   Use the following command to check if the Bastion Host is an active Bastion Host:

   ```
   $ is_active_bastion
   ```

   The system displays the following output if the Bastion Host is an active Bastion Host:
   `IS active-bastion`

   The system displays the following output if the Bastion Host isn't an active Bastion Host:

   `NOT active-bastion`

2. Navigate to the `/var/occne/cluster/${OCCNE_CLUSTER}/artifacts/maintenance` directory:

   ```
   $ cd /var/occne/cluster/${OCCNE_CLUSTER}/artifacts/maintenance
   ```

3. Run the following command to get the list of nodes and identify the node to be removed:

   > **Note**
   > - The node that you want to remove must be present in the list and must be a worker node.
   > - The node can be in any status (Ready or NotReady).

   ```
   $ kubectl get nodes
   ```

   Sample output:

   ```
   NAME                              STATUS      ROLES           AGE
   VERSION
   k8s-master-1.test.us.oracle.com   Ready       control-plane   7d15h
   v1.25.6
   k8s-master-2.test.us.oracle.com   Ready       control-plane   7d15h
   v1.25.6
   k8s-master-3.test.us.oracle.com   Ready       control-plane   7d15h
   v1.25.6
   k8s-node-1.test.us.oracle.com     Ready       <none>          7d15h
   v1.25.6
   k8s-node-2.test.us.oracle.com     Ready       <none>          7d15h
   v1.25.6
   k8s-node-3.test.us.oracle.com     NotReady    <none>          7d15h
   v1.25.6
   k8s-node-4.test.us.oracle.com     Ready       <none>          7d15h
   v1.25.6
   ```

   For example, consider that the `k8s-node-3.test.us.oracle.com` node that is in `NotReady` state is removed in this procedure.

4. Run the following command to remove the targeted worker node:

> **Note**
>
> - Remove one node at a time. If you want to remove multiple worker nodes, remove them one by one.
> - Removing a node can take several minutes. Once the script is run, don't cancel it in between; wait for the script to complete.

```
$ ./removeBmWorkerNode.py -nn <NODE_FULL_NAME>
```

where, `<NODE_FULL_NAME>` is the name of the node that you want to remove (in this case, `k8s-3.test.us.oracle.com`).
For example:

```
$ ./removeBmWorkerNode.py -nn k8s-3.test.us.oracle.com
```

Sample output:

```
Beginning remove worker node: k8s-3.test.us.oracle.com
 - Backing up configuration files.
    - Backup folder: /var/occne/cluster/littlefinger/backupConfig
 - Checking if the rook-ceph toolbox deployment already exists.
    - rook-ceph toolbox deployment does not exist.
    - Creating the rook-ceph toolbox deployment.
       - rook-ceph toolbox created.
 - Waiting for Toolbox pod to be Running.
    - Waiting for toolbox pod in namespace rook-ceph to be in Running
state, current state:
    - ToolBox pod in namespace rook-ceph is now in Running state.
 - Getting the OSD Id.
    - Found OSD ID: 3, for worker node: k8s-3.test.us.oracle.com
 - Scaling down the rook-ceph-operator deployment.
    - rook-ceph-operator deployment scaled down.
 - Scaling down the OSD deployment.
    - rook-ceph-osd-3 deployment scaled down.
 - Purging OSD Id.
    - Unable to purge OSD.3 on attempt 1/12, retrying in 5 seconds...
    - OSD.3 purged.
    - OSD host k8s-3-test-us-oracle-com removed.
 - Deleting OSD deployment.
    - rook-ceph-osd-3 deleted.
 - Scaling up the rook-ceph-operator deployment.
    - rook-ceph-operator deployment scaled up.
 - Checking if the rook-ceph toolbox deployment exists.
    - Removing the rook-ceph toolbox deployment.
       - rook-ceph toolbox removed.
 - Removing BM worker node, could take considerable amount of time.
    - k8s-3.test.us.oracle.com is reachable
       - Removing... Please Wait. Do not cancel.
       - Removed. Check /var/occne/cluster/test/
removeBmWkrNodeCapture-06162023_155106.log for details.
 - Cleanup leftover rook-ceph-mon deployment
    - Found pod rook-ceph-mon-g-64b489c6ff-h4bcr, getting deployment...
```

```
        - Found deployment rook-ceph-mon-g, deleting...
    - rook-ceph-mon cleanup completed.
Worker node: k8s-3.test.us.oracle.com removed Successfully
```

5. Run the following command to verify if the targeted worker node is removed (in this case, `k8s-3.test.us.oracle.com`):

   ```
   $ kubectl get nodes
   ```

   Sample output:

   ```
   NAME                              STATUS   ROLES           AGE     VERSION
   k8s-master-1.test.us.oracle.com   Ready    control-plane   7d15h   v1.25.6
   k8s-master-2.test.us.oracle.com   Ready    control-plane   7d15h   v1.25.6
   k8s-master-3.test.us.oracle.com   Ready    control-plane   7d15h   v1.25.6
   k8s-node-1.test.us.oracle.com     Ready    <none>          7d15h   v1.25.6
   k8s-node-2.test.us.oracle.com     Ready    <none>          7d15h   v1.25.6
   k8s-node-4.test.us.oracle.com     Ready    <none>          7d15h   v1.25.6
   ```

   The `k8s-3.test.us.oracle.com` node deleted in step 5 is not present in the sample output.

6. Check if the cluster is healthy by checking the pods and services:

   ```
   $ kubectl get pods -A
   $ kubectl get svc -A
   ```

   There are many ways to check the health of a cluster. Checking the pods and services is one of the simple options. You can use your preferred way to check the cluster health.

7. Navigate to the `/var/occne/cluster/${OCCNE_CLUSTER}/` directory:

   ```
   $ cd /var/occne/cluster/${OCCNE_CLUSTER}/
   ```

8. Edit the `hosts.ini` file and remove the lines containing the details of the node that is removed (in this case, `k8s-3.test.us.oracle.com`):

   a. Open the `host.ini` file in edit mode:

      ```
      $ vi hosts.ini
      ```

   b. Remove the lines containing the details of the node that is removed, from both the hardware header [host_hp_gen_X]/[host_netra_X] and [kube-node] header.

      > **Note**
      >
      > As per the example considered in this procedure, the node that is removed is `k8s-node-3.test.us.oracle.com`. The lines to be removed are highlighted in bold in the following sample output.

      ```
      [host_hp_gen_X]/[host_netra_X] # Host header may vary depending on your hardware
      k8s-host-1.test.us.oracle.com ansible_host=179.1.5.2 hp_ilo=172.16.9.44
      mac=a2-27-3d-d3-b4-00 oam_host=10.75.216.13
      k8s-host-2.test.us.oracle.com ansible_host=179.1.5.3 hp_ilo=172.16.9.45 mac=4d-
      ```

```
d9-1a-e2-7e-e8 oam_host=10.75.216.14
k8s-host-3.test.us.oracle.com ansible_host=179.1.5.4 hp_ilo=172.16.9.46
mac=e1-15-b4-1d-32-10

k8s-node-1.test.us.oracle.com ansible_host=179.1.5.5 hp_ilo=172.16.9.47 mac=3b-
d2-2d-f6-1e-20
k8s-node-2.test.us.oracle.com ansible_host=179.1.5.6 hp_ilo=172.16.9.48
mac=a8-1a-37-b1-c0-dc
k8s-node-3.test.us.oracle.com ansible_host=179.1.5.7 hp_ilo=172.16.9.49 mac=a4-
be-2d-3f-21-f0
k8s-node-4.test.us.oracle.com ansible_host=179.1.5.8 hp_ilo=172.16.9.35 mac=3a-
d9-2c-e6-35-18
.
.
.
[kube-node]
k8s-node-1.test.us.oracle.com
k8s-node-2.test.us.oracle.com
k8s-node-3.test.us.oracle.com
k8s-node-4.test.us.oracle.com
```

c. Save the file and exit.

7.4.7 Adding a New External Network

This section provides the procedure to add a new external network that applications can use to talk to external clients, by adding a Peer Address Pool (PAP) in a virtualized CNE (vCNE) and Bare Metal, after installation in CNE.

The cluster must be in good working condition. All pods and services must be running and no existing LBVMs must be in a FAILED state. Use the following command to run a full cluster test before starting the procedure:

```
OCCNE_STAGES=(TEST) pipeline.sh
```

7.4.7.1 Adding a New External Network in vCNE

The following procedure provides the steps to add a single Peer Address Pool (PAP) in vCNE.

Each time the script is run, the system creates a log file with a timestamp. The format of the log file is, `addpapCapture-<mmddyyyy_hhmmss>.log`. For example, `addPapCapture-09172021_000823.log`. The log includes the output from the Terraform and the pipeline call to configure the new LBVMs.

Each time the script is run, the system creates a new directory, `addPapSave-<mmddyyyy-hhmmss>`. The following files from the `/var/occne/cluster/<cluster_name>` directory are saved in the `addPapSave-<mmddyyyy-hhmmss>` directory:

- lbvm/lbCtrlData.json
- metallb.auto.tfvars
- mb_resources.yaml
- terraform.tfstate
- hosts.ini
- cluster.tfvars

Prerequisites

1. On an OpenStack deployment, run the following steps to source the OpenStack environment file. This step is not required for a VMware deployment as the credential settings are derived automatically.
   a. Log in to Bastion Host and change the directory to the cluster directory:

      ```
      $ cd /var/occne/cluster/${OCCNE_CLUSTER}
      ```

   b. Source the OpenStack environment file:

      ```
      $ source openrc.sh
      ```

Procedure

1. Update the `/var/occne/cluster/<cluster_name>/<cluster_name>/cluster.tfvars` file to include the new pool that is required.

   For more information about the following, see *Oracle Communications Cloud Native Core, Cloud Native Environment Installation, Upgrade, and Fault Recovery Guide*:

   - Updating the `cluster.tfvars` file.
   - Adding a Peer Address Pool (PAP) object to the existing `cluster.tfvars` file.
   - Adding the additional PAP name to the `occne_metallb_peer_addr_pool_names` list.
   - Adding the additional pool objects to the `occne_metallb_list` object.

2. Run the `addPeerAddrPools.py` script:

   The script is located on the Bastion Host at `/var/occne/cluster/<cluster_name>/scripts`. The script is fully automated. When the script completes, the new LBVM pair is fully integrated into the deployment. The script doesn't require any parameter and performs all the steps required to complete the addition of the new Peer Address Pool (PAP) including a validation at the end. All output echos back to the terminal.

   It is recommended to perform the following before running the script:

   - Start a terminal output capture process to ensure that all data output is captured (for example, the script).
   - Run `tail -f` on the current `addPapCapture` log file to see how the Terraform and pipeline is running.

   a. Run the following commands to run the `addPeerAddrPools.py` script:

      ```
      $ cd /var/occne/cluster/<cluster_name>
      $ ./scripts/addPeerAddrPools.py
      ```

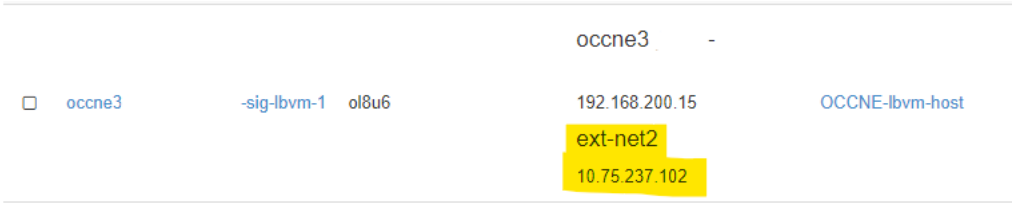
      Sample output to show a new PAP, "sig", added. The cluster_name in the sample output is considered as occne1-cluster:

      ```
      - Capturing current lbCtrlData.json data...

       - Generating new metallb.auto.tfvars file...
         Successfully generated the metallb.auto.tfvars file.

       - Executing terraform apply to generate new LBVM and port resources...
         . terraform apply attempt 1
         Successful execution of terraform apply -auto-approve -compact-
      ```

```
warnings -var-file=/var/occne/cluster/occne1-cluster/occne1-cluster/
cluster.tfvars

 - Generating new mb_resources.yaml file...
   Successfully updated the /var/occne/cluster/occne1-cluster/
mb_resources.yaml file.

 - Applying new mb_resources.yaml file...
   Successfully applied mb_resources.yaml file.

 - Generating new lbvm/lbCtrlData.json file (can take longer due to
number of existing LBVM pairs)...
   Successfully updated the lbvm/lbCtrlData.json file.

 - Generating new pool list...

 - Updating hosts.ini with new LBVMs...
   Successfully updated hosts.ini.

 - Running pipeline to config the new LBVMs (can take 10 minutes or
more)...
   Successfully updated the new LBVM config.

 - Updating /etc/hosts with new LBVMs...
   Successfully updated /etc/hosts

 - Generating new config maps for lb-controller...
   . Updating config map: lb-controller-ctrl-data from file: /var/occne/
cluster/occne1-cluster/lbvm/lbCtrlData.json
     Successfully applied new config map: lb-controller-ctrl-data from
file: /var/occne/occne1-cluster/lbvm/lbCtrlData.json
     Pausing for 30 seconds to allow completion of config map: lb-
controller-ctrl-data update from file: /var/occne/cluster/occne1-
cluster/lbvm/lbCtrlData.json.
   . Updating config map: lb-controller-master-ip from file: /etc/hosts
     Successfully applied new config map: lb-controller-master-ip from
file: /etc/hosts
     Pausing for 30 seconds to allow completion of config map: lb-
controller-master-ip update from file: /etc/hosts.
   . Updating config map: lb-controller-mb-resources from file: /var/
occne/cluster/occne1-cluster/mb_resources.yaml
     Successfully applied new config map: lb-controller-mb-resources
from file: /var/occne/cluster/occne1-cluster/mb_resources.yaml
     Pausing for 30 seconds to allow completion of config map: lb-
controller-mb-resources update from file: /var/occne/cluster/occne1-
cluster/mb_resources.yaml.
     Successfully restarted deployment: occne-lb-controller-server
     Waiting for occne-lb-controller-server deployment to return to
Running status.
     Deployment "occne-lb-controller-server" successfully rolled out

 - Generating the LBVM HAProxy templates...
   Successfully updated the templates files in the LBVMs.

 - Updating the lb-controller Db...
   Successfully added new pools to the lb_controller Db.
```

```
 - Restarting occne-egress-controller pods...
     Successfully restarted daemonset: occne-egress-controller
     Waiting for occne-egress-controller daemonset to return to Running
status.
   Daemon set "occne-egress-controller" successfully rolled out

 - Restarting occne-metallb-controller pod...
     Successfully restarted deployment: occne-metallb-controller
     Waiting for occne-metallb-controller deployment to return to
Running status.
   Deployment "occne-metallb-controller" successfully rolled out

 - Restarting occne-metallb-speaker pods...
     Successfully restarted daemonset: occne-metallb-speaker
     Waiting for occne-metallb-speaker daemonset to return to Running
status.
   Daemon set "occne-metallb-speaker" successfully rolled out

 - Validating LBVM configuration
   . Validating IPAddressPool CRD for pool: sig
     IPAddressPool CRD validated successfully for pool: sig
   . Validating BGPAdvertisements CRD for pool: sig
     BGPAdvertisements bgpadvertisement1 validated successfully for
pool: sig
   . Validating ACTIVE LBVM for pool: sig
     ACTIVE LBVM haproxy.cfg validation successful for pool: sig
   . Validating LB Controller Db for pool: tme
     LB Controller Db validation successful for pool: sig

Successfully added new LBVMs and ports for the new Peer Address Pool.
```

b. Log in to the OpenStack GUI and validate if the new pool LBVMs are created and the new external egress port is attached to the Active LBVM.



7.4.7.2 Adding a New External Network in Bare Metal

This section describes the procedure to add an additional network to an existing bare metal installation of CNE.

> **Note**
>
> Run all commands in this procedure from the Bastion Host.

1. Navigate to the `/var/occne/cluster/${OCCNE_CLUSTER}` directory.
2. Edit the `mb_resources.yaml` file to reflect the new network configuration. Add the following new address pool configuration as per the directions and example provided in the "Populate the MetalLB Configuration" section of *Oracle Communications Cloud Native Core, Cloud Native Environment Installation, Upgrade, and Fault Recovery Guide*:
   - Add the new IPAddressPool section.
   - Add the new pool's name to the BGPAdvertisement description.
3. Run the following command to apply the new metalLB configuration file:

   ```
   $ kubectl apply -f mb_resources.yaml
   ```

4. Perfrom the following steps to delete the address pool:
   a. Remove the address pool from the `IPAddressPool` section of the `mb_resources.yaml` file.
   b. Remove the address pool's name from the `BGPAdvertisement` description.
   c. Run the following command to effect the removal of the advertisement:

      ```
      kubectl apply -f mb_resources.yaml
      ```

   d. Run the following command to delete the now orphaned IPAddressPool resource description:

      ```
      kubectl delete IPAddressPool <pool-name>  -n <namespace>
      ```

7.4.8 Renewing the Platform Service Mesh Root Certificate

This section describes the procedure to renew the root certificate used by the platform service mesh to generate certificates for Mutual Transport Layer Security (mTLS) communication when the Intermediate Certification Authority (ICA) issuer type is used.

Prerequisites

1. The CNE platform service mesh must have been configured to use the Intermediate CA issuer type.
2. A network function configured with the platform service mesh, commonly istio, must be available.

Procedure

1. Renew the root CA certificate
   a. Obtain a new certificate and key

      Generate a new signing certificate and key from the external CA that generated the original root CA certificate and key. The generated certificate and key values must be base64 encoded.

      Check the Certificate Authority documentation for generating the required certificate and key.

   b. Set required environment variables

Set the OCCNE_NEW_CA_CERTIFICATE and OCCNE_NEW_CA_KEY as follows:

```
$ OCCNE_NEW_CA_CERTIFICATE=<base64 encoded root CA certificate>
$ OCCNE_NEW_CA_KEY=<base64 encoded root CA key>
```

> **Note**
>
> Ensure that for each certificate and key is encoded with a base64 value.

Example:

```
OCCNE_NEW_CA_CERTIFICATE=LS0tLS1CRUdJTiBDRVJUSUZJQ0F...0tRU5EIENFUlRJRkl
DQVRFLS0tLS0K
OCCNE_NEW_CA_KEY=LS0tLS1CRUdJTiBSU0EgUFJJVkFURSBLRVk...CBSU0EgUFJJVkFURS
BLRVktLS0tLQo
```

c. Renew the root certificate

Run the following script to renew the root certificate:

```
$ /var/occne/cluster/${OCCNE_CLUSTER}/artifacts/
renew_istio_root_certificate.sh ${OCCNE_NEW_CA_CERTIFICATE} $
{OCCNE_NEW_CA_KEY}
```

2. Verify that root certificate is renewed

a. Check that the Kubernetes Secret containing the certificate and key is updated

Run the following command to retrieve the *istio-ca* secret:

```
$ kubectl -n istio-system get secret istio-ca -o
jsonpath="{.data['tls\.crt']}"
```

Verify that the retrieved certificate and key match with the ones provided in the previous step.

b. Verify that the service mesh is configured to use the new certificate:

i. Run the following command to retrieve the Istio configured root certificate:

```
$ kubectl -n istio-system get cm istio-ca-root-cert -o
jsonpath="{.data['root-cert\.pem']}" | base64 -w0; echo
```

ii. Verify that the retrieved certificate and key match with the ones provided in the previous step.

c. Verify that NF pods are aware of the new root certificate.

Check that NF pod is updated with new root certificate value. It must be same as $ {OCCNE_NEW_CA_CERTIFICATE}. Repeat the step for other NF pods as follows:

```
# set the istio binary path
$ ISTIO_BIN=/var/occne/cluster/${OCCNE_CLUSTER}/artifacts/istio-$
(cat /var/occne/cluster/${OCCNE_CLUSTER}/artifacts/
CFG_container_images.txt | grep istio/pilot | cut -d: -f 2)/bin/istioctl
```

```
# set the pod name
$ POD_NAME=<nf-pod-name>

# set the namespace
$ NAME_SPACE=<nf-namespace>

# execute istio proxy-config command to get the root certificate of the
NF pod
$ ${ISTIO_BIN} proxy-config secret ${POD_NAME} -n ${NAME_SPACE} -o json
| grep -zoP 'trustedCa.*\n.*inlineBytes.*' | tail -n 1 | awk -d:
'{ print $2 }' | tr -d \"
```

7.4.9 Performing an etcd Data Backup

This section describes the procedure to back up the etcd database.

A backup copy of the etcd database is required to restore the CNE Kubernetes cluster in case all controller nodes fail simultaneously. You must back up the etcd data in the following scenarios:

- After a 5G NF is installed, uninstalled, or upgraded
- Before and after CNE is upgraded

This way the backed-up etcd data can be used to recover the CNE Kubernetes cluster during disaster scenarios. The etcd data is consistent across all controller nodes within a cluster. Therefore it is sufficient to take a backup from any one of the active Kubernetes controller node.

Procedure

1. Find Kubernetes controller hostname: Run the following command to get the names of Kubernetes controller nodes. The backup must be taken from any one of the controller nodes that is in **Ready** state.

   ```
   $ kubectl get nodes
   ```

2. Run the etcd-backup script:
   a. On the Bastion Host, switch to the `/var/occne/cluster/${OCCNE_CLUSTER}/artifacts` directory:

      ```
      $ cd /var/occne/cluster/${OCCNE_CLUSTER}/artifacts
      ```

   b. Run the `etcd_backup.sh` script:

      ```
      $ ./etcd_backup.sh
      ```

      On running the script, the system prompts you to enter the **k8s-ctrl node** name. Enter the name of the controller node from which you want to back up the etcd data.

      > **Note**
      >
      > The script keeps only three backup snapshots in the PVC and automatically deletes the older snapshots.

7.5 Updating OpenStack Credentials

This section describes the procedure to update the OpenStack credentials for vCNE.

Prerequisites

1. You must have access to active Bastion Host of the cluster.
2. All commands in this procedure must be run from the active CNE Bastion Host.
3. You must have knowledge of kubectl and handling base64 encoded and decoded strings.

Modifying Password for Cinder Access

Kubernetes uses the *cloud-config* secret when interacting with OpenStack Cinder to acquire persistent storage for applications. The following steps describe how to update this secret to include the new password.

1. Run the following command to decode and save the current *cloud-config* secret configurations in a temporary file:

   ```
   $ kubectl get secret cloud-config -n kube-system -o
   jsonpath="{.data.cloud\.conf}" | base64 --decode > /tmp/
   decoded_cloud_config.txt
   ```

2. Run the following command to open the temporary file in vi editor and update the username and password fields in the file with required values:

   ```
   $ vi /tmp/decoded_cloud_config.txt
   ```

   Sample to edit the username and password:

   ```
   username="new_username"
   password="new_password"
   ```

   After updating the credentials, save and exit from the file.

3. Run the following command to re-encode the `cloud-config` secret in Base64. Save the encoded output to use it in the following step.

   ```
   $ cat /tmp/decoded_cloud_config.txt | base64 -w0
   ```

4. Run the following command to edit the `cloud-config` Kubernetes secret:

   ```
   $ kubectl edit secret cloud-config -n kube-system
   ```

   Refer to the following sample to edit the `cloud-config` Kubernetes secret:

> **Note**
>
> Replace <encoded-output> in the following sample with the encoded output that you saved in the previous step.

```
# Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving
this file will be
# reopened with the relevant failures.
#
apiVersion: v1
data:
 cloud.conf: <encoded-output>
kind: Secret
metadata:
 annotations:
 kubectl.kubernetes.io/last-applied-configuration: |
 {"apiVersion":"v1","data":{"cloud.conf":"<encoded-
output>"},"kind":"Secret","metadata":{"annotations":{},"name":"cloud-
config","namespace":"kube-system"}}
 creationTimestamp: "2022-01-12T02:34:52Z"
 name: cloud-config
 namespace: kube-system
 resourceVersion: "2225"
 uid: 0994b024-6a4d-41cf-904c
type: Opaque
```

Save the changes and exit the editor.

5. Run the following command to remove the temporary file:

```
$ rm /tmp/decoded_cloud_config.txt
```

Modifying Password for OpenStack Cloud Controller Access

Kubernetes uses the `external-openstack-cloud-config` secret when interacting with the OpenStack Controller. The following steps describe the procedure to update the secret to include the new credentials.

1. Run the following command to decode the current `external-openstack-cloud-config` secret configurations in a temporary file:

```
$ kubectl get secret external-openstack-cloud-config -n kube-system -o
jsonpath="{.data.cloud\.conf}" | base64 --decode > /tmp/
decoded_external_openstack_cloud_config.txt
```

2. Run the following command to open the temporary file in vi editor and update the username and password fields in the file with required values:

```
$ vi /tmp/decoded_external_openstack_cloud_config.txt
```

Sample to edit the username and password:

```
username="new_username"
password="new_password"
```

After updating the credentials, save and exit from the file.

3. Run the following command to re-encode `external-openstack-cloud-config` in Base64. Save the encoded output to use it in the following step.

```
$ cat /tmp/decoded_external_openstack_cloud_config.txt | base64 -w0
```

4. Run the following command to edit the Kubernetes Secret named, `external-openstack-cloud-config`:

```
$ kubectl edit secret external-openstack-cloud-config -n kube-system
```

Refer to the following sample to edit the `external-openstack-cloud-config` Kubernetes Secret with the new encoded value:

> **Note**
>
> - Replace `<encoded-output>` in the following sample with the encoded output that you saved in the previous step.
> - An empty file aborts the edit. If an error occurs while saving, the file reopens with the relevant failures.

```
apiVersion: v1
data:
  ca.cert:
  cloud.conf:<encoded-output>
kind: Secret
metadata:
  annotations:
    kubectl.kubernetes.io/last-applied-configuration: |
      {"apiVersion":"v1","data":{"ca.cert":" ","cloud.conf":"<encoded-
output>"},"kind":"Secret","metadata":{"annotations":{},"name":"external-
openstack-cloud-config","namespace":"kube-system"}}
  creationTimestamp: "2022-07-21T17:05:26Z"
  name: external-openstack-cloud-config
  namespace: kube-system
  resourceVersion: "16"
  uid: 9c18f914-9c78-401d-ae79
type: Opaque
```

Save the changes and exit the editor.

5. Run the following command to remove the temporary file:

```
$ rm /tmp/decoded_external_openstack_cloud_config.txt
```

Restarting Affected Pods to Use the New Password

Services that use OpenStack credentials must be restarted to use the new password. The following steps describe how to restart the services.

> **Note**
>
> Before restarting the services, verify that all the affected Kubernetes resources to be restarted are in healthy state.

1. Perform the following steps to restart Cinder Container Storage Interface (Cinder CSI) controller plugin:
   a. Run the following command to restart Cinder Container Storage Interface (Cinder CSI) deployment:

      ```
      $ kubectl rollout restart deployment csi-cinder-controllerplugin -n
      kube-system
      ```

      Sample output:

      ```
      deployment.apps/csi-cinder-controllerplugin restarted
      ```

   b. Run the following command to get the pod and verify if it is running:

      ```
      $ kubectl get pods -l app=csi-cinder-controllerplugin -n kube-system
      ```

      Sample output:

      ```
      NAME                                          READY   STATUS
      RESTARTS   AGE
      csi-cinder-controllerplugin-7c9457c4f8-88sbt   6/6     Running
      0          19m
      ```

   c. [Optional]: If the pod is not up or if the pod is in the `crashloop` state, get the logs from the `cinder-csi-plugin` container inside the `csi-cinder-controller` pod using labels and validate the logs for more information:

      ```
      $ kubectl logs -l app=csi-cinder-controllerplugin -c cinder-csi-plugin -
      n kube-system
      ```

      Sample output to show a successful log retrieval:

      ```
      I0904 21:36:09.162886       1 server.go:106] Listening for connections
      on address: &net.UnixAddr{Name:"/csi/csi.sock", Net:"unix"}
      ```

      Sample output to show a log retrieval failure:

      ```
      W0904 21:34:34.252515       1 main.go:105] Failed to
      GetOpenStackProvider: Authentication failed
      ```

2. Perform the following steps to restart Cinder Container Storage Interface (Cinder CSI) nodeplugin daemonset:
   a. Run the following command to restart Cinder Container Storage Interface (Cinder CSI) nodeplugin daemonset:

      ```
      $ kubectl rollout restart -n kube-system daemonset csi-cinder-nodeplugin
      ```

      Sample output:

      ```
      daemonset.apps/csi-cinder-nodeplugin restarted
      ```

   b. Run the following command to get the pod and verify if it is running:

      ```
      $ kubectl get pods -l app=csi-cinder-nodeplugin -n kube-system
      ```

      Sample output:

      ```
      NAME                          READY   STATUS    RESTARTS   AGE
      csi-cinder-nodeplugin-pqqww   3/3     Running   0          3d19h
      csi-cinder-nodeplugin-vld6m   3/3     Running   0          3d19h
      csi-cinder-nodeplugin-xg2kj   3/3     Running   0          3d19h
      csi-cinder-nodeplugin-z5vck   3/3     Running   0          3d19h
      csi-cinder-nodeplugin-z5vck   3/3     Running   0          3d19h
      csi-cinder-nodeplugin-z5vck   3/3     Running   0          3d19h
      csi-cinder-nodeplugin-z5vck   3/3     Running   0          3d19h
      ```

   c. [Optional]: If the pod is not up or if the pod is in the `crashloop` state, verify the logs for more information
3. Run the following command to restart the OpenStack cloud controller daemonset:
   a. Run the following command to restart the OpenStack cloud controller daemonset:

      ```
      $ kubectl rollout restart -n kube-system daemonset openstack-cloud-
      controller-manager
      ```

      Sample output:

      ```
      daemonset.apps/openstack-cloud-controller-manager restarted
      ```

   b. Run the following command to get the pod and verify if it is running:

      ```
      $ kubectl get pods -l k8s-app=openstack-cloud-controller-manager -n
      kube-system
      ```

      Sample output:

      ```
      NAME                                       READY   STATUS    RESTARTS
      AGE
      openstack-cloud-controller-manager-qtfff   1/1     Running   0
      38m
      openstack-cloud-controller-manager-sn2pg   1/1     Running   0
      38m
      ```

```
openstack-cloud-controller-manager-w5dcv   1/1     Running   0
38m
```

c. [Optional]: If the pod is not up, or is in the `crashloop` state, verify the logs for more information.

Changing Inventory File

When you perform the steps to modify password for Cinder access and modify password for OpenStack cloud controller access, you modify the Kubernetes secrets to contain the new credentials. However, running the pipeline (for example, performing a standard upgrade or adding a new node to the cluster) takes the current credentials stored in the `occne.ini` file, causing the changes to be overridden. Therefore, it is important to update the `occne.ini` file with the new credentials.

1. Navigate to the cluster directory:

   ```
   $ cd /var/occne/cluster/${OCCNE_CLUSTER}/
   ```

2. Open the `occne.ini` file:

   ```
   $ vi occne.ini
   ```

3. Update External Openstack credentials (both username and password) as shown below:

   ```
   external_openstack_username = USER
   external_openstack_password = PASSWORD
   ```

4. Update Cinder credentials (both username and password) as shown below:

   ```
   cinder_username = USER
   cinder_password = PASSWORD
   ```

Updating Credentials for lb-controller-user

> **Note**
>
> Run all the commands in this section from Bastion Host.

1. Run the following commands to update lb-controller-user credentials:

   ```
   $ echo -n "<Username>" | base64 -w0 | xargs -I{} kubectl -n occne-infra
   patch --type=merge secret lb-controller-user --patch '{"data":
   {"USERNAME":"{}"}}'


   $ echo -n "<Password>" | base64 -w0 | xargs -I{} kubectl -n occne-infra
   patch --type=merge secret lb-controller-user --patch '{"data":
   {"PASSWORD":"{}"}}'
   ```

   where:

   - <Username>, is the new OpenStack username.

- <Password>, is the new OpenSatck password.

2. Run the following command to restart lb-controller-server to use the new credentials:

   ```
   $ kubectl rollout restart deployment occne-lb-controller-server -n occne-infra
   ```

3. Wait until the `lb-controller` restarts and run the following command to get the lb-controller pod status using labels. Ensure that only one pod is in the Running status:

   ```
   $ kubectl get pods -l app=lb-controller -n occne-infra
   ```

   Sample output:

   ```
   NAME                                          READY   STATUS    RESTARTS
   AGE
   occne-lb-controller-server-74fd947c7c-vtw2v   1/1     Running   0
   50s
   ```

4. Validate the new credentials by printing the username and password directly from the new pod's environment variables:

   ```
   $ kubectl exec -it $(kubectl get pod -n occne-infra | grep lb-controller-server | cut -d " " -f1) -n occne-infra -- bash -c "echo -n \$USERNAME"
   $ kubectl exec -it $(kubectl get pod -n occne-infra | grep lb-controller-server | cut -d " " -f1) -n occne-infra -- bash -c "echo -n \$PASSWORD"
   ```

7.6 Updating the Guest or Host OS

You must update the host OS (for Bare Metal installations) or guest OS (for virtualized installations) periodically so that CNE has the latest Oracle Linux software. If the CNE is not upgraded recently, or there are known security patches then perform an update by referring to the upgrade procedures in *Oracle Communications Cloud Native Core, Cloud Native Environment Installation, Upgrade, and Fault Recovery Guide*.

7.7 CNE Grafana Dashboards

Grafana is an observability tool available as open source and enterprise versions. Grafana supports number of data sources such as Prometheus from where it can read data for analytics. You can find the official list of supported data sources at Grafana Datasources.

CNE offers the following default Grafana dashboards:

- CNE Kubernetes dashboard
- CNE Prometheus dashboard
- CNE logging dashboard
- CNE persistent storage dashboard (only for Bare Metal)

> **Note**
>
> The Grafana dashboards provisioned by CNE are read-only. Refrain from updating or modifying these default dashboards.

You can clone these dashboards to customize them as per your requirement and save the customized dashboards in JSON format. This section provides details about the features offered by the open source Grafana version to add the required observability framework to CNE.

7.7.1 Accessing Grafana Interface

This section provides the procedure to access Grafana web interface.

1. Perform the following steps to get the Load Balancer IP address and port number for accessing the Grafana web interface:
   a. Run the following command to get the Load Balancer IP address of the Grafana service:

      ```
      $ export GRAFANA_LOADBALANCER_IP=$(kubectl get services occne-kube-prom-stack-grafana --namespace occne-infra -o jsonpath="{.status.loadBalancer.ingress[*].ip}")
      ```

   b. Run the following command to get the LoadBalancer port number of the Grafana service:

      ```
      $ export GRAFANA_LOADBALANCER_PORT=$(kubectl get services occne-kube-prom-stack-grafana --namespace occne-infra -o jsonpath="{.spec.ports[*].port}")
      ```

   c. Run the following command to get the complete URL for accessing Grafana in an external browser:

      ```
      $ echo http://$GRAFANA_LOADBALANCER_IP:$GRAFANA_LOADBALANCER_PORT/$OCCNE_CLUSTER/grafana
      ```

      Sample output:

      ```
      http://10.75.225.60:80/mycne-cluster/grafana
      ```

2. Use the URL obtained in the previous step (in this case, http://10.75.225.60:80/mycne-cluster/grafana) to access the Grafana home page.
3. Click **Downloads** and select **Browse**.
4. Expand the CNE folder to view the CNE dashboards.

> **Note**
>
> CNE doesn't support user access management on Grafana.

7.7.2 Cloning a Grafana Dashboard

This section describes the procedure to clone a Grafana dashboard.

1. Open the dashboard that you want to clone.
2. Click the **Share dashboard or panel** icon next to the dashboard name.
3. Select **Export** and click **Save to file** to save the dashboard in JSON format in your local system.
4. Perform the following steps to import the saved dashboard to Grafana:
   a. Click **Dashboards** and select **Import**.
   b. Click **Upload JSON file** and select the dashboard that you saved in step 3.
   c. Change the name and UID of the dashboard.
   You have cloned the dashboard successfully. You can now use the cloned dashboard to customize the options as per your requirement.

7.7.3 Restoring a Grafana Dashboard

The default Grafana dashboards provided by CNE are stored as configmap in the CNE cluster and artifact directory to restore them to their default state. This section describes the procedure to restore a Grafana dashboard.

> **Note**
>
> - This procedure is used to restore the dashboards to the default state (that is, the default dashboards provided by CNE).
> - When you restore the dashboards, you lose all the customizations that you made on the dashboards. You can't use this procedure to restore the customizations that you made on top of the CNE default dashboards.
> - You can't use this procedure to restore other Grafana dashboards that you created.

1. Navigate to the `occne-grafana-dashboard` directory:

   ```
   $ cd /var/occne/cluster/${OCCNE_CLUSTER}/artifacts/occne-grafana-dashboard
   ```

2. Run the following command to restore all the dashboards present in the `occne-grafana-dashboard` directory to its default state. The command uses the YAML files of the dashboards in the directory to restore them.

   ```
   $ kubectl -n occne-infra apply -R -f occne-grafana-dashboard
   ```

   You can also restore a specific dashboard by providing a specific YAML file name in the command. For example, you can use the following command to restore only the CNE Kubernetes dashboard:

   ```
   $ kubectl -n occne-infra apply -f occne-grafana-dashboard/occne-k8s-cluster-dashboard.yaml
   ```

7.8 Managing 5G NFs

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

7.8.1 Installing an NF

This section describes the procedure to install an NF in the CNE Kubernetes cluster.

Prerequisites

- Load container images and Helm charts onto Central Server repositories.
  Container and Helm repositories are created on a Central Server for easy CNE deployment at multiple customer sites. These repositories store all of the container images and Helm charts required to install CNE. When necessary, Helm pulls container images and Helm charts to the central server repositories on the local CNE Bastion Hosts. Similarly, NF installation uses Helm so that the container images and Helm charts needed to install NFs are loaded onto the same Central Server repositories. This procedure assumes that all container images and Helm charts required to install the NF are already loaded onto the Central Server repositories.
- Determine the NF deployment parameters
  The following values determine the NF's identity and where it is deployed. These values are used in the following procedure:

Table 7-15 NF Deployment Parameters

Parameters	Value	Description
nf-namespace	Any valid namespace name	The namespace where you want to install the NF. Typically each NF is installed in its own namespace.
nf-deployment-name	Any valid Kubernetes deployment name	The name that this NF instance is known to the Kubernetes.

Load NF artifacts onto Bastion Host repositories

All the steps in this section are run on the CNE Bastion Host where the NF installation happens.

Load NF container images

1. Create a file **container_images.txt** listing the Container images and tags as required by the NF:

   ```
   <image-name>:<release>
   ```

   Example:

   ```
   busybox:1.29.0
   ```

2. Run the following command to load the container images into the CNE Container registry:

   ```
   $ retrieve_container_images.sh <external-container-repo-name>:<external-container-repo-port> ${HOSTNAME}:5000 < container_images.txt
   ```

Example:

```
$ retrieve_container_images.sh mycentralrepo:5000 ${HOSTNAME%%.*}:5000 <
container_images.txt
```

Load NF Helm charts

1. Create a file **helm_charts.txt** listing the Helm chart and version:

   ```
   <external-helm-repo-name>/<chart-name> <chart-version>
   ```

   Example:

   ```
   mycentralhelmrepo/busybox 1.33.0
   ```

2. Run the following command to load the charts into the CNE Helm chart repository:

   ```
   $ retrieve_helm.sh /var/www/html/occne/charts http://<external-helm-repo-
   name>/occne/charts [helm_executable_full_path_if_not_default] <
   helm_charts.txt
   ```

   Example:

   ```
   $ retrieve_helm.sh /var/www/html/occne/charts http://mycentralrepo/occne/
   charts < helm_charts.txt
   ```

Install the NF

1. On the Bastion Host, create a YAML file named `<nf-short-name>-values.yaml` to contain the values to be passed to the NF Helm chart.
2. Add NF-specific values to file
   See the NF installation instructions to understand which keys and values must be included in the values file.
3. Additional NF configuration
   Before installing the NF, see the installation instructions to understand the requirements of additional NF configurations along with Helm chart values.
4. Run the following command to install the NF:

   ```
   $ helm install --namespace <nf-namespace> --create-namespace -f <nf-short-
   name>-values.yaml <nf-deployment-name> <chart-or-chart-location>
   ```

7.8.2 Upgrading an NF

This section describes the procedure to upgrade a 5G network function that was previously installed in the CNE Kubernetes cluster.

Prerequisites

Load container images and Helm charts onto Central Server repositories.

Container and Helm repositories are created on a Central Server for easy CNE deployment at multiple customer sites. These repositories store all of the container images and Helm charts required to install CNE. When necessary, Helm pulls container images and Helm charts to the

central server repositories on the local CNE Bastion Hosts. Similarly, Network Function (NF) installation uses Helm so that the container images and Helm charts needed to install NFs are loaded onto the same Central Server repositories. This procedure assumes that all container images and Helm charts required to install the NF are already loaded onto the Central Server repositories.

Procedure

Perform the following steps to upgrade an NF. All commands must be run from the Bastion Host.

1. Load NF artifacts onto Bastion Host repositories.
2. Upgrade the NF.

Load NF artifacts onto Bastion Host repositories

All the steps in this section are run on the CNE Bastion Host where the NF installation happens.

Load NF container images

1. Create a file **container_images.txt** listing the Container images and tags as required by the NF:

   ```
   <image-name>:<release>
   ```

   Example:

   ```
   busybox:1.29.0
   ```

2. Run the following command to load the container images into the CNE Container registry:

   ```
   $ retrieve_container_images.sh <external-container-repo-name>:<external-container-repo-port> ${HOSTNAME%%.*}:5000 < container_images.txt
   ```

   Example:

   ```
   $ retrieve_container_images.sh mycentralrepo:5000 ${HOSTNAME%%.*}:5000 < container_images.txt
   ```

Load NF Helm charts

1. Create a file **helm_charts.txt** listing the Helm chart and version:

   ```
   <external-helm-repo-name>/<chart-name> <chart-version>
   ```

   Example:

   ```
   mycentralhelmrepo/busybox 1.33.0
   ```

2. Run the following command to load the charts into the CNE Helm chart repository:

   ```
   $ retrieve_helm.sh /var/www/html/occne/charts http://<external-helm-repo-name>/occne/charts [helm_executable_full_path_if_not_default] < helm_charts.txt
   ```

Example:

```
$ retrieve_helm.sh /var/www/html/occne/charts http://mycentralrepo/occne/
charts < helm_charts.txt
```

Upgrade the NF

Change Helm input values used in previous NF release
To change any input value in the Helm chart during the upgrade, refer to the NF-specific upgrade instructions in *<NF-specific> Installation, Upgrade, and Fault Recovery Guide* for the new release. If any new input parameters are added in the new release, then run the following steps:

1. On the Bastion Host, create a YAML file named `<nf-short-name>-values.yaml` to contain the values to be passed to the NF Helm chart.
2. Create a YAML file that contains new and changed values needed by the NF Helm chart. See the NF installation instructions to understand which keys and values must be included in the values file. Only values for parameters that were not included in the Helm input values applied to the previous release, or parameters whose names changed from the previous release, must be included in this file.
3. If the yaml file is created for this upgrade, run the following command to upgrade the NF with new values:

   ```
   $ helm upgrade -f <nf-short-name>-values.yaml <nf-deployment-name> <chart-
   name-or-chart-location>
   ```

> **Note**
>
> The `nf-deployment-name` value must match the value used when installing the NF.

Retain all Helm input values used in previous NF release
If there is no requirement to change the Helm chart input values during upgrade and if no new input parameters were added in the new release, then run the following command to upgrade and retain all the NF values:

```
$ helm upgrade --reuse-values <nf-deployment-name> <chart-name-or-chart-
location>
```

> **Note**
>
> The `nf-deployment-name` value must match the value used when installing the NF.

7.8.3 Uninstalling an NF

This section describes the procedure to uninstall a 5G network function that was previously installed in the CNE Kubernetes cluster.

Prerequisites

- Determine the NF deployment parameters. The following values determine the NF's identity and where it is deployed:

Table 7-16 NF Deployment Parameters

Variable	Value	Description
nf-namespace	Any valid namespace name	The namespace where you want to install the NF. Typically each NF is installed in its own namespace.
nf-deployment-name	Any valid Kubernetes deployment name	The name by which the Kubernetes identifies this NF instance.

- All commands in this procedure must be run from the Bastion Host.

Procedure

1. Run the following command to uninstall an NF:

   ```
   $ helm uninstall <nf-deployment-name> --namespace <nf-namespace>
   ```

2. If there are remaining NF resource such as PVCs and namespace, run the following command to remove them:
   a. Run the following command to remove residual PVCs:

      ```
      $ kubectl --namespace <nf-namespace> get pvc | awk '{print $1}'| xargs -
      L1 -r kubectl --namespace <nf-namespace> delete pvc
      ```

   b. Run the following command to delete namespace:

      ```
      $ kubectl delete namespace <nf-namespace>
      ```

> **Note**
>
> Steps a and b are used to remove all the PVCs from the <nf-namespace> and delete the <nf-namespace>, respectively. If there are other components running in the <nf-namespace>, manually delete the PVCs that need to be removed and skip the `kubectl delete namespace <nf-namespace>` command.

7.8.4 Update Alerting Rules for an NF

This section describes the procedure to add or update the alerting rules for any Cloud Native Core 5G NF in Prometheus Operator and OSO.

Prerequisites

- For CNE Prometheus Operator, a YAML file containing an PrometheusRule CRD defining the NF-specific alerting rules is available. The YAML file must be an ordinary text file in a valid YAML format with the extension `.yaml`.
- 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.

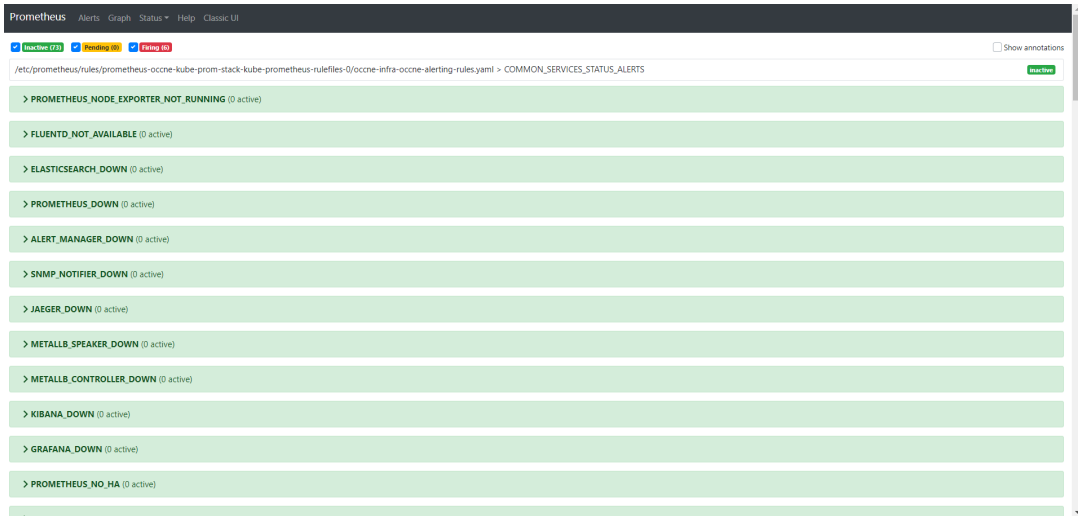
Procedure for Prometheus Operator

1. To copy the NF-specific alerting rules file from your computer to the */tmp* directory on the Bastion Host, see the Accessing the Bastion Host procedure.
2. Run the following command to create or update the PrometheusRule CRD containing the alerting rules for the NF:

```
$ kubectl apply -f /tmp/rules_file.yaml -n occne-infra
# To verify the creation of the alert-rules CRD, run the following command:
$ kubectl get prometheusrule -n occne-infra
NAME                                AGE
occne-alerting-rules                43d
occne-dbtier-alerting-rules         43d
test-alerting-rules                 5m
```

   The alerting rules automatically loads into all running Prometheus instances within 1 minute.
3. In the Prometheus GUI, select the Alerts tab. Select individual rules from the list to view the alert details and verify if the new rules are loaded.

Figure 7-1 New Alert Rules are loaded in Prometheus GUI



Procedure for OSO

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

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

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

2. Check and add the NF alert file name inside Prometheus configuration map. The NF alert file names vary from NF to NF. Retrieve the name of the NF alert rules file to add the name

in Prometheus configuration map. Once you retrieve the file name, run the following commands to add the NF alert file name inside Prometheus configuration map:

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

3. Update 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)"
```

7.8.5 Configuring Egress NAT for an NF

This section provides information about configuring NF microservices that originate egress requests to ensure compatibility with CNE.

Annotation for Specifying Egress Network

Starting CNE 22.4.x, egress requests do not get the IP address of the Kubernetes worker node assigned to the source IP field. Instead, each microservice that originates egress requests specifies an egress network through an annotation. An IP address from the indicated network is inserted into the source IP field for all egress requests.

For each microservice that originates egress requests, add the following annotation to the deployment specification and its pods:

```
annotations:
  oracle.com.cnc/egress-network: "oam"
```

Note

- The value of the annotation must match the name of an external network configured.
- This annotation must not be added for microservices that do not originate egress requests, as it leads to decreased CNE performance.
- CNE does not allow any microservice to pick a separate IP address. When CNE is installed, a single IP address is selected for each network.
- All pods in a microservice get the same source IP address attached to all egress requests.
- CNE 22.4.x supports this annotation in vCNE deployments only.

Configuring Egress Controller Environment

Egress controller runs as Kubernetes resource of type DaemonSet. The following table provides details about the environmental variables that can be edited in the Egress controller manifest file:

> **Note**
>
> Do not edit any variables that are not listed in the following table.

Table 7-17 Egress Controller Environment Configuration

Environment Variable	Default Value	Possible Value	Description
DAEMON_MON_TIME	0.5	Between 0.1 and 5	This value reflects the time in seconds and highlights the frequency at which the Egress controller checks the cluster status.

Configuring Egress NAT for Destination Subnet or IP Address

Destination subnet or IP address must be specified to route traffic trough a particular network. The destination subnet or IP address is specified in the form of a dictionary, where the pools are the dictionary keys and the lists of subnets or IP addresses are the dictionary values.

The following annotations show different scenarios of adding pools and destination subnets or IP addresses with examples:

- Specifying annotation for destination subnet:

  ```
  annotations:
    oracle.com.cnc/egress-destination: '{"<pool>" : ["<subnet_ip_address>/
  <subnet_mask>"]}'
  ```

  For example:

  ```
  annotations:
    oracle.com.cnc/egress-destination: '{"oam" : ["10.20.30.0/24"]}'
  ```

- Specifying annotation for destination IP address:

  ```
  annotations:
    oracle.com.cnc/egress-destination: '{"<pool>" : ["<ip_address>"]}'
  ```

  For example:

  ```
  annotations:
    oracle.com.cnc/egress-destination: '{"oam" : ["10.20.30.40"]}'
  ```

- Specifying annotation for multiple pools:

```
annotations:
  oracle.com.cnc/egress-destination: '{"<pool_one>" :
["<subnet_ip_address>/<subnet_mask>"], "<pool_two>" :
["<subnet_ip_address>/<subnet_mask>"]}'
```

For example:

```
annotations:
  oracle.com.cnc/egress-destination: '{"oam" : ["10.20.30.0/24"], "sig" :
["30.20.10.0/24"]}'
```

- Specifying annotation for multiple pools and multiple destinations:

```
annotations:
  oracle.com.cnc/egress-destination: '{"<pool_one>" :
["<subnet_ip_address>/<subnet_mask>", "<subnet_ip_address>/
<subnet_mask>"], "<pool_two>" : ["<subnet_ip_address>/<subnet_mask>",
"<ip_address>"]}'
```

For example:

```
annotations:
  oracle.com.cnc/egress-destination: '{"oam" : ["10.20.30.0/24",
"100.200.30.0/22"], "sig" : ["30.20.10.0/24", "20.10.5.1"]}'
```

Compatibility Between Egress NAT and Destination Egress NAT

Both Egress NAT and Destination Egress NAT annotations are independent and compatible. This means that they can be used independently or combined to create more specific rules. Egress NAT is enabled to route all traffic from a particular pod through a particular network. Whereas, Destination Egress NAT permits traffic to be routed using a destination subnet or IP address before regular Egress NAT rules are matched within the routing table. This feature allows more granularity to route traffic through a particular network.

For example, the following annotations show both the features combined to route a pod's traffic through the `sig` network, except the traffic destined for `10.20.30.0/24` subnet, which is routed through the `oam` network:

```
annotations:
  oracle.com.cnc/egress-destination: '{"oam" : ["10.20.30.0/24"]}'
  oracle.com.cnc/egress-network: sig
```

A
References

This section lists the additional topics that are referred to while performing some of the procedures in the document.

A.1 Changing Retention Size of Prometheus

This section details the procedures for changing the retention size of Prometheus. Prometheus is configured to delete data only in the following conditions:

- Data is older than the configured retention period (14 days) and size.
- The PV is at least 90% full.

The retention level does not consider the write-ahead log (WAL) in its calculations. If the WAL size is greater than 10% of the PV size, then the PV fills up before Prometheus detect the "too full" condition. Therefore, it is recommended to calculate the WAL growth and adjust the retention size of Prometheus accordingly.

Calculating WAL Growth

1. Refer to the WAL space calculation document for all the formulas that are used in this section for calculating the WAL growth.
2. Users must have a benchmarking of the metrics size with each release to calculate the approximate WAL size Growth. Consider the following parameters for analyzing and understanding the metrics size:

   > **Note**
   > - The metrics size must be calculated for a time period of three hours.
   > - The traffic has to be active and running at the Maximum Transaction Per Second (MAX TPS). The TPS considered in this example is 5000.

   a. Samples scraped per scrape:

   ```
   sum(scrape_samples_scraped{kubernetes_namespace="<namespace>"})
   ```

   Example for CNE Samples:

   ```
   sum(scrape_samples_scraped{kubernetes_namespace="occne-infra"})
   ```

   Example for UDR Samples:

   ```
   sum(scrape_samples_scraped{kubernetes_namespace="udr"})
   ```

b. Samples must remain constant irrespective of the TPS rate. However, for some of the NFs, the samples may grow with increasing TPS. In such cases, collect the growth rate of samples.

c. PV disk growth with the increasing sample size. This metric is collected only for observation and is not used in calculating the metrics size.

3. After gathering the data as described in step 2, use the following formula to calculate WAL growth.
The formula considers the following values:

- scrape_samples_scraped = total samples gathered in Step 2
- One Sample Size = 13 bytes
- Time period = 3 hours, that is 180min (average of two and four hours is the generally observed compaction interval)
- Default scrape interval provided by CNE = 60s

Formula to calculate WAL growth

```
WAL Growth = sum(scrape_samples_scraped{kubernetes_namespace="occne-
infra"}) * 13 * 180 * (60/scrape-interval)
```

Example 1

Considering the following values:

- Samples scraped per scrape at 5000 TPS with 4 worker nodes (CNE+UDR) = 75,000
- scrape-interval = 60s
- Total samples scraped = 75,000 * 13 * 180 * 1 = 175,500,000 which approximates to 175MB

the upper limit of WAL growth can be considered as 250MB in 3 hours.

Example 2

Considering the following values:

- Samples scraped per scrape at 5000 TPS with 5 worker nodes (CNE+UDR) = 88,200
- scrape-interval = 10s
- Total samples scraped = 88200 * 13 * 180 * (60/10) = 1153282 which approximates to 1.15GB

the upper limit of WAL growth can be considered as 1.2GB in 3 hours.

> **Note**
>
> Prometheus runs a garbage collection only after PV is full by 40% (for more information, see Prometheus-Storage). The threshold garbage collection is not initiated before PV is 40% full. For example, if Prometheus PV is full by 90%, Prometheus doesn't start purging old persisted data immediately. Instead, it waits until the next cycle of Garbage Collection. Therefore, there is no fixed time interval and the entire process is internal to Prometheus. To avoid Prometheus getting completely full and crashing before garbage collector is called, CNE provides a buffer of 10% in retention size.

Calculating Retention Size

The following examples provides details on calculating retention size.

Example 1

Considering the following values:

- Size of PV equal to 8GB
- WAL growth in 3 hours greter than or equal to 250MB (0.25GB), which is approximately 5% when rounded to the nearest multiple of 5
- Buffer left for Garbage Collection (GC) cycle equal to 10%, which is greater than or equal to 0.8GB

the retention size is calculated as 85% (100 - 5 - 10), which is greater than or equal to 6.8GB (85% of 8GB).

Example 2

Considering the following values:

- Size of PV equal to 8GB
- WAL growth in 3 hours greater than or equal to 1.2GB, which is approximately 15% when rounded to the nearest multiple of 5
- Buffer left for Garbage Collection (GC) cycle equal to 10%, which is greater than or equal to 0.8GB

the retention size is calculated as 75% (100 - 15 - 10), which is greater than or equal to 6GB (75% of 8GB).

Changing the Retention Value for CNE 1.8.x and below

1. Perform the following steps to scale down Prometheus deployment to 0.

   > **⚠ Caution**
   >
   > This step can lead to loss of data as Prometheus do not scrape or store metrics during the down time.

   a. Run the following command to get the Prometheus deployment:

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

   Example:

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

   Sample output:

   ```
   NAME                                          READY   UP-TO-DATE
   AVAILABLE   AGE
   occne-elastic-exporter-elasticsearch-exporter   1/1     1
   1           9d
   occne-grafana                                 1/1     1
   ```

```
1           9d
occne-kibana                                  1/1     1
1           9d
occne-metrics-server                          1/1     1
1           9d
occne-prometheus-kube-state-metrics           1/1     1
1           9d
occne-prometheus-pushgateway                  1/1     1
1           9d
occne-prometheus-server                       1/1     1
1           9d
occne-snmp-notifier                           1/1     1
1           9d
occne-tracer-jaeger-collector                 1/1     1
1           9d
occne-tracer-jaeger-query                     1/1     1
1           9d
```

b. Run the following command to scale down the deployment to 0:

```
$ kubectl scale deploy occne-prometheus-server --replicas 0 -n occne-
infra
```

Sample output:

```
deployment.apps/occne-prometheus-server scaled
```

2. Edit the retention size of Prometheus deployment and save the deployment:

```
$ kubectl edit deploy occne-prometheus-server -n <namespace>
```

Example:

```
$ kubectl edit deploy occne-prometheus-server -n occne-infra
```

Sample output:

```
317       - args:
   318         - --storage.tsdb.retention.time=14d
   319         - --config.file=/etc/config/prometheus.yml
   320         - --storage.tsdb.path=/data
   321         - --web.console.libraries=/etc/prometheus/console_libraries
   322         - --web.console.templates=/etc/prometheus/consoles
   323         - --web.enable-lifecycle
   324         - --web.external-url=http://localhost/prometheus
   325         - --storage.tsdb.retention.size=8GB
```

> **Note**
>
> Edit line number 325 in the sample output, which displays the retention size, to 6.8GB (as per example 1 in Calculating Retention Size).

3. Scale up the Prometheus:

   ```
   $ kubectl scale deployment occne-prometheus-server --replicas 1 -n
   <namespace>
   ```

   Example:

   ```
   $ kubectl scale deploy occne-prometheus-server --replicas 1 -n occne-infra
   ```

   Sample output:

   ```
   deployment.apps/occne-prometheus-server scaled
   ```

4. Check if the pod is up and running:

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

   Example:

   ```
   $ kubectl get pods -n occne-infra | grep prometheus-server
   ```

   Sample output:

   ```
   NAME                                                              READY
   STATUS      RESTARTS   AGE
   occne-prometheus-server-58d4d5c459-7dq2d                          2/2
   Running     0          19h
   ```

5. Perform the following steps to check if the deployment reflects the updated retention size:
   a. Run the following command to get the Prometheus deployment:

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

      Sample output:

      ```
      NAME                                          READY   UP-TO-DATE
      AVAILABLE   AGE
      occne-elastic-exporter-elasticsearch-exporter   1/1     1
      1           9d
      occne-grafana                                 1/1     1
      1           9d
      occne-kibana                                  1/1     1
      1           9d
      occne-metrics-server                          1/1     1
      1           9d
      occne-prometheus-kube-state-metrics           1/1     1
      1           9d
      occne-prometheus-pushgateway                  1/1     1
      1           9d
      occne-prometheus-server                       1/1     1
      1           9d
      occne-snmp-notifier                           1/1     1
      ```

```
1           9d
occne-tracer-jaeger-collector                  1/1     1
1           9d
occne-tracer-jaeger-query                      1/1     1
1           9d
```

b. Run the following commands to open the `occne-prometheus-server.yaml` file and verify the updated retention size:

```
$ kubectl get deploy occne-prometheus-server -n occne-infra -o yaml >
occne-prometheus-server.yaml
$ cat occne-prometheus-server.yaml
```

Sample output:

```
317       - args:
318           - --storage.tsdb.retention.time=14d
319           - --config.file=/etc/config/prometheus.yml
320           - --storage.tsdb.path=/data
321           - --web.console.libraries=/etc/prometheus/console_libraries
322           - --web.console.templates=/etc/prometheus/consoles
323           - --web.enable-lifecycle
324           - --web.external-url=http://localhost/prometheus
325           - --storage.tsdb.retention.size=6.8GB
```

> **Note**
>
> Use search or Grep "retention.size" in the `occne-prometheus-server.yaml` file to check if the retention size is updated to 6.8GB.

Changing the Retention Value for CNE 1.9.x and above

> **Note**
>
> As CNE 1.9.x and above uses HA for Prometheus, there is no service interruption or loss of data while performing this procedure. This is because one Prometheus pod is always up while performing the procedure.

1. Run the following command to list the Prometheus component:

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

Example:

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

Sample output:

```
NAME                                    VERSION   REPLICAS   AGE
occne-kube-prom-stack-kube-prometheus   v2.24.0   2          7d5h
```

2. Run the following command to edit the Prometheus component:

   ```
   $ kubectl edit prometheus occne-kube-prom-stack-kube-prometheus -n
   <namespace>
   ```

   Example:

   ```
   $ kubectl edit prometheus occne-kube-prom-stack-kube-prometheus -n occne-
   infra
   ```

   Sample output:

   ```
   51   replicas: 2
        52   retention: 14d
        53   retentionSize: 8GB
        54   routePrefix: /prometheus
        55   ruleNamespaceSelector: {}
        56   ruleSelector:
        57     matchLabels:
   ```

   Note

   a. Edit line number 53 in the sample output, which displays the retention size, to 6.8GB (as per example 1 in Calculating Retention Size).

   b. This initiates rolling update for the prometheus-occne-kube-prom-stack-kube-prometheus pods which takes a few minutes be complete.

3. Check if the pods are up and running.

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

   Example:

   ```
   $ kubectl get pods -n occne-infra | grep kube-prometheus
   ```

   Sample output:

   ```
   prometheus-occne-kube-prom-stack-kube-prometheus-0              2/2
   Running            1          4m22s
   prometheus-occne-kube-prom-stack-kube-prometheus-1              2/2
   Running            1          4m58s
   ```

4. Perform the following steps to check if the deployment reflects the updated retention size:

   a. Run the following command to get the Statefulset of the deployment:

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

Sample output:

```
NAME                                                 READY   AGE
alertmanager-occne-kube-prom-stack-kube-alertmanager 2/2     7d5h
occne-elastic-elasticsearch-client                   3/3     7d5h
occne-elastic-elasticsearch-data                     3/3     7d5h
occne-elastic-elasticsearch-master                   3/3     7d5h
prometheus-occne-kube-prom-stack-kube-prometheus     2/2     7d5h
```

b. Run the following commands to open the `occne-kube-prom-stack.yaml` file and verify the updated retention size:

```
$ kubectl get sts prometheus-occne-kube-prom-stack-kube-prometheus -n
occne-infra -o yaml > occne-kube-prom-stack.yaml
$ cat occne-kube-prom-stack.yaml
```

Sample output:

```
- args:
       - --web.console.templates=/etc/prometheus/consoles
       - --web.console.libraries=/etc/prometheus/console_libraries
       - --storage.tsdb.retention.size=6.8GB
```

> **Note**
>
> Use search or Grep "retention.size" in the `occne-kube-prom-stack.yaml` file to check if the retention size is updated to 6.8GB.