

Oracle® Communications

Cloud Native Core, Cloud Native Environment

User Guide



Release 26.1.200
G59019-01
July 2026

ORACLE®

Copyright © 2021, 2026, Oracle and/or its affiliates.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software, software documentation, data (as defined in the Federal Acquisition Regulation), or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs) and Oracle computer documentation or other Oracle data delivered to or accessed by U.S. Government end users are "commercial computer software," "commercial computer software documentation," or "limited rights data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, reproduction, duplication, release, display, disclosure, modification, preparation of derivative works, and/or adaptation of i) Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs), ii) Oracle computer documentation and/or iii) other Oracle data, is subject to the rights and limitations specified in the license contained in the applicable contract. The terms governing the U.S. Government's use of Oracle cloud services are defined by the applicable contract for such services. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle®, Java, MySQL, and NetSuite are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Inside are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Epyc, and the AMD logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

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

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

Contents

1	Introduction	
1.1	Overview	1
1.2	References	1
1.3	Key Terms	1
2	Deployment Models	
2.1	BareMetal Deployment	1
2.2	Virtualized Deployment	1
3	CNE Services	
3.1	Runtime Environment	1
3.1.1	Kubernetes Cluster	1
3.1.2	Networking	2
3.1.3	Storage	3
3.1.4	Traffic Segregation	3
3.1.5	DNS Query Services	3
3.1.5.1	Local DNS	4
3.2	Automation	4
3.2.1	Deployment	4
3.2.2	Upgrade	4
3.2.3	Maintenance	4
3.3	Security	5
3.4	Redundancy	5
3.5	Observability	7
3.5.1	Metrics	7
3.5.2	Alerts	7
3.5.3	Logs	8
3.5.4	Traces	8
3.6	Maintenance Access	9
3.7	Frequently Used Common Services	9

4	Resource Utilization	
5	Metrics	
5.1	Cloud Native Load Balancer Metrics	1
5.2	CNE Metrics	4
6	Alerts	
6.1	Kubernetes Alerts	1
6.2	Common Services Alerts	6
6.3	Bastion Host Alerts	17
7	CNE Supported Features	
8	Maintenance Procedures	
8.1	Accessing the CNE	1
8.1.1	Accessing the Bastion Host	1
8.1.1.1	Logging in to the Bastion Host	1
8.1.1.2	Copying Files to the Bastion Host	2
8.1.1.3	Managing Bastion Host	2
8.1.1.4	Troubleshooting Bastion Host	4
8.2	General Configuration	6
8.2.1	Configuring SNMP Trap Destinations	6
8.2.2	Changing Network MTU	8
8.2.3	Changing Metrics Storage Allocation	17
8.2.4	Changing OpenSearch Storage Allocation	21
8.2.5	Changing the RAM and CPU Resources for Common Services	25
8.2.5.1	Changing the Resources for Prometheus	25
8.2.5.2	Changing the Resources for Alertmanager	26
8.2.5.3	Changing the Resources for Grafana	27
8.2.5.4	Changing the Resources for Kube State Metrics	27
8.2.5.5	Changing the Resources for OpenSearch	28
8.2.5.6	Changing the Resources for OpenSearch Dashboard	29
8.2.5.7	Changing the Resources for Fluentd OpenSearch	30
8.2.5.8	Changing the Resources for Jaeger Agent	30
8.2.5.9	Changing the Resources for Jaeger Query	31
8.2.6	Activating and Configuring Local DNS	32
8.2.6.1	Activating Local DNS	32
8.2.6.2	Deactivating Local DNS	36

8.2.6.3	Adding and Removing DNS Records	38
8.2.6.4	Adding and Removing Forwarding Nameservers	49
8.2.6.5	Reloading Local or Core DNS Configurations	52
8.2.6.6	Other Local DNS API Endpoints	54
8.2.6.7	Troubleshooting Local DNS	56
8.3	Managing the Kubernetes Cluster	66
8.3.1	Creating CNE Cluster Backup	67
8.3.1.1	Creating a Backup of Bastion Host and Kubernetes Data	68
8.3.1.2	Verifying Backup in S3 Bucket	69
8.3.2	Renewing Kubernetes Certificates	70
8.3.2.1	Renewing Kyverno Certificates	79
8.3.2.2	Renewing Kubelet Server Certificates	80
8.3.3	Renewing the Kubernetes Secrets Encryption Key	81
8.3.4	Adding a Kubernetes Worker Node	84
8.3.5	Removing a Kubernetes Worker Node	95
8.3.6	Adding a New External Network	104
8.3.6.1	Adding a New External Network in Bare Metal	104
8.3.7	Performing an etcd Data Backup	105
8.4	Managing Kyverno	106
8.4.1	Accessing Cluster Policy Report	107
8.4.2	Migrating from PSP to Kyverno	108
8.4.3	Adding Policy Exclusions and Exceptions	109
8.4.4	Managing Kyverno Metrics	110
8.4.5	Validating Kyverno Compliance	110
8.5	Updating Grafana Password	119
8.6	Updating OpenStack Credentials	121
8.7	Updating OpenStack TLS Certificate	127
8.8	Updating the Guest or Host OS	128
8.9	CNE Grafana Dashboards	128
8.9.1	Accessing Grafana Interface	128
8.9.2	Cloning a Grafana Dashboard	129
8.9.3	Restoring a Grafana Dashboard	129
8.10	Managing 5G NFs	130
8.10.1	Installing an NF	130
8.10.2	Upgrading an NF	132
8.10.3	Uninstalling an NF	134
8.10.4	Update Alerting Rules for an NF	135
8.10.5	Configuring Egress NAT for an NF	137
8.11	Updating CNLB Networks	139
8.11.1	Adding a New Network	154
8.11.2	Deleting a network	158
8.11.3	Adding an IP to the service_ip_addr list of an existing network	161

8.11.4	Deleting an IP from the service_ip_addr list of an existing network	164
8.11.5	Changing an IP in the service_ip_addr list of an existing network	168
8.11.6	Adding a CIDR to the egress_dest field of an existing network	171
8.11.7	Deleting a CIDR from the egress_dest field of an existing network	174
8.11.8	Common Issues and Solutions	176
8.12	Secure DNS Zone Customization through CNLB	176
8.12.1	Determining Active Bastion Host	178
8.12.2	Defining Variables for External DNS Server	178
8.12.3	Enabling TLS in Zones	180
8.12.4	Enabling DNS Enhancement	185
8.12.5	Verifying Packet Flow Through the CNLB Interface	188
8.12.6	Rolling back coredns and nodelocaldns deployment and configmap	192
8.12.7	Updating DNS Enhancement	192
8.12.8	Validating DNS Enhancement	193
8.13	Enabling Public Endpoint for CNE (Kubernetes) API Server	196
8.13.1	Enabling occne-apiserver-proxy Service	198
8.13.2	Using occne-apiserver-proxy	200
8.13.3	Customizing the Nginx Configuration	201
8.14	Grafana Dashboards	202
8.14.1	Custom Metrics	202
8.14.2	CNLB Dashboard	207
8.14.3	Kyverno Dashboard	210
8.14.4	Fluentd Logging Dashboard	222
8.14.5	Persistence Storage Dashboard	223
8.15	Changing Bastion Registry Credentials	225
8.16	Log Externalization Support	226
8.17	Using CNLB Command Line Utility	232

A References

A.1	Changing Retention Size of Prometheus	A-1
-----	---------------------------------------	-----

Preface

- [Documentation Accessibility](#)
- [Diversity and Inclusion](#)
-

Diversity and Inclusion

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

Documentation Accessibility

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

Conventions

Specific text conventions are used in this document.

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

My Oracle Support

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

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

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

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

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

What's New in This Guide

This section introduces the documentation updates for release 26.1.2xx.

Release 26.1.200 - G59019-01, July 2026

Feature Updates

- **Generic Updates**

Updated the versions of the following common services in the [Frequently Used Common Services](#) section:

- Helm - 3.20.0
- Kubernetes - 1.35.6
- podman - 5.6.0
- containerd - 2.2.1
- Calico - 3.30.6
- MetalLB - 0.15.2
- Prometheus - 3.9.0
- OCI-Grafana - 7.5.17
- Jaeger - 2.14.1
- Kyverno - 1.16.1

- Added the [CNE Supported Features](#) section, which lists CNE and CNE deployment features, their default enabled or disabled status, configuration source, and applicable licensing information.

- **Maintenance Procedures**

- CNE provides Grafana dashboards that combine CNE custom node metrics with Kubernetes default health/utilization views to monitor cluster readiness, capacity, and workload distribution. For more information, see the [Grafana Dashboards](#) section.
- CNE supports changing Bastion container registry credentials to access image (for example, push, pull, list, and so on). For more information, see the [Changing Bastion Registry Credentials](#) section.
- Added guidance for removing or replacing vApp networks on VCD 10.4.1 and later in the [Updating CNLB Networks](#) section.
- CNE supports externalization of logs using which users do not require to install OpenSearch and Jaeger as part of their CNE install or upgrade. For more information, see the [Log Externalization Support](#) section.
- CNE provides a new CNLB binary utility, CNLBCTL which can be downloaded on a Bastion Host and install. For more information, see the [Using CNLB Command Line Utility](#) section.

1

Introduction

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

1.1 Overview

Cloud Native Environment (CNE) is a Container-as-a-Service (CaaS) platform designed for on-premises deployments, catering to both production and lab environments. It primarily supports telecommunications applications, including 3GPP 5G Network Functions, Session Delivery, Charging, and Network Analytics.

Key features include:

- **Unified Solution Services:** Offers a consistent platform that meets diverse telco application needs.
- **Scalability:** Designed to efficiently scale with network demands.
- **Uniform Distribution:** Ensures balanced resource allocation across deployments.
- **Automation:** Streamlines deployment processes and supports in-service upgrades, minimizing downtime for critical services.
- **Deployment Flexibility:** Supports both Bare Metal and Virtualization platforms, accommodating various customer deployment models.
- **Hardware and IaaS Support:** Compatible with multiple hardware configurations and Infrastructure-as-a-Service environments.
- **Cloud-Native and Observability Services:** Provides modern cloud-native capabilities along with monitoring and observability tools for operational insights.

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*
- *Oracle Communications Cloud Native Core Release Notes*
- *Oracle Communications Cloud Native Core Licensing Information User Guide*
- *Oracle Communications Cloud Native Core Solution Upgrade Guide*
- *Oracle Communications Cloud Native Core Security Guide*

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.
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 for BareMetal deployments. Whereas, for vCNE deployments the Bastion is a separate virtual machine within the OpenStack or VMWare Cloud. 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.

Table 1-1 (Cont.) Key Terms

Term	Definition
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) offers different deployment strategies for each vCNE and bare metal platform. These strategies account for a combined approach that leverages heterogeneous nodes, as well as the option to dedicate nodes to specific components. For more information, see the "CNLB Deployment Strategies" section in *Oracle Communications Cloud Native Core, Cloud Native Environment Installation, Upgrade, and Fault Recovery Guide*.

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 BareMetal Deployment

CNE BareMetal deployment is hardware agnostic and can be installed on any servers such as HP Gen 10 and Oracle Netra X8-2. CNE requires the following servers for BareMetal deployments:

- Three Kubernetes controller servers
- Six Kubernetes worker servers. However, CNE supports BareMetal deployment using three worker servers for getting started with CNE.

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

CNE allows you to set up your BareMetal cluster using any ToR switch for routing. For more information about configuring the ToR switches, see the "Configuring Top of Rack Switches" section in *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 OpenTofu 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
- A minimum of 2 CNLB nodes

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 26.1.2xx supports Kubernetes version 1.35.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 manages control node failures as follows:

- Kubernetes sustains the loss of one controller node with no impact to the CNE service.
- Kubernetes tolerates the loss of two controller nodes without losing the cluster configuration data. In a case where two controller nodes are lost, application workload continues to run on the worker node. However, no changes can be made to the cluster configuration data. Due to this, the system experiences the following constraints:
 - New applications cannot be installed.
 - Changes to application-specific custom resources, that are used to change an application's configuration, cannot be performed.
 - Most `kubectl` commands, especially those that require interacting with the control plane, doesn't work. Commands that don't depend on realtime data from the control plane can still work depending on the situation.

It is recommended to have a minimum of six worker nodes for a proper functioning of CNE. The maximum number of worker nodes depend on your requirement. However, CNE allows you to setup a BareMetal deployment with a minimal resources of three worker nodes which is only ideal for testing and getting started with CNE. For more information about installing and upgrading CNE with bare minimum resources, see the "Installing BareMetal CNE using Bare Minimum Servers" and "Upgrading BareMetal CNE Deployed using Bare Minimum Servers" sections respectively in *Oracle Communications Cloud Native Core, Cloud Native Environment Installation, Upgrade, and Fault Recovery Guide*.

Kubernetes distributes workloads across all the available worker nodes. The Kubernetes placement algorithm attempts to balance workloads across all available worker nodes, such 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.

Limited Kubernetes Cluster Nodes for CNLB

For CNLB based deployments of CNE, limited Kubernetes nodes can be used to run CNLB nodes. This allows to create CNE cluster with limited number of CNLB compliant Kubernetes nodes. It is useful when there is a constraint on the number of external IP addresses available (every Kubernetes node in cluster will require Ingress/Egress external addressees if the option is not enabled). This option helps in scenarios where cluster size is larger compared to the number of CNLB replicas.

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*.

Support for Floating IPs in OpenStack

Floating IPs are additional public IP addresses that are associated with instances such as control nodes, worker nodes, and Bastion Host. Floating IPs can be quickly re-assigned and switched from one instance to another using an API interface, thereby ensuring high availability and less maintenance. You can activate the Floating IP feature after installing or upgrading CNE. For information about enabling or disabling the Floating IP feature, 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 Traffic Segregation

CNE uses Cloud Native Load Balancer (CNLB) to segregate Ingress and Egress traffic.

Ingress Traffic Distribution

Ingress traffic distribution handles inbound traffic only. Traffic enters the CNLB application through an external interface service IP address and is routed internally through internal interfaces within CNLB networks.

Egress traffic delivery

Egress traffic delivery handles outbound traffic only. An NF pod can initiate traffic and route it through a CNLB application, which translates the source IP address to an external egress IP address.

An Egress Network Attachment Definition contains the network information required to create interfaces for NF pods and routes to external subnets.

For more information about enabling and configuring CNLB and selecting the external network for egress requests, see *Oracle Communications Cloud Native Core, Cloud Native Environment Installation, Upgrade, and Fault Recovery Guide*.

3.1.5 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.

3.1.5.1 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 multilevel 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.
- Kubernetes control plane and Kubernetes worker nodes communicate over secure communication channels to protect sensitive cluster data.
- Kyverno policy framework is deployed in CNE with base line policies. This ensures that the containers and resources running in CNE are compliant.
- Kyverno pod security policies ensure that malicious applications don't corrupt the Kubernetes controller, worker nodes, and cluster data. Kyverno provides the set of policies equivalent to those provided by the built-in Kubernetes PSP as Kubernetes PSP is deprecated.
- Bastion Host container registry is secured with TLS and can be accessed from CNE only.
- CNE supports both TLS 1.2 and TLS 1.3 for establishing secure connections. Currently, the minimum supported TLS version for CNE internal and external communication is TLSv12. However, in the upcoming releases, CNE will support only TLSv13 for security and governance compliance.
- 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.

Note

The occne-logstash indices which used to collect occne-infra logs is disabled from 23.1.x release onwards. By default, CNE supports the following indices:

- logstash: consists of 12 primary shards and 1 replica, and rotates daily
- jaeger-span: consists of 5 primary shards and 1 replica, and rotates daily
- jaeger-service: consists of 5 primary shards and 1 replica, and rotates daily

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

- CNE uses CNLB to manage ingress and egress traffic for both vCNE and Bare Metal deployments.
- CNLB App Pods process ingress and egress traffic. For each configured external network, CNLB App Pods run as an active-standby pair. The active pod processes traffic, while the standby pod provides redundancy.
- The CNLB Manager manages CNLB App Pods and their configuration. It coordinates the active and standby CNLB App Pods for each external and internal network.
- If an active CNLB App Pod becomes unavailable, traffic handling can fail over to the paired standby CNLB App Pod so ingress and egress traffic delivery can continue. When the failed CNLB App Pod is restored or replaced, CNLB redundancy is restored.

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.

① 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 26.1.2xx. 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	Usage
AlertManager	0.30.0	Alertmanager is a component that works in conjunction with Prometheus to manage and dispatch alerts. It handles the routing and notification of alerts to various receivers.
Grafana	7.5.17-1.0.4	Grafana is a popular open-source platform for monitoring and observability. It provides a user-friendly interface for creating and viewing dashboards based on various data sources.
Prometheus	3.9.0	Prometheus is a popular open-source monitoring and alerting toolkit. It collects and stores metrics from various sources and allows for alerting and querying.
Calico	3.30.6	Calico provides networking and security for NFs in Kubernetes with scalable, policy-driven connectivity.
Containerd	2.2.1	Containerd manages container lifecycles for running NFs efficiently in Kubernetes.
Fluentd - OpenSearch	1.19.0	Fluentd is an open-source data collector that streamlines data collection and consumption, allowing for improved data utilization and comprehension.
Helm	3.20.0	Helm, a package manager, simplifies deploying and managing network functions (NFs) on Kubernetes with reusable, versioned charts for easy automation and scaling.

Table 3-1 (Cont.) Frequently Used Common Services

Common Service	Supported Version	Usage
Jaeger	2.14.1	Jaeger provides distributed tracing for 5G NFs, enabling performance monitoring and troubleshooting across microservices.
Kubernetes	1.35.6	Kubernetes orchestrates scalable, automated network function (NF) deployments for high availability and efficient resource utilization.
Kyverno	1.16.1	Kyverno is a Kubernetes policy engine that helps manage and enforce policies for resource configurations within a Kubernetes cluster.
MetalLB	0.15.2	MetalLB provides load balancing and external IP management for 5G NFs in Kubernetes environments.
Oracle OpenSearch	2.19.1	OpenSearch provides scalable search and analytics for 5G NFs, enabling efficient data exploration and visualization.
Oracle OpenSearch Dashboard	2.19.1	OpenSearch Dashboard visualizes and analyzes data for 5G NFs, offering interactive insights and custom reporting.
Prometheus Operator	0.88.1	The Prometheus Operator is used for managing Prometheus monitoring systems in Kubernetes. Prometheus Operator, simplifies the configuration and management of Prometheus instances.
Velero	1.17.1	Velero backs up and restores Kubernetes clusters for 5G NFs, ensuring data protection and disaster recovery.
metrics-server	0.8.0	Metrics Server is used in Kubernetes for collecting resource usage data from pods and nodes.
snmp-notifier	2.0.0	snmp-notifier sends SNMP alerts for 5G NFs, providing real-time notifications for network events.

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, Fleuntld, 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
Grafana	2000	2000	4096	4096	1
MetalLB Controller (for CNLB disabled CNE only)	100	100	100	100	1
MetalLB Speaker (for CNLB disabled CNE only)	100	100	100	100	1 per worker node
Egress Controller (for CNLB disabled CNE only)	100	1000	200	500	1 per worker node
Bastion Controller	10	200	128	256	1
Kyverno	100	200	256	512	3
CNLB-App (Dedicated) vCNE only	1000	1000	1024	1024	Recommended: 4 Min required: 2
CNLB-App (Dedicated) vCNE only	16000	16000	16384	16384	4
CNLB-Manager	500	4000	1024	1024	1
Multus Thick	250	500	256	512	1 per node

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.

Note

When querying a metric in the Prometheus UI, queries that return a very large number of samples can fail with an Invalid JSON response error. To reduce the number of returned samples, apply an appropriate label filter. For example, instead of querying `some_test_metric`, query `some_test_metric{label="value"}`, using a label and value appropriate for the metric.

5.1 Cloud Native Load Balancer Metrics

CNE provides custom CNLB metrics that can be used to observe the cluster and monitor the cluster health from Prometheus, these metrics can be used in Grafana for interactive monitoring of the cluster.

Egress Network Metrics

Table 5-1 `cnlb_egr_outpkt`

Field	Details
Description	Total outgoing packets from the egress IP.
Type	Counter
Dimensions	egress_ip: active_pod_ip:

Table 5-2 `cnlb_egr_inpkt`

Field	Details
Description	Total incoming packets to the egress IP.
Type	Counter
Dimensions	egress_ip: active_pod_ip:

Table 5-3 `cnlb_egr_inbytes`

Field	Details
Description	Total incoming bytes to the egress IP.
Type	Counter

Table 5-3 (Cont.) cnlb_egr_inbytes

Field	Details
Dimensions	egress_ip: active_pod_ip:

Table 5-4 cnlb_egr_outbytes

Field	Details
Description	Total outgoing bytes from the egress IP.
Type	Counter
Dimensions	egress_ip: active_pod_ip:

Internal Multus (Interface) Metrics**Table 5-5 cnlb_interface_int_inpkt**

Field	Details
Description	Total incoming packets on internal multus master interface.
Type	Counter
Dimensions	active_cnlb_pod: service_ip: backend_ip: app_name:

Table 5-6 cnlb_interface_int_outpkt

Field	Details
Description	Total outgoing packets from internal multus master interface.
Type	Counter
Dimensions	active_cnlb_pod: service_ip: backend_ip: app_name:

Table 5-7 cnlb_interface_ext_inpkt

Field	Details
Description	Total incoming packets on external multus master interface.
Type	Counter
Dimensions	active_cnlb_pod: service_ip: backend_ip: app_name:

Table 5-8 cnlb_interface_ext_outpkt

Field	Details
Description	Total outgoing packets from external multus master interface.
Type	Counter
Dimensions	active_cnlb_pod: service_ip: backend_ip: app_name:

Service Ingress IPVS & Connection Metrics**Table 5-9 connections_total**

Field	Details
Description	Total connection count for backend_ip (active tracked for each backend/service).
Type	Gauge
Dimensions	active_cnlb_pod: service_ip: backend_ip: app_name:

Table 5-10 cnlb_inbyte

Field	Details
Description	Total incoming bytes from remote to service IP.
Type	Gauge
Dimensions	active_cnlb_pod: service_ip: backend_ip: app_name:

Table 5-11 cnlb_inpkt

Field	Details
Description	Total incoming packets received from remote to service IP (scrape interval).
Type	Gauge
Dimensions	active_cnlb_pod: service_ip: backend_ip: app_name:

Table 5-12 cnlb_outbyte

Field	Details
Description	Total outgoing bytes from app pod to remote.
Type	Gauge
Dimensions	active_cnlb_pod: service_ip: backend_ip: app_name:

Table 5-13 cnlb_outpkt

Field	Details
Description	Total outgoing packets sent from app pod to remote.
Type	Gauge
Dimensions	active_cnlb_pod: service_ip: backend_ip: app_name:

5.2 CNE Metrics

CNE offers custom metrics that enable more detailed monitoring of node behavior. These metrics are gathered from cluster nodes and provide insights into each vCPU as well as every network interface on each node.

Mpstat Metrics

These metrics are collected by saving the output of the `mpstat` command in a list and processing it as needed. After Prometheus scrapes the gauges, the list is cleared, retaining only the most recent value. Each metric displays data for every vCPU on the selected host.

Table 5-14 cne_metrics_mpstat_idle_average

Field	Details
Description	Captures the <code>%idle</code> value every second, stores it on a list and returns the average of it.
Type	Gauge
Dimensions	host: vcpu:

Table 5-15 cne_metrics_mpstat_idle_low_watermark

Field	Details
Description	Captures the <code>%idle</code> value every second and stores it on a list and returns the lowest value.
Type	Gauge

Table 5-15 (Cont.) cne_metrics_mpstat_idle_low_watermark

Field	Details
Dimensions	host: vcpu:

Table 5-16 cne_metrics_mpstat_isoft_average

Field	Details
Description	Captures the % <i>soft</i> value every second, stores it on a list and returns the average of it.
Type	Gauge
Dimensions	host: vcpu:

Table 5-17 cne_metrics_mpstat_soft_high_watermark

Field	Details
Description	Captures the % <i>idle</i> value every second and stores it on a list and returns the highest value.
Type	Gauge
Dimensions	host: vcpu:

Network Metrics

These metrics are collected by storing the output of the `ip -s -s link ls {interface}` command in a list. The Gauge value is updated using the delta between the most recent entry and the previous one in the list. After Prometheus scrapes the gauges, the list is cleared, keeping only the latest value. Each metric displays data for every network interface on the selected host.

Table 5-18 cne_metrics_network_bytes_sent

Field	Details
Description	Captures the <i>delta</i> value for <i>TX</i> : <i>bytes</i> sent by each interface every second.
Type	Gauge
Dimensions	host: interface:

Table 5-19 cne_metrics_network_packets_sent

Field	Details
Description	Captures the <i>delta</i> value for <i>TX</i> : <i>packets</i> sent by each interface every second.
Type	Gauge

Table 5-19 (Cont.) cne_metrics_network_packets_sent

Field	Details
Dimensions	host: interface:

Table 5-20 cne_metrics_network_errors_sent

Field	Details
Description	Captures the <i>delta</i> value for <i>TX: errors</i> sent by each interface every second.
Type	Gauge
Dimensions	host: interface:

Reference: For more information on how to visualize these metrics on Grafana, see the "Grafana Dashboard" section.

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.
Severity	Critical
SNMP Trap ID	1001
Affects Service (Y/N)	N
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 shrunk 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 }}

Table 6-2 (Cont.) CPU_LOAD_HIGH

Field	Details
Cause	CPU load is more than 80% of the allocated resources on the node.
Severity	Major
SNMP Trap ID	1002
Affects Service (Y/N)	N
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.
Severity	Major
SNMP Trap ID	1007
Affects Service (Y/N)	N
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.
Severity	Critical
SNMP Trap ID	1008
Affects Service (Y/N)	Y
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.
Severity	Minor
SNMP Trap ID	1009
Affects Service (Y/N)	N
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: <code>sudo su;</code> 3. Run the following command: <code>systemctl restart chronyd;</code> 4. Watch chronyc tracking. 5. Run the following command: <code>sudo reboot</code> • 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.
Severity	Critical
SNMP Trap ID	1010
Affects Service (Y/N)	Y
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.
Severity	Major
SNMP Trap ID	1011
Affects Service (Y/N)	N

Table 6-7 (Cont.) PVC_NEARLY_FULL

Field	Details
Recommended Actions	1. Manually clean up PVC data for Prometheus: <pre>\$ 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 *</pre> For Oracle OpenSearch: <pre>\$ 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 *</pre> 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.
Severity	Critical
SNMP Trap ID	1012
Affects Service (Y/N)	Y
Recommended Actions	NA

Table 6-9 NODE_UNAVAILABLE

Field	Details
Description	Kubernetes node {{ \$labels.kubernetes_node }} is not in Ready state.
Summary	Kubernetes node {{ \$labels.kubernetes_node }} is unavailable.
Cause	Node is not in ready state.
Severity	Critical
SNMP Trap ID	1013
Affects Service (Y/N)	Y
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.
Severity	Critical
SNMP Trap ID	1014
Affects Service (Y/N)	Y
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.
Severity	Major
SNMP Trap ID	1036
Affects Service (Y/N)	N
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.
Severity	Critical
SNMP Trap ID	1037
Affects Service (Y/N)	Y
Recommended Actions	Contact Oracle support .

Table 6-13 CEPH_OSD_DOWN

Field	Details
Description	Storage node {{ \$labels.ceph_daemon }} is down.
Summary	Storage node {{ \$labels.ceph_daemon }} is down.
Cause	Ceph OSD is down.
Severity	Major
SNMP Trap ID	1038
Affects Service (Y/N)	Y
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.
Severity	Critical
SNMP Trap ID	1042
Affects Service (Y/N)	Y
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.

Table 6-15 (Cont.) OPENSEARCH_CLUSTER_HEALTH_RED

Field	Details
Cause	Some or all of the shards (primary) are not ready.
Severity	Critical
SNMP Trap ID	1043
Affects Service (Y/N)	Y
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: <pre>kubectl -n occne-infra exec -it occne-opensearch-client-0 -- curl localhost:9200/_cat/indices</pre> 2. Run the following command to delete the indices in yellow or red state: <pre>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/{}</pre> 3. Run the following command to verify if the indices with yellow or red state are deleted: <pre>kubectl -n occne-infra exec -it opensearch-client-0 -- curl localhost:9200/_cat/indices</pre> 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.

Table 6-16 (Cont.) OPENSEARCH_CLUSTER_HEALTH_YELLOW

Field	Details
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.
Severity	Major
SNMP Trap ID	1044
Affects Service (Y/N)	N
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. <pre>PUT /logstash-2021.08.21/_settings { "index" : { "number_of_replicas":0 } }</pre>

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.
Severity	Critical
SNMP Trap ID	1045
Affects Service (Y/N)	N
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 5, the value of the <code>opensearch_data_replicas_count = 5</code> 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 O/1 or "ImagePullBackOff" state due to insufficient memory/ CPU, or issues with image used by pod.
Severity	Critical
SNMP Trap ID	1006
Affects Service (Y/N)	Y
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 O/1 or "ImagePullBackOff" state due to insufficient memory/ CPU, or issues with image used by pod.
Severity	Critical
SNMP Trap ID	1050
Affects Service (Y/N)	Y
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	- 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. - OpenSearch cluster is unavailable.
Severity	Critical
SNMP Trap ID	1047
Affects Service (Y/N)	Y
Recommended Actions	- 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. - 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	- 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. - PVC is full.
Severity	Critical
SNMP Trap ID	1017
Affects Service (Y/N)	Y

Table 6-21 (Cont.) PROMETHEUS_DOWN

Field	Details
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. <pre>kubectl edit prometheus occne-kube-prom-stack-kube-prometheus -n occne-infra</pre> 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: <pre>\$ 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 *</pre>

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 "ImagePullBackOff" state due to insufficient memory/ CPU, or issues with image used by pod.
Severity	Critical
SNMP Trap ID	1018
Affects Service (Y/N)	Y

Table 6-22 (Cont.) ALERT_MANAGER_DOWN

Field	Details
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. <pre>kubectl edit alertmanager occne-kube-prom-stack-kube-alertmanager -n occne-infra</pre> 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.
Severity	Critical
SNMP Trap ID	1019
Affects Service (Y/N)	Y
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.
Severity	Critical
SNMP Trap ID	1020
Affects Service (Y/N)	N

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.
Severity	Major
SNMP Trap ID	1021
Affects Service (Y/N)	N
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.
Severity	Critical
SNMP Trap ID	1022

Table 6-26 (Cont.) METALLB_CONTROLLER_DOWN

Field	Details
Affects Service (Y/N)	Y
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.
Severity	Major
SNMP Trap ID	1024
Affects Service (Y/N)	Y
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 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.
Severity	Major
SNMP Trap ID	1028
Affects Service (Y/N)	N

Table 6-28 (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. <pre>kubectl edit prometheus occne-kube-prom-stack-kube-prometheus -n occne-infra</pre> 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: <pre>\$ 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 *</pre>

Table 6-29 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 "ImagePullBackOff" state due to insufficient memory/CPU or issues with the image used by pod.
Severity	Major
SNMP Trap ID	1029
Affects Service (Y/N)	N

Table 6-29 (Cont.) ALERT_MANAGER_NO_HA

Field	Details
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. <pre>kubectl edit alertmanager occne-kube-prom-stack-kube-alertmanager -n occne-infra</pre> 2. If resource utilization is not the issue, then contact Oracle support.

Table 6-30 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 "ImagePullBackOff" state due to insufficient memory/CPU, or issues with image used by pod.
Severity	Major
SNMP Trap ID	1032
Affects Service (Y/N)	Y
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 6-31 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.
Severity	Critical
SNMP Trap ID	1042
Affects Service (Y/N)	Y

Table 6-31 (Cont.) VMWARE_CSI_CONTROLLER_FAILED

Field	Details
Recommended Actions	Contact Oracle support .

Table 6-32 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.
Severity	Major
SNMP Trap ID	1051
Affects Service (Y/N)	Y
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 6-33 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 failed to respond to liveness tests.
Severity	Major
SNMP Trap ID	1040
Affects Service (Y/N)	N
Recommended Actions	Contact Oracle support .

Table 6-34 ALL_BASTION_HOSTS_FAILED

Field	Details
Description	All Bastion Hosts are unavailable.
Summary	All Bastion Hosts are unavailable.
Cause	All Bastion Hosts fail to respond to liveness tests.
Severity	Critical

Table 6-34 (Cont.) ALL_BASTION_HOSTS_FAILED

Field	Details
SNMP Trap ID	1041
Affects Service (Y/N)	Y
Recommended Actions	Contact Oracle support .

7

CNE Supported Features

This section explains about the CNE supported features.

CNE Features List with Configuration Status

The following tables list CNE features and their configuration status.

Table 7-1 CNE Features List with Configuration Status

Feature Name	Enabled or Disabled by default	Configuration File (occne.ini, cnlb.ini, cnlb-manager file, or a combination)	Licensing SKU Details
Log Externalization Support	Disabled	Configuration is managed through CNE inventory file (occne.ini)	Oracle Communications Cloud Native Core, Cloud Native Environment – 25K Active Subscribers Perpetual
CNLBCTL Utility Installation	Enabled	None	Oracle Communications Cloud Native Core, Cloud Native Environment – 25K Active Subscribers Perpetual
CNE API Server Proxy or Public Kubernetes API Endpoint	Disabled	Configuration is managed through CNE inventory file (occne.ini)	Oracle Communications Cloud Native Core, GitOps - Instance Perpetual
DNS Enhancement Feature	Disabled	Configuration is managed through CNE inventory file (occne.ini)	Oracle Communications Cloud Native Core, Advanced Cloud Native Environment – 25K Active Subscribers Perpetual
Activate or Deactivate Local DNS feature	Disabled	Configuration is managed through CNE inventory file (occne.ini)	Oracle Communications Cloud Native Core, Cloud Native Environment – 25K Active Subscribers Perpetual

Table 7-1 (Cont.) CNE Features List with Configuration Status

Feature Name	Enabled or Disabled by default	Configuration File (occne.ini, cnlb.ini, cnlb-manager file, or a combination)	Licensing SKU Details
Add or Remove DNS Record	Enabled	Configuration is managed using local DNS API	Oracle Communications Cloud Native Core, Cloud Native Environment – 25K Active Subscribers Perpetual

The following table lists the CNE deployment features and their corresponding configuration status. For more information on these features, refer to *Oracle Communications Cloud Native Environment Installation, Upgrade, and Fault Recovery Guide*.

Table 7-2 CNE Deployment Features List with Configuration Status

Feature Name	Enabled or Disabled by default	Configuration File (occne.ini, cnlb.ini, cnlb-manager file, or a combination)	Licensing SKU Details
CNLB Bypass for Bare Metal	Disabled	Configuration is managed through CNE inventory file (cnlb.ini)	Oracle Communications Cloud Native Core, Cloud Native Environment – 25K Active Subscribers Perpetual
CNLB Bypass for Bare Metal	Disabled	Configuration is managed through CNE inventory file (cnlb.ini)	Oracle Communications Cloud Native Core, Cloud Native Environment – 25K Active Subscribers Perpetual
CNLB Dynamic Network Attachment Support	Disabled	Configuration is managed through CNE inventory file (cnlb.ini)	Oracle Communications Cloud Native Core, Cloud Native Environment – 25K Active Subscribers Perpetual
CNLB Bypass for OpenStack	Disabled	Configuration is managed through CNE inventory file (cnlb.ini)	Oracle Communications Cloud Native Core, Cloud Native Environment – 25K Active Subscribers Perpetual
Floating IP Enable/Disable for OpenStack	Disabled	Configuration is managed through cluster.tfvars file	Oracle Communications Cloud Native Core, Cloud Native Environment – 25K Active Subscribers Perpetual

Table 7-2 (Cont.) CNE Deployment Features List with Configuration Status

Feature Name	Enabled or Disabled by default	Configuration File (occne.ini, cnlb.ini, cnlb-manager file, or a combination)	Licensing SKU Details
CNLB Nodes for vCNE OpenStack	Disabled	Configuration is managed through CNE inventory file (cnlb.ini)	Oracle Communications Cloud Native Core, Cloud Native Environment – 25K Active Subscribers Perpetual
CNLB Nodes for vCNE VMware	Disabled	Configuration is managed through cnlb.ini and cluster.tfvars files	Oracle Communications Cloud Native Core, Cloud Native Environment – 25K Active Subscribers Perpetual
CNLB Nodes for Bare Metal	Disabled	Configuration is managed through CNLB inventory file (cnlb.ini)	Oracle Communications Cloud Native Core, Cloud Native Environment – 25K Active Subscribers Perpetual
CNLB Dedicated Nodes for vCNE OpenStack	Disabled	Configuration is managed through occne.ini and cnlb.ini files	Oracle Communications Cloud Native Core, Cloud Native Environment – 25K Active Subscribers Perpetual
CNLB Dedicated Nodes for vCNE VMware	Disabled	Configuration is managed through occne.ini and cnlb.ini files	Oracle Communications Cloud Native Core, Cloud Native Environment – 25K Active Subscribers Perpetual
Enabling Contrack feature	Disabled	Configuration is managed using cnlb-manager file	Oracle Communications Cloud Native Core, Advanced Cloud Native Environment – 25K Active Subscribers Perpetual
Disabling Contrack feature	Disabled	Configuration is managed using cnlb-manager file	Oracle Communications Cloud Native Core, Advanced Cloud Native Environment – 25K Active Subscribers Perpetual
Scale CNLB App pods	Enabled	Configuration is managed using the cnlb.ini file	Oracle Communications Cloud Native Core, Cloud Native Environment – 25K Active Subscribers Perpetual

Table 7-2 (Cont.) CNE Deployment Features List with Configuration Status

Feature Name	Enabled or Disabled by default	Configuration File (occne.ini, cnlb.ini, cnlb-manager file, or a combination)	Licensing SKU Details
N+1 upgrade support	Enabled	None	Oracle Communications Cloud Native Core, Cloud Native Environment – 25K Active Subscribers Perpetual
N+2 Upgrade support	Enabled	None	Oracle Communications Cloud Native Core, Cloud Native Environment – 25K Active Subscribers Perpetual

8

Maintenance Procedures

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

8.1 Accessing the CNE

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

8.1.1 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.

8.1.1.1 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>
```

8.1.1.2 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>
```

8.1.1.3 Managing Bastion Host

The Bastion Host comes with the following built-in scripts to manage the Bastion Hosts. These scripts are available in the `/var/ocne/cluster/$OCCNE_CLUSTER/artifacts/` directory:

- `is_active_bastion`
- `get_active_bastion`
- `get_other_bastions`
- `update_active_bastion.sh`

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. Therefore, the commands will not run if the Kubernetes API is not reachable.

Subsequent sections provide the description for each of the commands.

For more information about the possible errors and troubleshooting, see [Troubleshooting Bastion Host](#).

8.1.1.3.1 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
```

8.1.1.3.2 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
```

8.1.1.3.3 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
```

8.1.1.3.4 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
```

8.1.1.4 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/var/occne/
cluster/occne1-rainbow/artifacts
```

8.2 General Configuration

This section describes the general configuration tasks for CNE.

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

- 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
```

8.2.2 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.

Prerequisites:

- Ensure that the cluster runs in a healthy state.
- The commands in this procedure must be primarily run from the active CNE Bastion Host unless specified otherwise.

Note

Ensure that the `secrets.ini` file is present and is rightly configured. Refer to the "Configuring `secrets.ini` and `occne.ini` Files" section in *Oracle Communications Cloud Native Core, Cloud Native Environment Installation, Upgrade, and Fault Recovery Guide*.

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

1. From the active Bastion, run the following command to instantiate a container that is used as a workspace and contains the necessary tools:

```
$ podman run -it --rm --rmi --network host --name DEPLOY_${OCCNE_CLUSTER} -
v /var/occne/cluster/${OCCNE_CLUSTER}:/host -e 'OCCNEINV=/host/hosts -i /
host/occne.ini' ${CENTRAL_REPO}:5000/occne/provision:${OCCNE_VERSION} bash
```

Note

The prompt changes from Bastion's prompt to Provision Container's prompt. For example, `bash-5.1`.

2. Run the following command to print the default primary interface name and make a note of it to be used in the following step:

Note

- The default interface name may vary depending on the platform.
- Ansible sets a primary interface as the default. This can be used to determine the default connection name for the internal network.
- The interface name must be consistent across all nodes in a cluster. This is the expected and default setting and you cannot proceed with the procedure if the interface names are different.

```
$ ansible -i $OCCNEINV k8s-cluster -m ansible.builtin.setup -a
'gather_subset=!all,network filter=ansible_default_ipv4' | grep -e
interface -e SUCCESS
```

Sample output:

```
occne-k8s-ctrl-1 | SUCCESS => {
  "interface": "eth0",
occne-k8s-ctrl-3 | SUCCESS => {
  "interface": "eth0",
occne-k8s-node-4 | SUCCESS => {
  "interface": "eth0",
occne-k8s-node-3 | SUCCESS => {
  "interface": "eth0",
occne-k8s-node-2 | SUCCESS => {
  "interface": "eth0",
```

```
occne-k8s-node-1 | SUCCESS => {
    "interface": "eth0",
occne-k8s-ctrl-2 | SUCCESS => {
    "interface": "eth0",
```

In this example, the default interface name is "eth0" and all the nodes have the same default interface name as expected.

3. Set up an environment variable using the default interface name obtained in the previous step:

```
$ export DEFAULT_NIC_NAME="<value collected from the above step>"
```

For example:

```
$ export DEFAULT_NIC_NAME="eth0"
```

You can verify the environment variable by running the following command:

```
$ echo $DEFAULT_NIC_NAME
```

4. Perform the following steps to modify the MTU value using ansible:
 - a. Run the following command to set the new MTU value as desired:

```
$ export NEW_MTU=<MTU value>
```

- b. Run the following ansible command to modify each interface using the environment variable:

```
$ ansible -i $OCCNEINV k8s-cluster -m shell -a 'sudo nmcli con mod "$
(nmcli -f GENERAL.CONNECTION -m tabular -t dev show
'$DEFAULT_NIC_NAME')" 802-3-ethernet.mtu '$NEW_MTU'; sudo nmcli con up
"$(nmcli -f GENERAL.CONNECTION -m tabular -t dev show
'$DEFAULT_NIC_NAME')"'
```

Sample output:

```
occne-k8s-node-3 | CHANGED | rc=0 >>
Connection successfully activated (D-Bus active path: /org/freedesktop/
NetworkManager/ActiveConnection/6)
occne-k8s-node-4 | CHANGED | rc=0 >>
Connection successfully activated (D-Bus active path: /org/freedesktop/
NetworkManager/ActiveConnection/6)
occne-k8s-node-2 | CHANGED | rc=0 >>
Connection successfully activated (D-Bus active path: /org/freedesktop/
NetworkManager/ActiveConnection/6)
occne-k8s-node-1 | CHANGED | rc=0 >>
Connection successfully activated (D-Bus active path: /org/freedesktop/
NetworkManager/ActiveConnection/6)
occne-k8s-ctrl-3 | CHANGED | rc=0 >>
Connection successfully activated (D-Bus active path: /org/freedesktop/
NetworkManager/ActiveConnection/6)
occne-k8s-ctrl-2 | CHANGED | rc=0 >>
```

```

Connection successfully activated (D-Bus active path: /org/freedesktop/
NetworkManager/ActiveConnection/6)
occne-k8s-ctrl-1 | CHANGED | rc=0 >>
Connection successfully activated (D-Bus active path: /org/freedesktop/
NetworkManager/ActiveConnection/6)

```

5. Run the following ansible commands to print the newly set MTU value across the cluster:

- a. Run the following command to show the connection MTU:

```

$ ansible -i $OCCNEINV k8s-cluster -m shell -a 'nmcli -f
GENERAL.NAME,802-3-ethernet.mtu con show "$(nmcli -f GENERAL.CONNECTION
-m tabular -t dev show '$DEFAULT_NIC_NAME')"'

```

Sample output:

```

occne-k8s-node-3 | CHANGED | rc=0 >>
802-3-ethernet.mtu:                8980
GENERAL.NAME:                      System eth0
occne-k8s-node-2 | CHANGED | rc=0 >>
802-3-ethernet.mtu:                8980
GENERAL.NAME:                      System eth0
occne-k8s-node-4 | CHANGED | rc=0 >>
802-3-ethernet.mtu:                8980
GENERAL.NAME:                      System eth0
occne-k8s-node-1 | CHANGED | rc=0 >>
802-3-ethernet.mtu:                8980
GENERAL.NAME:                      System eth0
occne-k8s-ctrl-1 | CHANGED | rc=0 >>
802-3-ethernet.mtu:                8980
GENERAL.NAME:                      System eth0
occne-k8s-ctrl-3 | CHANGED | rc=0 >>
802-3-ethernet.mtu:                8980
GENERAL.NAME:                      System eth0
occne-k8s-ctrl-2 | CHANGED | rc=0 >>
802-3-ethernet.mtu:                8980
GENERAL.NAME:                      System eth0

```

- b. Run the following command to show the MTU using IP command:

```

$ ansible -i $OCCNEINV k8s-cluster -m shell -a "ip link
show $DEFAULT_NIC_NAME"

```

Sample output:

```

occne-k8s-node-3 | CHANGED | rc=0 >>
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 8980 qdisc fq_codel
state UP mode DEFAULT group default qlen 1000
    link/ether fa:16:3f:eb:1b:f5 brd ff:ff:ff:ff:ff:ff
    altname enp0s3
    altname ens3
occne-k8s-node-4 | CHANGED | rc=0 >>
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 8980 qdisc fq_codel
state UP mode DEFAULT group default qlen 1000
    link/ether fa:16:3f:fd:b4:c6 brd ff:ff:ff:ff:ff:ff

```

```

    altname enp0s3
    altname ens3
occne-k8s-node-2 | CHANGED | rc=0 >>
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 8980 qdisc fq_codel
state UP mode DEFAULT group default qlen 1000
    link/ether fa:16:3f:82:0b:d7 brd ff:ff:ff:ff:ff:ff
    altname enp0s3
    altname ens3
occne-k8s-ctrl-1 | CHANGED | rc=0 >>
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 8980 qdisc fq_codel
state UP mode DEFAULT group default qlen 1000
    link/ether fa:16:3f:b5:41:e6 brd ff:ff:ff:ff:ff:ff
    altname enp0s3
    altname ens3
occne-k8s-node-1 | CHANGED | rc=0 >>
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 8980 qdisc fq_codel
state UP mode DEFAULT group default qlen 1000
    link/ether fa:16:3f:01:5f:c6 brd ff:ff:ff:ff:ff:ff
    altname enp0s3
    altname ens3
occne-k8s-ctrl-3 | CHANGED | rc=0 >>
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 8980 qdisc fq_codel
state UP mode DEFAULT group default qlen 1000
    link/ether fa:16:3f:48:1d:ac brd ff:ff:ff:ff:ff:ff
    altname enp0s3
    altname ens3
occne-k8s-ctrl-2 | CHANGED | rc=0 >>
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 8980 qdisc fq_codel
state UP mode DEFAULT group default qlen 1000
    link/ether fa:16:3f:d4:bb:23 brd ff:ff:ff:ff:ff:ff
    altname enp0s3
    altname ens3

```

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

① 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>
```

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"
```

Note

- The following `sudo nmcli con up` commands takes effect on the MTU values modified in the previous step. Ensure that you perform the following steps in the given order. Not following the correct order can make the host-2 and bastion-2 unreachable.
- The base interface has higher MTU during the sequence. For the `bond0` interface, `bond0` MTU is greater than or equal to the `bondbr0` MTU and `vlan` bridge interface. The `bond0.<vlan_id>` interface MTU is modified when `vlan<vlan_id>-br` interface is modified and restarted.

5. Run the following commands if the `<MTU value>` is lower than the old value:

```
$ sudo nmcli con up "vlan<mgmt vlan id>-br"
$ sudo nmcli con up bondbr0
$ sudo nmcli con up bond0
```

6. Run the following commands if the `<MTU value>` is higher than the old value:

```
$ sudo nmcli con up bond0
$ sudo nmcli con up bondbr0$
$ sudo nmcli con up "vlan<mgmt vlan id>-br"
```

7. 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.

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

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

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

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

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

10. Exit from host-2 and return to bastion-1. Wait until bastion-2 is reachable and run the following command to SSH to bastion-2:

```
$ ssh bastion-2
```

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

```
$ nmcli con show
```

12. 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 2.

```
$ 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"
```

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

```
$ ssh bastion-2
```

14. Repeat steps 9 and 10 to change the MTU value on k8s-host-1 and bastion-1.

15. 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/ocne/cluster/$
{OCCNE_CLUSTER}:/host winterfell:5000/ocne/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

vxlan.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 to perform a rollout restart:

```
$ 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/ocne/cluster/${OCCNE_CLUSTER}:/host winterfell:5000/ocne/provision:${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. Run the following command from the Bastion Host to edit configmap calico-config:

```
$ kubectl edit configmap calico-config -n kube-system "mtu": 1500, →
"mtu": <MTU value>,
```

2. Run the following command to restart daemonset:

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

3. The change in Calico interface MTU takes effect when a new pod on the node is started. The following codeblock provides an example to restart occne-kube-prom-stack-grafana deployment. Verify that the deployment is READY 1/1 before you delete and reapply:

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

4. 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    <none>      <none>
```

- Use SSH to log in to the node and check the calico interface change. Only the last interface MTU changes due to new pod for the services. Other Calico interface MTU changes when other services are changed.

```
$ ssh 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
```

5. Reboot all worker nodes if possible. In this case, all pods get restarted and all cali* interfaces have the new MTU:

```
$ ssh <worker node>$ sudo reboot
```

8.2.3 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	RECONCILED	AVAILABLE	AGE	VERSION	DESIRED	READY
occne-kube-prom-stack-kube-prometheus	True	True	20h	v2.44.0	2	2

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 RWX 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

```

8.2.4 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 (<code>log_trace_active_storage</code>)
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
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      RWX
Delete      Bound      occne-infra/occne-opensearch-cluster-master-
occne-opensearch-cluster-master-2  occne-esmaster-sc  17h
pvc-9d9897c1-b7b9-43a3-bf78-f03b91ea4d72  40Gi      RWX
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
RWX      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    RW0    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    RW0
Delete    Bound    occne-infra/occne-opensearch-cluster-data-occne-opensearch-cluster-data-2    occne-esdata-sc    17h
pvc-80a56d73-d7b7-417f-a7a7-c8484bc8171d    20Gi    RW0
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,...).

8.2.5 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 be 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.

8.2.5.1 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		

8.2.5.2 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		

8.2.5.3 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	

8.2.5.4 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
```

8.2.5.5 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.

8.2.5.6 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

```

8.2.5.7 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
```

8.2.5.8 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		

8.2.5.9 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.

8.2.6 Activating and Configuring Local DNS

This section provides information about activating and configuring local DNS.

8.2.6.1 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 25.1.20x.
- Once you activate Local DNS, you cannot rollback or deactivate the feature.

8.2.6.1.1 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 25.1.20x. 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.

8.2.6.1.2 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 the Active Bastion Host (for example, Bastion 1). Verify if the current Bastion is active:

```
$ 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
```

Determining the current state of the bastion_http_server service API

① Note

By default, after the post-installation procedure the bastion_http_server service is disabled and shut down.

1. Verify the current status of the bastion_http_server API by running the following command:

```
$ systemctl status bastion_http_server.service
```

The following sample output displays the service as inactive and disabled:

```
bastion_http_server.service - Bastion http server
   Loaded: loaded (/etc/systemd/system/bastion_http_server.service;
   disabled; preset: disabled)
   Active: inactive (dead)

May 06 23:01:39 occne-test-bastion-1.novalocal systemd[1]: Stopping
Bastion http server...
May 06 23:01:40 occne-test-bastion-1.novalocal unicorn[82034]:
[2025-05-06 23:01:39 +0000] [82034] [INFO] Handling signal: term
May 06 23:01:40 occne-test-bastion-1.novalocal unicorn[82057]:
[2025-05-06 23:01:39 +0000] [82057] [INFO] Worker exiting (pid: 82057)
May 06 23:01:40 occne-test-bastion-1.novalocal unicorn[82056]:
[2025-05-06 23:01:39 +0000] [82056] [INFO] Worker exiting (pid: 82056)
May 06 23:01:40 occne-test-bastion-1.novalocal unicorn[82055]:
[2025-05-06 23:01:39 +0000] [82055] [INFO] Worker exiting (pid: 82055)
May 06 23:01:41 occne-test-bastion-1.novalocal unicorn[82034]:
[2025-05-06 23:01:41 +0000] [82034] [INFO] Shutting down: Master
May 06 23:01:42 occne-test-bastion-1.novalocal systemd[1]:
bastion_http_server.service: Deactivated successfully.
```

```
May 06 23:01:42 occne-test-bastion-1.novalocal systemd[1]: Stopped Bastion
http server.
May 06 23:01:42 occne-test-bastion-1.novalocal systemd[1]:
bastion_http_server.service: Consumed 3min 37.802s CPU time.
```

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. If the variable is present, it should be included under the `occne:vars` header. If the variable setting is otherwise, then Local DNS is disabled.

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.

8.2.6.1.3 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. Run the following commands on Bastion Host to enable and start the `bastion_http_server` service:

```
$ sudo systemctl enable bastion_http_server.service
$ sudo systemctl start bastion_http_server.service
```

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

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

8.2.6.1.4 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/ocne/cluster/${OCCNE_CLUSTER}
```

2. Run the `validateLocalDns.py` script:

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

Sample output:

```
Beginning local DNS validation
- Validating local DNS configuration in ocne.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](#).

8.2.6.2 Deactivating Local DNS

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

1. Log in to Bastion Host (for example, Bastion 1) and determine if that Bastion Host is active or not. If the current Bastion is not active, then log in to the mate Bastion and verify if the mate Bastion is active or not.

Run the following command to check if Bastion Host is active or not:

```
$ is_active_bastion
IS active-bastion
```

2. Deactivating local DNS:

- a. On the active Bastion Host, change to the cluster directory from the existing directory.

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

- b. Edit the `ocne.ini` (vCNE) or `hosts.ini` (Bare metal) file.
Example for vCNE:

```
$ vi ocne.ini
```

Example for Bare Metal:

```
$ vi hosts.ini
```

- c. Set the `local_dns_enabled` variable under the `ocne:vars` header to `False`. If the `local_dns_enabled` variable is not present under the `ocne: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,

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

- d. Delete the CoreDNS configmap using the following command:

Note

Running the following command will delete all the records added using the API as part of Local DNS.

```
$ kubectl delete cm coredns -n kube-system
```

- e. Run the following commands on the bastion host to stop and disable the `bastion_http_server` service:

```
$ sudo systemctl stop bastion_http_server.service
$ sudo systemctl disable bastion_http_server.service
```

- f. Update the cluster with the new settings in the ini file:

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

8.2.6.3 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.

8.2.6.3.1 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.

8.2.6.3.2 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.

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

Table 8-1 Adding an A Record

Request URL	HT TP Me tho d	Content Type	Request Body	Response Code	Sample Response
http://localhost:8000/occne/dns/a	POST	application/json	<pre>{ "name": "string", "ttl": "integer", "ip-address": "string" }</pre> Note: Define each field in the request body within double quotes (""). Sample request: <pre>curl -X POST http:// localhost:8000/ occne/dns/a \ -H 'Content- Type: application/ json' \ -d '{"name":"occne.l ab.oracle.com","t tl":"3600","ip- address":"175.80. 200.20"}'</pre>	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 8-2 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.

8.2.6.3.3 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 must 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 8-3 Deleting an A Record

Request URL	HT TP Me tho d	Content Type	Request Body	Response Code	Sample Response
http://localhost:8000/occne/dns/a	DELETE	application/json	<pre>{ "name": "string", "ip-address": "string" }</pre> Note: Define each field in the request body within double quotes (""). Sample request: <pre>curl -X DELETE http:// localhost:8000/ occne/dns/a \ -H 'Content- Type: application/ json' \ -d '{"name":"occne.l ab.oracle.com","i p- address":"175.80. 200.20"}'</pre>	• 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 8-4 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.

8.2.6.3.4 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 8-5 Adding an SRV Record

Request URL	HT TP Me tho d	Content Type	Request Body	Response Code	Sample Response
https:// localhost:8000/ occne/dns/srv	PO ST	applicati on/json	{ "service": "string", "protocol": "string", "dn": "string", "ttl": "integer" "priority": "integer", "weight": "integer", "port": "integer", "server": "string", "a_record": "string" }	• 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.

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","s
erver":"occne","p
riority":"10","a_
```

Table 8-5 (Cont.) Adding an SRV Record

Request URL	HT TP Me tho d	Content Type	Request Body	Response Code	Sample Response
			record": "ocne.la b.oracle.com"}'		

The following table provides details about the request body parameters:

Table 8-6 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 8-6 (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 8-6 (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.

8.2.6.3.5 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 8-7 Deleting an SRV Record

Request URL	HT TP Me tho d	Content Type	Request Body	Response Code	Sample Response
https:// localhost:8000/ occne/dns/srv	DE LE TE	applicati on/json	{ "service": "string", "protocol": "string", "dn": "string", "ttl": "integer" "priority": "integer", "weight": "integer", "port": "integer", "server": "string" "a_record": "string" }	• 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

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","s
erver":"occne","p
riority":"10","a_
```

Table 8-7 (Cont.) Deleting an SRV Record

Request URL	HT TP Me tho d	Content Type	Request Body	Response Code	Sample Response
			<code>record": "ocne.la b.oracle.com"}'</code>		

The following table provides details about the request body parameters:

Table 8-8 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 8-8 (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.

8.2.6.4 Adding and Removing Forwarding Nameservers

The forward section added in the coreDNS forwards the queries to the nameservers if they are not resolved by coreDNS. This section provides information about adding or removing forwarding nameservers using the forward endpoint provided by Local DNS API.

8.2.6.4.1 Adding Forwarding Nameservers

This section provides information about adding forward section in coreDNS using the forward endpoint provided by Local DNS API.

Note

- Add forward section in coreDNS by using the payload data.
- Ensure that the forward section is not added already.

The following table provides details on how to use the Local DNS API endpoint to add forward section:

Table 8-9 Adding Forwarding Nameserver

Request URL	HT TP Me tho d	Content Type	Request Body	Response Code	Sample Response
https:// localhost:8000/ occne/dns/ forward	PO ST	applicati on/json	<pre>{ "ip-address": "string", }</pre> Note: Define each field in the request body within double quotes (""). Sample request to add forward section: <pre>curl -X POST http:// localhost:8000/ occne/dns/ forward \ -H 'Content- Type: application/ json' \ -d '{"ip- address":"192.168 .100.25,10.75.100 .5"}'</pre>	• 200: Forward successfully configured. • 400: Request payload/parameters incomplete or not valid. • 503: Could not configure, internal error.	200: SUCCESS: Forwarding nameserver list added to coredns successfully

The following table provides details about the request body parameters:

Table 8-10 Request Body Parameters

Parameter	Required or Optional	Type	Description
ip-address	Required	string	The IP addresses to forward the requests. You can define up to 15 IP addresses. The IP addresses must be valid IPv4 addresses and they must be defined as a comma separated list without extra space.

8.2.6.4.2 Removing Forwarding Nameservers

This section provides information about deleting forward section in coreDNS using the forward endpoint provided by Local DNS API.

① Note

- CNE doesn't support updating the forward section. If you want to update the forward nameservers, then remove the existing forward section and add a new one with the updated data.
- Before removing the forward section, ensure that there is already a forward section to delete.

The following table provides details on how to use the Local DNS API endpoint to remove forward section:

Table 8-11 Removing Forwarding Nameserver

Request URL	HT TP Me tho d	Content Type	Request Body	Response Code	Sample Response
https://localhost:8000/occne/dns/forward	DELETE	NA	Sample request to delete forward section: curl -X DELETE http://localhost:8000/occne/dns/forward	• 200: Forward configuration successfully removed. • 400: Request payload/parameters incomplete or not valid. Not configured, unable to remove. • 503: Could not configure, internal error.	200: SUCCESS: Forwarding nameserver list deleted from coredns successfully

8.2.6.5 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 8-12 Reloading Local or Core DNS Configurations

Request URL	HT TP Me tho d	Content Type	Request Body	Response Code	Sample Response
http://localhost:8000/occne/coredns/reload	POST	application/json	<pre>{ "deployment-name": "string", "namespace": "string" }</pre> 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): <pre>curl -X POST http:// localhost:8000/ occne/coredns/ reload</pre> Sample request to reload the core DNS using the payload: <pre>curl -X POST http:// localhost:8000/ occne/coredns/ reload \ -H 'Content-</pre>	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 8-12 (Cont.) Reloading Local or Core DNS Configurations

Request URL	HT TP Me tho d	Content Type	Request Body	Response Code	Sample Response
			<pre>Type: application/ json' \ -d '{"deployment- name":"coredns", " namespace":"kube- system"}'</pre>		

The following table provides details about the request body parameters:

Table 8-13 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 <i>coredns</i> .
namespace	Required	string	The namespace where the deployment exists. The value must be a valid Kubernetes namespace name. The default value is <i>kube-system</i> .

8.2.6.6 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 8-14 Get Local DNS or Core DNS Configurations

Request URL	HT TP Me tho d	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: <pre>[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.co m ' '201307231 3600 10800 86400 3600\n' 'occne1.us.orac le.com. , '3600 IN A ' '10.65.200.182\n , '_sip._tcp.lab.o</pre>

Table 8-14 (Cont.) Get Local DNS or Core DNS Configurations

Request URL	HT TP Me tho d	Content Type	Request Body	Response Code	Sample Response
					rac1e.com 30 IN SRV 10 102 32061 '
					'ocncne.lab.orac1 e.com.\n'
					'ocncne.lab.orac1 e.com. ,
					'3600 IN A '
					'175.80.200.20\n , ... # Output Omitted ...

8.2.6.7 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.

8.2.6.7.1 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 unicorn[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 unicorn[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 unicorn[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 unicorn[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 unicorn[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 unicorn[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 unicorn[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 unicorn[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 unicorn[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 8-15 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 have the same name. 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.

8.2.6.7.2 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-5nrvv -----
[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
```

8.2.6.7.3 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.

Note

After adding multiple records, use the API to reload the CoreDNS configuration so that the changes are applied.

Another example uses a pod created in a different namespace. This pod is not bundled with the CNE cluster by default.

- Run the following command from a Bastion server to query an A record. In this example, the A record for occne1.us.oracle.com was previously added 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
```

- 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 test-app-pod | awk '{print $1}') -- nslookup not-in.oracle.com
```

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 query resolves successfully through `kubernetes.default.svc.test` because the `kubernetes` service exists only in the default namespace.

- 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 test-app-pod | awk '{print $1}') -- nslookup -type=svr
kubernetes.default.svc.test
```

Sample output:

```
Server:      169.254.25.10
Address:     169.254.25.10:53
```

```
** server can't find _sip._tcp.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 test-app-pod | awk '{print $1}') -- nslookup kubernetes
```

Sample output:

```
erver:      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 test-app-pod | awk '{print $1}') -- nslookup -type=svr
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.

8.2.6.7.4 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'},
```

8.2.6.7.5 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:

Note

Refer to the procedure in [Activating Local DNS](#) that explains how to enable local DNS and provides the `validateLocalDns.py` validation script.

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 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

- Validating local DNS configuration in `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 8-16 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: <i>name:occne.dns.local.com, ip-address: 10.0.0.3</i>	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: <i>name:occne.dns.local.com, ip-address: 10.0.0.3</i>	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 .
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.

8.3 Managing the Kubernetes Cluster

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

8.3.1 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.
- The backup will only create an CNE Cluster backup that contains Bastion Host data, including Kubernetes.
- Cluster must be in a good state, that is all content included in the following namespaces must be up and running:
 - `occne-infra`
 - `kube-system`
 - `rook-ceph` (for Bare Metal)
- All bastion-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
 - kube-system
 - rook-ceph (for Bare Metal)
- You must take the Bastion backup in the ACTIVE Bastion only.

8.3.1.1 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.

Note

Before creating a backup, Velero must be enabled.

Cluster backup procedure consists of the following two main stages:

1. Bastion Backup
2. Velero Kubernetes Backup

Both are created in order when the `createClusterBackup.py` initializes.

Procedure

1. Access the active Bastion using `ssh`.
2. 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
```

Note

If the active Bastion is not being used, log in to the other Bastion, and continue this procedure.

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

Note

The boto3 library is required for the backup procedure. This is only required for CNE versions 23.3 and higher.

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

Sample output:

```
Initializing cluster backup occne-example-20250310-145503
```

```
Creating bastion backup: 'occne-example-20250310-145503'
```

```
Successfully created bastion backup
```

```
GENERATED LOG FILE AT: /var/occne/cluster/occne-example/logs/velero/backup/  
createBastionBackup-20250310-145503.log
```

```
Creating velero backup: 'occne-example-20250310-145503'
```

```
Successfully created velero backup
```

```
Successfully created cluster backup
```

```
GENERATED LOG FILE AT: /var/occne/cluster/occne-example/logs/velero/backup/  
createBastionBackup-20250310-145503.log
```

4. Verify that the backup tar file was generated at /home/\${USER}.

```
$ ls ~
```

Sample output:

```
createBastionBackup-20250310-144552.log  occne4-  
example-20250310-145503.tar             OCCNE_PIPELINE_2025-03-06_234008.log  
createBastionBackup-20250310-145503.log  
OCCNE_PIPELINE_2025-03-06_200544.log  
OCCNE_PIPELINE_2025-03-08_010753.log  
createClusterBackup-20250310-144552.log  
OCCNE_PIPELINE_2025-03-06_222802.log  
createClusterBackup-20250310-145503.log  
OCCNE_PIPELINE_2025-03-06_233319.log
```

8.3.1.2 Verifying Backup in S3 Bucket

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

Log in into the S3 cloud storage that was used to save the backup for verifying that the Bastion backup was successfully uploaded.

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*.

8.3.2 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. These 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 8-17 Kubernetes Internal Certificates and Validity Period

Node Type	Component Name	.crt File Path	Validity (in years)	.pem File Path	Validity (in years)
Kubernetes Controller	etcd	<code>/etc/pki/ca-trust/source/anchors/etcd-ca.crt</code>	100	<code>/etc/ssl/etcd/ssl/admin-<node_name>.pem</code>	100
Kubernetes Controller	etcd	NA	NA	<code>/etc/ssl/etcd/ssl/ca.pem</code>	100
Kubernetes Controller	etcd	NA	NA	<code>/etc/ssl/etcd/ssl/member-<node_name>.pem</code>	100

Table 8-17 (Cont.) Kubernetes Internal Certificates and Validity Period

Node Type	Component Name	.crt File Path	Validity (in years)	.pem File Path	Validity (in years)
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
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.

Prerequisites

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.

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 `kubect`. 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
Running            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. Recreate the secrets that contains the `admin.config`:

- a. **CNLB only:**

- i. Delete the existing `cnlb-man-admin-config` secret and recreate a new one from the new `admin.conf` file.

Run the following command to delete the existing `cnlb-man-admin-config` secret:

```
kubectl -n occne-infra delete secret cnlb-man-admin-config
```

Run the following command to recreate a new `cnlb-man-admin-config` secret from the new `admin.conf` file.

```
kubectl -n occne-infra create secret generic cnlb-man-admin-config
--from-file=/var/occne/cluster/${OCCNE_CLUSTER}/artifacts/admin.conf
```

- ii. Patch the `cnlb-man-admin-config` secret and restart the `cnlb-manager` pod and `cnlb-app` pods:

```
$ kubectl -n occne-infra patch --type=merge secret cnlb-man-admin-
config --patch '{"data":{"admin.conf":"'$(kubectl -n occne-infra
get secret cnlb-man-admin-config -o jsonpath='{.data.admin\conf}'
| base64 -d | sed 's/https:\\\\lb-apiserver.kubernetes.local:6443/
https:\\\\kubernetes.default:443/g' | base64 -w0)'"}}'
```

Run the following command to obtain the number of replicas of the `cnlb` app pods:

```
$ kco get deploy | grep cnlb
```

Run the following command to remove the `cnlb` manager and app pods:

```
$ kco scale deploy cnlb-manager --replicas=0
$ kco scale deploy cnlb-app --replicas=0
```

Restore the `cnlb` manager and app pods:

```
$ kco scale deploy cnlb-manager --replicas=1
$ kco scale deploy cnlb-app --replicas=<Number of replicas obtained
previously>
$ kco scale deploy cnlb-app --replicas=<Number of replicas obtained
previously>
```

8.3.2.1 Renewing Kyverno Certificates

Kyverno 1.9.0 doesn't support automatic certificate renewal. Therefore, if you are using Kyverno 1.9.0, you must renew the certificates manually. This section provides the procedure to renew Kyverno certificates.

1. Renew the Kyverno certificates by deleting the secrets from the kyverno namespace:

```
$ 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
```

2. 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	AGE	TYPE
occne-kyverno-svc.kyverno.svc.kyverno-tls-ca	2	kubernetes.io/tls
occne-kyverno-svc.kyverno.svc.kyverno-tls-pair	2	kubernetes.io/tls
sh.helm.release.v1.occne-kyverno-policies.v1	1	helm.sh/release.v1
sh.helm.release.v1.occne-kyverno.v1	1	helm.sh/release.v1

- 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 $1'}); do currdte=$(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 $currdte ]; 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
```

8.3.2.2 Renewing Kubelet Server Certificates

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.

Note

Perform this procedure from the active Bastion.

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. 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.
```

8.3.3 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

Note

Secret encryption is enabled by default during CNE installation or upgrade.

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_Z0J1Hf50Cx
          secret: l+CaDTmMkC85LwJRiWJ0LQPYVt0yZ0TdtNZ2ij+kuGA=
        - name: key
          secret: ZXJ1Ulk2U0xSbWkwejdreTlJWkFrZmpJZjhBRzg4U00=
    - identity: {}
```

- 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"
```

- 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/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/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-config 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-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 encounter some errors on the output depending on how the secret is created. You can ignore these errors and verify that the encrypted secret's key is replaced with a new one using the following steps.

4. Run the following command from a controller node with any cert pem and key pem pair files to show all existing secrets:

```

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 --keys-only=true --
prefix /registry/secrets

```

Select any secret path you want to verify from the output.

5. To verify that the secret is using the newly generated key, run the following command from a controller node. Replace <cert pem file>, <key pem file> and <secret-path> 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 <secret-path> -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_Z0J1Hf50Cx:&9\x90\u007f'*6\x0e\xf8]\x98\xd7t1\xa9  
|\x90\x93\x88\xebc\xa9\xfe\x82<\xebhxaa\x17$  
\xa4\x14%mb7<\x1d\xf7N\b\xa7\xbaZ\xb0\xd4#\xbbev)  
\x1bv9\x19\xdel\xab\x89@\xe7\xaf$L\xb8)\xc9\x1b\x13\xc1V\x1b\xf7\bX\x88\xe  
7ue131\x1dG\xe2_\x04\xa2\xf1n\xf5\x1dP\  
\4\xe7)^ \x81go\x99\x98b\xbb\x0ehxc0R;>η  
j\xeeV54\xac\x06ht\x1b9\xd5N\xa77\xd9\x03 \x05\xfb%\xa1\x81\xd5\x0e  
\xcax\xc4\x1cz6\xf3\xd8\xf9?Щ\x9a%  
\x9b\xe5\xa7 \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\xd50l\xb7\x87BW\tY;寄  
ηxcaηx87Y;\n;/  
\xf2\x89\xa1\xcc\xc3\xc9\xe3\xc5\v\x1b\x88\x84  
ηxc6\x00\xb4\xed\xa5\xe2\xfa\xa9\xff \xd9kηxf2\x04\x8f\x81, l"
```

This example shows a new key, key_Z0J1Hf50Cx, being used to encrypt secret1 secret.

8.3.4 Adding a Kubernetes Worker Node

This section describes how to add a Kubernetes node (formally called a "worker" node) to an existing CNE Kubernetes cluster. The procedure is applicable to both CNLB and non-CNLB clusters. However, certain steps may differ depending on that.

Note

- For a BareMetal installation, ensure that you are familiar with the inventory file preparation procedure. For more information about this procedure, see the "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.

Limitations

- **vcNE CNLB deployment:** Node names are defined as a list of string identifiers in `cluster.tfvars`. Each identifier can contain up to four characters. By default, a single-character identifier is used and must be enclosed in quotation marks.

- **BareMetal Deployment:** Only one node can be added at a time. Nodes are defined in the `hosts.ini` file, and no specific sequence is required. However, duplicate entries are not allowed. If two or more nodes were removed using the `removeBmWorkerNode.py` script, comment out the other removed nodes in `hosts.ini` and add them back one at a time.
- Ensure that the cluster is in a healthy state before performing this procedure. No nodes should be in a `NotReady` state. Remove any unreachable nodes before proceeding.

Adding a Kubernetes Worker Node on vCNE (OpenStack and VMware)

This section provides procedures to add Kubernetes worker nodes on vCNE CNLB and vCNE VMWare deployments.

Pre-procedure:

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.
Run 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/ocne/cluster/${OCNE_CLUSTER}/
```

3. Ensure the `secrets.ini` file is present and properly populated.

Perform the following steps to add a Kubernetes node for vCNE CNLB deployments

Note

CNE now supports the addition of medium Kubernetes worker nodes and CNLB nodes. Adding multiple nodes at a time is also supported.

If this feature was enabled during installation, you can use the script to add a standard Kubernetes worker node, a medium Kubernetes worker node, or a CNLB node to the cluster.

If this feature is disabled, you can use the script to add only a standard Kubernetes worker node or a medium Kubernetes worker node.

1. Run the following command to open the `$OCNE_CLUSTER/cluster.tfvars` file.
Modify only one of the fields at a time, and add a new "name" to the lists of node names. The items in a given list cannot be repeated within the same list, and each item must have a maximum of 4 characters (lowercase). OpenTofu sorts each list alphabetically, turning it into a set, a new node can be added regardless of being first, second or last in the set.

Table 8-18 Node Names

Field	Example: Old value	Example: New value	Valid only if
occne_node_names	["1", "2", "3"]	["1", "2", "3", "4"] or ["0", "1", "2", "3"]	All types of vCNE CNLB deployments
medium_node_names	["1", "2", "3", "4", "5"]	["1", "2", "3", "4", "5", "6"] or ["1", "2", "a", "3", "4", "5"]	This is an optional field. All types of vCNE CNLB deployments.
cnlb_node_names	["1", "2", "3", "4", "5"]	["1", "2", "3", "4", "5", "6"] or ["1", "2", "a", "3", "4", "5"]	CNLB Nodes feature is enabled: cnlb_node_label = True

Edit the `cluster.tfvars` file.

```
$ vi $OCCNE_CLUSTER/cluster.tfvars
```

Sample output:

Adding a new node name at the end of the list

```
...
# K8s nodes
#
occne_node_names = ["1", "2", "3", "4"]
...
```

Adding a new node name at the middle of the list

```
...
# Medium nodes
#
occne_medium_node_names = ["1", "2", "a", "3", "4", "5"]
```

Adding a new node name at the end of the list

```
...
# CNLB nodes
#
cnlb_node_names = ["0", "1", "2", "3", "4", "5"]
```

2. Run the `addK8sNode.py` script.

Note

- Log file location:** Information will be displayed on screen and logged to a log file. This file is located at the path, `/var/ocne/cluster/${OCCNE_CLUSTER}/logs/maintenance/add_cnlb_k8s_node_<yyyy-mm-dd_hhmmss>.log`.
 Long-running processes (such as cluster tests or the node provision) will not appear in the file until they have finished. The process must complete before its entry is recorded.

 The standard output (stdout) and standard error output (stderr) commands will be logged to the file, each on a separate line.
- Backup file location:** The script creates a backup of the `cluster.tfvars`, `terraform.tfstate` files and saves them to the `/var/ocne/cluster/${OCCNE_CLUSTER}/backup_config/add_cnlb_k8s_node_<yyyy-mm-dd_hhmmss>/` location.
 By default, each run of the script creates a dedicated directory and copy of the files.

 The `terraform.tfstate` file is renamed to `terraform.tfstate.ORIG` to prevent it from being used by the inventory.

- 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.

- Source the `openrc.sh` file.

```
$ source openrc.sh
```

- 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>:
```

- Run the `addK8sNode.py` script. Adding any type of node will produce similar outputs, regardless of whether it is a CNLB node or a regular Kubernetes node.

```
$ ./scripts/addK8sNode.py
```

Sample output for both OpenStack and VMware:

- Initializing Add K8s Node(s) Process
- Backing up configuration files...
 Backup files location: `/var/ocne/cluster/test-cluster/backup_config/add_cnlb_k8s_node_2025-09-05_162745`
- Log file location: `/var/ocne/cluster/test-cluster/logs/maintenance/add_cnlb_k8s_node_2025-09-05_162745.log`, long-running processes will write to it as they finish.
- Initializing tofu...

```

Tofu successfully initialized.

- Update tfstate (tofu plan)...
  Tofu successfully planned.

- List of new or affected nodes: test-cluster-k8s-node-4

- Testing cluster health...
  Test pipeline completed successfully.

- Setting maintenance banner...
  Maintenance banner set successfully.

- Updating infrastructure (tofu apply)...
  Tofu successfully applied.

- Updating /etc/hosts files...
  Successfully updated file: /etc/hosts on all servers.

- Running provision container (pipeline.sh) - may take considerable time
to complete...
  Provision pipeline completed successfully.

- Running Cluster Test: Provision...
  Test pipeline completed successfully.

- Running k8s_install container (pipeline.sh) - may take considerable
time to complete...
  K8s pipeline completed successfully.

- Running Cluster Test: Kubernetes...
  Test pipeline completed successfully.

- Running Cluster Test: Common Services...
  Test pipeline completed successfully.

- Restoring default banner...
  Successfully restored default banner.

- Add K8s Node(s) successfully completed - Check the log file for
details: /var/ocne/cluster/test-cluster/logs/maintenance/
add_cnlb_k8s_node_2025-09-05_162745.log

```

Adding a Kubernetes Worker Node in BareMetal Deployment (MetalLB and CNLB)

Pre-procedure

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.
Run 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/ocne/cluster/${OCCNE_CLUSTER}/
```

3. Ensure the `secrets.ini` file is present and properly populated.
4. Ensure the correct Oracle Linux ISO image is available.
 - a. Verify that the Oracle Linux ISO image exists and that it corresponds to the desired version for the new node.
 - b. Verify its permissions and ownership so that Ansible can read it.

```
$ ls -l /var/ocne/os
```

Sample output:

```
total 12305408
-rw----- 1 admusr admusr 12600737792 Mar 21 22:34 OracleLinux-R9-U7-
x86_64-dvd.iso
```

- c. If the ISO is missing, download it from the central repository before proceeding.
5. Verify that the cluster is healthy.

Note

vCNE CNLB script includes this pre-test.

```
$ OCCNE_STAGES=(TEST) pipeline.sh
```

1. Edit the `hosts.ini` file by adding a new line for `<NODE_FULL_NAME>` along with all required properties. Add it under the appropriate hardware sections:

- `[host_hp_gen_X]` for HP
- `[host_netra_X]` for Netra hardware

The `hosts.ini` file must follow these validation rules:

- a. The required parameters are `<NODE_FULL_NAME>`, `ansible_host`, either `hp_ilo` or `netra_ilm`, and `mac`. The values for these parameters must be unique across the `hosts.ini` file.
- b. The `<mac-address>` must be a string of six two-digit hexadecimal numbers separated by a dash.
- c. All IP address values must use valid IPv4 format.
- d. Optional properties include: `pxe_config_ks_nic`, `pxe_config_nic_list`, `pxe_uefi`, and any extra properties not listed in the documentation.
- e. All properties, whether required or optional, must follow the format, `<KEY>=<VALUE>`. There must be no spaces before or after the equals sign. For example, `ansible_host=192.168.1.10`

- f. Comments must be added on a separate line using #. Inline comments are not allowed.
For more information on how to fill the inventory file, see the "Inventory file preparation" section in *Oracle Communication Cloud Native Core, Cloud Native Environment Installation, Upgrade, and Fault Recovery Guide*.

- i. Open the `hosts.ini` file.

```
$ vi hosts.ini
```

- ii. Identify the hosts group:

```
[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
```

Example:

```
<NODE_FULL_NAME>=k8s-node-5.test.us.oracle.com
```

Sample output:

```
...
[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
...
```

- 2. Add the `<NODE_FULL_NAME>` under the `[kube-node]` header, then save and exit.

- a. Open the `hosts.ini` file.

```
$ vi hosts.ini
```

- b.** Add the new node name:

```
[kube-node]
k8s-node.example.oracle.com
```

Example:

<NODE_FULL_NAME>=k8s-node-5.test.us.oracle.com

Sample output:

```
...
[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
...
```

3. **For BareMetal CNLB only:** Add the new node in the `hosts.ini` `[cnlb_node]` section if it is to be added as `cnlb_node`.

- a. Open the `hosts.ini` file.

```
$ vi hosts.ini
```

- b.** Add the new node name:

```
[cnlb_node]
k8s-node.example.oracle.com
```

Example:

<NODE_FULL_NAME>=k8s-node-5.test.us.oracle.com

Sample output:

```
...
[cnlb_node]
k8s-node-2.test.us.oracle.com
k8s-node-4.test.us.oracle.com
# New node
k8s-node-5.test.us.oracle.com
```

```
[cnlb:children]
k8s-cluster
occne_bastion
```

4. Run the `addBmWorkerNode.py` script using the same `<NODE_FULL_NAME>` that was added to the `hosts.ini` file.

Note

The script displays information on the screen and also logs it to a file at the following location:

```
/var/occne/cluster/${OCCNE_CLUSTER}/addBmWkrNodeCapture-  
<mmddyyyy_hhmmss>.log
```

Long-running processes, such as node provisioning, are written to the log file only after they complete.

```
$ scripts/maintenance/addBmWorkerNode.py -nn <NODE_FULL_NAME>
```

Example:

```
$ scripts/maintenance/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 does not exist.
  - Creating the rook-ceph toolbox deployment.
    - rook-ceph toolbox created.
- Wait for Toolbox pod
- Waiting for Toolbox pod to be in Running state.
  - Waiting for toolbox pod in namespace rook-ceph to be in Running
state, current state: Pending
  - 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

Post Verification

1. Run the following commands to verify if the node is added successfully:

```
$ kubectl get nodes
```

Example output for BareMetal:

NAME	STATUS	ROLES	AGE	VERSION
k8s-master-1.test.us.oracle.com	Ready	control-plane	8d10h	v1.33.1
k8s-master-2.test.us.oracle.com	Ready	control-plane	8d10h	v1.33.1
k8s-master-3.test.us.oracle.com	Ready	control-plane	8d10h	v1.33.1
k8s-node-1.test.us.oracle.com	Ready	<none>	8d10h	v1.33.1
k8s-node-2.test.us.oracle.com	Ready	<none>	8d10h	v1.33.1
k8s-node-4.test.us.oracle.com	Ready	<none>	8d10h	v1.33.1
k8s-node-5.test.us.oracle.com	Ready	<none>	15m	

v1.33.1 # <----- Example: New node

Example output for vCNE OpenStack CNLB (CNLB Nodes enabled):

NAME	STATUS	ROLES	AGE	VERSION
test-cluster-k8s-ctrl-1	Ready	control-plane	45h	v1.33.1
test-cluster-k8s-ctrl-2	Ready	control-plane	45h	v1.33.1
test-cluster-k8s-ctrl-3	Ready	control-plane	45h	v1.33.1
test-cluster-k8s-node-1	Ready	<none>	45h	v1.33.1
test-cluster-k8s-node-2	Ready	<none>	45h	v1.33.1
test-cluster-k8s-node-a	Ready	<none>	9m	v1.33.1

<----- Example: New node

test-cluster-k8s-node-cnlb-1	Ready	<none>	45h	v1.33.1
test-cluster-k8s-node-cnlb-2	Ready	<none>	45h	v1.33.1
test-cluster-k8s-node-cnlb-3	Ready	<none>	45h	v1.33.1
test-cluster-k8s-node-cnlb-4	Ready	<none>	45h	v1.33.1
test-cluster-k8s-node-cnlb-5	Ready	<none>	45h	v1.33.1

2. Verify that the cluster is healthy. The following are simple examples.
 - a. Retrieve all nodes from all namespaces and verify that they are all up and running.

```
$ kubectl get pods -A
```

- b. For MetalLB Only, verify that all services are assigned and accessible.

```
$ kubectl get svc -A
```

Troubleshooting

If an issue occurs while adding a new node, it is strongly recommended to contact the support team.

Advanced Options for vCNE CNLB

The vCNE CNLB script, `addK8sNode.py`, supports additional arguments that can help troubleshoot common issues or resume a node addition process after a previous failure.

Use the script's help option for more information about each argument.

```
$ scripts/addK8sNode.py -h
```

Sample output:

```
usage: addK8sNode.py [-h] [-d] [-nb] [-nr] [-r] [-s {PRE,PROV,K8S,CFG}
[{{PRE,PROV,K8S,CFG} ...}] [-R] [-D]
```

Add one or more K8s nodes or CNLB K8s nodes (control plane nodes are not supported) to an OCCNE cluster.

Optional arguments:

- d, --debug Debug mode.
- nb, --no-backup Do not create backup.
- nr, --include-not-ready Include NotReady nodes in the provisioning and k8s_install procedure. Allows the script to resume from a previous failure or unhealthy cluster state. Use with caution.
- r, --resume Allow the script to resume after a previous failure.
- s {PRE,PROV,K8S,CFG} [{{PRE,PROV,K8S,CFG} ...}], --skip-test {PRE,PROV,K8S,CFG} [{{PRE,PROV,K8S,CFG} ...}] Do not test the cluster health. By default, all tests are run; specify individual tests to be skipped.
- R, --refresh Allow OpenTofu to synchronize its state with current infrastructure resources. Internally sends a -refresh-only command to both tofu plan and apply commands.
- D, --dry-run Test mode. Identifies new or affected nodes without applying changes. Stops in the init phase.

Examples:

```
./addK8sNode.py
./addK8sNode.py --debug
./addK8sNode.py -nb -d -nr -r
./addK8sNode.py --debug --no-backup --dry-run
./addK8sNode.py --include-not-ready --resume
./addK8sNode.py --resume --refresh
./addK8sNode.py -d -nr --skip-test PRE CFG --resume -R
```

8.3.5 Removing a Kubernetes Worker Node

This section describes how to remove a Kubernetes node (formally called a "worker" node) to an existing CNE Kubernetes cluster. The procedure is applicable to both CNLB and non-CNLB clusters. However, certain steps may differ depending on that.

Prerequisites

- 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.

Points to note

- For dedicated CNLB nodes in a CNE cluster, a minimum of three nodes is recommended to provide resilience. At least two CNLB nodes are required for the CNE cluster to function.
- **vcNE CNLB deployment:** Only one node can be removed at a time.
- **BareMetal deployment:** Only one node can be removed at a time.
- **All platforms:** It is possible to run this procedure multiple times in a row to remove various nodes from the cluster, however there must be a minimum number/set of nodes available for the cluster to function correctly. See the prerequisites section for more information.
- **Bare Metal:** CNE does not support standard recovery procedures when more than one node enters the **NotReady** state. In such cases, the cluster is considered **impaired**, and the customer must raise a support ticket for further assistance.

Pre-procedure:

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.
Run 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. Ensure the `secrets.ini` file is present and properly populated.

Perform the following steps to remove a Kubernetes node for vcNE CNLB deployments (Openstack and VMware)

Note

- The procedure and script are applicable to “CNLB nodes”.
- If this feature was previously enabled (only during installation), the script can be used to remove either a new Kubernetes node or a CNLB node to the cluster.
- If the feature is disabled, this script can be used to remove a standard Kubernetes node.

1. Run the following command to open the `$OCCNE_CLUSTER/cluster.tfvars` file. Modify only one of the fields at a time, and add a new "name" to the lists of node names. The items in a given list cannot be repeated within the same list, and each item must have a maximum of 4 characters (lowercase). OpenTofu sorts each list alphabetically, turning it into a set, a new node can be added regardless of being first, second or last in the set.

Table 8-19 Node Names

Field	Example: Old value	Example: New value	Valid only if
occne_node_names	["1", "2", "3"]	["1", "2", "3", "4"] or ["0", "1", "2", "3"]	All types of vCNE CNLB deployments
cnlb_node_names	["1", "2", "3", "4", "5"]	["1", "2", "3", "4", "5", "6"] or ["1", "2", "a", "3", "4", "5"]	CNLB Nodes feature is enabled: <code>cnlb_node_label = True</code>

Edit the `cluster.tfvars` file.

```
$ vi $OCCNE_CLUSTER/cluster.tfvars
```

Sample output: Removing a node name at the end of the list

```
...
# K8s nodes
#
occne_node_names = ["1", "2", "3", "4"]
...
```

Sample output: Removing a node name at the beginning of the list

```
...
# CNLB nodes
#
cnlb_node_names = ["2", "3", "4", "5"]
```

2. Run the `removeK8sNode.py` script.

Note

- **Log file location:** Information will be displayed on screen and logged to a log file. This file is located at the path, `/var/ocne/cluster/${OCCNE_CLUSTER}/logs/maintenance/remove_cnlb_k8s_node_<yyyy-mm-dd_hhmmss>.log`. Long-running processes (such as cluster tests or the node provision) will not appear in the file until they have finished. The process must complete before its entry is recorded.

The standard output (stdout) and standard error output (stderr) commands will be logged to the file, each on a separate line.
- **Backup file location:** The script creates a backup of the `cluster.tfvars`, `terraform.tfstate` files and saves them to the `/var/ocne/cluster/${OCCNE_CLUSTER}/backup_config/remove_cnlb_k8s_node_<yyyy-mm-dd_hhmmss>/` location.
By default, each run of the script creates a dedicated directory and copy of the files.

The `terraform.tfstate` file is renamed to `terraform.tfstate.ORIG` to prevent it from being used by the inventory.

3. **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>:
```

Note

It is assumed that the provided credentials and the `openrc.sh` file are current, valid, and have the necessary permissions.

4. Verify that the cluster is healthy.

Note

As mentioned in the introduction, while it is possible to run this procedure even if the node to be removed is considered unhealthy, it is recommended to perform this step to check the cluster health. The vCNE CNLB script includes this health pre-check by default.

```
$ OCCNE_STAGES=(TEST) pipeline.sh
```

5. Run the `removeK8sNode.py` script. removing any type of node will produce similar outputs, regardless of whether it is a CNLB node or a regular Kubernetes node.
Run the following script to remove a Kubernetes node:

```
./scripts/removeK8sNode.py
```

Sample output for both OpenStack and VMware:

Initializing Remove K8s Node(s) Process

- Backing up configuration files...
Backup files location: `/var/ocne/cluster/test-cluster/backup_config/remove_cnlb_k8s_node_2025-09-10_003841/`
- Log file location: `/var/ocne/cluster/test-cluster/logs/maintenance/remove_cnlb_k8s_node_2025-09-10_003841.log`, long-running processes will write to it as they finish.
- Initializing tofu...
Tofu successfully initialized.
- Update tfstate (tofu plan)...
Tofu successfully planned.
- List of nodes to be removed: `cluster-test-k8s-node-cnlb-2`
- PRE: Testing cluster health...
Test pipeline completed successfully.
- Setting maintenance banner...
Maintenance banner set successfully.
- Cordoning node(s)...
Node(s) successfully cordoned.
- Draining node(s) - can take considerable time to complete...
Node(s) successfully drained.
- Removing node(s) from Kubernetes cluster...
Node(s) removed successfully from Kubernetes.
- Updating infrastructure (tofu apply)...
Tofu successfully applied.
- Update `/etc/hosts` files on all remaining servers...
Update `/etc/host` successful.
- POST: Testing cluster health...
Test pipeline completed successfully.
- Restoring default banner...
Successfully restored default banner.
- Remove K8s Node(s) successfully completed - Check the log file for

```
details: /var/ocne/cluster/test-cluster/logs/maintenance/  
remove_cnlb_k8s_node_2025-09-10_003841.log
```

Remove a Kubernetes Worker Node in BareMetal Deployment (MetalLB and CNLB)

Note

For any failure or successful run, the system maintains all pipeline outputs in the `/var/ocne/cluster/${OCCNE_CLUSTER}/removeWrkNodeCapture-<mmddyyy_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/ocne/cluster/${OCCNE_CLUSTER}/scripts/maintenance` directory:

```
$ cd /var/ocne/cluster/${OCCNE_CLUSTER}/scripts/maintenance
```

3. Run the following command to remove the targeted worker node:

Note

- **Log file location:** Information will be displayed on screen and logged to a log file. This file is located at the path, `/var/occne/cluster/${OCCNE_CLUSTER}/removeBmWkrNodeCapture-<mmddyyyy_hhmmss>.log`. Long-running processes (such as node provisioning) will not appear in the file until they have finished. The process must complete before its entry is recorded.
- **Backup file location:** The script creates a backup of the `lbCtrlData.json`, `/etc/hosts`, `registries.conf`, and `hosts.ini` files and saves them to the `/var/occne/cluster/${OCCNE_CLUSTER}/backUpConfig/` location.

```
$ artifacts/maintenance/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:

```
$ artifacts/maintenance/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/delta/backupConfig
- Removing BM worker node, could take considerable amount of
time.
  - k8s-3.test.us.oracle.com is not reachable
    - Performing node cleanup.
      - k8s-3.test.us.oracle.com pods are scheduled for
deletion.
        - Node cleanup completed.
          - Removing... Please Wait. Do not cancel.
            - Removed. Check /var/occne/cluster/delta/
removeBmWkrNodeCapture-05312024_215105.log for details.
  - 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: 1, 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-1 deployment scaled down.
```

```

- Purging OSD Id.
  - OSD.1 purged.
  - OSD host k8s-3.test.us.oracle.com removed.
  - OSD auth key osd.1 removed.
- Deleting OSD deployment.
  - rook-ceph-osd-1 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.
- Cleanup leftover rook-ceph-mon replicaset
  - Found pod rook-ceph-mon-c-74f5d4d498-tw4d6, getting
    replica set...
    - Found deployment rook-ceph-mon-c, deleting...
    - rook-ceph-mon cleanup completed.
Worker node: k8s-3.test.us.oracle.com removed Successfully

```

- Edit the `hosts.ini` file, remove the lines that contain `<NODE_FULL_NAME>`, must be the same node used the previous step. Run the following command to open the `hosts.ini` file:

```
$ vi hosts.ini
```

For the following sample output, `<NODE_FULL_NAME>` is `k8s-node-3.test.us.oracle.com`

```

[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 # <----- Remove this
whole line
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 # <----- Remove this whole line
k8s-node-4.test.us.oracle.com

```

Save and exit.

Post verification

This procedure provides steps to verify if the node is removed successfully. Perform the following steps to verify the node is no longer part of Kubernetes cluster:

```
$ kubectl get nodes
```

Before performing the procedure, the node `k8s-node-4.test.us.oracle.com` was part of the cluster. The following sample output for BareMetal shows that the node is removed:

NAME	STATUS	ROLES	AGE	VERSION
k8s-master-1.test.us.oracle.com	Ready	control-plane	8d10h	v1.33.1
k8s-master-2.test.us.oracle.com	Ready	control-plane	8d10h	v1.33.1
k8s-master-3.test.us.oracle.com	Ready	control-plane	8d10h	v1.33.1
k8s-node-1.test.us.oracle.com	Ready	<none>	8d10h	v1.33.1
k8s-node-2.test.us.oracle.com	Ready	<none>	8d10h	v1.33.1
k8s-node-3.test.us.oracle.com	Ready	<none>	8d10h	v1.33.1
k8s-node-5.test.us.oracle.com	Ready	<none>	8d10h	v1.33.1
k8s-node-6.test.us.oracle.com	Ready	<none>	8d10h	v1.33.1

Run the following command to check the node is no longer part of the `/etc/hosts` file:

```
$ cat /etc/hosts
```

Before performing the procedure, the node `test-cluster-k8s-node-6` was part of the cluster. The following sample output for vCNE shows that the node is removed:

```
127.0.0.1    localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6

10.1.1.3 centralrepo
192.168.202.3 test-cluster-bastion-2.novalocal test-cluster-bastion-2
192.168.202.4 test-cluster-k8s-node-1.novalocal test-cluster-k8s-node-1
192.168.202.5 test-cluster-k8s-node-2.novalocal test-cluster-k8s-node-2
192.168.200.6 test-cluster-k8s-node-3.novalocal test-cluster-k8s-node-3
192.168.203.7 test-cluster-k8s-node-4.novalocal test-cluster-k8s-node-4
192.168.201.8 test-cluster-k8s-node-5.novalocal test-cluster-k8s-node-5
192.168.201.9 test-cluster-k8s-ctrl-1.novalocal test-cluster-k8s-ctrl-1
192.168.201.10 test-cluster-k8s-ctrl-2.novalocal test-cluster-k8s-ctrl-2
192.168.201.11 test-cluster-k8s-ctrl-3.novalocal test-cluster-k8s-ctrl-3
127.0.0.1 occne-repo-host test-cluster-bastion-1
192.168.201.9 lb-apiserver.kubernetes.local
192.168.201.10 lb-apiserver.kubernetes.local
192.168.201.11 lb-apiserver.kubernetes.local
```

Run the following command to check the node is no longer part of the dynamic inventory (hosts script) file:

```
$ ./hosts --hostfile
```

Before performing the procedure, the node `cluster-test-k8s-node-3` was part of the cluster. The following sample output for vCNE (CNLB Nodes) shows that the node is removed:

```
## begin hosts generated by terraform.py ##
10.123.154.116      cluster-test-bastion-1
10.123.154.60       cluster-test-bastion-2
192.168.202.217     cluster-test-k8s-node-cnlb-1
192.168.202.33      cluster-test-k8s-node-cnlb-2
192.168.200.219     cluster-test-k8s-node-cnlb-3
192.168.203.179     cluster-test-k8s-node-cnlb-4
192.168.201.229     cluster-test-k8s-node-cnlb-5
192.168.201.204     cluster-test-k8s-ctrl-1
192.168.201.130     cluster-test-k8s-ctrl-2
192.168.201.206     cluster-test-k8s-ctrl-3
192.168.203.205     cluster-test-k8s-node-1
192.168.203.205     cluster-test-k8s-node-2
192.168.203.205     cluster-test-k8s-node-4
## end hosts generated by terraform.py ##
```

Verify that the cluster is healthy. The following are simple examples.

Run the following command to retrieve all node from all namespaces and verify that they are all up and running:

```
$ kubectl get pods -A
```

Run the following command to verify that all services are assigned and accessible.

```
$ kubectl get svc -A
```

```
kco get deploy,sts,ds -o json | jq -r '
.items[]
| select(.spec.template.metadata.annotations["oracle.com.cnc/cnlb"])
| "\(.metadata.name): \(.spec.template.metadata.annotations."oracle.com.cnc/
cnlb" | fromjson[0].cnlbIp)'"
```

Troubleshooting

Advanced options (vCNE CNLB)

The vCNE CNLB script (`removeK8sNode.py`) can accept additional arguments that are useful for troubleshooting common issues or for resuming a node deletion process if it failed previously.

```
$ scripts/removeK8sNode.py -h
```

Note

Use the script's help option for more information about each argument.

Sample output:

```
usage: removeK8sNode.py [-h] [-d] [-nb] [-i] [-iq] [-s {PRE,POST}
[{{PRE,POST} ...}] [-R] [-D]
```

Remove one or more K8s nodes or CNLB K8s nodes (control plane nodes are not supported) from an OCCNE cluster.

Optional arguments:

```
-d, --debug           Debug mode.
-nb, --no-backup      Do not create backup.
-i, --ignore-disruption
                      Ignores pod disruption budget while draining a node.
```

This could lead to service disruption use with caution or manually shift load around before attempting to

remove a node.

```
-iq, --ignore-quota  Allows the script to remove a node even if the number
of nodes would be too low to execute its load. Use it only to replace a node
(add it back immediately after
```

the removal is complete).

```
-s {PRE,POST} [{{PRE,POST} ...}, --skip-test {PRE,POST} [{{PRE,POST} ...]
```

Do not test the cluster health. By default, all tests are run; specify individual tests to be skipped.

```
-R, --refresh        Allow OpenTofu to synchronize its state with current
infrastructure resources. Internally sends a -refresh-only command to both
tofu plan and apply commands.
```

```
-D, --dry-run        Test mode. Identifies new or affected nodes without
applying changes. Stops in the init phase.
```

Examples:

```
./removeK8sNode.py
./removeK8sNode.py --debug
./removeK8sNode.py -nb -d
./removeK8sNode.py --debug --no-backup --dry-run
./removeK8sNode.py --ignore-disruption
./removeK8sNode.py --ignore-quota --refresh
./removeK8sNode.py -d --skip-test PRE POST -R
```

8.3.6 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 Bare Metal after installation in CNE.

The cluster must be in good working condition and all pods and services must be running. Use the following command to run a full cluster test before starting the procedure:

```
OCCNE_STAGES=(TEST) pipeline.sh
```

8.3.6.1 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/ocne/cluster/${OCNE_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. Perform 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>
```

8.3.7 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 nodes.

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.

8.4 Managing Kyverno

Policies are used to audit workloads running on Kubernetes. Kubernetes 1.21.x deprecated its policy framework PodSecurityPolicies (PSP) and halted its support in Kubernetes release 1.25.x. Therefore, CNE supports Kyverno, an open-source tool, as an alternate to PSP for managing Kubernetes policies. This section provides information about Kyverno deployment in CNE, migrating from PodSecurityPolicies (PSP) to Kyverno, and using Kyverno to manage Kubernetes policies.

Note

- PodSecurityPolicies (PSP) are not supported from CNE 23.2.x onwards (that is, Kubernetes 1.25.x onwards).
- If you are creating your own custom Kyverno policies, ensure that you exclude `occne-infra` namespace as shown in the following example. This ensures that the existing `occne-infra` deployments are not affected.

```
policyExclude:
  disallow-capabilities:
    any:
      - resources:
          kinds:
            - Pod
            - DaemonSet
          namespaces:
            - kube-system
            - occne-infra
            - rook-ceph
```

Kyverno Deployment

Kyverno framework is deployed as a common service in CNE. All PSP based policies are removed. CNE is configured with number of baseline policies that are applied across CNE clusters.

The following table provides information about the policies that are deployed in CNE:

Table 8-20 Kyverno Policies

Policy	Policy Level (23.1.x)	Policy Level (23.2.x to 24.2.x)	Policy Level (24.3.x onwards)
disallow-capabilities	audit	enforced	enforced
disallow-host-namespaces	audit	enforced	enforced
disallow-host-path	audit	enforced	enforced
disallow-host-ports	audit	enforced	enforced
disallow-host-process	audit	enforced	enforced
disallow-privileged-containers	audit	enforced	enforced
disallow-proc-mount	audit	enforced	enforced
disallow-selinux	audit	enforced	enforced
restrict-apparmor-profiles	audit	enforced	enforced
restrict-seccomp	audit	enforced	enforced
restrict-sysctls	audit	enforced	enforced
require-emptydir-requests-and-limits	NA	NA	audit

In release 23.1.x, the baseline policies are set to the "audit" level. This means that, the cluster keeps record of the policy failures, but doesn't cause non-compliant resources to fail. Starting 23.2.x, the policies are set to the "enforced" level. This means that, any resource that are non-compliant to any policy will fail. When a new policy is added in a release, CNE determines the level of the policy to control how a non-compliant scenario is handled.

References:

- Kyverno pod security policy: <https://github.com/kyverno/kyverno/tree/main/charts/kyverno-policies>
- Kyverno policy validation and enforcement: <https://kyverno.io/docs/writing-policies/validate/>
- Kyverno policy writing: <https://kyverno.io/docs/kyverno-policies/>
- Kyverno reporting: <https://kyverno.io/docs/policy-reports/>
- Kyverno metrics: <https://kyverno.io/docs/monitoring/#installation-and-setup>

8.4.1 Accessing Cluster Policy Report

Kyverno performs regular scans and updates validation status in "ClusterPolicyReport". When a new resource is created by a user or process, Kyverno checks the properties of the resource against the validation rule. This section provides information about access ClusterPolicyReport.

Run the following command to access ClusterPolicyReport :

```
$ kubectl get policyreport -A
```

Sample output:

NAMESPACE	NAME	PASS	FAIL	WARN	ERROR	SKIP
AGE						
ingress-nginx	polr-ns-ingress-nginx	11	1	0	0	0
10d						
occne-infra	polr-ns-occne-infra	263	13	0	0	0
10d						

You can use the `describe policyreport` command to get details about the policies that is violated. For example, in the above example, the `ingress-nginx` deployment reports one violation. You can use the following command to get details about the policy violation in `ingress-nginx`:

```
$ kubectl describe policyreport polr-ns-ingress-nginx -n ingress-nginx
```

Sample output:

```
Name:      polr-ns-ingress-nginx
Namespace: ingress-nginx
Labels:    managed-by=kyverno
Annotations: <none>
API Version: wgpolicyk8s.io/v1alpha2
Kind:      PolicyReport
.
.
.
Results:
  Category: Pod Security Standards (Baseline)
  Message:  validation error: Use of host ports is disallowed. The fields
spec.containers[*].ports[*].hostPort ,
spec.initContainers[*].ports[*].hostPort, and
spec.ephemeralContainers[*].ports[*].hostPort must either be unset or set to
`0`. Rule autogen-host-ports-none failed at path /spec/template/spec/
containers/0/ports/0/hostPort/
  Policy:   disallow-host-ports
  Resources:
    API Version: apps/v1
    Kind:        DaemonSet
    Name:        ingress-nginx-controller
    Namespace:   ingress-nginx
```

The sample output shows that the `ingress-nginx-controller` resource violates the `disallow-host-ports` policy.

8.4.2 Migrating from PSP to Kyverno

Every component that runs on CNE 23.2.x and above must migrate all PSP based policy resources to Kyverno pod security policies. For information and examples about performing a

migration from PSP to Kyverno, see [Kyverno pod security policy](#) and [Kyverno source](#) documentations.

8.4.3 Adding Policy Exclusions and Exceptions

This section provides details about adding exclusions and exceptions to Kyverno policies.

Adding Exclusions

Exclusions simplify policy management when used alongside Pod Security Admission (PSA). Instead of creating multiple policies for each control in PSA, you can leverage Kyverno to provide detailed and selective exclusions, reducing policy overhead and enhancing overall manageability. You can add exclusion to policies by editing the current policies or running `kubectl patch`. For more information about policy exclusions, see [Kyverno documentation](#).

Adding Exceptions

There are scenarios where users need to deploy resources that require privileged access. This can be done by adding exceptions to the Kyverno policies for individual resources. The following code block provides a sample YAML file with the required information to create a `PolicyException` resource:

```
apiVersion: v1
items:
- apiVersion: kyverno.io/v2alpha1
  kind: PolicyException
  metadata:
    name: cncc-exception
    namespace: occne-infra
  spec:
    exceptions:
    - policyName: disallow-capabilities
      ruleNames:
      - adding-capabilities
    match:
      any:
      - resources:
          kinds:
          - Pod
          - Deployment
          names:
          - cncc-debug-tool*
          namespaces:
          - cncc
  kind: List
  metadata:
    resourceVersion: ""
```

In this example, `PolicyException` allows the pods named `cncc-debug-tool*` (the `*` character indicates the system to include any resources belonging to a deployment to be allowed) in the "cncc" namespace to bypass the "disallow-capabilities > adding-capabilities" policy.

After creating the policy exception file (in this case, `policyException.yaml`), proceed to create the resource:

```
$ kubectl create -f policyException.yaml
```

8.4.4 Managing Kyverno Metrics

When you install Kyverno using Helm, additional services are created inside the Kyverno namespace that expose metrics on port 8000. This section provides details about managing Kyverno metrics.

By default, the service type of these services is ClusterIP.

Run the following command to get the Kyverno metric services:

```
$ kubectl -n kyverno get svc
```

Sample output:

NAME	EXTERNAL-IP	PORT(S)	AGE	TYPE	CLUSTER-IP
occne-kyverno-reports-controller-metrics	<none>	8000/TCP	10d	ClusterIP	10.233.21.143
occne-kyverno-svc	<none>	443/TCP	10d	ClusterIP	10.233.20.88
occne-kyverno-svc-metrics	<none>	8000/TCP	10d	ClusterIP	10.233.6.173

8.4.5 Validating Kyverno Compliance

This section provides the steps to validate if the CNE cluster runs Kyverno policy management tool and if all CNE resources run in compliance mode.

Prerequisites

Before validating Kyverno compliance, ensure that the following prerequisites are met:

- CNE cluster must be healthy.
- The CNE cluster version must be at least 23.2.0.
- All pods in the cluster must be running.
- There must be no violations in the Kyverno report.

1. Verify Kyverno Policies:

- a. Run the following command to list all clusterpolicy configured by Kyverno:

```
$ kubectl get clusterpolicy -n kyverno
```

Sample output:

NAME	BACKGROUND	VALIDATE	ACTION	READY
AGE				
disallow-capabilities	true	Enforce		true

10h				
disallow-host-namespaces	true	Enforce	true	
10h				
disallow-host-path	true	Enforce	true	
10h				
disallow-host-ports	true	Enforce	true	
10h				
disallow-host-process	true	Enforce	true	
10h				
disallow-privileged-containers	true	Enforce	true	
10h				
disallow-proc-mount	true	Enforce	true	
10h				
disallow-selinux	true	Enforce	true	
10h				
restrict-apparmor-profiles	true	Enforce	true	
10h				
restrict-seccomp	true	Enforce	true	
10h				
restrict-sysctls	true	Enforce	true	
10h				

- b. Run the following command and ensure that there are no policy violations in any policy report in any namespace:

```
$ kubectl get policyreport -A
```

Sample output:

NAMESPACE	NAME	PASS	FAIL
WARN	ERROR	SKIP	AGE
ingress-nginx	cpol-disallow-capabilities	4	0
0	0	0	10h
ingress-nginx	cpol-disallow-host-namespaces	4	0
0	0	0	10h
ingress-nginx	cpol-disallow-host-path	4	0
0	0	0	10h
ingress-nginx	cpol-disallow-host-process	5	0
0	0	0	10h
ingress-nginx	cpol-disallow-privileged-containers	4	0
0	0	0	10h
ingress-nginx	cpol-disallow-proc-mount	5	0
0	0	0	10h
ingress-nginx	cpol-disallow-selinux	10	0
0	0	0	10h
ingress-nginx	cpol-restrict-apparmor-profiles	5	0
0	0	0	10h
ingress-nginx	cpol-restrict-seccomp	5	0
0	0	0	10h
ingress-nginx	cpol-restrict-sysctls	5	0
0	0	0	10h
kube-system	cpol-disallow-host-process	62	0
0	0	0	10h
kube-system	cpol-disallow-proc-mount	62	0
0	0	0	10h

kube-system	cpol-disallow-selinux	124	0
0 0	0 10h		
kube-system	cpol-restrict-apparmor-profiles	62	0
0 0	0 10h		
kube-system	cpol-restrict-seccomp	62	0
0 0	0 10h		
kube-system	cpol-restrict-sysctls	62	0
0 0	0 10h		
kyverno	cpol-disallow-capabilities	3	0
0 0	0 9h		
kyverno	cpol-disallow-host-namespaces	3	0
0 0	0 9h		
kyverno	cpol-disallow-host-path	3	0
0 0	0 9h		
kyverno	cpol-disallow-host-ports	3	0
0 0	0 9h		
kyverno	cpol-disallow-host-process	5	0
0 0	0 9h		
kyverno	cpol-disallow-privileged-containers	3	0
0 0	0 9h		
kyverno	cpol-disallow-proc-mount	5	0
0 0	0 9h		
kyverno	cpol-disallow-selinux	10	0
0 0	0 9h		
kyverno	cpol-restrict-apparmor-profiles	5	0
0 0	0 9h		
kyverno	cpol-restrict-seccomp	5	0
0 0	0 9h		
kyverno	cpol-restrict-sysctls	5	0
0 0	0 9h		
occne-infra	cpol-disallow-host-process	86	0
0 0	0 9h		
occne-infra	cpol-disallow-proc-mount	86	0
0 0	0 9h		
occne-infra	cpol-disallow-selinux	172	0
0 0	0 9h		
occne-infra	cpol-restrict-apparmor-profiles	86	0
0 0	0 9h		
occne-infra	cpol-restrict-seccomp	86	0
0 0	0 9h		
occne-infra	cpol-restrict-sysctls	86	0
0 0	0 9h		

- c. Run the following command to verify that there are no pods running in the cluster due to Kyverno clusterpolicy:

```
$ kubectl get pods -A | grep -v Runn
```

Sample output:

NAMESPACE	NAME	READY	STATUS	RESTARTS	AGE
-----------	------	-------	--------	----------	-----

2. Verify the daemonset, statefulset, and deployment in the cluster:

- a. Run the following command to verify all the daemonset (ds) in the cluster:

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

Sample output:

NAME	READY	UP-TO-DATE	AVAILABLE	NODE SELECTOR	DESIRED	CURRENT
occne-kube-prom-stack-prometheus-node-exporter	7	7	7		7	7
occne-metallb-speaker	4	4	4	kubernetes.io/os=linux	4	4
occne-tracer-jaeger-agent	4	4	4		4	4

- b. Run the following command to verify all the statefulset (sts) in the cluster:

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

Sample output:

NAME	READY
alertmanager-occne-kube-prom-stack-kube-alertmanager	2/2
occne-opensearch-cluster-client	3/3
occne-opensearch-cluster-data	3/3
occne-opensearch-cluster-master	3/3
prometheus-occne-kube-prom-stack-kube-prometheus	2/2

- c. Run the following command to verify all the deploy in the cluster:

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

Sample output:

NAME	READY	UP-TO-DATE
occne-bastion-controller	1/1	1

```

11h
occne-kube-prom-stack-grafana          1/1      1
1
11h                                     occne-metallb-controller
occne-kube-prom-stack-kube-operator    1/1      1
1
11h
occne-kube-prom-stack-kube-state-metrics 1/1      1
1
11h
occne-metallb-controller                1/1      1
1
11h
occne-metrics-server                    1/1      1
1
11h
occne-opensearch-dashboards             1/1      1
1
11h
occne-promxy                            1/1      1
1
11h
occne-promxy-apigw-nginx                2/2      2
2
11h
occne-snmp-notifier                     1/1      1
1
11h
occne-tracer-jaeger-collector            1/1      1
1
11h
occne-tracer-jaeger-query                1/1      1
1          11h

```

3. Scale down and scale up the deployments to verify any impact of Kyverno clusterpolicy:

a. Run the following command to scale down the deployment in the cluster:

```
kubectl scale --replicas=0 deploy --all -n occne-infra
```

- b. Run the following command to scale up the deployment in the cluster:

```
$ kubectl scale --replicas=1 deploy --all -n occne-infra
$ kubectl scale --replicas=2 deploy occne-promxy-apigw-nginx -n occne-infra
```

- c. Run the following command to get the deployment list:

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

Sample output:

NAME AVAILABLE AGE	READY	UP-TO-DATE
occne-bastion-controller 1 12h	1/1	1
occne-kube-prom-stack-grafana 1 12h	1/1	1
occne-kube-prom-stack-kube-operator 1 12h	1/1	1
occne-kube-prom-stack-kube-state-metrics 1 12h	1/1	1
occne-metallb-controller 1 12h	1/1	1
occne-metrics-server 1 12h	1/1	1
occne-opensearch-dashboards 1 12h	1/1	1
occne-promxy 1 12h	1/1	1
occne-promxy-apigw-nginx 2 12h	2/2	2
occne-snmp-notifier 1 12h	1/1	1

```

occne-tracer-jaeger-collector      1/1      1
1
12h

occne-tracer-jaeger-query          1/1      1
1      12h

```

- d. Run the following command to scale down the statefulset (sts) in the cluster:

```
$ kubectl scale --replicas=0 sts --all -n occne-infra
```

Sample output:

```

statefulset.apps/alertmanager-occne-kube-prom-stack-kube-alertmanager
scaled
statefulset.apps/occne-opensearch-cluster-client scaled
statefulset.apps/occne-opensearch-cluster-data scaled
statefulset.apps/occne-opensearch-cluster-master scaled
statefulset.apps/prometheus-occne-kube-prom-stack-kube-prometheus scaled

```

- e. Run the following command to scale up the statefulset (sts) in the cluster:

```
$ kubectl scale sts occne-opensearch-cluster-master occne-opensearch-
cluster-data occne-opensearch-cluster-client -n occne-infra --replicas 3
```

- f. Run the following command to get the list of statefulset:

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

Sample output:

NAME	READY	AGE
alertmanager-occne-kube-prom-stack-kube-alertmanager	2/2	2d19h
occne-opensearch-cluster-client	3/3	2d19h
occne-opensearch-cluster-data	3/3	2d19h
occne-opensearch-cluster-master	3/3	2d19h
prometheus-occne-kube-prom-stack-kube-prometheus	2/2	2d19h

- g. Run the following command to restart the daemonset (ds):

```
$ kubectl rollout restart ds -n occne-infra
```

Sample output:

```

daemonset.apps/occne-kube-prom-stack-prometheus-node-exporter restarted
daemonset.apps/occne-metallb-speaker restarted
daemonset.apps/occne-tracer-jaeger-agent restarted

```

- h. Run the following command to get the list of daemonset:

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

Sample output:

NAME	READY	UP-TO-DATE	AVAILABLE	NODE SELECTOR	DESIRED	CURRENT AGE
occne-kube-prom-stack-prometheus-node-exporter	7	7	7	<none>	7	7
occne-metallb-speaker	4	4	4	kubernetes.io/os=linux	4	4
occne-tracer-jaeger-agent	4	4	4	<none>	4	4
						2d19h
						2d19h
						2d19h

4. Reverify Kyverno Policies:

- a. Run the following command to list all the cluster policies configured by Kyverno:

```
$ kubectl get clusterpolicy -n kyverno
```

Sample output:

NAME	BACKGROUND	VALIDATE	ACTION	READY
AGE				
disallow-capabilities	true	Enforce		true
12h				
disallow-host-namespaces	true	Enforce		true
12h				
disallow-host-path	true	Enforce		true
12h				
disallow-host-ports	true	Enforce		true
12h				
disallow-host-process	true	Enforce		true
12h				
disallow-privileged-containers	true	Enforce		true
12h				
disallow-proc-mount	true	Enforce		true
12h				
disallow-selinux	true	Enforce		true
12h				
restrict-apparmor-profiles	true	Enforce		true
12h				
restrict-seccomp	true	Enforce		true
12h				
restrict-sysctls	true	Enforce		true
12h				

- b. Run the following command and ensure that there are no policy violations in any policy report in any namespace:

```
$ kubectl get policyreport -A
```

Sample output:

NAMESPACE	NAME	PASS	FAIL
WARN	ERROR	SKIP	AGE
ingress-nginx	cpol-disallow-capabilities	4	0
0	0	0	2d15h
ingress-nginx	cpol-disallow-host-namespaces	4	0
0	0	0	2d15h
ingress-nginx	cpol-disallow-host-path	4	0
0	0	0	2d15h
ingress-nginx	cpol-disallow-host-process	5	0
0	0	0	2d15h
ingress-nginx	cpol-disallow-privileged-containers	4	0
0	0	0	2d15h
ingress-nginx	cpol-disallow-proc-mount	5	0
0	0	0	2d15h
ingress-nginx	cpol-disallow-selinux	10	0
0	0	0	2d15h
ingress-nginx	cpol-restrict-apparmor-profiles	5	0
0	0	0	2d15h
ingress-nginx	cpol-restrict-seccomp	5	0
0	0	0	2d15h
ingress-nginx	cpol-restrict-sysctls	5	0
0	0	0	2d15h
kube-system	cpol-disallow-host-process	62	0
0	0	0	2d15h
kube-system	cpol-disallow-proc-mount	62	0
0	0	0	2d15h
kube-system	cpol-disallow-selinux	124	0
0	0	0	2d15h
kube-system	cpol-restrict-apparmor-profiles	62	0
0	0	0	2d15h
kube-system	cpol-restrict-seccomp	62	0
0	0	0	2d15h
kube-system	cpol-restrict-sysctls	62	0
0	0	0	2d15h
kyverno	cpol-disallow-capabilities	3	0
0	0	0	2d15h
kyverno	cpol-disallow-host-namespaces	3	0
0	0	0	2d15h
kyverno	cpol-disallow-host-path	3	0
0	0	0	2d15h
kyverno	cpol-disallow-host-ports	3	0
0	0	0	2d15h
kyverno	cpol-disallow-host-process	5	0
0	0	0	2d15h
kyverno	cpol-disallow-privileged-containers	3	0
0	0	0	2d15h
kyverno	cpol-disallow-proc-mount	5	0
0	0	0	2d15h

kyverno	cpol-disallow-selinux	10	0
0 0	0 2d15h		
kyverno	cpol-restrict-apparmor-profiles	5	0
0 0	0 2d15h		
kyverno	cpol-restrict-seccomp	5	0
0 0	0 2d15h		
kyverno	cpol-restrict-sysctls	5	0
0 0	0 2d15h		
occne-infra	cpol-disallow-host-process	88	0
0 0	0 2d15h		
occne-infra	cpol-disallow-proc-mount	88	0
0 0	0 2d15h		
occne-infra	cpol-disallow-selinux	176	0
0 0	0 2d15h		
occne-infra	cpol-restrict-apparmor-profiles	88	0
0 0	0 2d15h		
occne-infra	cpol-restrict-seccomp	88	0
0 0	0 2d15h		
occne-infra	cpol-restrict-sysctls	88	0
0 0	0 2d15h		

- c. Run the following command to verify that there are no pods running in the cluster due to Kyverno clusterpolicy:

```
$ kubectl get pods -A | grep -v Runn
```

Sample output:

NAMESPACE	NAME	READY	STATUS	RESTARTS	AGE
-----------	------	-------	--------	----------	-----

8.5 Updating Grafana Password

This section describes the procedure to update the password of Grafana which is used to access the graphical user interface of Grafana dashboard.

Prerequisites

1. A CNE cluster of version 22.x or above must be deployed.
2. The Grafana pod must be up and running.
3. The Load Balancer IP of Grafana must be accessible through browser.

Limitations and Expectations

- This procedure works on the existing Grafana deployments instantly. However, this procedure requires a restart of the Grafana pod causing temporary unavailability of the service.
- The grafana-cli is used to update the password by running an exec to get into the pod, which means it's ephemeral.
- The newly set password is not persistent. Therefore, if the Grafana pod restarts or crashes due to any unfortunate incidents, then rerun this procedure to set the password. Otherwise, the GUI displays the Invalid Username or Password errors.

- Only the password that is updated during an installation is persistent. However, if you change this password impromptu and patch the new password into the secret again, the password change doesn't take effect. To update and persist a password, use this procedure during installation of the CNE cluster.

Procedure

1. Log in to the Bastion Host of your CNE cluster.
2. Get the name of the Grafana pod in the occne-infra namespace:

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

For example:

```
$ kubectl get pods | grep grafana
```

Sample output:

```
occne-prometheus-grafana-7f5fb7c4d4-lxqsr 3/3 Running 0 12m
```

3. Use `exec` to get into the Grafana pod:

```
$ kubectl exec -it occne-prometheus-grafana-7f5fb7c4d4-lxqsr bash
```

4. Run the following command to update the password:

Note

Use a strong unpredictable password consisting of complex strings of more than 10 mixed characters.

```
$ grafana-cli admin reset-admin-password <password>
```

where, <password> is the new password to be updated.

For example:

```
occne-prometheus-grafana-7f5fb7c4d4-lxqsr:/usr/share/grafana$ grafana-cli  
admin reset-admin-password samplepassword123#
```

Sample output:

```
INFO [01-29|07:44:25] Starting Grafana logger=settings version= commit=  
branch= compiled=1970-01-01T00:00:00Z  
WARN [01-29|07:44:25] "sentry" frontend logging provider is deprecated and  
will be removed in the next major version. Use "grafana" provider instead.  
logger=settings  
INFO [01-29|07:44:25] Config loaded from logger=settings file=/usr/share/  
grafana/conf/defaults.ini  
INFO [01-29|07:44:25] Config overridden from Environment variable  
logger=settings var="GF_PATHS_DATA=/var/lib/grafana/"
```

```

INFO [01-29|07:44:25] Config overridden from Environment variable
logger=settings var="GF_PATHS_LOGS=/var/log/grafana"
INFO [01-29|07:44:25] Config overridden from Environment variable
logger=settings var="GF_PATHS_PLUGINS=/var/lib/grafana/plugins"
INFO [01-29|07:44:25] Config overridden from Environment variable
logger=settings var="GF_PATHS_PROVISIONING=/etc/grafana/provisioning"
INFO [01-29|07:44:25] Config overridden from Environment variable
logger=settings var="GF_SECURITY_ADMIN_USER=admin"
INFO [01-29|07:44:25] Config overridden from Environment variable
logger=settings var="GF_SECURITY_ADMIN_PASSWORD=*****"
INFO [01-29|07:44:25] Target logger=settings target=[all]
INFO [01-29|07:44:25] Path Home logger=settings path=/usr/share/grafana
INFO [01-29|07:44:25] Path Data logger=settings path=/var/lib/grafana/
INFO [01-29|07:44:25] Path Logs logger=settings path=/var/log/grafana
INFO [01-29|07:44:25] Path Plugins logger=settings path=/var/lib/grafana/
plugins
INFO [01-29|07:44:25] Path Provisioning logger=settings path=/etc/grafana/
provisioning
INFO [01-29|07:44:25] App mode production logger=settings
INFO [01-29|07:44:25] Connecting to DB logger=sqlstore dbtype=sqlite3
INFO [01-29|07:44:25] Starting DB migrations logger=migrator
INFO [01-29|07:44:25] migrations completed logger=migrator performed=0
skipped=484 duration=908.661µs
INFO [01-29|07:44:25] Envelope encryption state logger=secrets
enabled=true current provider=secretKey.v1

```

Admin password changed successfully η

5. After updating the password, log in to the Grafana GUI using the updated password. Ensure that you are aware of the limitations listed in [Limitations and Expectations](#).

8.6 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
```

8.7 Updating OpenStack TLS Certificate

When the Certificate Authority (CA) certificate for OpenStack is updated, you can use this procedure to update the certificate which CNE uses to access OpenStack.

Note

This procedure is applicable only to CNE instances running on OpenStack.

Procedure to update the certificate

Note

All commands must be run from the default CNE user account (for example, `cloud-user`) on the active Bastion host.

1. Use SSH to log in to the active Bastion and run the following command to confirm if it is the active Bastion:

```
$ is_active_bastion
```

Sample output:

```
IS active-bastion
```

2. Get the renewed OpenStack `cacert.pem` file from the Openstack admin.
3. Copy the new `.pem` certificate file to `/var/ocne/cluster/${OCCNE_CLUSTER}/openstack-cacert.pem`, replacing the existing file.
4. Change to cluster directory and read the `.pem` certificate file into variables:

```
$ cd cl
$ PEM=$(cat openstack-cacert.pem)
$ ENCODED_PEM=$(base64 -w0 openstack-cacert.pem)
```

5. Write the new PEM content into the `.pem` files used for OpenStack access throughout the cluster:

```
$ ocne_all.sh "echo -n '${PEM}' | sudo tee /etc/kubernetes/cinder-cacert.pem /etc/kubernetes/external-openstack-cacert.pem" k8s-cluster
```

6. On one of the control nodes, update the Kubernetes secret that stores the base64-encoded PEM data.

```
$ SECRET_PATH=/etc/kubernetes/external-openstack-cloud-config-secret.yml
$ ocne_all.sh "if [ -f '${SECRET_PATH}' ]; then sudo sed -i 'ca.cert/c\ ca.cert: ${ENCODED_PEM}' ${SECRET_PATH}; sudo -E env 'PATH=$PATH' kubectl apply -f ${SECRET_PATH}; fi" kube-master
```

7. Restart all OpenStack controller and cinder pods:

```
$ kubectl -n kube-system rollout restart ds/openstack-cloud-controller-manager ds/csi-cinder-nodeplugin deployment/csi-cinder-controllerplugin
```

8.8 Updating the Guest or Host OS

You must update the host OS (for Bare Metal installations) or guest OS (for both Bare Metal and virtualized installations) periodically so that CNE has the latest Oracle Linux software. If the cluster is not upgraded recently, or there are known security patches then perform an update by referring to the "Performing an Upgrade" section in *Oracle Communications Cloud Native Core, Cloud Native Environment Installation, Upgrade, and Fault Recovery Guide*. Ensure that you follow the specific steps for performing an OS update (the cluster version remains the same), and not a Cluster Upgrade.

8.9 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.

8.9.1 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.

8.9.2 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.

8.9.3 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
```

8.10 Managing 5G NFs

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

8.10.1 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 8-21 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/ocne/charts http://<external-helm-repo-  
name>/ocne/charts [helm_executable_full_path_if_not_default] <  
helm_charts.txt
```

Example:

```
$ retrieve_helm.sh /var/www/html/ocne/charts http://mycentralrepo/ocne/
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>
```

8.10.2 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.

8.10.3 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 8-22 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.

8.10.4 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:

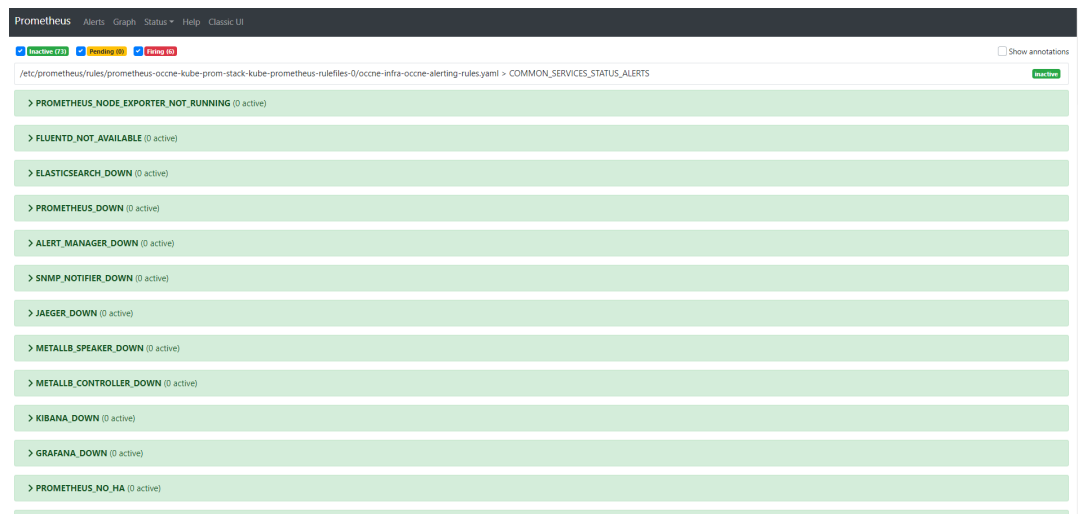
```
$ kubectl apply -f /tmp/rules_file.yaml -n occne-infra
# To verify the creation of the alert-rules CRD, run the following command:
$ kubectl get prometheusrule -n occne-infra
NAME                                AGE
occne-alerting-rules                43d
```

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 8-1 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)"
```

8.10.5 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 8-23 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 through 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
```

8.11 Updating CNLB Networks

This section provides information on how to update the network in an existing CNLB deployment and also explains the non-disruptive and disruptive network updates.

Note

Network attachment updates are supported for BareMetal, vCNE OpenStack and vCNE VMware deployments only.

Prerequisites

Before updating the network, ensure that the following prerequisites are met:

1. CNLB cluster must be installed and configured.
2. All nodes are up, reachable, and have the correct network settings.
3. For VMware, this procedure applies to CNLB deployments on VCD 10.4.3 and VCD 10.6.0. VCD 10.2 is not supported.
4. For vCNE on VMware, it is recommended that before running this procedure, while performing any disruptive update, application traffic must be diverted to another georedundant site.

Expectations

Before updating the network, ensure to note the following points:

1. This procedure applies to vCNE on OpenStack, vCNE on VMware, and Bare Metal deployments.

2. For vCNE on VMware, when performing a Disruptive Update, it is strongly recommended to divert application traffic to another georedundant site before running this procedure. For more information, see the [#unique_24/unique_24_Connect_42_GUID-CA5CE3C4-54F5-4F53-BDAD-768213A4448D](#) section.
3. This procedure assumes that the user is fully familiar with configuring the `cnlb.ini` file.

Disruptive Updates

Disruptive updates remove or modify the settings in the `cnlb.ini` file, which can affect the traffic flow due to the current cloud configuration. In such instances, the impact varies depending on the platform and may include an outage of all applications or NFs' that use CNLB.

The following table lists the operations that are performed for a disruptive network update:

Table 8-24 Disruptive Updates

Description	Assumptions	Type	Platform	How to prevent outage
Adding a new network	All the applications or NFs (for example, Deployments and StatefulSets) including CNE common services must be restarted after the script completes. This allows CNLB to refresh its status from each annotation.	Add	VMware	This change will switch around the order and network interface names (mac addresses and IPs may also be impacted). This is an inherent limitation that arises due to VMware Cloud Director (VCD) and OpenTofu providers. The outage will occur as soon as OpenTofu applies the new configuration (as part of the script), and will resolve only after the script has successfully completed and all the applications or NFs or common services have been restarted.

Table 8-24 (Cont.) Disruptive Updates

Description	Assumptions	Type	Platform	How to prevent outage
Removing an unused network	All the applications or NFs (for example, Deployments and StatefulSets) including CNE common services must be restarted after the script is run. This allows CNLB to refresh its status from each annotation.	Remove	VMware	This change will switch around the order and network interface names (mac addresses and IPs can be impacted). This is an inherent limitation that arises due to VMware Cloud Director (VCD) and OpenTofu providers. The outage will occur as soon as OpenTofu applies the new configuration (as part of the script), and will resolve only after the script has successfully finished and all the applications or NFs or common services have been restarted.

Table 8-24 (Cont.) Disruptive Updates

Description	Assumptions	Type	Platform	How to prevent outage
Changing an existing network range or CIDR	The existing network remains configured in <code>cnlb.ini</code> , but its network range is changed. For example, <code>internal_network_range</code> changes from <code>/25</code> to <code>/24</code> .	Replace	VMware	On VMware, the outage starts when OpenTofu applies the new configuration as part of the script. This change can cause OpenTofu to replace a VCD vApp network because of provider-level changes, such as gateway or netmask replacement. This is not supported as an in-service CNLB network update on VCD 10.4.1 and later. The outage is resolved only after the script completes successfully and the applications, NFs, and CNE Common Services are restarted so CNLB can refresh the annotation status.
Removing a network	The network is used by annotated applications or NFs.	Replace	BareMetal, OpenStack and VMware	Before performing this action, all applications or NFs annotated with this network must be reconfigured, that is change the annotation to point to a different active network to avoid an outage.

Table 8-24 (Cont.) Disruptive Updates

Description	Assumptions	Type	Platform	How to prevent outage
Removing an IP address from <code>service_ip_addr</code> list of an existing network	The service IP is used by an annotated application or NF.	Remove	BareMetal, OpenStack and VMware	Before performing this action, the application or NF using this <code>service_ip</code> must be reconfigured, that is change the annotation to point to a different <code>service_ip</code> in the same network or another network, to avoid an outage.
Removing an IP address from an <code>egress_dest</code> list of an existing network	The <code>egress_dest</code> is used by one or more annotated applications or NFs. There can be more than one <code>egress_dest</code> in the list.	Remove	BareMetal, OpenStack and VMware	Before performing this action, all applications or NFs annotated with the <code>egress_dest</code> list must be reconfigured, that is change the annotation to point to a different <code>egress_dest</code> .
Replacing an existing IP address from the <code>service_ip_addr</code> list of an existing network	There is an application or NF (for example, Deployments or StatefulSets) with annotation that is actively using this service IP address. The annotations are updated immediately after the script finishes.	Remove	BareMetal, OpenStack and VMware	To clear the outage, the application or NFs annotations must be reconfigured.

Table 8-24 (Cont.) Disruptive Updates

Description	Assumptions	Type	Platform	How to prevent outage
Updating an existing network <code>network_id</code> or <code>subnet_id</code>	The network is being used by annotated applications or NFs. The annotations (for all the applicable applications or NFs) are updated immediately after the script execution is complete. Fields such as <code>service_ip_address</code> and <code>egress_dest</code> must also be updated to match the new network.	Replace	BareMetal, OpenStack and VMware	To clear the outage, the application or NFs' annotations must be reconfigured.
Modifying shared-internal variable from <code>true</code> to <code>false</code> (or vice versa)	The network is being used by annotated applications or NFs. Network attachment definitions (for all the applicable applications or NFs) are updated immediately after the script is run completely. Restart annotated applications or NFs for the changes made to network attachment definitions to reflect.	Replace	BareMetal, OpenStack and VMware	To clear the outage, the application deployments using modified network attachment definitions must be restarted to reflect the updated configuration.

Non-Disruptive Updates

Non-disruptive updates perform network operations on an existing network that does not affect the current traffic flow. For example, Adding a new network, Adding an IP address to an existing network, and so on.

Note

Deleting a network or an IP address can be non-disruptive depending on how the network or IP address is configured in the cluster.

The following table lists the operations that are performed for a non-disruptive network update:

Table 8-25 Non-Disruptive Updates

Description	Assumptions	Type	Platform
Adding a new network	None	Add	BareMetal and OpenStack
Adding an IP address to the <code>service_ip_addr</code> list of an existing network	None	Add	BareMetal, OpenStack and VMware
Adding an IP address to the <code>egress_dest</code> list of an existing network	None	Add	BareMetal, OpenStack and VMware
Removing an unused network	There are no applications or NFs (for example, Deployments or StatefulSets) with annotations pointing to this network.	Remove	BareMetal and OpenStack
Removing an IP address to the <code>service_ip_addr</code> list of an existing network	There is no application or NF (for example, Deployment or StatefulSets) with an annotation that uses this service IP address.	Remove	BareMetal, OpenStack and VMware
Removing an IP address to the <code>egress_dest</code> list of an existing network	There is no application or NF (for example, Deployments or StatefulSets) with an annotation that uses this IP address.	Remove	BareMetal, OpenStack and VMware
Replacing an existing IP address from the <code>service_ip_addr</code> list of an existing network	There is no application or NF (for example, Deployments or StatefulSets) with an annotation that uses this service IP address.	Replace	BareMetal, OpenStack and VMware
Changing an existing network range or CIDR	The change does not remove or replace the network resource, and there are no applications/NFs using addresses or annotations affected by this change.	Replace	BareMetal and OpenStack

Procedure to update a network attachment

To add, delete, or update a CNLB network, perform the following procedure.

Update the `cnlb.ini` file before running the script, located at `/var/ocne/cluster/${OCCNE_CLUSTER}/installer/updNetwork.py`. Once the script is run successfully, verification must be performed.

Note

It is recommended to identify the changes to be made in the `cnlb.ini` file before running the `updNetwork.py` script. Running the script with updates that include deletion and/or updates of configuration made to the `cnlb.ini` file, can be disruptive to the current CNLB configuration. In such cases, validation can be performed only on the syntax of the `cnlb.ini` file.

Running the `updNetwork.py` script, it performs the following steps:

1. The `cnlb.ini` file is validated, ensuring the changes made are syntactically correct and that there are no unsupported duplicate IPs used.
2. The updated `cnlb.auto.tfvars` file is generated.
3. The `OpenTofu` is run to generate the new configuration resources.
4. The `installCnlb.py` is run to generate the Kubernetes objects in the cluster.

The following two files are generated which show the results of the script after it completes. Check these files to ensure that the script is successful.

1. `updNetwork-<timestamp>.log`
This file must include the output from the **OpenTofu** run.
2. `installer.log`
This file must include any logs generated by the `installCnlb.py` script.

Note

The `-h` option in the command syntax, displays the help.

Note

For this release, only `occne-infra` namespace is operational. If the script is run without any parameters, the default namespace used is `occne-infra`. The `-ns/--namespace` option can be used, but must be set to `occne-infra` namespace only. It is recommended not to insert any other namespace.

Run the following command to display the help option.

```
$ installer/updNetwork.py -h
```

The following sample output shows the usage of the configuration options:

```
usage: updNetwork.py [-h] [-ns NAMESPACE] [-db]
```

Used to update the network on an existing CNLB deployment.

Parameters:

Required parameters:
None

Optional Parameters:

```
-ns/--namespace: Namespace for network
-db/--debug: Print tracebacks on error
```

optional arguments:

```
-h, --help            show this help message and exit
-ns NAMESPACE, --namespace NAMESPACE
                        namespace for network
-db, --debug
```

Examples:

```
./updNetwork.py
./updNetwork.py -ns occne-infra -db
```

Updating an Existing CNLB Network Configuration

This procedure provides the steps to follow when updating an existing CNLB network configuration. The following example demonstrates how to add a network to the existing CNLB Network configuration.

For specific examples, refer to *Examples cases* section in this chapter. After running the `updNetwork.py` script, all changes made must be manually verified. Refer to the section, to validate the update.

1. Edit the `cnlb.ini` file located at `/var/occne/cluster/$OCCNE_CLUSTER/` to add the configuration for the new network. In this case, the network named **sig** is added to the `cnlb.ini` file, to the existing **oam** network. This includes adding the necessary service IPs along with all other necessary fields for the new network. The following sample code snippet shows how to add a **sig** network.

Run the following command to add the configurations required for the new network:

```
$ vi /var/occne/cluster/$OCCNE_CLUSTER/cnlb.ini
```

Below is a sample `cnlb.ini` file:

```
[cnlb]

[cnlb:vars]
cnlb_replica_cnt = 4

[cnlb:children]
oam
sig

[oam:vars]
service_network_name = "oam"
subnet_id = "43df8249-8316-48ed-a6b7-79de5413ddb"
network_id = "18e0s112-ac40-4c57-bedf-a16b9df497cf"
external_network_range = "10.199.180.0/24"
external_default_gw = "10.199.180.1"
egress_ip_addr = ["10.199.180.11", "10.199.180.189"]
internal_network_range = "132.16.0.0/24"
service_ip_addr = ["10.199.180.128", "10.199.180.10", "10.199.180.247",
"10.199.180.174", "10.199.180.131", "10.199.180.4"]
egress_dest = ["10.199.180.0/24", "10.123.155.0/25"]

[sig:vars]
```

```

service_network_name = "sig"
subnet_id = "2f317932-91bd-4440-a2fe-f3e5fc0af49c"
network_id = "7375bbd7-a787-4f33-ae60-dbcc58c075c2"
external_network_range = 10.199.201.0/24
external_default_gw = "10.199.201.1"
egress_ip_addr = ["10.199.201.10", "10.199.201.11"]
internal_network_range = 172.16.0.0/24
service_ip_addr = ["10.199.201.15", "10.199.201.16"]
shared_internal = False

```

2. For OpenStack-only deployment, source the `openrc.sh` file in the cluster directory and provide the OpenStack credentials as prompted.

```
$ source openrc.sh
```

3. Run the `updNetwork.py` script to add a **sig** network.

```
$ ./installer/updNetwork.py
```

Sample output:

```

Updating CNLB network as indicated in the cnlb.ini file
- Validating the cnlb.ini file
  - Successfully validated cnlb.ini file
- Generating new cnlb.auto.tfvars file
  - Successfully created cnlb.auto.tfvars file
- Initializing and running tofu apply
  - Tofu initialized
  - Running tofu apply... (may take several minutes)
    - Apply complete! Resources: 14 added, 0 changed, 0 destroyed
    - Successful run of tofu apply - check /var/ocne/cluster/ocne-user/
    updNetwork-08122024_164714.log for details
- Running installCnlb.py
  - Successfully ran installCnlb.py
- Restarting cnlb-manager and cnlb-app deployments
  - Deployment: cnlb-manager was restarted. Please wait for the pods
  status to be 'running'
  - Deployment: cnlb-app was restarted. Please wait for the pods status
  to be 'running'
- Network update successfully completed

```

VCD vApp Network Removal or Replacement Maintenance Workflow For vCNE VMware only:

On VCD 10.4.1 and later, removing a VCD vApp network from a powered-on vApp is not supported.

If a `cnlb.ini` change would remove or replace a VCD vApp network, `updNetwork.py` script blocks the change from being applied as an in-service CNLB network update. This block occurs before stale interface cleanup or `tofu apply` is run.

Procedure to Identify and Manually Remove or Replace VCD vApp Networks:

1. Run the `updNetwork.py` script first to identify the VCD vApp networks that must be manually removed while the vApp is powered off. The script detects and lists unsafe actions without applying changes or interrupting traffic.

In this example, the script detected two VCD vApp networks that require manual removal: sig-network and occne-test-dia-int-net. The first planned action removes a network. The second planned action replaces a network because of gateway or netmask changes. Use the reported VCD network name values when selecting networks in the VCD GUI.

Following is an example for blocked output:

```
$ ./installer/updNetwork.py
```

Example output:

```
ERROR: Unable to update network - err:
Unsafe VCD vApp network change detected in OpenTofu plan.
VCD 10.4.1+ requires the parent vApp to be powered off before removing a
vApp network.
Powering off the parent vApp will also power off the bastion nodes and the
entire cluster.
This operation is not safe and is blocked. Consult the documentation.
Planned VCD vApp network removal/replacement:
- module.isolated_network.vcd_vapp_network.isolated["sig"]
(vcd_vapp_network): actions=[delete], replace_paths=[N/A]
  VCD network name: sig-network
- module.isolated_network.vcd_vapp_network.isolated["dia"]
(vcd_vapp_network): actions=[delete, create], replace_paths=[gateway,
netmask]
  VCD network name: occne-test-dia-int-net
```

2. Perform the following manual vApp actions in the VMware Cloud Director (VCD) GUI: Navigate to All Virtual Data Centers > Data center: <target data center> > Compute > vApps > <target vApp>.
 - a. Power off the target VCD vApp.
 - b. In the target vApp, select **Networks**.
 - c. Select each affected VCD vApp network reported by updNetwork.py, and click **Delete**.
 - d. Power on the target VCD vApp.
3. Wait for all non-CNLB components to be up and running. The user is responsible for verifying cluster health after the vApp is powered on again.

Note

CNLB application pods are expected to be in CrashLoopBackOff. This is acceptable and is caused by the network removal. The subsequent script run restores the CNLB configuration.

- a. Verify that all nodes are Ready.


```
kubectl get nodes
```
- b. Verify that pods across all namespaces are up and running.


```
kubectl get pods -A
```

4. Run the `updNetwork.py` script again.

```
./installer/updNetwork.py
```

5. Restart applications or NFs and CNE Common Services so CNLB can refresh annotation status.

6. Verify that CNLB pods return to **Running**:

- a. Verify the CNLB App pods.

```
kubectl -n occne-infra get pods -l app=cnlb-app
```

- b. Verify the CNLB Manager pods.

```
kubectl -n occne-infra get pods -l app=cnlb-manager
```

Validating Updates

This procedure validates the CNLB network configuration update after running the `updNetwork.py` script.

All changes must be manually verified using the following commands. The required validation commands depend on the changes made in the `cnlb.ini` file.

Procedure

1. Run the following `describe` command to retrieve and display the data from `cnlb-manager-networks`.

```
kubectl describe cm cnlb-manager-networks -n occne-infra
```

Sample output:

```
Name:          cnlb-manager-networks
Namespace:     occne-infra
Labels:        <none>
Annotations:   <none>

Data
====
cnlb_manager_net_cm.json:
----
{
  "networks": [
    {
      "external-network": [
        {
          "range": "10.199.180.0/24",
          "gatewayIp": "10.199.180.1"
        }
      ],
      "internal-network": [
        {
          "range": "132.16.0.0/24",
          "egressIp": [

```

```

        "10.199.180.11",
        "132.16.0.193"
    ],
    [
        "10.199.180.189",
        "132.16.0.194"
    ]
]
}
],
"networkName": "oam"
},
{
    "external-network": [
        {
            "range": "10.199.201.0/24",
            "gatewayIp": "10.199.201.1"
        }
    ],
    "internal-network": [
        {
            "range": "172.16.0.0/24",
            "egressIp": [
                [
                    "10.199.201.10",
                    "172.16.0.193"
                ],
                [
                    "10.199.201.11",
                    "172.16.0.194"
                ]
            ]
        }
    ]
},
"networkName": "sig"
}
]
}
}

```

BinaryData

====

- The data can also be retrieved and displayed using the `kubectl get net-attach-def -A` command. This command retrieves the set of network attachment objects created for the **oam** and **sig** networks.

```
$ kubectl get net-attach-def -A
```

Sample output:

NAMESPACE	NAME	AGE
default	nf-oam-egr1	3h5m
default	nf-oam-egr2	3h5m
default	nf-oam-ie1	3h5m
default	nf-oam-ie2	3h5m

default	nf-oam-ie3	3h5m
default	nf-oam-ie4	3h5m
default	nf-oam-ie5	3h5m
default	nf-oam-ie6	3h5m
default	nf-oam-int1	3h5m
default	nf-oam-int2	3h5m
default	nf-oam-int3	3h5m
default	nf-oam-int4	3h5m
default	nf-oam-int5	3h5m
default	nf-oam-int6	3h5m
default	nf-sig-egr1	26m
default	nf-sig-egr2	26m
default	nf-sig-ie1	26m
default	nf-sig-ie2	26m
default	nf-sig-int1	86m
default	nf-sig-int2	86m
occne-infra	lb-oam-ext	3h5m
occne-infra	lb-oam-int	3h5m
occne-infra	lb-sig-ext	86m
occne-infra	lb-sig-int	86m

3. Validate the following:

- There is 1 "ie" object and 1 "int" for each service_ip_addr list.
- There is 1 "int" and 1 "egr" for each egress_ip_addr list.
- There are lb objects for external and internal networks for **oam** and **sig**.

4. **For Multiple networks only:** If SIN was enabled or disabled, verify the change by selecting one network attachment definition, such as int1, and running the following command. Confirm that the interfaces for the previously SIN-configured networks were updated accordingly.

Following is an example for SIN enabled:

Note

SIN-enabled networks share the same interface as OAM.

```
for x in oam sig; do kubectl describe net-attach-def nf-${x}-ie1 | egrep
"Name:|Config:";echo; done
```

Sample output:

```
Name:          nf-oam-ie1
Config:  {"cniVersion": "0.4.0", "name": "nf-oam-ie1", "plugins":
[{"type": "macvlan", "mode": "bridge", "master": "oam-int", "ipam":
{"type": "whereabouts", "range": "142.16.0.0/24", "range_start":
"142.16.0.129", "range_end": "142.16.0.190", "gateway": "142.16.0.1",
"routes": [{"dst": "10.75.212.0/23", "gw": "142.16.0.1"}]}}]
```

```
Name:          nf-sig-ie1
Config:  {"cniVersion": "0.4.0", "name": "nf-sig-ie1", "plugins":
[{"type": "macvlan", "mode": "bridge", "master": "oam-int", "ipam":
{"type": "whereabouts", "range": "143.16.0.0/24", "range_start":
```

```
"143.16.0.129", "range_end": "143.16.0.190", "gateway": "143.16.0.1",
"routes": [{"dst": "10.75.212.0/23", "gw": "143.16.0.1"}]}}}]}
```

Following is an example for SIN disabled:

Note

Networks disabled use their own interface.

```
$ for x in oam sig; do kubectl describe net-attach-def nf-${x}-ie1 | egrep
"Name:|Config:";echo; done
```

Sample output:

```
Name:          nf-oam-ie1
Config: {"cniVersion": "0.4.0", "name": "nf-oam-ie1", "plugins": [{"type":
"macvlan", "mode": "bridge", "master": "oam-int", "ipam": {"type":
"whereabouts", "range": "142.16.0.0/24", "range_start": "142.16.0.129",
"range_end": "142.16.0.190", "gateway": "142.16.0.1", "routes": [{"dst":
"10.75.212.0/23", "gw": "142.16.0.1"}]}}]}
```

```
Name:          nf-sig-ie1
Config: {"cniVersion": "0.4.0", "name": "nf-sig-ie1", "plugins": [{"type":
"macvlan", "mode": "bridge", "master": "sig-int", "ipam": {"type":
"whereabouts", "range": "143.16.0.0/24", "range_start": "143.16.0.129",
"range_end": "143.16.0.190", "gateway": "143.16.0.1", "routes": [{"dst":
"10.75.212.0/23", "gw": "143.16.0.1"}]}}]}
```

Example cases:

The examples provided in this section are for an OpenStack cluster deployment. The flow of the script is same for both the platforms, however, VMware includes additional configurations for the infrastructure after running **OpenTofu Apply**.

Note

The examples provided in this section, indicate at the beginning, if it corresponds to the output of the script ran on a vCNE cluster or on a VMware cluster.

Refer to the section, to validate the update of the CNLB network configuration.

Note

These are examples only. The IPs used must be changed as per the system on which these commands are run.

The subsequent sections provide in detail the following operations with a sample output:

- [Adding a New Network](#)
- [Deleting a network](#)

- [Adding an IP to the service_ip_addr list of an existing network](#)
- [Changing an IP in the service_ip_addr list of an existing network](#)
- [Deleting an IP from the service_ip_addr list of an existing network](#)
- [Adding a CIDR to the egress_dest field of an existing network](#)
- [Deleting a CIDR from the egress_dest field of an existing network](#)

8.11.1 Adding a New Network

This section describes the steps to add a new network to an existing CNLB deployment.

Prerequisites

Before adding a network, ensure that the **oam** network is added during CNE installation.

Procedure

The following procedure provides an example output of a script run on a VMware vCNE cluster.

1. Edit the `cnlb.ini` file to add the **sig** network to the `cnlb.ini` file.

```
[cnlb]

[cnlb:vars]
# if set to true, it will use list of worker nodes mentioned in cnlb group
to host lb client pods, if not set then use all
cnlb_replica_cnt = 4

[cnlb:children]

# Network names to be used for pod and external
# networking, these networks and ports
# will be created and attached to hosts
# in cnlb host group

oam
sig

[oam:vars]
service_network_name = "oam"
subnet_id = "cnlb-dhcp2"
network_id = "cnlb-dhcp2"
external_network_range = 10.199.180.0/25
external_default_gw = "10.199.180.1"
egress_ip_addr = ["10.199.180.76","10.199.180.77"]
internal_network_range = 132.16.0.0/24
service_ip_addr =
["10.199.180.78","10.199.180.79","10.199.180.80","10.199.180.81","10.199.18
0.82","10.199.180.83"]
# Optional variable. Uncomment and specify external destination subnets to
communicate with.
# If egress NADs are not needed for a network, then below variable should
not be added.
egress_dest = ["10.199.180.0/25"]
```

```
[sig:vars]
service_network_name = "sig"
subnet_id = "cnlb-dhcp2"
network_id = "cnlb-dhcp2"
external_network_range = 10.199.201.0/25
external_default_gw = "10.199.201.1"
egress_ip_addr = ["10.199.201.2", "10.199.201.6"]
internal_network_range = 172.16.0.0/24
service_ip_addr = ["10.199.201.16", "10.199.201.28"]
# Optional variable. Uncomment and specify external destination subnets to
communicate with.
# If egress NADs are not needed for a network, then below variable should
not be added.
egress_dest = ["10.199.201.0/25"]
shared_internal = False
```

2. Run the `updNetwork.py` script to update the CNLB network.

```
$ ./installer/updNetwork.py
```

Sample output:

```
Updating CNLB network as indicated in the cnlb.ini file
- Validating the cnlb.ini file
  - Successfully validated cnlb.ini file
- Generating new cnlb.auto.tfvars file
  - Successfully created cnlb.auto.tfvars file
- Removing Stale Network Interfaces
  - No stale interfaces found to remove
- Initializing and running tofu apply
  - Tofu initialized
  - Running tofu apply... (may take several minutes)
    - Apply complete! Resources: 1 added, 4 changed, 0 destroyed
    - Successful run of tofu apply - check /var/ocne/cluster/ocne-user/
    updNetwork-08092024_183741.log for details
- Renew DHCP Lease
  - Renewing DHCP lease for host ocne-user-k8s-node-1...
  - Renewing DHCP lease for host ocne-user-k8s-node-2...
  - Renewing DHCP lease for host ocne-user-k8s-node-3...
  - Renewing DHCP lease for host ocne-user-k8s-node-4...
  - DHCP lease was successfully renewed
- Validate Number of Networks and NICs match
  - Compare active connections against TF state number of networks
  - Success: Correct number of Active Connections across all nodes
- Refresh tofu state, refresh IP allocation
  - Successful run of tofu refresh - check /var/ocne/cluster/ocne-user/
    updNetwork-08092024_183741.log for details
- Running installCnlb.py
  - Successfully ran installCnlb.py
- Restarting cnlb-manager and cnlb-app deployments
  - Deployment: cnlb-manager was restarted. Please wait for the pods
    status to be 'running'
  - Deployment: cnlb-app was restarted. Please wait for the pods status
    to be 'running'
- Network update successfully completed
```

3. Verify the changes using the following command.

```
$ kubectl describe cm cnlb-manager-networks -n occne-infra
```

Sample output:

```
Name:          cnlb-manager-networks
Namespace:     occne-infra
Labels:        <none>
Annotations:   <none>

Data
====
cnlb_manager_net_cm.json:
----
{
  "networks": [
    {
      "external-network": [
        {
          "range": "10.199.180.0/24",
          "gatewayIp": "10.199.180.1"
        }
      ],
      "internal-network": [
        {
          "range": "132.16.0.0/24",
          "egressIp": [
            [
              "10.199.180.76",
              "132.16.0.193"
            ],
            [
              "10.199.180.77",
              "132.16.0.194"
            ]
          ]
        }
      ]
    },
    {
      "networkName": "oam"
    }
  ],
  {
    "external-network": [
      {
        "range": "10.199.201.0/24",
        "gatewayIp": "10.199.201.1"
      }
    ],
    "internal-network": [
      {
        "range": "172.16.0.0/24",
        "egressIp": [
          [
            "10.199.201.2",
            "172.16.0.193"
          ]
        ]
      }
    ]
  }
}
```

```

        ],
        [
            "10.199.201.6",
            "172.16.0.194"
        ]
    ]
}
],
"networkName": "sig"
}
]
}

BinaryData
====

```

After the verification, run the following command to fetch the additional default network attachment definitions `ie1` and `ie2` for **sig** network.

```
$ kubectl get net-attach-def -A
```

Sample output:

NAMESPACE	NAME	AGE
default	nf-oam-egr1	118m
default	nf-oam-egr2	118m
default	nf-oam-ie1	118m
default	nf-oam-ie2	118m
default	nf-oam-ie3	118m
default	nf-oam-ie4	118m
default	nf-oam-ie5	118m
default	nf-oam-ie6	118m
default	nf-oam-int1	118m
default	nf-oam-int2	118m
default	nf-oam-int3	118m
default	nf-oam-int4	118m
default	nf-oam-int5	118m
default	nf-oam-int6	118m
default	nf-sig-int1	19m
default	nf-sig-int2	19m
occne-infra	lb-oam-ext	118m
occne-infra	lb-oam-int	118m
occne-infra	lb-sig-ext	19m
occne-infra	lb-sig-int	19m

8.11.2 Deleting a network

This section describes the steps to delete a network.

Prerequisites

Before deleting a network, ensure that the **sig** network is added as explained in the section, [Adding a New Network](#) or ensure that the cluster was initially deployed with this network.

Procedure

The following procedure provides an example output of a script run on a VMware vCNE cluster.

1. Edit the `cnlb.ini` file to delete the **sig** network. Remove the **sig** name under the `cnlb:children` and the `sig:vars` groups with all the indicated fields as shown below.

```
[cnlb]

[cnlb:vars]
# if set to true, it will use list of worker nodes mentioned in cnlb group
to host lb client pods, if not set then use all
cnlb_replica_cnt = 4

[cnlb:children]

# Network names to be used for pod and external
# networking, these networks and ports
# will be created and attached to hosts
# in cnlb host group

oam
sig

[oam:vars]
service_network_name = "oam"
subnet_id = "cnlb-dhcp2"
network_id = "cnlb-dhcp2"
external_network_range = 10.199.180.0/25
external_default_gw = "10.199.180.1"
egress_ip_addr = ["10.199.180.76", "10.199.180.77"]
internal_network_range = 132.16.0.0/24
service_ip_addr =
["10.199.180.78", "10.199.180.79", "10.199.180.80", "10.199.180.81", "10.199.180.82", "10.199.180.83"]
# Optional variable. Uncomment and specify external destination subnets to
communicate with.
# If egress NADs are not needed for a network, then below variable should
not be added.
egress_dest = ["10.199.180.0/25"]
```

2. Run the **updNetwork.py** script to update the CNLB network.
For VMware on VCD 10.4.1 and later, the first run blocks before `tofu` apply and reports the affected VCD vApp network name. Refer to the section "VCD vApp Network Removal or

Replacement Maintenance Workflow", to perform the required manual VCD vApp network changes, and then rerun `updNetwork.py`.

After the VCD vApp network has been manually removed and `updNetwork.py` is rerun, the script completes the CNLB configuration update.

```
$ ./installer/updNetwork.py
```

Sample output:

Updating CNLB network as indicated in the `cnlb.ini` file

- Validating the `cnlb.ini` file
 - Successfully validated `cnlb.ini` file
- Generating new `cnlb.auto.tfvars` file
 - Successfully created `cnlb.auto.tfvars` file
- Initializing tofu
 - Tofu initialized
- Planning tofu changes
 - Saved temporary tofu plan: `/var/occne/cluster/occne-user/cnlb-tofu-plan.tmp`
- Checking VCD vApp network plan safety
 - Inspecting tofu plan for VCD vApp network safety check
 - No VCD vApp network removal or replacement detected
- Removing Stale Network Interfaces
 - No stale interfaces found to remove
- Running tofu apply
 - Running tofu apply using saved plan... (may take several minutes)
 - Apply complete! Resources: 0 added, 4 changed, 0 destroyed
 - Successful run of tofu apply - check `/var/occne/cluster/occne-user/updNetwork-08092024_183741.log` for details
- Activating disconnected VCD network devices
- Renew DHCP Lease
 - Renewing DHCP lease for host `occne-user-k8s-node-1...`
 - Renewing DHCP lease for host `occne-user-k8s-node-2...`
 - Renewing DHCP lease for host `occne-user-k8s-node-3...`
 - Renewing DHCP lease for host `occne-user-k8s-node-4...`
 - DHCP lease was successfully renewed
- Validate Number of Networks and NICs match
 - Compare active connections against expected network count per node type
 - Success: Correct number of Active Connections across all nodes
- Refresh tofu state, refresh IP allocation
 - Successful run of tofu refresh - check `/var/occne/cluster/occne-user/updNetwork-08092024_183741.log` for details
- Validating network interfaces health (after Tofu apply)
- Validating number of interfaces across nodes.
- Number of interfaces is consistent across nodes.
- Validating interfaces names and networks consistency across nodes.
 - Interfaces names and neworks are consistent across nodes
- Running `installCnlb.py`
 - Successfully ran `installCnlb.py`
- Restarting `cnlb-manager` and `cnlb-app` deployments
 - Deployment: `cnlb-manager` was restarted. Please wait for the pods status to be 'running'
 - Deployment: `cnlb-app` was restarted. Please wait for the pods status to be 'running'
- Network update successfully completed

3. Verify the changes using the following command:

```
$ kubectl describe cm cnlb-manager-networks -n occne-infra
```

Sample output:

```
Name:          cnlb-manager-networks
Namespace:     occne-infra
Labels:        <none>
Annotations:   <none>

Data
====
cnlb_manager_net_cm.json:
----
{
  "networks": [
    {
      "external-network": [
        {
          "range": "10.199.180.0/24",
          "gatewayIp": "10.199.180.1"
        }
      ],
      "internal-network": [
        {
          "range": "132.16.0.0/24",
          "egressIp": [
            [
              "10.199.180.76",
              "132.16.0.193"
            ],
            [
              "10.199.180.77",
              "132.16.0.194"
            ]
          ]
        }
      ]
    }
  ],
  "networkName": "oam"
}
```

After the verification, run the following command to fetch the additional default network attachment definitions for **oam** ie1 and ie2.

```
$ kubectl get net-attach-def -A
```

Sample output:

NAMESPACE	NAME	AGE
default	nf-oam-egr1	3h24m
default	nf-oam-egr2	3h24m
default	nf-oam-ie1	3h24m
default	nf-oam-ie2	3h24m
default	nf-oam-ie3	3h24m
default	nf-oam-ie4	3h24m
default	nf-oam-ie5	3h24m
default	nf-oam-ie6	3h24m
default	nf-oam-int1	3h24m
default	nf-oam-int2	3h24m
default	nf-oam-int3	3h24m
default	nf-oam-int4	3h24m
default	nf-oam-int5	3h24m
default	nf-oam-int6	3h24m
occne-infra	lb-oam-ext	3h24m
occne-infra	lb-oam-int	3h24m

8.11.3 Adding an IP to the `service_ip_addr` list of an existing network

This section describes the steps to add an IP to the `service_ip_addr` list of an existing network.

Prerequisites

Before adding an IP to a network, ensure that the **sig** network is added as explained in the section, [Adding a New Network](#).

Procedure

1. Edit the `cnlb.ini` file and add a valid external IP to the end of the `service_ip_addr` field for the **sig** network.
Below is a sample `cnlb.ini` file.

```
[cnlb]

[cnlb:vars]
# if set to true will use list of worker nodes mentioned in cnlb group to
host lb client pods, if not set then use all
cnlb_replica_cnt = 4

[cnlb:children]
# Network names to be used for pod and external
# networking, these networks and ports
# will be created and attached to hosts
# in cnlb host group
oam
sig

[oam:vars]
service_network_name = "oam"
subnet_id = "43df8249-8316-48ed-a6b7-79de5413ddb"
network_id = "18e0s112-ac40-4c57-bedf-a16b9df497cf"
```

```

external_network_range = "10.199.180.0/24"
external_default_gw = "10.199.180.1"
egress_ip_addr = ["10.199.180.11", "10.199.180.189"]
internal_network_range = "132.16.0.0/24"
service_ip_addr = ["10.199.180.128", "10.199.180.10", "10.199.180.247",
"10.199.180.174", "10.199.180.131", "10.199.180.4"]
egress_dest = ["10.199.180.0/24", "10.123.155.0/25"]

[sig:vars]
service_network_name = "sig"
subnet_id = "2f317932-91bd-4440-a2fe-f3e5fc0af49c"
network_id = "7375bbd7-a787-4f33-ae60-dbcc58c075c2"
external_network_range = 10.199.201.0/24
external_default_gw = "10.199.201.1"
egress_ip_addr = ["10.199.201.10", "10.199.201.11"]
internal_network_range = 172.16.0.0/24
service_ip_addr = ["10.199.201.15", "10.199.201.16", "10.199.201.76"]
shared_internal = False

```

The IP 10.199.201.76 is added.

2. Run the **updNetwork.py** script to update the CNLB network.

```
$ ./installer/updNetwork.py
```

Sample output:

```

Updating CNLB network as indicated in the cnlb.ini file
- Validating the cnlb.ini file
  - Successfully validated cnlb.ini file
- Generating new cnlb.auto.tfvars file
  - Successfully created cnlb.auto.tfvars file
- Initializing and running tofu apply
  - Tofu initialized
  - Running tofu apply... (may take several minutes)
    - Apply complete! Resources: 1 added, 0 changed, 0 destroyed
    - Successful run of tofu apply - check /var/ocne/cluster/ocne-user/
    updNetwork-08122024_171433.log for details
- Running installCnlb.py
  - Successfully ran installCnlb.py
- Restarting cnlb-manager and cnlb-app deployments
  - Deployment: cnlb-manager was restarted. Please wait for the pods
  status to be 'running'
  - Deployment: cnlb-app was restarted. Please wait for the pods status
  to be 'running'
- Network update successfully completed

```

3. Verify the changes using the following command:

```
$ kubectl describe cm cnlb-manager-networks -n ocne-infra
```

Sample output:

```
Name:      cnlb-manager-networks
Namespace: occne-infra
Labels:    <none>
Annotations: <none>

Data
====
cnlb_manager_net_cm.json:
----
{
  "networks": [
    {
      "external-network": [
        {
          "range": "10.199.180.0/24",
          "gatewayIp": "10.199.180.1"
        }
      ],
      "internal-network": [
        {
          "range": "132.16.0.0/24",
          "egressIp": [
            [
              "10.199.180.11",
              "132.16.0.193"
            ],
            [
              "10.199.180.189",
              "132.16.0.194"
            ]
          ]
        }
      ]
    }
  ],
  "networkName": "oam"
},
{
  "external-network": [
    {
      "range": "10.199.201.0/24",
      "gatewayIp": "10.199.201.1"
    }
  ],
  "internal-network": [
    {
      "range": "172.16.0.0/24",
      "egressIp": [
        [
          "10.199.201.10",
          "172.16.0.193"
        ],
        [
          "10.199.201.11",
          "172.16.0.194"
        ]
      ]
    }
  ]
}
```

```

    ]
  },
  "networkName": "sig"
}
]
}

```

```

BinaryData
====

```

4. After the verification, run the following command to fetch the additional default network attachment definitions.

```
$ kubectl get net-attach-def -A
```

Sample output:

NAMESPACE	NAME	AGE
default	nf-oam-egr1	132m
default	nf-oam-egr2	132m
default	nf-oam-ie1	132m
default	nf-oam-ie2	132m
default	nf-oam-ie3	132m
default	nf-oam-ie4	132m
default	nf-oam-ie5	132m
default	nf-oam-ie6	132m
default	nf-oam-int1	132m
default	nf-oam-int2	132m
default	nf-oam-int3	132m
default	nf-oam-int4	132m
default	nf-oam-int5	132m
default	nf-oam-int6	132m
default	nf-sig-int1	33m
default	nf-sig-int2	33m
default	nf-sig-int3	6m11s
occne-infra	lb-oam-ext	132m
occne-infra	lb-oam-int	132m
occne-infra	lb-sig-ext	33m
occne-infra	lb-sig-int	33m

8.11.4 Deleting an IP from the `service_ip_addr` list of an existing network

This section describes the steps to delete an IP from the `service_ip_addr` list of an existing network.

Prerequisites

Before proceeding to delete an IP from a network, ensure that the **sig** network is added as explained in the section, [Adding a New Network](#) in addition to the service IP 10.199.201.76.

Procedure

1. Edit the `cnlb.ini` file to remove the external IP 10.199.201.76 from the `service_ip_addr` field for the **sig** network.
Below is a sample `cnlb.ini` file.

```
[cnlb]

[cnlb:vars]
# if set to true, it will use list of worker nodes mentioned in cnlb group
to host lb client pods, if not set then use all
cnlb_replica_cnt = 4

[cnlb:children]

# Network names to be used for pod and external
# networking, these networks and ports
# will be created and attached to hosts
# in cnlb host group
oam
sig

[oam:vars]
service_network_name = "oam"
subnet_id = "43df8249-8316-48ed-a6b7-79de5413ddb"
network_id = "18e0s112-ac40-4c57-bedf-a16b9df497cf"
external_network_range = "10.199.180.0/24"
external_default_gw = "10.199.180.1"
egress_ip_addr = ["10.199.180.11", "10.199.180.189"]
internal_network_range = "132.16.0.0/24"
service_ip_addr = ["10.199.180.128", "10.199.180.10", "10.199.180.247",
"10.199.180.174", "10.199.180.131", "10.199.180.4"]
egress_dest = ["10.199.180.0/24", "10.123.155.0/25"]

[sig:vars]
service_network_name = "sig"
subnet_id = "2f317932-91bd-4440-a2fe-f3e5fc0af49c"
network_id = "7375bbd7-a787-4f33-ae60-dbcc58c075c2"
external_network_range = 10.199.201.0/24
external_default_gw = "10.199.201.1"
egress_ip_addr = ["10.199.201.10", "10.199.201.11"]
internal_network_range = 172.16.0.0/24
service_ip_addr = ["10.199.201.15", "10.199.201.16"]
shared_internal = False
```

2. Run the **updNetwork.py** script to update the CNLB network.

```
$ ./installer/updNetwork.py
```

Sample output:

```
Updating CNLB network as indicated in the cnlb.ini file
- Validating the cnlb.ini file
  - Successfully validated cnlb.ini file
```

- Generating new cnlb.auto.tfvars file
 - Successfully created cnlb.auto.tfvars file
- Initializing and running tofu apply
 - Tofu initialized
 - Running tofu apply... (may take several minutes)
 - Apply complete! Resources: 0 added, 0 changed, 1 destroyed
 - Successful run of tofu apply - check /var/ocne/cluster/ocne-user/updNetwork-08122024_172857.log for details
- Running installCnlb.py
 - Successfully ran installCnlb.py
- Restarting cnlb-manager and cnlb-app deployments
 - Deployment: cnlb-manager was restarted. Please wait for the pods status to be 'running'
 - Deployment: cnlb-app was restarted. Please wait for the pods status to be 'running'
- Network update successfully completed

3. Verify the changes using the following command.

```
$ kubectl describe cm cnlb-manager-networks -n occne-infra
```

Sample output:

```
Name:          cnlb-manager-networks
Namespace:     occne-infra
Labels:        <none>
Annotations:   <none>

Data
====
cnlb_manager_net_cm.json:
----
{
  "networks": [
    {
      "external-network": [
        {
          "range": "10.199.180.0/24",
          "gatewayIp": "10.199.180.1"
        }
      ],
      "internal-network": [
        {
          "range": "132.16.0.0/24",
          "egressIp": [
            [
              "10.199.180.11",
              "132.16.0.193"
            ],
            [
              "10.199.180.189",
              "132.16.0.194"
            ]
          ]
        }
      ]
    }
  ]
}
```

```

    ],
    "networkName": "oam"
  },
  {
    "external-network": [
      {
        "range": "10.199.201.0/24",
        "gatewayIp": "10.199.201.1"
      }
    ],
    "internal-network": [
      {
        "range": "172.16.0.0/24",
        "egressIp": [
          [
            "10.199.201.10",
            "172.16.0.193"
          ],
          [
            "10.199.201.11",
            "172.16.0.194"
          ]
        ]
      }
    ]
  },
  "networkName": "sig"
}
]
}

BinaryData
====

```

After the verification, run the following command to fetch the additional default network attachment definitions.

```
$ kubectl get net-attach-def -A
```

Sample output:

NAMESPACE	NAME	AGE
default	nf-oam-egr1	143m
default	nf-oam-egr2	143m
default	nf-oam-ie1	143m
default	nf-oam-ie2	143m
default	nf-oam-ie3	143m
default	nf-oam-ie4	143m
default	nf-oam-ie5	143m
default	nf-oam-ie6	143m
default	nf-oam-int1	143m
default	nf-oam-int2	143m
default	nf-oam-int3	143m
default	nf-oam-int4	143m
default	nf-oam-int5	143m
default	nf-oam-int6	143m

default	nf-sig-int1	44m
default	nf-sig-int2	44m
occne-infra	lb-oam-ext	143m
occne-infra	lb-oam-int	143m
occne-infra	lb-sig-ext	44m
occne-infra	lb-sig-int	44m

8.11.5 Changing an IP in the `service_ip_addr` list of an existing network

This section describes the steps to change an IP in the `service_ip_addr` list of an existing network.

Prerequisites

Before proceeding to delete an IP from a network, ensure that the **sig** network is added as explained in the section, [Adding a New Network](#) in addition to the service IP 10.199.201.76.

Procedure

1. Edit the `cnlb.ini` file to change the service IP 10.199.201.76 to 10.199.201.16 in the **service_ip_addr** field for the **sig** network.
Below is a sample `cnlb.ini` file.

```
[cnlb]

[cnlb:vars]
# if set to true will use list of worker nodes mentioned in cnlb group to
host lb client pods, if not set then use all
cnlb_replica_cnt = 4

[cnlb:children]

# Network names to be used for pod and external
# networking, these networks and ports
# will be created and attached to hosts
# in cnlb host group
oam
sig

[oam:vars]
service_network_name = "oam"
subnet_id = "43df8249-8316-48ed-a6b7-79de5413ddb"
network_id = "18e0s112-ac40-4c57-bedf-a16b9df497cf"
external_network_range = "10.199.180.0/24"
external_default_gw = "10.199.180.1"
egress_ip_addr = ["10.199.180.11", "10.199.180.189"]
internal_network_range = "132.16.0.0/24"
service_ip_addr = ["10.199.180.128", "10.199.180.10", "10.199.180.247",
"10.199.180.174", "10.199.180.131", "10.199.180.4"]
egress_dest = ["10.199.180.0/24", "10.123.155.0/25"]

[sig:vars]
service_network_name = "sig"
subnet_id = "2f317932-91bd-4440-a2fe-f3e5fc0af49c"
network_id = "7375bbd7-a787-4f33-ae60-dbcc58c075c2"
external_network_range = 10.199.201.0/24
```

```
external_default_gw = "10.199.201.1"
egress_ip_addr = ["10.199.201.10", "10.199.201.11"]
internal_network_range = 172.16.0.0/24
service_ip_addr = ["10.199.201.15", "10.199.201.16", "10.199.201.16"]
shared_internal = False
```

2. Run the **updNetwork.py** script to update the CNLB network.

```
$ ./installer/updNetwork.py
```

Sample output:

```
Updating CNLB network as indicated in the cnlb.ini file
- Validating the cnlb.ini file
  - Successfully validated cnlb.ini file
- Generating new cnlb.auto.tfvars file
  - Successfully created cnlb.auto.tfvars file
- Initializing and running tofu apply
  - Tofu initialized
  - Running tofu apply... (may take several minutes)
    - Apply complete! Resources: 1 added, 0 changed, 1 destroyed
    - Successful run of tofu apply - check /var/ocne/cluster/ocne-user/
    updNetwork-08122024_173855.log for details
- Running installCnlb.py
  - Successfully ran installCnlb.py
- Restarting cnlb-manager and cnlb-app deployments
  - Deployment: cnlb-manager was restarted. Please wait for the pods
  status to be 'running'
  - Deployment: cnlb-app was restarted. Please wait for the pods status
  to be 'running'
- Network update successfully completed
```

3. Verify the changes using the following commands.

```
$ kubectl describe cm cnlb-manager-networks -n ocne-infra
$ kubectl get net-attach-def -A
```

Sample output:

```
Name:          cnlb-manager-networks
Namespace:     ocne-infra
Labels:        <none>
Annotations:    <none>

Data
====
cnlb_manager_net_cm.json:
----
{
  "networks": [
    {
      "external-network": [
        {
          "range": "10.199.180.0/24",
```

```

        "gatewayIp": "10.199.180.1"
    },
    ],
    "internal-network": [
        {
            "range": "132.16.0.0/24",
            "egressIp": [
                [
                    "10.199.180.11",
                    "132.16.0.193"
                ],
                [
                    "10.199.180.189",
                    "132.16.0.194"
                ]
            ]
        }
    ],
    ],
    "networkName": "oam"
},
{
    "external-network": [
        {
            "range": "10.199.201.0/24",
            "gatewayIp": "10.199.201.1"
        }
    ],
    "internal-network": [
        {
            "range": "172.16.0.0/24",
            "egressIp": [
                [
                    "10.199.201.10",
                    "172.16.0.193"
                ],
                [
                    "10.199.201.11",
                    "172.16.0.194"
                ]
            ]
        }
    ],
    ],
    "networkName": "sig"
}
]
}

```

BinaryData
====

Run the following command:

```
$ kubectl get net-attach-def -A
```

NAMESPACE	NAME	AGE
default	nf-oam-egr1	153m
default	nf-oam-egr2	153m
default	nf-oam-ie1	153m
default	nf-oam-ie2	153m
default	nf-oam-ie3	153m
default	nf-oam-ie4	153m
default	nf-oam-ie5	153m
default	nf-oam-ie6	153m
default	nf-oam-int1	153m
default	nf-oam-int2	153m
default	nf-oam-int3	153m
default	nf-oam-int4	153m
default	nf-oam-int5	153m
default	nf-oam-int6	153m
default	nf-sig-int1	53m
default	nf-sig-int2	53m
occne-infra	lb-oam-ext	153m
occne-infra	lb-oam-int	153m
occne-infra	lb-sig-ext	53m
occne-infra	lb-sig-int	53m

8.11.6 Adding a CIDR to the egress_dest field of an existing network

This section describes the steps to add a Classless Inter-Domain Routing (CIDR) to the egress_dest field of an existing network.

Prerequisites

Before adding a CIDR to the egress_dest field, ensure that the **sig** network is added as explained in the section, [Adding a New Network](#).

Validation commands

The following commands are used to validate the changes.

Table 8-26 Validation commands

Command	Purpose
<code>kubectl describe cm cnlb-manager-networks -n occne-infra</code>	This command is used to retrieve and display the data from cnlb-manager-networks.
<code>kubectl get net-attach-def -A</code>	This command is used to get the network attachment definition through out the cluster. Here, -A represents all namespaces.
<code>kubectl describe net-attach-def nf-sig-egr1 -n default</code>	This command is used to retrieve the contents of the network attachment definition, nf-sig-egr1 in default namespace.

Procedure

1. Edit the `cnlb.ini` file to add the IP, `10.199.100.0/25` to the `egress_dest` field for the **sig** network.

For more information on how to add **sig** network, see [Adding a New Network](#).

Below is a sample `cnlb.ini` file.

```
[cnlb]

[cnlb:vars]
# if set to true, it will use list of worker nodes mentioned in cnlb group
to host lb client pods, if not set then use all
cnlb_replica_cnt = 4

[cnlb:children]

# Network names to be used for pod and external
# networking, these networks and ports
# will be created and attached to hosts
# in cnlb host group
oam
sig

[oam:vars]
service_network_name = "oam"
subnet_id = "43df8249-8316-48ed-a6b7-79de5413ddb"
network_id = "18e0s112-ac40-4c57-bedf-a16b9df497cf"
external_network_range = "10.199.180.0/24"
external_default_gw = "10.199.180.1"
egress_ip_addr = ["10.199.180.11", "10.199.180.189"]
internal_network_range = "132.16.0.0/24"
service_ip_addr = ["10.199.180.128", "10.199.180.10", "10.199.180.247",
"10.199.180.174", "10.199.180.131", "10.199.180.4"]
egress_dest = ["10.199.180.0/24", "10.123.155.0/25"]

[sig:vars]
service_network_name = "sig"
subnet_id = "2f317932-91bd-4440-a2fe-f3e5fc0af49c"
network_id = "7375bbd7-a787-4f33-ae60-dbcc58c075c2"
external_network_range = "10.199.201.0/24"
external_default_gw = "10.199.201.1"
egress_ip_addr = ["10.199.201.10", "10.199.201.11"]
internal_network_range = "172.16.0.0/24"
service_ip_addr = ["10.199.201.15", "10.199.201.16", "10.199.201.16"]
egress_dest = ["10.199.201.0/25", "10.199.100.0/25"]
shared_internal = False
```

2. Run the **updNetwork.py** script to update the CNLB network.

```
$ ./installer/updNetwork.py
```

Sample output:

```
Updating CNLB network as indicated in the cnlb.ini file
- Validating the cnlb.ini file
  - Successfully validated cnlb.ini file
- Generating new cnlb.auto.tfvars file
  - Successfully created cnlb.auto.tfvars file
- Initializing and running tofu apply
  - Tofu initialized
  - Running tofu apply... (may take several minutes)
    - Apply complete! Resources: 0 added, 0 changed, 0 destroyed
    - Successful run of tofu apply - check /var/ocne/cluster/ocne-user/
updnetwork-08122024_174922.log for details
- Running installCnlb.py
  - Successfully ran installCnlb.py
- Restarting cnlb-manager and cnlb-app deployments
  - Deployment: cnlb-manager was restarted. Please wait for the pods
status to be 'running'
  - Deployment: cnlb-app was restarted. Please wait for the pods status
to be 'running'
- Network update successfully completed
```

3. Verify the changes using the following command.

```
$ kubectl describe net-attach-def nf-sig-egr1 -n default
```

The parameter net-attach-def route from the configuration is added to the /Spec/ Config section and also as shown below. See **"routes"** below where {"dst": "10.199.100.0/25", "gw": "172.16.0.193"} is added.

Sample output:

```
Name:          nf-sig-egr1
Namespace:     default
Labels:        <none>
Annotations:   <none>
API Version:   k8s.cni.cncf.io/v1
Kind:          NetworkAttachmentDefinition
Metadata:
  Creation Timestamp:  2024-08-12T17:48:09Z
  Generation:         2
  Resource Version:    55141
  UID:                f3720ac4-f49e-487b-a0cc-5b2096d12432
Spec:
  Config:  {"cniVersion": "0.4.0", "name": "nf-sig-egr1", "plugins":
[{"type": "macvlan", "mode": "bridge", "master": "eth3", "ipam": {"type":
"whereabouts", "range": "132.16.0.0/24", "range_start": "132.16.0.129",
"range_end": "132.16.0.190", "gateway": "132.16.0.193", "routes": [{"dst":
"10.199.201.0/23", "gw": "132.16.0.193"}, {"dst": "10.199.100.0/25", "gw":
"172.16.0.193"}]}]}]
Events:    <none>
```

4. Restart the affected NF pods so that the updated CNLB network configuration is picked up by the application or NF.

8.11.7 Deleting a CIDR from the `egress_dest` field of an existing network

This section describes the procedure to delete a CIDR from the `egress_dest` field from an existing network

Prerequisites

Before deleting a CIDR from the `egress_dest` field, ensure that the **sig** network is added as explained in the section, [Adding a New Network](#).

Procedure

1. Edit the `cnlb.ini` file and delete IP `10.199.100.0/25` from the `egress_dest` field for the **sig** network.
Below is a sample `cnlb.ini` file.

```
[cnlb]

[cnlb:vars]
# if set to true, it will use list of worker nodes mentioned in cnlb group
to host lb client pods, if not set then use all
cnlb_replica_cnt = 4

[cnlb:children]
# Network names to be used for pod and external
# networking, these networks and ports
# will be created and attached to hosts
# in cnlb host group
oam
sig

[oam:vars]
service_network_name = "oam"
subnet_id = "43df8249-8316-48ed-a6b7-79de5413ddb"
network_id = "18e0s112-ac40-4c57-bedf-a16b9df497cf"
external_network_range = "10.199.180.0/24"
external_default_gw = "10.199.180.1"
egress_ip_addr = ["10.199.180.11", "10.199.180.189"]
internal_network_range = "132.16.0.0/24"
service_ip_addr = ["10.199.180.128", "10.199.180.10", "10.199.180.247",
"10.199.180.174", "10.199.180.131", "10.199.180.4"]
egress_dest = ["10.199.180.0/24", "10.123.155.0/25"]

[sig:vars]
service_network_name = "sig"
subnet_id = "2f317932-91bd-4440-a2fe-f3e5fc0af49c"
network_id = "7375bbd7-a787-4f33-ae60-dbcc58c075c2"
external_network_range = 10.199.201.0/24
external_default_gw = "10.199.201.1"
egress_ip_addr = ["10.199.201.10", "10.199.201.11"]
internal_network_range = 172.16.0.0/24
service_ip_addr = ["10.199.201.15", "10.199.201.16", "10.199.201.16"]
egress_dest = ["10.199.201.0/25"]
shared_internal = False
```

2. Run the **updNetwork.py** script to update the CNLB network.

```
$ ./installer/updNetwork.py
```

Sample output:

```
Updating CNLB network as indicated in the cnlb.ini file
- Validating the cnlb.ini file
  - Successfully validated cnlb.ini file
- Generating new cnlb.auto.tfvars file
  - Successfully created cnlb.auto.tfvars file
- Initializing and running tofu apply
  - Tofu initialized
  - Running tofu apply... (may take several minutes)
    - Apply complete! Resources: 0 added, 0 changed, 0 destroyed
    - Successful run of tofu apply - check /var/occone/cluster/occone-user/
      updNetwork-08122024_175715.log for details
- Running installCnlb.py
  - Successfully ran installCnlb.py
- Restarting cnlb-manager and cnlb-app deployments
  - Deployment: cnlb-manager was restarted. Please wait for the pods
    status to be 'running'
  - Deployment: cnlb-app was restarted. Please wait for the pods status
    to be 'running'
- Network update successfully completed
```

3. Verify the following changes by running the command `$ kubectl describe net-attach-def nf-sig-egr1 -n default`:
 - route net-attach-def is removed from /spec/config section as shown in the sample output below
 - Observe that in the routes field, the IPs {"dst": "10.199.100.0/25", "gw": "172.16.0.193"} are deleted.

Sample output:

```
Name:      nf-sig-egr1
Namespace: default
Labels:    <none>
Annotations: <none>
API Version: k8s.cni.cncf.io/v1
Kind:      NetworkAttachmentDefinition
Metadata:
  Creation Timestamp: 2024-08-12T17:48:09Z
  Generation:        2
  Resource Version:   55141
  UID:                f3720ac4-f49e-487b-a0cc-5b2096d12432
Spec:
  Config: {"cniVersion": "0.4.0", "name": "nf-sig-egr1", "plugins":
[{"type": "macvlan", "mode": "bridge", "master": "eth3", "ipam": {"type":
"whereabouts", "range": "132.16.0.0/24", "range_start": "132.16.0.129",
"range_end": "132.16.0.190", "gateway": "132.16.0.193", "routes": [{"dst":
"10.199.201.0/23", "gw": "132.16.0.193"}]}]}]}
Events:    <none>
```

4. Restart the affected NF pods so that the updated CNLB network configuration is picked up by the application or NF.

8.11.8 Common Issues and Solutions

This section outlines basic steps to resolve common errors encountered when running the network update procedure.

Failed to connect to nodes

This error typically occurs when one or more cluster nodes cannot be reached.

Cause	Solution
Nodes are powered off	Verify all nodes are powered on.
Nodes have incorrect or incomplete network settings	Ensure that network connectivity is enabled on all nodes and that network settings (IP, gateway, routes, DNS as applicable) are configured correctly. Confirm that the nodes are reachable over the management or control network (for example, using ping or SSH, depending on your environment).

Inconsistent interfaces found across nodes

This error typically occurs when one or more nodes have different network interface names than the rest of the cluster.

Cause	Solution
Network interface names changed on one or more nodes	Ensure interface names are consistent and enabled across all nodes. If naming is driven by udev rules, reloading udev rules may resolve the issue. A common approach is to safely reboot the affected nodes one at a time (drain each node before rebooting, if applicable to your cluster).
Network interface IP addresses are misconfigured	Verify the relevant interfaces are configured on the correct subnet and consistently across all nodes.

8.12 Secure DNS Zone Customization through CNLB

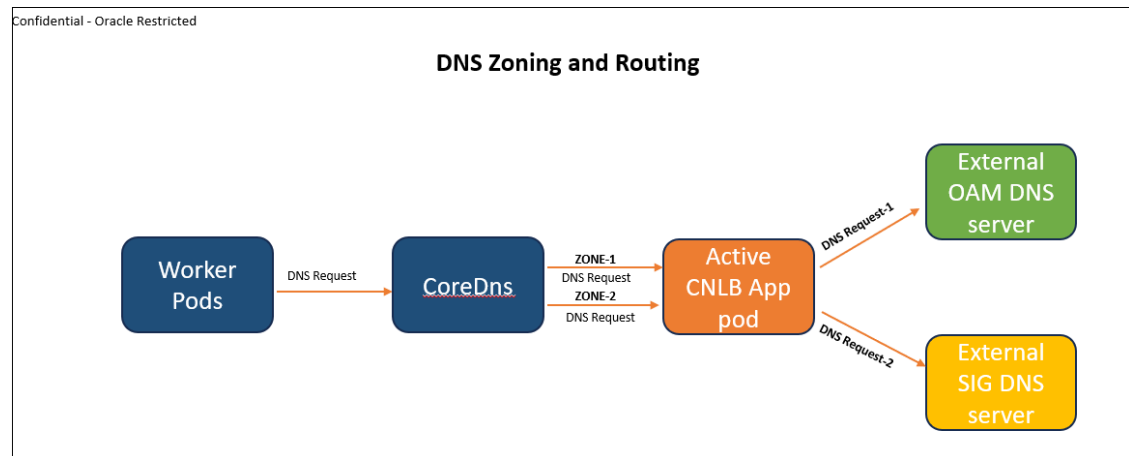
Overview

DNS Zone Customization functionality in the CNE environment enhances the security and scalability of DNS requests by routing traffic based on domain zones, such as OAM or Signaling. The feature isolates DNS traffic and forwards requests through Cloud Native Load Balancers (CNLB) to external DNS servers, which are managed by customers. CNE does not control these external servers, but ensures the secure routing of requests.

Note

When this feature is enabled, it is required that CoreDNS pods are not scheduled to run on control plane nodes.

DNS Zone Customization feature modifies the DNS resolution flow within the CNE environment to optimize name resolution and centralize DNS routing through the CNLB application. It improves the control over external DNS traffic by routing it through an active CNLB application, bypassing the Bastion-host-based resolution flow.



DNS requests originating from worker pods are routed through CoreDNS, which examines the requested domain zone (for example, OAM, Signaling). Depending on the zone type, the DNS request is forwarded through the appropriate CNLB (Cloud Native Load Balancer) application to the corresponding external DNS server.

Egress Network Attachment Definitions (NADs) isolate DNS traffic paths for different domain zones, such as OAM and Signaling. CoreDNS pods must not be running on control nodes when this feature is enabled, it must be restricted to worker nodes only, ensuring that DNS requests are handled within the local cluster network before being forwarded to external DNS servers.

DNS traffic is routed through specific CNLB interfaces, enabling detailed control over egress traffic and supporting customer-specific routing rules to ensure each zone's traffic uses a secure and dedicated path.

① Note

- The cluster should be CNLB enabled.
- External IPs of the DNS servers will be provided and maintained by the Customers.
- During the implementation of the DNS enhancement procedure, CNE cluster may experience a momentary service disruption due to transient DNS lookup unavailability affecting Network Functions (NFs) and other dependent components during the configuration switchover. The disruption is brief and self-limiting, with services expected to automatically stabilize upon completion of the procedure.

① Note

After activating the flow, no issues have been observed during subsequent self-update and upgrade processes. All components continue to function as expected. Additionally, the ConfigMap changes for NodeLocalDNS and CoreDNS remain intact.

Switch DNS Routing using CNLB

Switching DNS routing to utilize the Cloud Native Load Balancer (CNLB) improves scalability and reliability. The switchover procedure involves the following steps:

1. Determining the active Bastion Host
2. Defining variables to configure the External DNS server and zone settings in `occne.ini`
3. Enabling TLS in zones
4. Enabling DNS enhancement

8.12.1 Determining Active Bastion Host

1. Log in to either Bastion Host (typically Bastion 1).
2. Check if the Bastion Host is hosting the active cluster services by running the following command:

```
$ is_active_bastion
```

Sample output:

```
IS active-bastion
```

3. If the current Bastion is not active, log out and connect to the mate Bastion Host.
4. Repeat the check using the same command:

```
$ is_active_bastion
```

Sample output:

```
IS active-bastion
```

5. Confirm that the Bastion is hosting active services before proceeding with any further actions.

8.12.2 Defining Variables for External DNS Server

Populate the required values in the `occne.ini` file or `hosts.ini` for BM file.

For this step, ensure that the following values are correctly populated in the `[occne:vars]` section of the `occne.ini` file for vCNE deployments, or in the `hosts.ini` file for BareMetal deployments.

Table 8-27 Variables for External DNS Server

Field Name	Description	Required/Optional	Structure	Notes
enable_dns_enhancement	This field enables the DNS enhancement flow when set to True.	Required	enable_dns_enhancement: True	• Boolean value • Set to True to activate DNS enhancement logic • Set to False for rollout the DNS enhancement
upstream_dns_servers	Indicates if any request that does not match the zones defined in coredns_external_zones will be forwarded to these upstream servers for resolution.	Required	upstream_dns_servers: ["IP1"]	• Must be a list of IP addresses • If not set, non-matching queries will fail • Can include multiple IPs
coredns_egress_nad	Indicates the list of Network Attachment Definitions (NADs) for routing CoreDNS egress traffic.	Required	coredns_egress_nad = ["default/nad-egr@nad-egr"]	• Traffic is routed through CNLB interfaces • Can include one or more NAD names
coredns_external_zones	Each entry is a combination of the fields: zones, nameservers, and cache used to define how specific DNS queries must be resolved using external DNS servers.	Required	coredns_external_zones = [{"cache": 30, "zones": ["zone1"], "nameservers": ["IP1"]}]	• JSON array defining zone names, caching TTL, and their corresponding nameservers. Must be a valid stringified JSON. • defines a list of external DNS zones and their respective upstream nameservers to which CoreDNS will forward matching queries, optionally with rewrite rules.

The following table lists the fields to be added to the coredns_external_zones variable.

Table 8-28 coredns_external_zones Fields

Field Name	Description	Required/Optional	Structure	Notes
zones	List of zones (domains) for which DNS queries should be forwarded	Required	"zones": ["zone1"]	• Must be a list of strings • Multiple zones can be added as part of this list

Table 8-28 (Cont.) coredns_external_zones Fields

Field Name	Description	Required/Optional	Structure	Notes
nameservers	List of External DNS server IPs to which these zone queries should be forwarded	Required	"nameservers": ["IP1"]	- Must be a list of valid IP addresses - Multiple entries are supported
cache	TTL (in seconds) for caching responses from the DNS servers	Required	"cache": 30	- Cache cannot be 0 or a negative number - Useful to reduce DNS resolution time for frequently queried domains
rewrite	Rules to rewrite DNS queries before forwarding (for example, changing domain names)	Optional	"rewrite": ["rule1", "rule2"]	- Must be a list of strings - List of rewrite rules - Each rule follows CoreDNS syntax

Example:

```
[ocne:vars]
enable_dns_enhancement = True
upstream_dns_servers = ["10.**.**.*"]
coredns_egress_nad = ["default/egr-nad-name@egr-nad-name"]
coredns_external_zones = [{"zones":
["example1.com", "example2.io:1053"], "nameservers":
["1.1.1.1", "2.2.2.2"], "cache": 5}, {"zones":
["examplesvc1.local:4453"], "nameservers": ["192.168.0.53"], "cache": 9}, {"zones":
["exampledomain.tld"], "nameservers": ["10.233.0.3"], "cache": 5, "rewrite": ["name
stop example.tld example.namespace.svc.cluster.local"]}]}
```

Note

- Ensure that zones are written entirely in lowercase letters.
- The `enable_dns_enhancement` parameter requires an exact boolean value of `True` (case-sensitive) to activate DNS enhancement. Any deviation from this value, such as `true` or `1`, will result in the feature being disabled.
- DNS enhancement feature can also be enabled along with local DNS feature.
- If `enable_dns_enhancement` is set to `True`, then all the other 3 variables are required, else it will fail.

8.12.3 Enabling TLS in Zones

This section explains how to enable TLS for secure communication in the DNS Enhancement setup.

If TLS is not required for communication between CoreDNS and external DNS servers, this section can be skipped.

CoreDNS supports TLS connections with external DNS servers. DNS enhancement allows to enable TLS in each zone independently.

Enabling TLS in Zones

1. For vCNE deployments, edit the `occne.ini` file in the cluster directory:

```
$ vi /var/occne/cluster/${OCCNE_CLUSTER}/occne.ini
```

2. For BareMetal deployments, edit the `hosts.ini` file in the cluster directory:

```
$ vi /var/occne/cluster/${OCCNE_CLUSTER}/hosts.ini
```

3. To enable TLS in DNS enhancement, a JSON Object named `tls` must be defined in the `coredns_external_zones` variable.

The following table lists the fields to be added to the `tls` JSON Object.

Table 8-29 `tls` JSON Object Variables

Field	Description	Required/ Optional	Structure	Notes
<code>tls_port</code>	Port used for communication with the external DNS server via TLS	Optional	<code>"tls_port": "port_number"</code>	• Default port for TLS is 853 • Other port numbers may be used
<code>tls_servername</code>	Server name of the external DNS server used during the TLS handshake.	Required	<code>"tls_servername": "server_name"</code>	• Used to validate certificates for TLS handshake

Table 8-29 (Cont.) tls JSON Object Variables

Field	Description	Required/ Optional	Structure	Notes
tls_key_path	Relative path of client's key used during the TLS handshake.	Optional	"tls_key_path": "client.key"	• Depends on the TLS configuration of the external DNS server • If defined, tls_cert_path must be also defined • When omitted, no client authentication is performed • The TLS client key must be created or located under the /var/occne/cluster/\${OCCNE_CLUSTER}/installer/dns_enhancement directory. Sub directories are allowed.

Table 8-29 (Cont.) tls JSON Object Variables

Field	Description	Required/ Optional	Structure	Notes
tls_cert_path	Relative path of client's cert used during the TLS handshake.	Optional	"tls_cert_path": "client.crt"	• Depends on the TLS configuration of the external DNS server • If defined, tls_key_path must be also defined • When omitted, no client authentication is performed • The TLS client cert must be created or located under the /var/occne/cluster/\${OCCNE_CLUSTER}/installer/dns_enhancement directory. Sub directories are allowed.

Table 8-29 (Cont.) tls JSON Object Variables

Field	Description	Required/ Optional	Structure	Notes
tls_ca_path	Relative path of CA cert used during the TLS handshake.	Optional	"tls_ca_path": "ca.crt"	• Depends on the TLS configuration of the external DNS server • Can be used without defining tls_cert_path and tls_key_path. • The TLS CA cert must be created or located under the /var/occne/cluster/\${OCCNE_CLUSTER}/installer/dns_enhancement directory. Sub directories are allowed.

Note

- TLS configuration requires zones that have it enabled, to only have one nameserver defined.
- Ensure that `tls_servername` is always defined in the zones that have TLS enabled.
- If `tls_key_path`, `tls_cert_path`, or `tls_ca_path` uses subdirectories, the value must be specified using the following directory structure:

`sub_directory/second_sub_directory/client_key_cert_or_ca_cert`

Following is an example TLS configuration:

```
[occne:vars]...coredns_external_zones = [{"zones":
["example1.com", "example2.io:1053"], "nameservers":
["1.1.1.1"], "cache": 5, "tls":
{"tls_port": "853", "tls_key_path": "example1_zone_keys/
coredns.key", "tls_cert_path": "example1_zone_keys/certs/
coredns.crt", "tls_servername": "example.local"}}, {"zones":
["examplesvc1.local:4453"], "nameservers":
["192.168.0.53"], "cache": 5}, {"zones":
["exampledomain.tld"], "nameservers":
["10.233.0.3"], "cache": 5, "tls":
{"tls_ca_path": "ca.crt", "tls_servername": "exampledomain.local"}, "
rewrite": ["name stop example.tld
example.namespace.svc.cluster.local"]}]
```

8.12.4 Enabling DNS Enhancement

The `coreDnsEnhancement.py` script is used to enable the DNS enhancement feature.

This script automates the necessary configuration steps to update CoreDNS and related components as part of the DNS enhancement process.

Enabling DNS Enhancement

1. Navigate to the cluster directory.

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

2. Run the `coreDnsEnhancement.py` script.

The `coreDnsEnhancement.py` script configures and enhances the DNS setup based on the `occne.ini` file for vCNE deployments or `hosts.ini` file for BareMetal deployment.

```
./installer/dns_enhancement/coreDnsEnhancement.py
```

Sample output:

```
2025-07-24 09:06:55,077 CNLB_LOGGER:INFO: step-1: Load ini file into file
parser
2025-07-24 09:06:55,078 CNLB_LOGGER:INFO: step-2: Check for
```

```

enable_dns_enhancement parameter
2025-07-24 09:06:55,078 CNLB_LOGGER:INFO: step-3: Check for
upstream_dns_servers parameter
2025-07-24 09:06:55,078 CNLB_LOGGER:INFO: upstream_dns_servers is defined
with value: ['10.75.200.13']
2025-07-24 09:06:55,078 CNLB_LOGGER:INFO: step-4: Check for
coredns_egress_nad parameter
2025-07-24 09:06:55,078 CNLB_LOGGER:INFO: coredns_egress_nad is defined
with value: ['default/nf-oam-egr2@nf-oam-egr2']
2025-07-24 09:06:55,078 CNLB_LOGGER:INFO: NAD to be attached to coredns
is: nf-oam-egr2
2025-07-24 09:06:55,078 CNLB_LOGGER:INFO: step-5: Check for
coredns_external_zones parameter
2025-07-24 09:06:55,078 CNLB_LOGGER:INFO: step-6: Take back up of
configurations
2025-07-24 09:06:55,273 CNLB_LOGGER:INFO: step-7: Run kubernetes pipeline
to update coreDNS
2025-07-24 09:07:37,107 CNLB_LOGGER:INFO: step-8: update the zones file
with new data
2025-07-24 09:07:37,113 CNLB_LOGGER:INFO: step-9: restart resources for
adoption of changes
2025-07-24 09:07:37,322 CNLB_LOGGER:INFO: step-10: push dns validation
image to occne-repo-host
2025-07-24 09:07:42,134 CNLB_LOGGER:INFO: SUCCESS: DNS is enhanced, now
coredns can reach external zones

```

Using coreDnsEnhancement.py script:

```

[cloud-user@occne-cluster-name-bastion-1 cluster-name]$ ./installer/
dns_enhancement/coreDnsEnhancement.py -h
usage: coreDnsEnhancement.py [-h] [-nn TESTNODENAME] [-t TESTRECORD] [-sz]
[-d]

```

Used to update the network on an existing CNLB deployment.

Parameters:

Required parameters:

None

Optional Parameters:

```

-nn/--testNodeName: Node to be used for testing the DNS query (allowed
only with -t)
-t/--testRecord: runs test only
-sz/--syncZoneFile: Only syncs external zone file in bastion host
-d/--debugOn: sets the debug level as On

```

Following are some example arguments that can be passed with the script:

```

./coreDnsEnhancement.py
./coreDnsEnhancement.py -nn k8s-node-1 -t testrecord.oam
./coreDnsEnhancement.py -t testrecord.oam
./coreDnsEnhancement.py -sz
./coreDnsEnhancement.py -d

```

Following are some of the optional arguments:

- -h, --help show this help message and exit
- -nn TESTNODENAME, --testNodeName TESTNODENAME Node used for testing the DNS query
- -t TESTRECORD, --testRecord TESTRECORD runs only test of enhancement for the given record
- -sz, --syncZoneFile only syncs the zone file in bastion host
- -d, --debugOn sets the log level as Debug

Note

- The -sz argument syncs the zone file on the Bastion host with the `occne.ini` file for vCNE deployments or the `hosts.ini` file for BareMetal deployments.
- Test node name can only be sent with the -t option, else it will be ignored and the setup proceeds with the default parameters.

```
./installer/dns_enhancement/coreDnsEnhancement.py -t <zonename> -nn  
k8s-node-1
```

3. Testing DNS Resolution:

The DNS enhancement script includes a testing feature that enables verification of DNS resolution for a specified zone name. To perform a DNS test, run the script with the -t option followed by a valid zone name.

Example 1:

```
./installer/dns_enhancement/coreDnsEnhancement.py -t testrecord.abc
```

Sample output:

```
2025-07-24 09:11:45,489 CNLB_LOGGER:INFO: Update configurations in  
validationPod.yaml and kubectl apply it.  
2025-07-24 09:11:49,845 CNLB_LOGGER:INFO: waiting for dnsvalidation pod to  
be up  
2025-07-24 09:11:50,265 CNLB_LOGGER:INFO: nslookup of testrecord.pranavs,  
result: Server: 169.254.25.10  
Address: 169.254.25.10#53  
  
Name: testrecord.abc  
Address: 10.4.1.8  
Name: testrecord.abc  
Address: abbb:abbb::1  
2025-07-24 09:11:50,265 CNLB_LOGGER:INFO: Deleting the temporary test pod
```

Example 2:

```
./installer/dns_enhancement/coreDnsEnhancement.py -t dns1.abc
```

Sample output:

```
2025-07-24 09:13:45,955 CNLB_LOGGER:INFO: Update configurations in
validationPod.yaml and kubectl apply it.
2025-07-24 09:13:47,128 CNLB_LOGGER:INFO: waiting for dnsvalidation pod to
be up
2025-07-24 09:13:47,488 CNLB_LOGGER:INFO: nslookup of dns1.abc, result:
Server:          169.254.25.10
Address:         169.254.25.10#53

Name:   dns1.abc
Address: 10.4.1.1
2025-07-24 09:13:47,488 CNLB_LOGGER:INFO: Deleting the temporary test pod
```

This test checks DNS resolution for the specified zone name and provides insight into the functionality of the DNS enhancement. Additionally, the `-nn` option can be used to specify a particular node to test DNS connectivity, allowing for more targeted testing. For example:

```
./installer/dns_enhancement/coreDnsEnhancement.py -t <zonenumber> -nn
<node_name>
```

8.12.5 Verifying Packet Flow Through the CNLB Interface

This section explains about verifying packet flow through the CNLB interface to monitor network traffic in cloud-native environments.

1. Run the following command to get the coredns pod IPs (nf-oam-egr2):

```
kubectl -n kube-system describe po -l k8s-app=kube-dns | grep -A3 egr2

      "name": "default/nf-oam-egr2",
      "interface": "nf-oam-egr2",
      "ips": [
        "132.16.0.142"
      ],
    --
      k8s.v1.cni.cncf.io/networks: default/nf-oam-egr2@nf-
oam-egr2
      kubectl.kubernetes.io/restartedAt:
2025-04-03T09:10:41Z
Status: Running
SeccompProfile: RuntimeDefault
    --
      "name": "default/nf-oam-egr2",
      "interface": "nf-oam-egr2",
      "ips": [
        "132.16.0.139"
      ],
    --
      k8s.v1.cni.cncf.io/networks: default/nf-oam-egr2@nf-
oam-egr2
      kubectl.kubernetes.io/restartedAt:
2025-04-03T09:10:41Z
```

```
Status:           Running
SeccompProfile:   RuntimeDefault
```

In the above sample output, the coredns pod IPs are 132.16.0.142 and 132.16.0.139.

2. Run the following command to get the gateway IP from the net-attach-definition configuration:

```
kubectl describe net-attach-def nf-oam-egr2
```

Sample output:

```
Name:           nf-oam-egr2
Namespace:      default
Labels:         <none>
Annotations:    <none>
API Version:    k8s.cni.cncf.io/v1
Kind:           NetworkAttachmentDefinition
Metadata:
  Creation Timestamp:  2025-04-01T08:39:54Z
  Generation:         2
  Resource Version:    985125
  UID:                75c9c2ef-141b-49b7-929e-93fb16cc2d67
Spec:
  Config:  {"cniVersion": "0.4.0", "name": "nf-oam-egr2", "plugins":
[{"type": "macvlan", "mode": "bridge", "master": "eth2", "ipam": {"type":
"whereabouts", "range": "132.16.0.0/24", "range_start": "132.16.0.129",
"range_end": "132.16.0.190", "gateway": "132.16.0.194", "routes": [{"dst":
"10.75.201.0/24", "gw": "132.16.0.194"}]}]}]}
Events:    <none>
```

In the above sample output, the gateway IP is 132.16.0.194.

3. Run the following command to get the pods IP using the gateway IP:

```
$ kubectl -n occne-infra exec -it $(kubectl -n occne-infra get po --no-headers -l app=cnlb-manager -o custom-columns=:metadata.name) -- curl http://localhost:5001/net-info | python -m json.tool | jq
```

Sample output:

```
{
  "10.233.68.72": [
    {
      "egressIpExt": "10.75.200.91",
      "gatewayIp": "132.16.0.193",
      "networkName": "oam"
    }
  ],
  "10.233.68.73": [
    {
      "egressIpExt": "10.75.200.12",
      "gatewayIp": "132.16.0.194",
```

```

        "networkName": "oam"
    }
  ]
}

```

From the above sample output the pod IP is 10.233.68.73.

4. Run the following command to retrieve the pod name using the pod IP:

```
$ kubectl -n occne-infra get po -l app=cnlb-app -o wide | grep <active
cnlb app ip>
```

Sample output:

```

cnlb-app-fd566bffb-mhmg2    1/1      Running    0          145m
10.233.68.73
      occne3-prince-p-pranav-k8s-node-4    <none>

```

5. Get into the active pod to monitor the traffic.
The pod whose name matches the IP address is the current active one, where traffic can be monitored. To do so, run the following command in the same pod and monitor the traffic while performing the queries.

```
kubectl -n occne-infra exec -it <POD_NAME> -- bash
```

6. Run the following command to capture the packets:

```
bash-5.1# tcpdump -i lb-oam-int port 53 and udp -n -vv
```

Sample output:

```

dropped privs to tcpdump
tcpdump: listening on lb-oam-int, link-type EN10MB (Ethernet), snapshot
length 262144 bytes09:13:49.476867 IP (tos 0x0, ttl 64, id 60831, offset
0, flags [DF], proto UDP (17), length 75)
    132.16.0.142.58822 > 10.75.201.36.domain: [udp sum ok] 50867+ [1au] A?
testrecord.pranavs. ar: . OPT UDPsize=2048 DO (47)
09:13:49.479133 IP (tos 0x0, ttl 61, id 19793, offset 0, flags [none],
proto UDP (17), length 91)
    10.75.201.36.domain > 132.16.0.142.58822: [udp sum ok] 50867* q: A?
testrecord.pranavs. 1/0/1 testrecord.pranavs. A 10.4.1.8 ar: . OPT
UDPsize=1232 DO (63)
09:13:49.480514 IP (tos 0x0, ttl 64, id 60832, offset 0, flags [DF], proto
UDP (17), length 75)
    132.16.0.142.58822 > 10.75.201.36.domain: [udp sum ok] 676+ [1au]
AAAA? testrecord.pranavs. ar: . OPT UDPsize=2048 DO (47)
09:13:49.481049 IP (tos 0x0, ttl 61, id 19794, offset 0, flags [none],
proto UDP (17), length 103)
    10.75.201.36.domain > 132.16.0.142.58822: [udp sum ok] 676* q: AAAA?
testrecord.pranavs. 1/0/1 testrecord.pranavs. AAAA abbb:abbb::1 ar: . OPT
UDPsize=1232 DO (75)
09:13:49.701706 IP (tos 0x0, ttl 64, id 60833, offset 0, flags [DF], proto
UDP (17), length 73)
    132.16.0.142.58822 > 10.75.201.36.domain: [udp sum ok] 65302+ [1au] A?

```

```

ip6test.pranavs. ar: . OPT UDPsize=2048 DO (45)
09:13:49.702754 IP (tos 0x0, ttl 61, id 19871, offset 0, flags [none],
proto UDP (17), length 125)
    10.75.201.36.domain > 132.16.0.142.58822: [udp sum ok] 65302* q: A?
ip6test.pranavs. 0/1/1 ns: pranavs. SOA dns1.pranavs. hostmaster.pranavs.
3001 21600 3600 604800 86400 ar: . OPT UDPsize=1232 DO (97)
09:13:49.704475 IP (tos 0x0, ttl 64, id 60834, offset 0, flags [DF], proto
UDP (17), length 73)
    132.16.0.142.58822 > 10.75.201.36.domain: [udp sum ok] 63895+ [1au]
AAAA? ip6test.pranavs. ar: . OPT UDPsize=2048 DO (45)
09:13:49.704855 IP (tos 0x0, ttl 61, id 19873, offset 0, flags [none],
proto UDP (17), length 101)
    10.75.201.36.domain > 132.16.0.142.58822: [udp sum ok] 63895* q: AAAA?
ip6test.pranavs. 1/0/1 ip6test.pranavs. AAAA abbb:abbb::2 ar: . OPT
UDPsize=1232 DO (73)
09:13:49.825753 IP (tos 0x0, ttl 64, id 60835, offset 0, flags [DF], proto
UDP (17), length 70)
    132.16.0.142.58822 > 10.75.201.36.domain: [udp sum ok] 5680+ [1au] A?
alias.pranavs. ar: . OPT UDPsize=2048 DO (42)
09:13:49.826438 IP (tos 0x0, ttl 61, id 19888, offset 0, flags [none],
proto UDP (17), length 111)
    10.75.201.36.domain > 132.16.0.142.58822: [udp sum ok] 5680* q: A?
alias.pranavs. 2/0/1 alias.pranavs. CNAME testrecord.pranavs.,
testrecord.pranavs. A 10.4.1.8 ar: . OPT UDPsize=1232 DO (83)
09:13:49.942999 IP (tos 0x0, ttl 64, id 60836, offset 0, flags [DF], proto
UDP (17), length 64)
    132.16.0.142.58822 > 10.75.201.36.domain: [udp sum ok] 34442+ [1au]
MX? pranavs. ar: . OPT UDPsize=2048 DO (36)
09:13:49.943851 IP (tos 0x0, ttl 61, id 19911, offset 0, flags [none],
proto UDP (17), length 101)
    10.75.201.36.domain > 132.16.0.142.58822: [udp sum ok] 34442* q: MX?
pranavs. 1/0/2 pranavs. MX mail.pranavs. 10 ar: mail.pranavs. A
10.4.1.15, . OPT UDPsize=1232 DO (73)
09:13:50.067124 IP (tos 0x0, ttl 64, id 60837, offset 0, flags [DF], proto
UDP (17), length 64)
    132.16.0.142.58822 > 10.75.201.36.domain: [udp sum ok] 30499+ [1au]
TXT? pranavs. ar: . OPT UDPsize=2048 DO (36)
09:13:50.067926 IP (tos 0x0, ttl 61, id 20007, offset 0, flags [none],
proto UDP (17), length 104)
    10.75.201.36.domain > 132.16.0.142.58822: [udp sum ok] 30499* q: TXT?
pranavs. 1/0/1 pranavs. TXT "v=spf1 ip4:10.4.1.0/24 -all" ar: . OPT
UDPsize=1232 DO (76)
^C
14 packets captured
14 packets received by filter
0 packets dropped by kernel

```

Note

To capture the above packets, DNS server's external IP must be in the range of **egress_dest = ["10.*.*/*25"]**, defined in the `cnlb.ini` file. If it doesn't, change the `egress_dest` with the required subnet range and run `updNtework.py`. Delete (or restart) each `coredns` pod manually so that the new annotation can take effect. The `tcpdump` command does not process any traffic until pods are restarted.

8.12.6 Rolling back coredns and nodelocaldns deployment and configmap

This section explains how to rollback CoreDNS deployment and nodelocaldns deployment, as well as any associated ConfigMap or zone files that were modified or added. This ensures that the DNS resolution returns to its original state before the changes were made.

1. Run the following command to change the directories to the cluster directory:

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

2. The following values must be modified under the [ocne:vars] section in the ocne.ini file for vCNE deployments and in the hosts.ini file for BareMetal deployments:

```
[ocne:vars]
enable_dns_enhancement = False
#upstream_dns_servers = ["10.**.**.*"]
#coredns_egress_nad = ["default/egr-nad-name@egr-nad-name"]
#coredns_external_zones = [{"cache": 30, "zones":["sample1"], "nameservers":
["10.**.**.*"]}]
```

3. After commenting the above variables in ocne.ini file, run the script coreDnsEnhancement.py to trigger cleanup:

```
./installer/dns_enhancement/coreDnsEnhancement.py
```

Sample output:

```
2025-07-21 13:11:28,230 CNLB_LOGGER:INFO: step-1: Load ini file into file
parser
2025-07-21 13:11:28,231 CNLB_LOGGER:INFO: step-2: Check for
enable_dns_enhancement parameter
2025-07-21 13:11:28,232 CNLB_LOGGER:INFO: step-3: Running cleanup since
the enable_dns_enhancement variable is removed/made as False
2025-07-21 13:11:28,682 CNLB_LOGGER:INFO: Ran ./tests/setupZones.sh -c in
cloud-user@10.75.200.57 server
2025-07-21 13:11:28,682 CNLB_LOGGER:INFO: running k8s pipeline to update
coredns and nodelocaldns
```

8.12.7 Updating DNS Enhancement

If the DNS enhancement setup script has already been run and you need to run it again, either without changes or after modifying parameters such as nameservers, cache values, zone names, or other configuration variables in the ocne.ini file for vCNE or the hosts.ini file for BareMetal, delete the external_zones.yaml file generated under the dns_enhancement directory before running the script again. This step ensures that any updated values are properly applied and prevents stale or conflicting configuration from previous runs.

```
[cloud-user@cluster-name-bastion-1 cluster-name]$ cd /var/ocne/cluster/$
{OCCNE_CLUSTER}/installer/dns_enhancement
[cloud-user@cluster-name-bastion-1 dns_enhancement]$ ls
backUp  coreDnsEnhancement.py  external_zones.yaml  validationPod.yaml
```

```
[cloud-user@cluster-name dns_enhancement]$ rm external_zones.yaml

#to fetch new changes in dns enhancement
[cloud-user@cluster-name dns_enhancement]$ ./coreDnsEnhancement.py
```

8.12.8 Validating DNS Enhancement

To ensure the proper functioning of DNS Enhancement, it is necessary to check the following:

1. Ensure CoreDNS pods are in Running State
2. Verify CoreDNS ConfigMap
3. Ensure NodeLocalDNS pods are in Running State

Perform the following steps to validate DNS enhancement:

1. Ensure CoreDNS pods are in the running state. CoreDNS is an essential part of DNS enhancement therefore, CoreDNS pods must be healthy.
Run the following command to check the status of the CoreDNS pods:

```
$ kubectl -n kube-system get pod -l k8s-app=kube-dns
```

Sample output:

NAME	READY	STATUS	RESTARTS	AGE
coredns-5965687c46-4hjdk	1/1	Running	0	13h
coredns-5965687c46-8q9d4	1/1	Running	0	13h

2. Review the CoreDNS pods logs. See ALL logs from both CoreDNS pods.

```
$ 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-5nrvv -----
[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 =
2bc9e13e66182e6e829fe1a954359de92746468f433b8748589dfe16e1afd0e790e1ff75415
ad40ad17711abfc7a8348fdda2770af99962db01247526afbe24a
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 =
2bc9e13e66182e6e829fe1a954359de92746468f433b8748589dfe16e1afd0e790e1ff75415
ad40ad17711abfc7a8348fdda2770af99962db01247526afbe24a
CoreDNS-1.9.3
```

```
linux/amd64, go1.18.2, 45b0a11
```

```
### TIP ###
```

```
# Additionally, the above command can be piped to a file, for better readability and sharing purposes.
```

```
$ 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
```

3. Run the following command to verify the changes implemented by DNS Enhancement that can be viewed in the CoreDNS ConfigMap:

```
$ kubectl -n kube-system get cm coredns -o yaml
```

Sample output:

```
apiVersion: v1
data:
  Corefile: |
    example.com {
      log
      errors {
      }
      forward . tls://10.95.18.61:853 {
        tls /etc/ssl/example-com/tls.crt /etc/ssl/example-com/
        tls.key /etc/ssl/example-com/ca.crt
        tls_servername named.local
      }
      loadbalance
      cache 5
      reload
    }
    test.com {
      log
      errors {
      }
      forward . tls://10.12.15.16 {
        tls /etc/ssl/test-com/tls.crt /etc/ssl/test-com/tls.key /etc/ssl/
        test-com/ca.crt
        tls_servername named.local
      }
      loadbalance
      cache 5
      reload
    }
    .:53 {
      errors {
      }
      health {
        lameduck 5s
      }
      ready
      kubernetes occne-example in-addr.arpa ip6.arpa {
        pods insecure
```

```

        fallthrough in-addr.arpa ip6.arpa
    }
    prometheus :9153
    forward . 10.75.144.85 {
        prefer_udp
        max_concurrent 1000
    }
    cache 30

    loop
    reload
    loadbalance
}
kind: ConfigMap
metadata:
  annotations:
    kubectl.kubernetes.io/last-applied-configuration: |
      {"apiVersion":"v1","data":{"Corefile":"example.com {\n    log\n
errors {\n    }\n    forward . tls://10.75.180.161:853 {\n
tls /etc/ssl/example-com/tls.crt /etc/ssl/example-com/tls.key /etc/ssl/
example-com/ca.crt\n    tls_servername named.local\n    }\n
loadbalance\n    cache 5\n    reload\n}\ntest.com {\n    log\n    errors
{\n    }\n    forward . tls://10.75.180.161 {\n    tls /etc/ssl/test-com/
tls.crt /etc/ssl/test-com/tls.key /etc/ssl/test-com/ca.crt\n
tls_servername named.local\n    }\n    loadbalance\n    cache 5\n
reload\n}\n.:53 {\n    errors {\n    }\n    health {\n    lameduck
5s\n    }\n    ready\n    kubernetes ocne-example in-addr.arpa ip6.arpa
{\n    pods insecure\n    fallthrough in-addr.arpa ip6.arpa\n    }
\n    prometheus :9153\n    forward . 10.75.144.85 {\n
prefer_udp\n    max_concurrent 1000\n    }\n    cache 30\n\n
loop\n    reload\n    loadbalance\n}\n"},"kind":"ConfigMap","metadata":
{"annotations":{},"labels":{"addonmanager.kubernetes.io/
mode":"EnsureExists"},"name":"coredns","namespace":"kube-system"}}
    creationTimestamp: "2025-07-31T20:51:39Z"
  labels:
    addonmanager.kubernetes.io/mode: EnsureExists
  name: coredns
  namespace: kube-system
  resourceVersion: "1654250"
  uid: c5a4a0b6-3795-43ae-b0c2-1ec52212d0f5

```

4. Run the following command to ensure that NodeLocalDNS pods are in running state:

```
$ kubectl -n kube-system get pod -l k8s-app=node-local-dns
```

Sample output:

NAME	READY	STATUS	RESTARTS	AGE
nodelocaldns-65657	1/1	Running	0	14h
nodelocaldns-6hzn6	1/1	Running	0	14h
nodelocaldns-8lrd7	1/1	Running	0	14h
nodelocaldns-jdxct	1/1	Running	0	14h
nodelocaldns-ktjsx	1/1	Running	0	14h
nodelocaldns-qfvjx	1/1	Running	0	14h
nodelocaldns-xcn7j	1/1	Running	0	14h

8.13 Enabling Public Endpoint for CNE (Kubernetes) API Server

This section provides an overview of the implementation for exposing the Kubernetes API Server as a public endpoint.

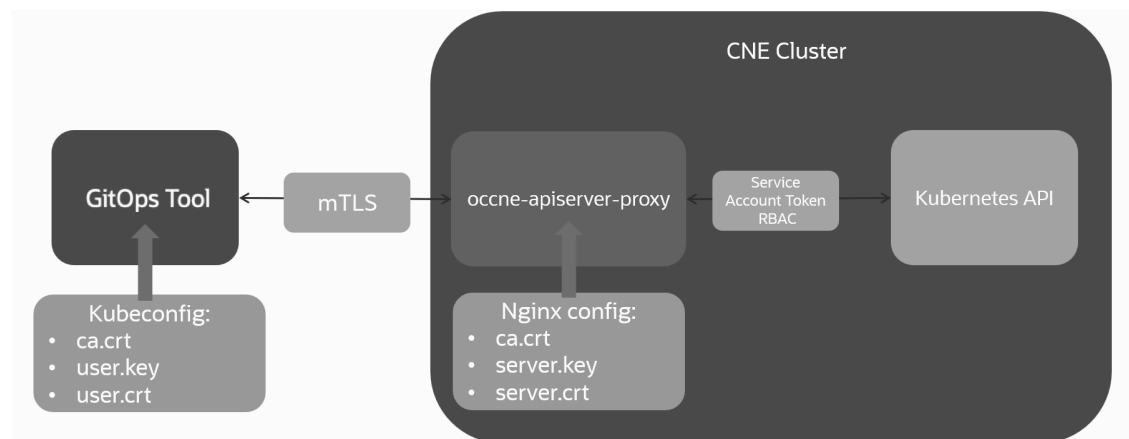
The *occne-apiserver-proxy* service is designed to enable GitOps integration by exposing the Kubernetes API Server to an external network. This is achieved by deploying an nginx pod configured as a reverse proxy with mTLS encryption. The nginx proxy securely redirects incoming external traffic to the Kubernetes API Server, while CNLB (Cloud Native Load Balancer) manages ingress to the Kubernetes control plane.

Note

After enabling *occne-apiserver-proxy*, any client certificate signed by the same CA that signed the server certificate will be accepted and will grant full administrative access to the CNE cluster.

Users accessing the public endpoint are required to authenticate using mTLS. In the current release, the nginx proxy uses a service account token with cluster admin-level privileges.

Figure 8-2 CNE Cluster



Note

Ensure that all certificate files are generated and signed by the same Certificate Authority (CA).

Prerequisites

To enable the mTLS communication for the *occne-apiserver-proxy* service, the following three files are required:

- CA certificate
- Server key

- Server certificate

The CA certificate, server key, and server certificate must be generated and signed by the same Certificate Authority (CA) that is used to create the user certificates for communication with the `occne-apiserver-proxy`.

Examples on how to create keys and certificates

The following are examples about how to create keys and certificates but can vary according to the CA and the algorithm that will be used to create them.

Example 1: Create a nginx server's key and certificate

Following is an example about how to create a service key and certificate using a Certificate Authority (CA) key.

```
$ cat server.conf
```

Sample output:

```
[ req ]
default_bits = 2048
distinguished_name = req_distinguished_name
req_extensions = req_ext
prompt = no

[ req_distinguished_name ]
C = US
ST = NC
L = Morrisville
O = Oracle
CN = occne6-cluster-apiserver-proxy # This should match the name that will be
resolved to the occne-apiserver-proxy in the DNS servers

[ req_ext ]
subjectAltName = @alt_names
keyUsage = critical, digitalSignature, keyEncipherment
extendedKeyUsage = serverAuth
basicConstraints = critical, CA:FALSE

[ alt_names ]
DNS.1 = lb-apiserver.kubernetes.local # DNS entry
DNS.2 = OCCNE_CLUSTER-apiserver-proxy
IP.1 = 10.75.200.111 # The actual IP for the occne-apiserver-proxy
```

Creating nginx key and certificate:

```
$ openssl genrsa -out server.key 4096
$ openssl req -new -key server.key -out server.csr -config server.conf
$ openssl x509 -req -in server.csr -CA ca.crt -CAkey ca_private.key -
CAcreateserial -out server.crt -days 365 -extensions req_ext -extfile
server.conf -passin pass:${CA_KEY_PASS}
```

Example 2: Create a user's key and certificate

Following is an example about how to create an user key and certificate using a Certificate Authority (CA) key:

```
$ cat client.conf
```

Sample output:

```
[ req ]
default_bits = 2048
distinguished_name = req_distinguished_name
req_extensions = req_ext
prompt = no[ req_distinguished_name ]
C = US
ST = NC
L = Morrisville
O = Oracle
CN = client.OCCNE_CLUSTER-apiserver-proxy

[ req_ext ]
keyUsage = critical, digitalSignature, keyEncipherment
extendedKeyUsage = clientAuth
basicConstraints = critical, CA:FALSE
```

Creating nginx key and certificate:

```
$ openssl genrsa -out client.key 4096
$ openssl req -new -key client.key -out client.csr -config client.conf
$ openssl x509 -req -in client.csr -CA ca.crt -CAkey ca_private.key -
CAcreateserial -out client.crt -days 365 -extensions req_ext -extfile
client.conf -passin pass:${CA_KEY_PASS}
```

Limitations

① Note

Once occne-apiserver-proxy is enabled, any client certificate signed by the same CA used to sign the server certificate will be trusted and will grant full administrative access to the CNE cluster.

1. CNE will not be in charge of creating or maintaining the certificates that will be used to connect to the occne-apiserver-proxy, instead, these certificates and keys will be provided, maintained and rotated by the OCCNE CLUSTER ADMINISTRATOR.
2. For Kubernetes API exposure, the nginx proxy uses a service account token which has cluster administrator equivalent privileges. Treat these certificates and access credentials with extreme caution as they grant full administrative control over the Kubernetes cluster.

8.13.1 Enabling occne-apiserver-proxy Service

Perform the following steps to enable occne-apiserver-proxy service:

1. **Provide required certificates**

- a. Create a directory to store the certificates. Run the following command on the bootstrap host to enable the service during installation, or on the Bastion host to enable it during an upgrade.

Create certificates directory

```
mkdir -m 700 -p /var/ocne/certificates/ocne-apiserver-proxy
```

- b. Place the certificates into the bootstrap or Bastion host. Run the following command to copy nginx server's key and certificate:

```
$ scp <user>@<system>:/<path_to_certificates>/ca.crt <user>@<bootstrap/  
bastion_ip>:/var/ocne/certificates/ocne-apiserver-proxy/  
$ scp <user>@<system>:/<path_to_certificates>/server.key  
<user>@<bootstrap/bastion_ip>:/var/ocne/certificates/ocne-apiserver-  
proxy/  
$ scp <user>@<system>:/<path_to_certificates>/server.crt  
<user>@<bootstrap/bastion_ip>:/var/ocne/certificates/ocne-apiserver-  
proxy/
```

2. Configure ocne-apiserver-proxy service

- a. Edit the ocne.ini file to enable the ocne-apiserver-proxy and specify the IP address that will be used for access.

```
$ vi /var/ocne/cluster/${OCCNE_CLUSTER}/ocne.ini
```

Sample output:

```
... TRUNCATED ...  
ocne_apiserver_cnlb = 10.75.200.111  
  
... TRUNCATED ...  
  
# Specify 'True' (case matters, no quotes) to expose ocne apiserver  
proxy publicly. Default is 'False'  
ocne_apiserver_proxy_enabled = True  
... TRUNCATED ...
```

- b. Ensure that the IP for the service is also defined in cnlb.ini file. Run the following command to enable the service as a standalone procedure:

```
$ vi /var/ocne/cluster/${OCCNE_CLUSTER}/cnlb.ini
```

Sample output:

```
... TRUNCATED ...  
[oam:vars]  
...  
service_ip_addr = [<IPs for other services>,"10.75.200.111"]  
... TRUNCATED ...
```

3. Enable ocne-apiserver-proxy

- a. **Enable ocne-apiserver-proxy during CNE cluster installation and upgrades:**

To enable the `occne-apiserver-proxy` service during CNE installation or upgrade, proceed with the standard installation or upgrade procedure.

b. Enable `occne-apiserver-proxy` as a standalone procedure:

To enable the service, run the following steps:

- i. If you added a new IP to the `service_ip_addr` variable in `cnlb.ini`, run the network upgrade script to create a new `NetworkAttachmentDefinition` for the service. If the IP used for the `occne-apiserver-proxy` already exists, skip this step.

```
cd /var/occne/cluster/${OCCNE_CLUSTER}/
./installer/updNetwork.py
```

- ii. Run the `pipeline.sh` script for installing the `occne-apiserver-proxy` deployment and its dependencies:

```
OCCNE_STAGES=(DEPLOY) OCCNE_CONTAINERS=(CFG) OCCNE_ARGS="--
tags=apiserver-proxy" pipeline.sh
```

8.13.2 Using `occne-apiserver-proxy`

To utilize the **`occne-apiserver-proxy`** service, you'll need to gather some essential information:

1. Retrieve `occne-apiserver-proxy` External Details:

On the cluster where the Kubernetes API is exposed, perform the following steps:

a. Obtain the Service's External IP

Retrieve the external IP address for **`occne-apiserver-proxy`**:

```
kubectl -n occne-infra get pod -l app.kubernetes.io/name=apiserver-
proxy -o jsonpath="{.items[].metadata.annotations['oracle\.com\.cnc\
/cnlb']}" | jq -r '.[].cnlbIp'
```

b. Encode Keys and Certificates

Encode the required certificate files in base64:

```
export ca_cert=$(cat ca.crt | base64 -w 0)
export client_cert=$(cat client.crt | base64 -w 0)
export client_key=$(cat client.key | base64 -w 0)
```

2. Create kubeconfig file

On the external or secondary Kubernetes cluster:

a. Create a Kubeconfig to connect into the exposed API cluster

Render the kubeconfig file with the previously obtained values:

```
apiVersion: v1
clusters:
- cluster:
  certificate-authority-data: <Value for $ca_cert>
  server: https://<Value for occne-apiserver-proxy IP>:6443
  name: <Value for $CLUSTER_NAME>
contexts:
- context:
  cluster: <Value for $CLUSTER_NAME>
  user: flux-user-<Value for $CLUSTER_NAME>
  name: flux-user-<Value for $CLUSTER_NAME>@<Value for $CLUSTER_NAME>
```

```

current-context: flux-user-<Value for $CLUSTER_NAME>@<Value
for $CLUSTER_NAME>
kind: Config
preferences: {}
users:
- name: flux-user-<Value for $CLUSTER_NAME>
  user:
    client-certificate-data: <Value for $client_cert>
    client-key-data: <Value for $client_key>

```

b. Connect to the exposed API cluster

Use the kubeconfig file that was created in the previous step to connect to the Kubernetes cluster using the occne-apiserver-proxy:

```
kubectl --kubeconfig ~/external_kubeconfig.conf get nodes
```

Sample output:

NAME	STATUS	ROLES	AGE	VERSION
occne-cluster-k8s-ctrl-1	Ready	control-plane	136m	v1.31.1
occne-cluster-k8s-ctrl-2	Ready	control-plane	136m	v1.31.1
occne-cluster-k8s-ctrl-3	Ready	control-plane	136m	v1.31.1
occne-cluster-k8s-node-1	Ready	<none>	135m	v1.31.1
occne-cluster-k8s-node-2	Ready	<none>	135m	v1.31.1
occne-cluster-k8s-node-3	Ready	<none>	135m	v1.31.1
occne-cluster-k8s-node-4	Ready	<none>	135m	v1.31.1

8.13.3 Customizing the Nginx Configuration

This section describes how to update the configmap that manages **nginx.conf** for occne-apiserver-proxy. This procedure is optional.

1. Retrieve the Current Nginx Configuration

The `nginx.conf` file is stored in the `occne-apiserver-proxy-nginx-config` configmap. To extract it, run the following command:

```
kubectl -n occne-infra get cm occne-apiserver-proxy-nginx-config -o json |
jq -r '.data."nginx.conf"' > nginx.conf
```

2. Edit the nginx.conf

Modify the `nginx.conf` file as needed, ensuring all changes use valid Nginx parameters (invalid configurations will cause the pod to fail):

```
vi nginx.conf
```

3. Apply the Updated Configuration

Delete and re-create the configmap with your modified `nginx.conf` file, then restart the deployment. Perform the following steps:

a. Run the following command to delete the configmap:

```
kubectl -n occne-infra delete cm occne-apiserver-proxy-nginx-config
```

Sample output:

```
configmap "occne-apiserver-proxy-nginx-config" deleted
```

- b. Run the following command to create the configmap:

```
kubectl -n occne-infra create cm occne-apiserver-proxy-nginx-config--  
from-file=data=nginx.conf
```

Sample output:

```
configmap/occne-apiserver-proxy-nginx-config created
```

- c. Run the following command to restart the API server:

```
kubectl -n occne-infra rollout restart deployment occne-apiserver-proxy
```

4. Verify the Update

Check the status of the pods to confirm that the update was successful:

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

Sample output:

```
occne-apiserver-proxy-7c55658548-ps469 1/1 Running 0 19s
```

8.14 Grafana Dashboards

The CNE Grafana dashboards combine CNE custom node metrics with Kubernetes default health or utilization views to monitor cluster readiness, capacity, and workload distribution.

8.14.1 Custom Metrics

CNE Metrics component is part of CNE common services. These metrics are derived from cluster node data and are not specific to CNLB.

CNE Metrics collects information for each node in the cluster, including details about every vCPU and each network interface. The component has its own Podman image, Helm chart, and Grafana dashboard panels.

Grafana CNE Metrics Dashboard Panels

A series of panels are available on the CNE Kubernetes Cluster Dashboard that display useful CNE Metrics data custom Metrics Panel.

CNE Metrics Variables

Table 8-30 CNE Metrics Variables

Variable Name	Description
Host Selector	This variable helps select which cluster node or host to monitor by displaying metric values for the selected node. The node or host value can be changed by the user from the top of the dashboard.
Bottom N Idle%	This variable helps identify the vCPUs with the lowest idle percentage by displaying the bottom N vCPUs with the least Idle%. The value of N can be changed by the user.
Top N Soft% IRQ	This variable helps identify the vCPUs with the highest Soft IRQ percentage by displaying the top N vCPUs with the most Soft% IRQ. The value of N can be changed by the user.

Mpstat Panels

The metrics displayed in the following panels are obtained by running the `mpstat` command on the cluster worker nodes:

Note

`mpstat` (multiprocessor statistics) is a Linux command that reports CPU activity for each processor or vCPU. In the CNE Grafana dashboard, it runs on cluster worker nodes to collect per-vCPU %idle and %soft (software interrupt) metrics.

- **Idle Percentage Average:** This metric captures %idle every second and stores the values in a list. During the same run, the gauge value is updated with the average of the collected values. The list is cleared after Prometheus scrapes the gauges, while preserving the last value. The %idle values are displayed for each vCPU of the selected host.
- **Bottom N vCPUs with Less Average Idle Percentage:** This panel helps identify the vCPUs with the lowest average idle percentage by displaying the bottom N vCPUs with the least average Idle%. The value of N is configured using the dashboard variable at the top.
- **Idle Percentage Low Watermark:** This metric captures %idle every second and stores the values in a list. During the same run, the gauge value is updated with the minimum value from the collected list. The list is cleared after Prometheus scrapes the gauges, while preserving the last value. The %idle values are displayed for each vCPU of the selected host.
- **Bottom N vCPUs with Less Low Watermark Idle Percentage :** This panel helps identify the vCPUs with the lowest low watermark idle percentage by displaying the bottom N vCPUs with the least low watermark Idle%. The value of N is configured using the dashboard variable at the top.
- **Soft IRQ Percentage Average:** This metric captures %soft every second and stores the values in a list. During the same run, the gauge value is updated with the average of the collected values. The list is cleared after Prometheus scrapes the gauges, while preserving the last value. The %soft values are displayed for each vCPU of the selected host.
- **Top N vCPUs with Most Average Soft IRQ Percentage :** This panel helps identify the vCPUs with the highest average Soft IRQ percentage by displaying the top N vCPUs with the most average Soft% IRQ. The value of N is configured using the dashboard variable at the top.

- **Soft IRQ Percentage High Watermark:** This metric captures %soft every two seconds and stores the values in a list. During the same run, the gauge value is updated with the maximum value from the collected list. The list is cleared after Prometheus scrapes the gauges, while preserving the last value. The %soft values are displayed for each vCPU of the selected host.
- **Top N vCPUs with Most High Watermark Soft IRQ Percentage:** This panel helps identify the vCPUs with the highest high watermark Soft IRQ percentage by displaying the top N vCPUs with the most high watermark Soft% IRQ. The value of N is configured using the dashboard variable at the top.

Network Panels

The metrics displayed in the following panels are obtained by reading data from the `/proc/net/dev` file on the cluster nodes:

- **Bytes Sent per Interface:** This metric shows the number of bytes transmitted per second for each network interface on the selected host. CNE reads the values in `/proc/net/dev`, waits one second, reads them again, and calculates the difference. The resulting values are displayed in bytes per second (BpS).
- **Bytes Received per Interface:** This metric shows the number of bytes received per second for each network interface on the selected host. CNE reads the values in `/proc/net/dev`, waits one second, reads them again, and calculates the difference. The resulting values are displayed in bytes per second (BpS).
- **Packets Sent per Interface:** This metric shows the number of packets transmitted per second for each network interface on the selected host. CNE reads the values in `/proc/net/dev`, waits one second, reads them again, and calculates the difference. The resulting values are displayed in packets per second (PpS).
- **Packets Received per Interface:** This metric shows the number of packets received per second for each network interface on the selected host. CNE reads the values in `/proc/net/dev`, waits one second, reads them again, and calculates the difference. The resulting values are displayed in packets per second (PpS).
- **Errors Sent per Interface:** This metric shows the number of transmission errors per second for each network interface on the selected host. CNE reads the values in `/proc/net/dev`, waits one second, reads them again, and calculates the difference. The resulting values are displayed in errors per second (EpS).
- **Errors Received per Interface:** This metric shows the number of receive errors per second for each network interface on the selected host. CNE reads the values in `/proc/net/dev`, waits one second, reads them again, and calculates the difference. The resulting values are displayed in errors per second (EpS).
- **Outgoing Gbps Bandwidth Usage:** This metric shows the total outgoing bandwidth usage, in gigabits per second (Gbps), for the selected host. It adds the transmitted bytes-per-second values from all network interfaces and converts the result to Gbps.
- **Incoming Gbps Bandwidth Usage:** This metric shows the total incoming bandwidth usage, in gigabits per second (Gbps), for the selected host. It adds the received bytes-per-second values from all network interfaces and converts the result to Gbps.

Grafana Kubernetes Default Dashboard Panels

These panels are part of the default Kubernetes dashboards used for cluster supervision.

Cluster Health and utilization panels

- **Kubernetes Version:** This panel displays the Kubernetes version running in the cluster, which helps verify cluster compatibility, confirm feature support, and identify when an upgrade may be required.

- **Nodes Ready/Nodes Not Ready:** This panel shows the number of cluster nodes that are currently in a Ready state and the number of nodes that are Not Ready. A healthy cluster is expected to have all worker and control plane nodes marked as Ready. Nodes in a Not Ready state may indicate issues with kubelet, network connectivity, or resource exhaustion.
- **Pods Running:** This panel displays the total number of active Pods that are scheduled and running across the cluster. It helps indicate current workload utilization, while a sudden drop in the number of running Pods may point to failures, evictions, or scheduling issues.
- **Cluster Max Pods Allowed:** This panel shows the maximum number of Pods that can be deployed in the cluster based on node capacity and kubelet configuration. It is useful for assessing cluster saturation and capacity planning by comparing the maximum allowed Pods with the current number of running Pods.
- **Cluster Memory Usage:** This panel monitors total memory consumption across all nodes in the cluster, including used memory, total memory, and overall usage percentage. Healthy utilization is typically shown in the green zone, while sustained high usage may indicate memory pressure that can affect Pod scheduling or lead to OOM kills.
- **Cluster CPU Usage:** This panel shows the overall CPU utilization across the entire cluster, including used CPU, total CPU, and overall usage percentage. Low CPU usage indicates available capacity for scaling, while sustained high CPU usage may indicate the need for additional nodes or performance tuning.
- **Cluster Disk Usage:** This panel tracks overall disk consumption across all nodes, including used disk, total disk, and overall usage percentage. Monitoring disk usage is important to ensure enough storage is available for Pods, logs, and container images. High disk usage may degrade node performance or cause scheduling issues.
- **Node Summary Table:** This table provides node-level details that help identify resource usage, capacity, and potential issues across the cluster.

Table 8-31 Node Summary Table

Column	Description
Instance	Displays the node IP address and kubelet port, which helps identify and troubleshoot specific nodes.
FQDN	Displays the fully qualified domain name of the node, helping map IP addresses to logical node identifiers.
Uptime	Displays how long each node has been running since the last reboot or restart. Consistent uptime across nodes indicates stability.
Cores	Displays the number of CPU cores available on each node.
Memory	Displays the total memory available on each node.
Disk Size	Displays the available disk capacity for each node.
CPU Usage	Displays a visual bar and percentage showing how much CPU each node is using. Values below approximately 80% are generally healthy, while spikes may indicate workload pressure or scheduling imbalance.

Table 8-31 (Cont.) Node Summary Table

Column	Description
Memory Usage	Displays a visual bar and percentage showing node memory consumption. High memory usage, typically around 80–90%, may indicate Pods with memory leaks or insufficient memory limits.
Disk Usage	Displays the percentage of disk space used on each node which helps to monitor node-level storage pressure.

Pods Overview

This section provides a detailed view of Pod distribution and resource usage across the Kubernetes cluster, helping administrators understand how workloads are distributed across nodes and identify possible imbalances or resource bottlenecks.

PODS on NODE

This panel displays the number of Pods currently running on each node in the cluster. It helps operators quickly review workload distribution and identify nodes that may be overloaded or underutilized.

Figure 8-3 PODS on NODE



Table 8-32 PODS on NODE

Column	Description
Node FQDN	Displays the fully qualified domain name of each node.
PodsRunning	Shows the number of Pods actively running on the node. Pod counts should generally be balanced across nodes to maintain even workload distribution. Large differences may indicate scheduling bias, resource constraints, or taints and tolerations affecting Pod placement.

POD Overview by Namespace

This panel displays Pod-level resource usage, including memory and CPU consumption, grouped by namespace. It helps operators monitor resource usage across workloads and identify Pods with high utilization.

Figure 8-4 Pod-level Resource Usage

namespace	pod	Memory Usage	CPU Usage
kube-system	calico-kube-controllers-57759d7bb7-nlr...	32.5 MB	0.0051 Cores
kube-system	calico-node-9rt4c	119 MB	0.0263 Cores
ocne-infra	cnlb-app-5d5fbc7f57-kcgv5	45.2 MB	0.0296 Cores
ocne-infra	cnlb-manager-74b786574-l2c2j	112 MB	0.5239 Cores
kube-system	coredns-659d77b4f8-5l1dv	24.5 MB	0.0019 Cores
kube-system	csi-cinder-controllerplugin-ff596cdf-jlw...	74.1 MB	0.0048 Cores
kube-system	csi-cinder-nodeplugin-wbbnq	26.9 MB	0.0006 Cores
kube-system	dns-autoscaler-76dbd8c669-cqzrw	10.8 MB	0.0006 Cores
kube-system	dynamic-networks-controller-ds-hljtn	10.1 MB	0.0028 Cores

Table 8-33 Pod-level Resource Usage

Column	Description
Namespace	Displays the logical grouping of Kubernetes resources to which the Pod belongs.
Pod	Displays the name of each Pod instance, helping identify the application or service associated with the resource usage.
Memory Usage	Shows the current memory consumption of each Pod, which is useful for detecting memory leaks or Pods exceeding expected memory limits.
CPU Usage	Shows the CPU usage of each Pod, expressed in cores, and helps identify CPU-bound workloads.

8.14.2 CNLB Dashboard

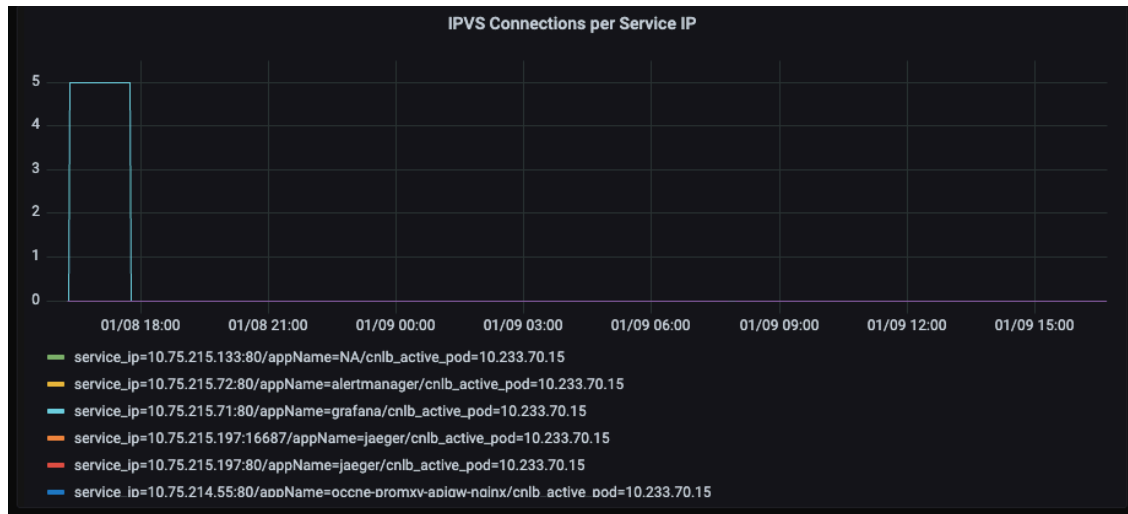
CNLB Dashboard in CNE is a common service that provides visibility into CNLB metrics across the cluster. It helps monitor the cnlb app pods and network status into the CNE observability stack.

CNLB Dashboard Panels Overview

IPVS Connections per Service IP

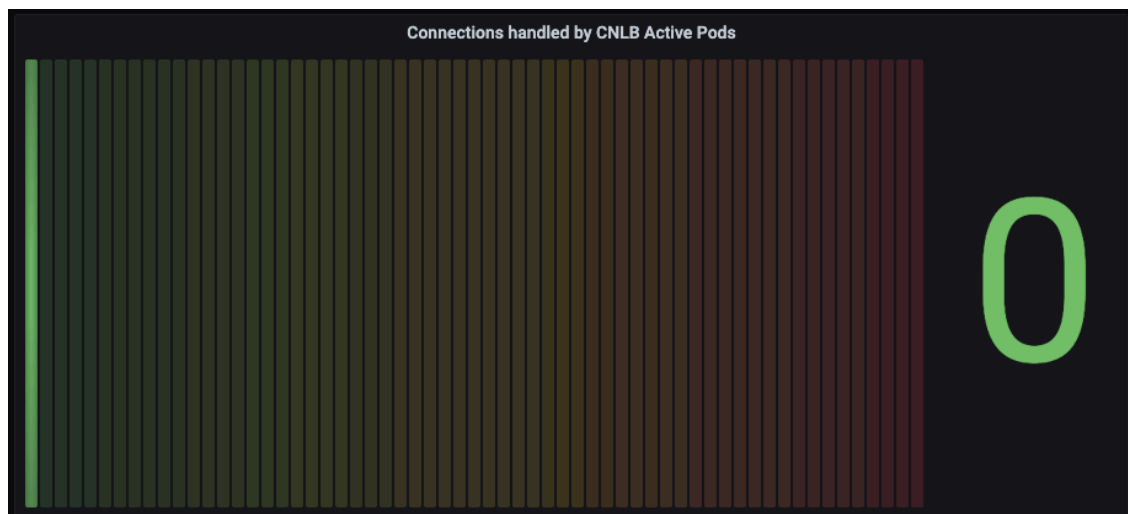
This helps monitor the number of connections established to each Service IP managed that expose the apps in the cluster. Users can track:

- Active connections per Service IP
- Historical connection trends

Figure 8-5 IPVS Connections per Service IP**Connections handled by CNLB Active Pods**

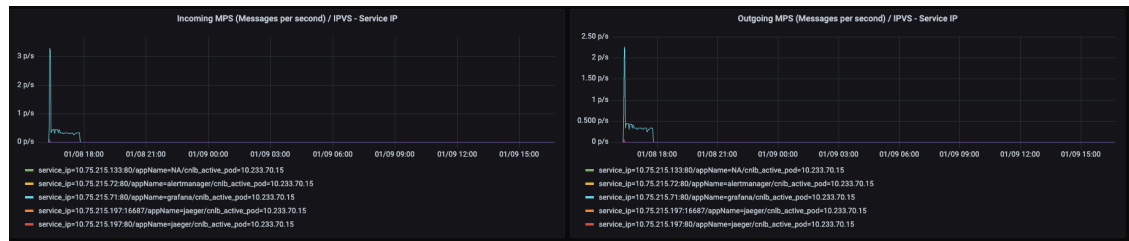
This helps to track how many live client connections are being managed currently by Cloud Native Load Balancer (CNLB) pods. Users can track:

- Number of connections per pod
- Total active connections across all active CNLB pods

Figure 8-6 Connections handled by CNLB Active Pods**Incoming or Outgoing MPS/IPVS - Service IP**

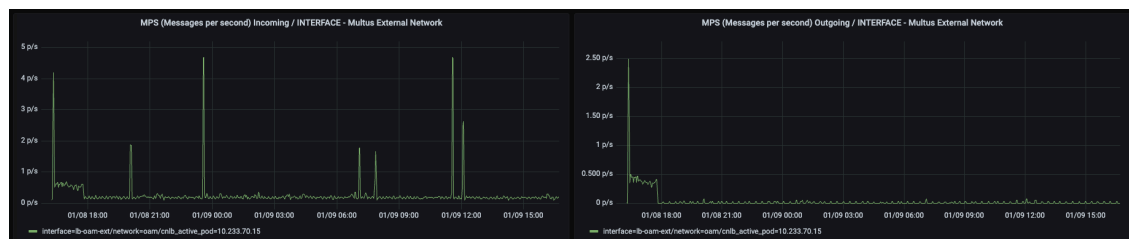
This helps to visualize the amount of messages handled by each service IP on a time chart. Users can track the Inbound/outbound MPS, per Service IP, over time.

Figure 8-7 Incoming/Outgoing MPS/IPVS - Service IP

**Incoming or Outgoing MPS/INTERFACE - Multus External Network**

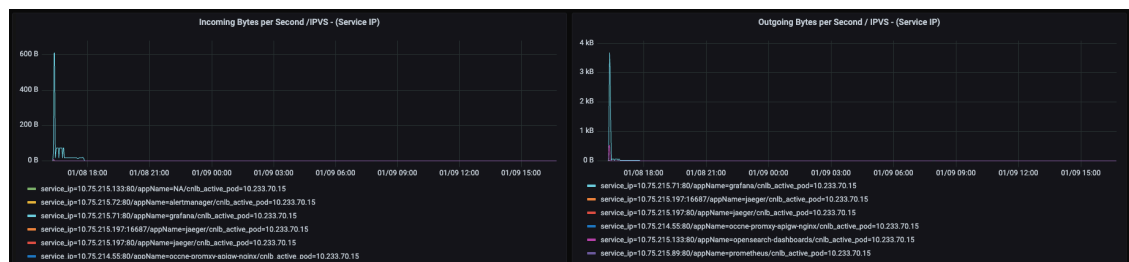
This displays the amount of messages handled by each multus external interface on a time chart. Users can track the Inbound or outbound MPS per external interface.

Figure 8-8 Incoming or Outgoing MPS or INTERFACE - Multus External Network

**Incoming or Outgoing Bytes per Second/IPVS - Service IP**

This helps to visualize the volume of bytes processed by each service IP over time using a time series chart. Users can track Incoming or Outgoing bytes per second, per Service IP, on a time chart.

Figure 8-9 Incoming or Outgoing Bytes per Second/IPVS

**Incoming or Outgoing MPS or INTERFACE - Multus Internal Network**

This helps to visualize the number of messages processed by each Multus internal interface over time using a time series chart. Users can track the Inbound/outbound MPS per internal interface.

Figure 8-10 Incoming or Outgoing MPS or INTERFACE - Multus Internal Network

8.14.3 Kyverno Dashboard

The Kyverno Dashboard in CNE is a common services that offers visibility into Kyverno policy metrics across the cluster. It helps track policy compliance, rule execution, and admission requests, all integrated with the CNE observability stack.

Grafana Kyverno Dashboard Panels Overview

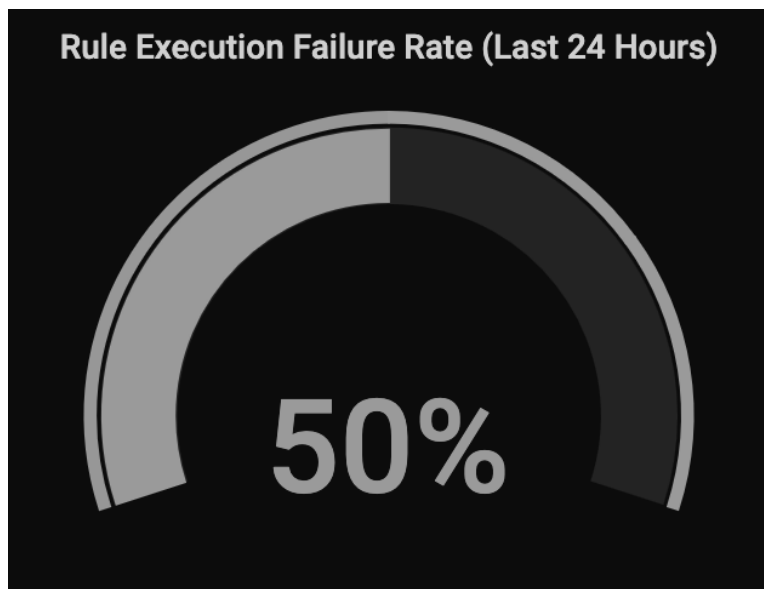
A series of panels have been added to the CNE Kyverno Policy Dashboard to display key Kyverno policy metrics, providing enhanced visibility and insight into policy enforcement and compliance across the cluster.

The following is an overview on CNE Kyverno section.

Latest Status

Rule Execution Failure Rate (Last 24 Hours)

This panel displays the percentage of Kyverno policy rule executions that have failed in the past 24 hours. This metric provides insight into the cluster's overall policy compliance status. An elevated failure rate may indicate possible misconfigurations or the presence of non-compliant resources.

Figure 8-11 Rule Execution Failure

Cluster Policies

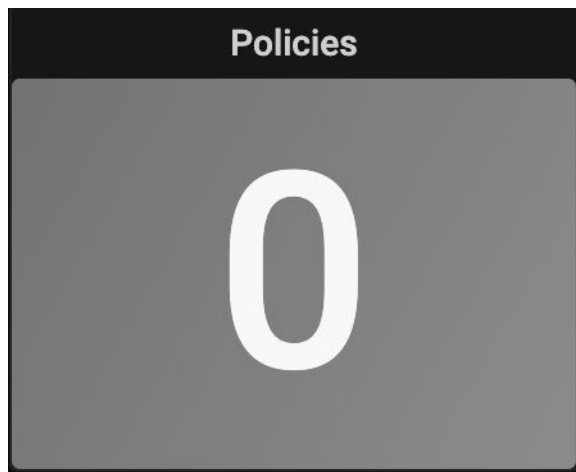
This panel displays the total number of active cluster-wide Kyverno policies applied throughout the Kubernetes environment. This metric helps users assess the overall scope of governance and security controls currently enforced at the cluster level.

Figure 8-12 Cluster Policies

**Policies**

This panel displays the total number of active Kyverno policies, encompassing both cluster-wide and namespaced policies. This provides a quick overview of how many governance and compliance rules are currently being enforced within the Kubernetes environment.

Figure 8-13 Policies

**Validate Rules**

This panel displays the total number of active Kyverno validation rules configured in the cluster. These rules verify that resource configurations adhere to defined compliance or security standards, helping users understand the breadth of validation checks enforced across the environment.

Figure 8-14 Validate Rules**Mutate Rules**

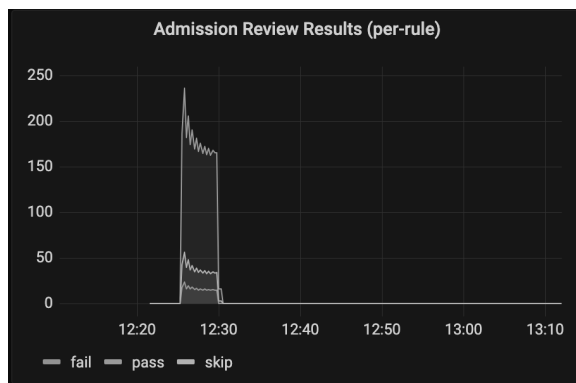
This panel displays the total number of active Kyverno mutation rules currently defined in the cluster. These rules automatically modify or add fields to Kubernetes resources during creation or updates, helping users track how many rules are actively ensuring consistent and compliant resource configurations.

Figure 8-15 Mutate Rules**Generate Rules**

This panel displays the total number of active Kyverno generation rules in the cluster. These rules automatically create or replicate Kubernetes resources according to predefined policies, helping users monitor how many automation rules are in place to ensure consistency and enforce desired configurations across namespaces.

Figure 8-16 Generate Rules**Policy-Rule Results (per rule)****Admission Review Results (per-rule)**

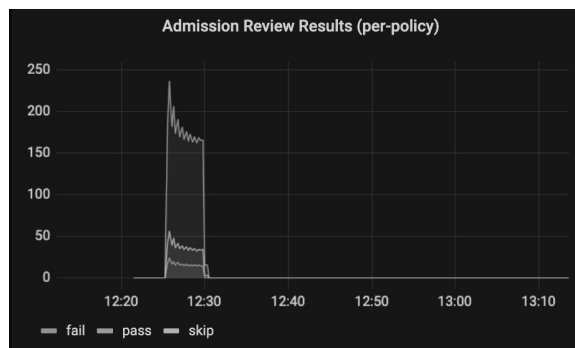
This panel displays the number of Kyverno policy rule evaluations triggered by admission requests, categorized by result (pass, fail, or skip). This provides visibility into the performance of individual rules during resource creation or modification events, helping administrators identify specific rules that frequently result in policy violations during admission reviews.

Figure 8-17 Admission Review Results per Rule**Policy Failures**

This panel displays the total number of failed Kyverno policy executions, grouped by policy type (Cluster or Namespaced). This helps users identify which category of policies is experiencing more compliance issues. Frequent failures may suggest misconfigured policies or workloads that do not comply with defined governance rules.

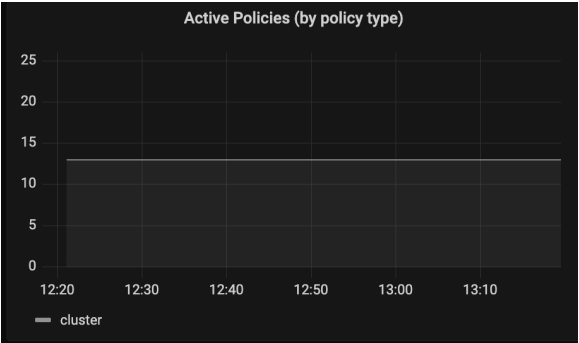
Figure 8-18 Policy Failures**Admission Review Results (per-policy)**

This panel displays the number of Kyverno admission review outcomes grouped by policy, indicating how many admission requests passed or failed for each policy. This helps users identify which policies are most frequently triggered and assess their effectiveness in enforcing compliance. This view also helps pinpoint policies that may need adjustment or review due to frequent failures.

Figure 8-19 Admission Review Results per Policy**Policy-Rule Info****Active Policies (by policy type)**

This panel displays the total number of active Kyverno policies, categorized by type—Cluster or Namespaced. This provides a clear overview of how policy enforcement is distributed across the cluster, helping users understand the balance between global and namespace-specific governance rules.

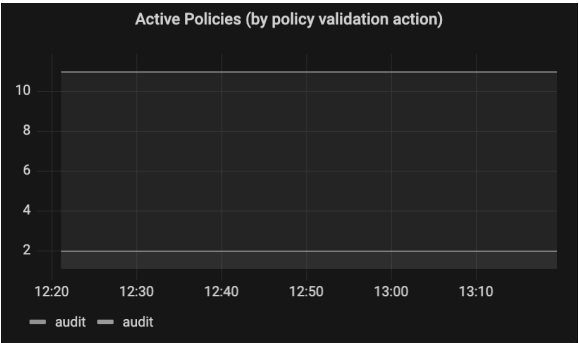
Figure 8-20 Active Policies by Policy Type



Active Policies (by policy validation action)

This panel displays the total number of active Kyverno policies grouped by their validation action, such as enforce or audit. This helps to understand how many policies are actively blocking non-compliant resources versus those that are only logging violations for monitoring and analysis.

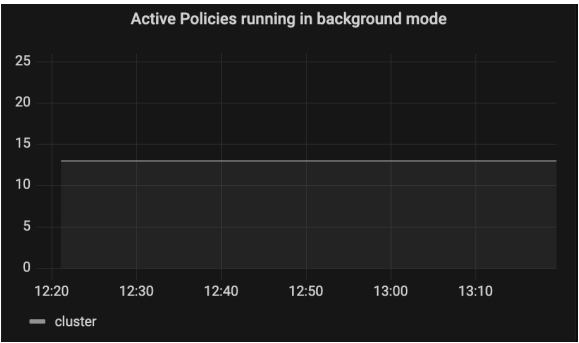
Figure 8-21 Active Policies by Policy Validation Action



Active Policies running in background mode

Active Policies running in background mode displays the number of Kyverno policies configured to perform periodic background scans on existing resources. This metric helps to understand how many policies continuously evaluate resource compliance beyond admission requests, ensuring ongoing enforcement throughout the cluster.

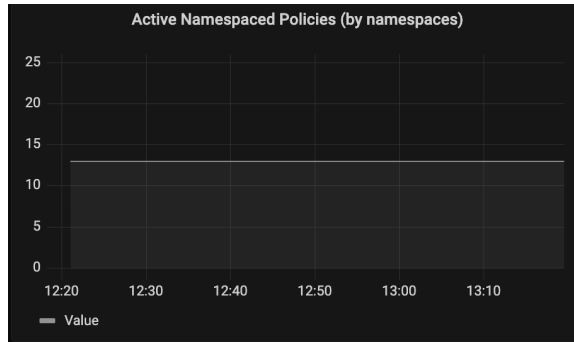
Figure 8-22 Active Policies Running in Background



Active Namespaced Policies (by namespace)

Active Policies running in background mode displays the number of Kyverno policies configured to perform periodic background scans on existing resources. This metric helps to understand how many policies continuously evaluate resource compliance beyond admission requests, ensuring ongoing enforcement throughout the cluster.

Figure 8-23 Active Namespaced Policies



Active Rules (by rule type)

Active Rules (by rule type) displays the total number of active Kyverno rules, categorized by type—such as validate, mutate, or generate. This helps users understand the makeup of rules enforcing compliance, automation, and configuration management within the cluster.

Figure 8-24 Active Rules by Rule Type



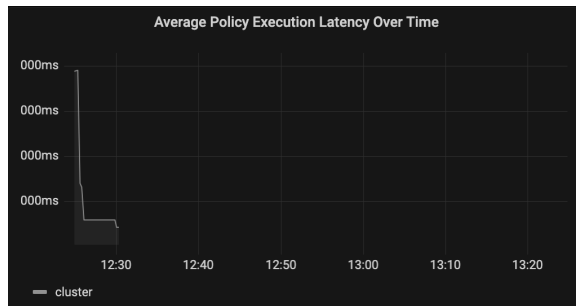
Policy-Rule Execution Latency

Average Rule Execution Latency over Time

This panel displays the average time Kyverno takes to execute individual policy rules during admission or background operations, segmented by rule type (validate, mutate, and generate). This helps to evaluate the performance and efficiency of each rule category over time, identifying any potential delays that might impact admission request processing or policy enforcement.

Figure 8-25 Average Rule Execution Latency over Time**Average Policy Execution Latency over Time**

This panel displays the average time Kyverno takes to execute entire policies—including all associated rules—during admission or background operations. The panel breaks down latency by policy type, distinguishing between Cluster Policies and Namespaced Policies, enabling users to compare performance between global and namespace-scoped policies. This helps identify latency trends that may impact policy enforcement and overall cluster responsiveness.

Figure 8-26 Average Policy Execution Latency over Time**Overall Average Rule Execution Latency**

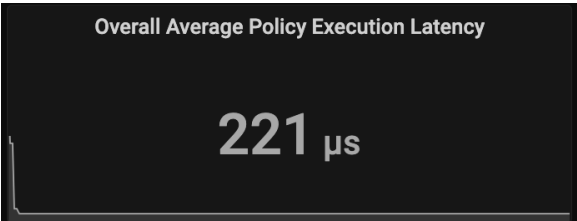
This panel displays the cumulative average time Kyverno takes to execute all policy rules across the cluster. This provides users with a high-level overview of rule processing performance and helps assess the overall efficiency of Kyverno's policy evaluations.

Figure 8-27 Overall Average Rule Execution Latency**Overall Average Policy Execution Latency**

This panel displays the cumulative average time Kyverno takes to execute all policies across the cluster. This offers users an overall measure of policy processing efficiency, helping to

identify performance trends that may impact admission control or background policy operations.

Figure 8-28 Overall Average Policy Execution Latency

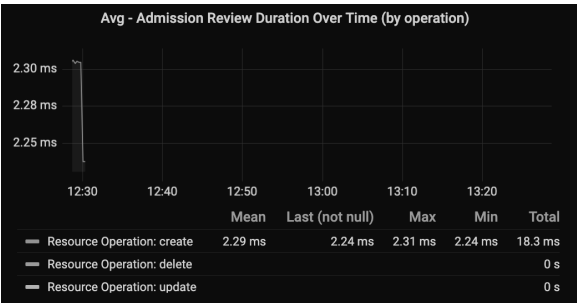


Admission Review Latency

Admission Review Latency Avg - Admission Review Duration over Time (by operation)

This panel displays the average time Kyverno takes to process admission review requests, categorized by operation type such as CREATE, UPDATE, or DELETE. This helps users assess the efficiency of Kyverno in handling different types of admission requests and identify any operation-specific latency issues.

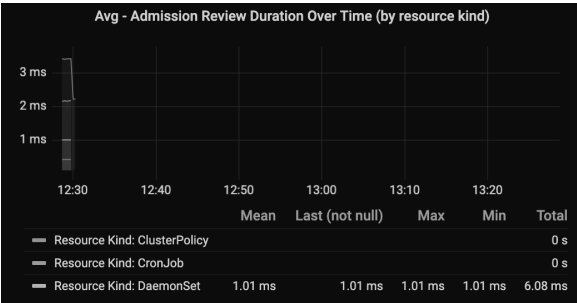
Figure 8-29 Admission Review Duration over Time



Avg - Admission Review Duration Over Time (by resource kind)

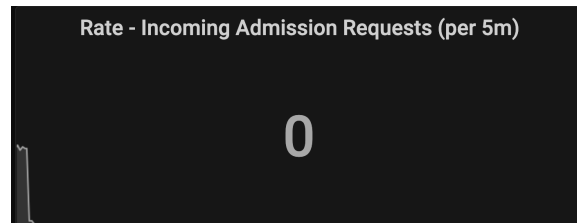
This panel displays the average time Kyverno takes to process admission review requests for various Kubernetes resource types, such as Pods, Deployments, or ConfigMaps. This helps users identify which resource kinds experience higher review latency, supporting performance analysis and optimization efforts.

Figure 8-30 Admission_Review_Duration_Time_Resourcekind

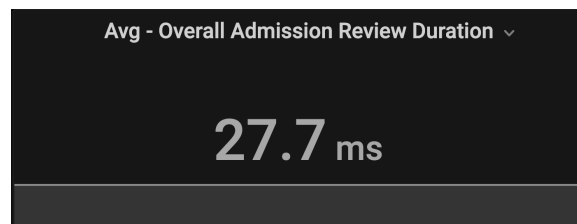


Rate - Incoming Admission Requests (per 5m)

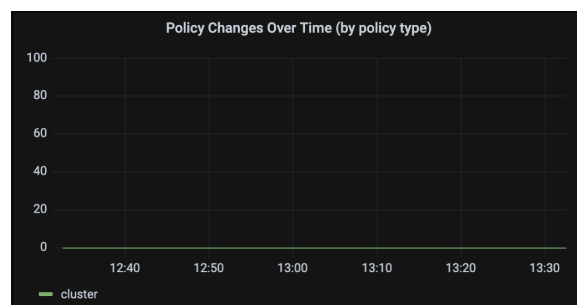
This panel displays the number of admission requests received and processed by Kyverno within each five-minute interval. This helps users monitor the workload and request traffic managed by Kyverno, offering insights into policy evaluation frequency and overall cluster admission activity levels.

Figure 8-31 Incoming Administration Requests**Avg - Overall Admission Review Duration**

This panel displays the average time Kyverno takes to process all admission review requests cluster-wide. This provides users with an overall measure of admission webhook performance, helping to identify latency trends that may impact resource creation or update operations.

Figure 8-32 Overall_Admission_Review_Duration**Policy Changes****Policy Changes Over Time (By Change Type)**

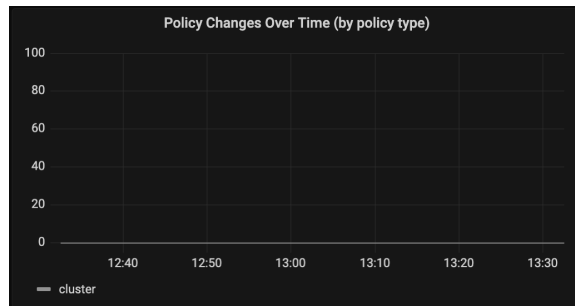
This panel displays the number of Kyverno policy modifications, categorized by change type such as create, update, or delete. This helps users track how often policies are changed over time, providing visibility into configuration activity and policy lifecycle management within the cluster.

Figure 8-33 Policy Changes over Time

Policy Changes Over Time (By Policy Type)

This panel displays the number of Kyverno policy modifications over time, categorized by policy type—Cluster or Namespaced. This helps users understand where most policy changes are taking place and whether updates are concentrated in global or namespace-specific governance rules.

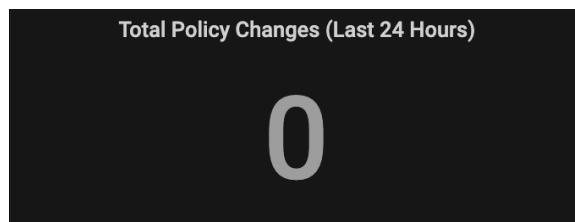
Figure 8-34 Policy_Changes_Over_Time_Policy_Type



Total Policy Changes (Last 24 Hours)

This panel displays the total number of Kyverno policy modifications —such as creations, updates, or deletions—recorded over the past 24 hours. This allows users to quickly assess recent policy activity and understand how frequently governance configurations are being updated within the cluster.

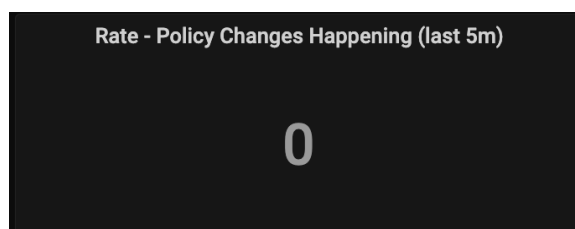
Figure 8-35 Total Policy Changes



Rate -Policy Changes (last 5m)

This panel displays the frequency of Kyverno policy modifications over the past five minutes. It helps users to monitor recent configuration activity, providing insight into how actively policies are being created, updated, or deleted in near real time.

Figure 8-36 Policy Changes (Last 5m)

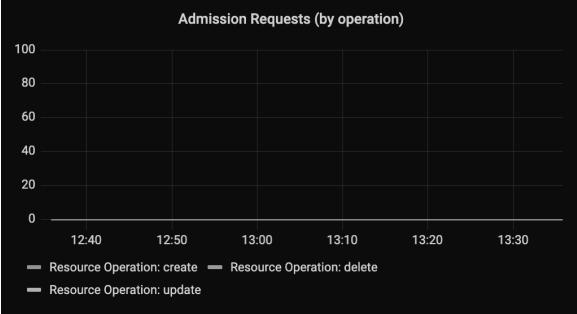


Admission Requests

Admission Requests (by operation)

This panel displays the number of admission requests processed by Kyverno, categorized by operation type such as CREATE, UPDATE, or DELETE. It helps users to understand the distribution of admission activities within the cluster and identify patterns or spikes in specific types of operations.

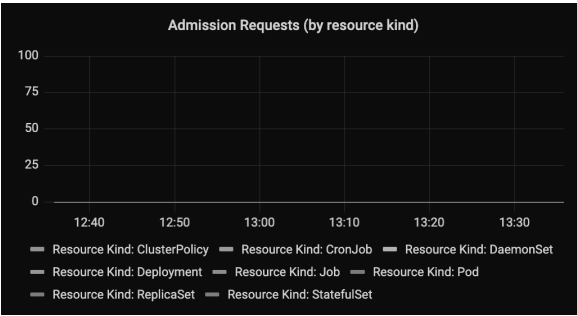
Figure 8-37 Admission Requests by Operation



Admission Requests (by resource kind)

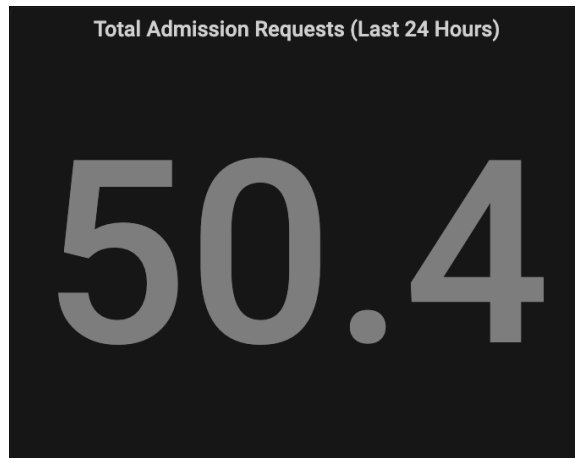
This panel displays the number of admission requests handled by Kyverno, grouped by Kubernetes resource type such as Pod, Deployment, or ConfigMap. It helps users to analyze which resource kinds generate the most admission traffic and assess how policy enforcement activity varies across different resource types.

Figure 8-38 Admission Request by Resource Kind



Total Admission Requests (Last 24 hrs)

This panel displays the total number of admission requests processed by Kyverno during the past 24 hours. It provides users with an overall measure of Kyverno’s activity level, helping to assess policy evaluation workload and cluster admission request volume over a daily period.

Figure 8-39 Total Admission Requests in Last 24 Hours

8.14.4 Fluentd Logging Dashboard

The Fluentd Log Ingestion Dashboard is a common service that offers visibility into Fluentd metrics across the cluster. It enables monitoring of logs, events, records, and pod status within the CNE observability stack.

Fluentd Monitoring Panels Overview

The Fluentd Monitoring dashboard has the following panels:

Fluentd Version

This panel displays the current Fluentd version by extracting it from the running pod metadata.

Fluentd Pods

This panel displays the count of active Fluentd pods.

Pod Write Operations Distribution

This panel helps visualize write operation rates per pod over 5-minute intervals.

Actual Buffer Status

This panel provides buffer-related metrics:

- Buffer Usage: Ratio of used vs. total buffer space.
- Buffer Total Size: Total buffer capacity in bytes.
- Buffer Queue Size: Size of the buffer queue in bytes.
- Buffer Stage Size: Size of staged (event) data in bytes.

Emitted Events Count by Pod

This panel displays the total events emitted per Fluentd pod.

Emitted Records Count by Pod

This panel displays the total records emitted per Fluentd pod.

Error Count by Pod

This panel displays the sum of errors recorded per pod.

Errors Count by Pod Over Time

This panel displays the timeline displaying error occurrences for each pod.

Total Retry Attempts Counts

This panel displays the total retry attempts by Fluentd, reflecting error handling activity.

Actual Retry Wait Time

This panel displays the average time Fluentd waits between retries, resetting after successful attempts.

Total Rollback Operations

This panel displays the total number of rollbacks, indicating error recovery actions.

Total Slow Flush Operations

This panel displays the count of slow flushes in the buffer, useful for bottleneck detection.

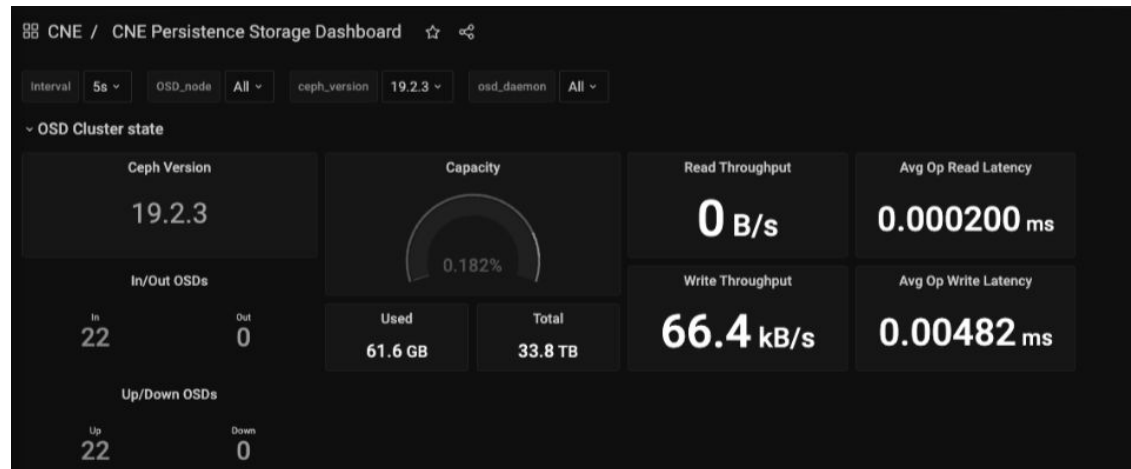
8.14.5 Persistence Storage Dashboard

The CNE Persistent Storage Dashboard, which is available in CNE as part of the common services, offers comprehensive visibility into Rook Ceph metrics across the cluster. It monitors OSD pods, provides a cluster and resource overview, and integrates with the CNE observability stack.

Persistence Storage Dashboard Panel Overview**OSD (Object Storage Daemons) Cluster State**

- Ceph Version: Displays the currently running Ceph cluster version.
- In/Out OSDs: Indicates the number of OSDs that are part of the cluster versus those that are out.
- Up/Down OSDs: Shows the number of OSDs that are operational versus those that are down or unresponsive.
- Capacity: Provides a visual gauge of the overall storage usage as a percentage.
- Used: Indicates the amount of storage space currently consumed.
- Total: Shows the total raw storage capacity available in the cluster.
- Read Throughput: Displays the current data read rate across the cluster.
- Write Throughput: Displays the current data write rate across the cluster.
- Avg Op Read Latency: Shows the average time required to complete read operations.
- Avg Op Write Latency: Shows the average time required to complete write operations.

Figure 8-40 Persistence Storage Dashboard



Cluster Summary

- **OSD:** Name of the Object Storage Daemon.
- **Node:** The node where the OSD resides.
- **Pod:** The pod responsible for managing the OSD.
- **Available:** Indicates whether the OSD is currently part of the cluster or marked as out.
- **Status:** Shows the current state of the OSD, such as running or not responding.

Figure 8-41 Cluster Summary

Cluster Summary				
OSD	Node	Pod	Available	Status
osd.1	k8s-node-2.stark.lab.us.oracle.com	rook-ceph-mgr-a-8d4748bf7-qs4rf	In	Up
osd.2	k8s-node-6.stark.lab.us.oracle.com	rook-ceph-mgr-a-8d4748bf7-qs4rf	In	Up
osd.4	k8s-node-2.stark.lab.us.oracle.com	rook-ceph-mgr-a-8d4748bf7-qs4rf	In	Up

Resource Usage Summary

1. Ceph Node: Name of the node hosting the Rook Ceph pod.
2. Pod: Name of the Rook Ceph pod.
3. Uptime: Duration the pod has been running and available.
4. Cores: Number of CPU cores allocated to the pod.
5. Memory: Amount of memory currently consumed by the pod.
6. CPU Usage: Amount of CPU resources currently used by the pod.

8.15 Changing Bastion Registry Credentials

The Bastion container registry can be configured to require authentication for image operations such as pull, push, and list. This allows registry access to be secured by requiring credentials for users or automation interacting with the registry.

This procedure describes how to update the Bastion container registry credentials after installation. It supports changing credentials from one existing set to another, as well as enabling credential-based access where no credentials were previously configured.

The update is designed to be non-disruptive. It does not require Kubernetes nodes or pods to restart and is not expected to impact the availability of services running on the Kubernetes cluster. Existing workloads and cluster-provided services should continue operating normally while the registry credential configuration is updated.

Enabling Credentials on a New Installation

To enable Bastion registry credentials on a new CNE installation, set the variables `occne_registry_user` and `occne_registry_pass` to your desired string values in the `[occne:vars]` section of the `secrets.ini` file, prior to the installation.

Note

Use a strong password when configuring `occne_registry_pass`. Choose a complex and unique password, and avoid common passwords to reduce the risk of unauthorized access.

Following is a sample `secrets.ini` file:

```
[occne:vars]
...
occne_registry_user=registryUser
occne_registry_pass=registryPass
...
```

Enabling or Changing Credentials on an Existing System

To enable Bastion registry credentials on an existing system running version 25.2.100 or later, perform the following steps:

1. Place the `secrets.ini` file in `/var/occne/cluster/${OCCNE_CLUSTER}/` on the active Bastion, alongside the inventory file, such as `occne.ini` or `hosts.ini`.
2. Set `occne_registry_user` and `occne_registry_pass` to your desired string values in the `[occne:vars]` section of the `secrets.ini` file.

Following is a sample `secrets.ini` file:

```
[occne:vars]
...
occne_registry_user=registryUser
occne_registry_pass=registryPass
...
```

- Plan a maintenance window before running the following command from the active Bastion. The command takes a few minutes to reconfigure the registries on the Bastions and update the credentials across the Kubernetes cluster.

```
OCCNE_STAGES=(DEPLOY) OCCNE_CONTAINERS=(PROV K8S) OCCNE_ARGS=--
tags=containerd,registry_setup pipeline.sh
```

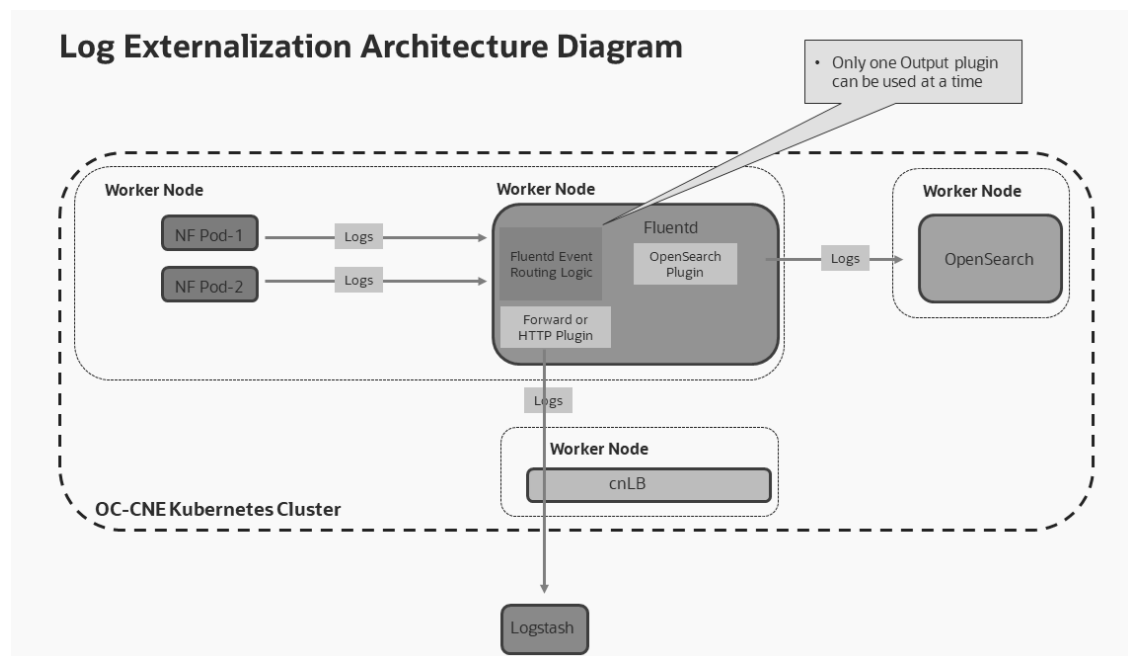
After completing this procedure, the registry requires the new credentials for image access. The default Bastion accounts, such as `admusr` or `cloud-user`, are automatically updated to use the new credentials, as are all Kubernetes nodes. If additional Bastion accounts access the registry, those users must run `podman login` with the appropriate credentials before accessing the registry.

8.16 Log Externalization Support

Overview

This section provides instructions on how to enable log externalization in CNE CNLB cluster. When log externalization is enabled, CNE and network function (NF) logs are no longer processed by the OpenSearch solution. Instead, these logs are exported from the CNE cluster to an external service provided by the customer through Fluentd using HTTP communication.

Figure 8-42 Log Externalization Architecture



Prerequisites

- This procedure must be performed on a cluster where CNLB is enabled.
- If the cluster is already deployed, you must perform the procedure from the active Bastion. Confirm you are on the active Bastion using the following command:

```
$ is_active_bastion
IS active-bastion
```

If the output does not indicate active-bastion, switch to the active bastion host before proceeding.

3. Users must know how to configure egress annotations (for example, which annotation keys or values are expected in your environment and where to apply them, for example, namespace, pod, deployment, and so on).

Limitations and Expectations

The following are limitations and expectations associated with enabling log externalization in CNE.

1. There is no rollback procedure.
2. Opensearch will not be installed in CNE.
3. Jaeger will not be installed in CNE.
4. Users must provide the correct configuration for Fluentd.
5. It is user's responsibility to configure dimensioning of Fluentd.
6. This procedure only offers to externalize logs; you cannot bring logs back in CNE.
7. CNE is not responsible for log loss resulting from external backend failures, misconfiguration, or capacity limitations beyond the configured retry mechanism.
8. Log externalization is enabled when:
 - OpenSearch and Jaeger are not installed in CNE
 - Fluentd installation is enabled

Configuring Log Externalization

Currently, CNE routes all logs to an in-cluster OpenSearch by default, which limits users from using centralized observability platforms (for example, Splunk or Elastic) and forces non-tracing users to allocate resources to components they may not need (such as Jaeger and OpenSearch).

If you are enabling log externalization before installation, perform the steps in the Bootstrap section. Otherwise, perform the steps in the Active Bastion section.

The following variables from `occne.ini` file will be modified in this procedure:

Table 8-34 `occne.ini` Variables

<code>occne.ini</code> Variable	Default Value	Log Externalization Enabled Value
<code>occne_opensearch_enabled</code>	True	False
<code>occne_jaeger_enabled</code>	True	False
<code>fluentd_logs_output_config_file_path</code>	None	Absolute path for output_plugin file. Example: <code>"/var/occne/cluster/my_cluster/fluentd_config/output_plugin"</code>
<code>occne_fluentd_egress_annotation</code>	None	Desired egress annotation. Example: <code>"nf-sigegr1@ nf-sig-egr1"</code>

OpenSearch

Disabling OpenSearch enables log externalization through Fluentd over HTTP. When OpenSearch is disabled, its Helm charts and related Kubernetes resources are not deployed in the cluster.

Perform the following steps to edit the `occne.ini` file to disable Opensearch:

1. Edit the `occne.ini` file inside the cluster directory:

```
$ vi /var/occne/cluster/$OCCNE_CLUSTER/occne.ini
```

2. Add and set the variable `occne_opensearch_enabled` to `False`. Inside the `occne.ini` file add the following line at the end:

```
occne_opensearch_enabled=False
```

Jaeger

Disabling Jaeger is an optional step for log externalization, but Jaeger cannot operate as a standalone component without OpenSearch. Therefore, if OpenSearch is disabled, Jaeger should also be disabled to optimize cluster resources; when Jaeger is disabled, its Helm charts and related Kubernetes resources are not deployed.

Note

OpenSearch can be deployed without Jaeger (enabling log search and analysis only), while deploying OpenSearch with Jaeger enables both log search and distributed tracing.

Perform the following steps to edit the `occne.ini` file to disable Jaeger:

1. Edit the `occne.ini` file in the cluster directory:

```
vi /var/occne/cluster/$OCCNE_CLUSTER/occne.ini
```

2. Add the following line at the end of the file to disable Jaeger:

```
occne_jaeger_enabled=False
```

Configuring Fluentd Output Plugin

Users must provide their own Fluentd `out_http` plugin configuration. The HTTP output plugin should be configured according to the official Fluentd documentation (Fluentd HTTP Output Plugin).

Note

In the official documentation, the parameter `endpoint_url` is shown as `endpoint`. Using `endpoint` can cause configuration errors, which will be visible in the Fluentd pod logs.

Perform the following steps to configure Fluentd output plugin:

1. Create the Fluentd directory:

Run the following command to create a `fluentd_config` directory inside the cluster directory:

```
mkdir /var/ocne/cluster/$OCCNE_CLUSTER/fluentd_config
```

2. Create the configuration file:

Inside the directory that was created in the previous step, create the configuration file `output_plugin` for Fluentd HTTP Output Plugin and insert its configuration:

```
vi /var/ocne/cluster/$OCCNE_CLUSTER/fluentd_config/output_plugin
```

Insert the Fluentd `out_http` configuration. Following is an example configuration:

```
<match **>
@type http
endpoint_url http://10.1.0.1:8080/
http_method post
serializer json
<buffer>
flush_interval 2s
</buffer>
</match>
```

3.

Setting the configuration file path in the `ocne.ini` file.

1. Edit the `ocne.ini` file in the cluster directory:

```
vi /var/ocne/cluster/$OCCNE_CLUSTER/ocne.ini
```

2. Add and set the variable `fluentd_logs_output_config_file_path` with the absolute path of the Fluentd configuration file:

Inside `ocne.ini` file add the next line at the end
`fluentd_logs_output_config_file_path="<Absolute path for output_plugin file>"`

Example:

```
fluentd_logs_output_config_file_path="/var/ocne/cluster/my_cluster/
fluentd_config/output_plugin"
```

Configuring Egress Annotation

This section is an optional step and may be skipped in order for Fluentd to externalize traffic through default network oam and default egress annotation `nf-oam-egr1@nf-oam-egr1`. In the case a different network is needed instead, perform the following steps:

Note

Ensure the Logstash server IP is present in `cnlb.ini` file, `egress_dest` variable for the egress annotation network.
Example:

```
Logstash server IP: 10.80.10.115
egress_dest = ["10.80.10.0/23"]
```

1. From the `cnlb.ini` file, obtain the available egress annotation network name: `nf-<service_network_name>-egr1`:

```
$ cat /var/ocne/cluster/$OCCNE_CLUSTER/cnlb.ini
```

Sample output:

```
...
[oam:vars]
service_network_name = "oam"
...
egress_dest = ["10.75.100.0/23"]
...
[sig:vars]
service_network_name = "sig"
...
egress_dest = ["10.80.10.0/23"]
```

2. With the previously identified network name, set the egress annotation by editing the `ocne.ini` and setting it in the variable `ocne_fluentd_egress_annotation`:

```
$ vi /var/ocne/cluster/$OCCNE_CLUSTER/ocne.ini
```

Inside the `ocne.ini` file, add the following line at the end with the required egress annotation:

```
ocne_fluentd_egress_annotation="<Egress annotation name>@<Egress
annotation name>"
```

Example:

```
ocne_fluentd_egress_annotation="nf-sig-egr1@nf-sig-egr1"
```

Applying the Configuration

This configuration can be done before deploying a cluster or on an already running cluster.

Deploying with Log Externalization changes

To deploy a cluster with Log Externalization enabled, perform the steps provided in *Oracle Communications Cloud Native Core, Cloud Native Environment Installation, Upgrade, and Fault Recovery Guide*.

Applying Log Externalization on a deployed cluster
Upgrade cluster to apply the changes:

To enable Log Externalization by upgrading the whole cluster, perform the steps provided in the "CNE Upgrade" section in *Oracle Communications Cloud Native Core, Cloud Native Environment Installation, Upgrade, and Fault Recovery Guide*.

Update only the services:

If these changes are needed on an cluster that is already deployed and a full upgrade is not needed, run the pipeline with the following parameters:

```
$ OCCNE_CONTAINERS=(CFG) OCCNE_STAGES=(UPGRADE) OCCNE_ARGS='--
tags=opensearch,fluentd-opensearch,jaeger' pipeline.sh
```

Validating the Fluentd pods configuration

Once the changes are applied by deploying, upgrading or just updating the services, validate the configuration:

Using Custom Log Handler

1. Run the following command to validate that the egress annotation is the selected one:

```
kco describe pod occne-fluentd-opensearch-<pod_id>
```

Example output:

```
Name:                occne-fluentd-opensearch-65f6f
Namespace:           occne-infra
Priority:              0
Priority Class Name:   default-priority-class
Service Account:      occne-fluentd-opensearch
Node:                 my_cluster-k8s-node-1/192.168.0.1
Start Time:           Thu, 26 Feb 2026 18:17:55 +0000
Labels:               app.kubernetes.io/instance=occne-fluentd-opensearch
                     app.kubernetes.io/name=fluentd
                     controller-revision-hash=5b4d5456d6
                     pod-template-generation=9
                     topology.kubernetes.io/region=RegionOne
                     topology.kubernetes.io/zone=nova

Annotations:
.
.
.
k8s.v1.cni.cncf.io/networks: default/nf-sig-egr1@nfsig-egr1
# "nf-sig-egr1" egress annotation present
```

2. Run the following command to validate the Fluentd output plugin configuration:

```
$ kco logs occne-fluentd-opensearch-<pod_id> -f | grep "<label @OUTPUT>" -
A 10
```

Sample output:

```
<label @OUTPUT>
  <match **>
```

```
@type http
endpoint_url "http://10.1.0.1:8080/"
http_method post
serializer json
<buffer>
  flush_interval 2s
</buffer>
</match>
```

Using Custom Log Handler

You can disable Fluentd and use an alternative log collection/forwarding solution.

1. OpenSearch and Jaeger cannot operate as standalone components without Fluentd in this workflow.
2. Disabling Fluentd will disable log externalization, because there will be no in-cluster log collection/tracking service.
3. When Fluentd is disabled, its Helm charts and related Kubernetes resources are not deployed.

If Fluentd is disabled, it is the user's responsibility to deploy and manage an equivalent application to handle log collection and forwarding.

Edit the `occne.ini` file to disable Fluentd

1. Edit the `occne.ini` file inside the cluster directory:

```
$ vi /var/occne/cluster/$OCCNE_CLUSTER/occne.ini
```

2. Add and set the variable `occne_fluentd_opensearch_enabled` to `False`. Inside the `occne.ini` file add the following line at the end:

```
occne_fluentd_opensearch_enabled=False
```

3. Apply the configuration as provided in the step "Applying the Configuration".

8.17 Using CNLB Command Line Utility

CNLB CLI (`cnlbctl`) is a lightweight, `kubectl`-style command-line utility used to inspect and validate the runtime state of CNLB in an CNE cluster. It connects to the CNLB manager pod (in the `occne-infra` namespace) and presents human-readable status tables for CNLB service sets, active/standby pods, frontend services, backend endpoint mappings, egress source IPs/ connection counts, and network attachment definitions—enabling quick health checks and troubleshooting from the Bastion host.

Overview

`cnlbctl` is a `kubectl`-like CLI that communicates with the CNLB manager pod in the `occne-infra` namespace and renders status tables for:

- service sets
- pods
- services
- backends

- egress mappings
- network attachments

All commands support a `--mock` flag for offline testing.

Installing cnlbctl Utility

Run the following commands from the Bastion host to install CNLBCTL_VERSION=1.0.0:

1. Download the binary:

```
wget http://$CENTRAL_REPO/ocne/binaries/cnlbctl/ocne-cnlbctl_${CNLBCTL_VERSION}.zip
```

2. Extract the `ocne-cnlbctl_${CNLBCTL_VERSION}.zip` file:

```
unzip ocne-cnlbctl_${CNLBCTL_VERSION}.zip
```

3. Change the file permissions:

```
chmod +x cnlbctl
```

4. Move to system path:

```
sudo mv cnlbctl /usr/local/bin/
```

5. Verify installation:

```
cnlbctl --help
```

Using cnlbctl Utility

The following are a few basic usage examples for `cnlbctl` utility:

Help

Use the following command to display the CLI's built-in help text, a summary of what `cnlbctl` is, the supported commands/subcommands:

```
cnlbctl --help
```

Mock mode

Use the following command to isolate CLI vs cluster issues:

```
cnlbctl --mock get info
```

Command Reference

Table 8-35 Command Reference

Command	What it shows	Common options
<code>cnlbctl get info</code>	Overall CNLB service-set sync status	<code>--mock</code>

Table 8-35 (Cont.) Command Reference

Command	What it shows	Common options
<code>cnlbctl get pods</code>	CNLB pods, role (active/standby), and node	<code>--mock</code>
<code>cnlbctl get services</code>	Frontend services exposed by CNLB	<code>--net <network></code>
<code>cnlbctl get backends</code>	Frontend IP → backend IP/ports mapping	<code>--net <network></code> <code>--svc <frontend-ip></code>
<code>cnlbctl get egress</code>	Pod-to-egress network mappings and connection counts	<code>--mock</code>
<code>cnlbctl get networks</code>	NADs, gateways, routes, CNLB pod assignments	<code>--mock</code>
<code>cnlbctl generate config</code>	Interactive generator for pod annotations	<code>--mock</code>

Get Command Examples

This section provides a few Get command usage examples that are run on Bastion host. These examples provide real time CLI output formats.

Command Name: `cnlbctl get info`

Usage: Run this command first to confirm which service sets are in Enabled sync state.

```
$ cnlbctl get info
```

Sample output:

```
SERVICE SET  SYNC STATUS  ACTIVE POD IP  ACTIVE POD NAME  STANDBY
POD NAME
serviceIpSet0 Enabled      10.233.94.141  cnlb-app-6d4889585d-m8rqb  cnlb-
app-6d4889585d-r8vgf
```

Description:

- Start here when validating CNLB health.
- Confirm that the service set is in Enabled state before checking pod or egress activity.

Command Name: cnlbctl get pods**Usage:** Run this command to view CNLB pods, their role, and the node they are running on.

```
$ cnlbctl get pods
```

Sample output:

SERVICE SET	POD IP	CNLB POD NAME	POD ROLE	NODE NAME
serviceIpSet0	10.233.94.141	cnlb-app-6d4889585d-m8rqb	active	occne1-luis-botello-ortega-k8s-node-cnlb-2
serviceIpSet0	10.233.104.144	cnlb-app-6d4889585d-r8vgf	standBy	occne1-luis-botello-ortega-k8s-node-cnlb-3

Description:

- The active pod IP should match the IP shown in cnlbctl get info.
- Standby rows appear without connection counts.

Command Name: cnlbctl get services**Usage:** Run this command to view the frontend services currently exposed by CNLB.

```
cnlbctl get services
```

Sample output:

SERVICE SET	APP NAME	CNLB POD NAME	FRONTEND IP	PORT	NETWORK NAME
serviceIpSet0	grafana	cnlb-app-6d4889585d-m8rqb	10.75.200.27	80	oam
serviceIpSet0	alertmanager	cnlb-app-6d4889585d-m8rqb	10.75.200.44	80	oam
serviceIpSet0	alertmanager	cnlb-app-6d4889585d-m8rqb	10.75.200.44	80	oam
serviceIpSet0	prometheus	cnlb-app-6d4889585d-m8rqb	10.75.200.51	80	oam
serviceIpSet0	tcp-test-app	cnlb-app-6d4889585d-m8rqb	10.75.200.112	80	sig

Description:

- Use this to verify which workloads are fronted by CNLB.
- This is also the quickest way to confirm the exposed frontend IP for an application.

Command Name: cnlbctl get backends**Usage:** Run this command to view the frontend-to-backend mappings, including backend ports and network details.

```
cnlbctl get backends
```

Sample output:

FRONTEND-IP	BACKEND-IPS	BACKEND-PORT-NAME	BACKEND-PORT
NETWORK-NAME	GATEWAY-IP	NETWORK-DEFINITION	
10.75.200.27	112.16.0.129	grafana	3000
oam	112.16.0.6	nf-oam-int3	
10.75.200.44	112.16.0.130	http-web	9093
oam	112.16.0.5	nf-oam-int2	
10.75.200.44	112.16.0.130	http-web	9093

```

oam          112.16.0.5  nf-oam-int2
10.75.200.51 112.16.0.135, 112.16.0.144 http-web      9090
oam          112.16.0.4  nf-oam-int1
10.75.200.112 113.16.0.129          test          8008
sig          113.16.0.4  nf-sig-int1

```

Description:

- Use `--net oam` to filter by network.
- Use `--svc 10.75.200.27` to inspect a specific frontend service.
- This command is useful when tracing frontend traffic to backend endpoints.

Command Name: `cnlbctl get egress`**Usage:** Run this command to view the pod-to-egress-network mappings and active connection counts.`cnlbctl get networks`

Sample output:

CNLB	POD NAME	POD IP	EGRESS SRC IP	NETWORK NAME	GATEWAY IP
EGR_CONN_COUNT					
cnlb-app-6d488958...	10.233.94.141	10.75.200.85	oam	112.16.0.193	
26					
cnlb-app-6d488958...	10.233.94.141	10.75.200.81	sig	113.16.0.193	
26					

Command Name: `cnlbctl get networks`**Usage:** Run this command to view NetworkAttachmentDefinitions (NADs), routes, and any CNLB pod assignments.`cnlbctl get networks`

Sample output:

NETWORK DEFINITION	GATEWAY IP	CNLB	POD IP	EGRESS SRC IP
ROUTES				
nf-oam-egr1	112.16.0.193	10.75.201.0/25,	10.75.201.0/25,	
10.75.201.0/25, ...	10.233.94.141	10.75.200.85		
nf-oam-ie1	112.16.0.59			
10.75.200.0/25		N/A		N/A
nf-oam-ie2	112.16.0.58			
10.75.200.0/25		N/A		N/A
nf-oam-ie3	112.16.0.57			
10.75.200.0/25		N/A		N/A
nf-oam-ie4	112.16.0.56			
10.75.200.0/25		N/A		N/A
nf-oam-ie5	112.16.0.55			
10.75.200.0/25		N/A		N/A
nf-oam-ie6	112.16.0.54			
10.75.200.0/25		N/A		N/A
nf-oam-int1				
112.16.0.4				N/A

```

N/A
...
nf-sig-egr1      113.16.0.193 10.75.253.0/25,
10.75.200.0/25      10.233.94.141 10.75.200.81
nf-sig-ie1      113.16.0.59
10.75.200.0/25      N/A      N/A
nf-sig-int1
113.16.0.4      N/A
N/A

```

Description:

- This view cross-references NADs with the pods currently consuming them.
- N/A indicates that the network is defined but not currently assigned to an active CNLB pod.

cnlbctl generate config Command

The `cnlbctl generate config` walks you through building the pod annotations required for ingress and egress connectivity. Below is a sample interactive session captured from the bastion host.

```
$ cnlbctl generate config
```

Sample output:

```

CNLB Configuration Generator
=====
_____ Available Networks
_____

Configured networks in cnlb.ini:

Include OAM network?
=====
  → 1. Yes - Include this network
  2. No - Skip this network
Enter your choice (1-2): 1
...
Include SIG network?
...
✓ Selected networks: OAM
Select Application Communication Type
=====
  → 1. IE - Ingress/Egress (bidirectional)
  2. EGR - Egress Only
  3. INT - Ingress Only
Enter your choice (1-3): 1
✓ Single network selected; using SN (Single Network) ingress mode
Enter backend port name and frontend port ...

```

Troubleshooting Common CLI Issues**Typo in subcommand**

If a subcommand is mistyped, the CLI returns a validation error with valid choices.

Example:

```
cnlbctl get egrss
```

Sample output:

```
invalid choice: 'egrss' (choose from ...)
```

Missing connection counts

EGR_CONN_COUNT is shown only when both of the following are true:

```
syncStatus = Enabled
```

```
podRole = active
```

Missing networks in generate config

Ensure that the `cnlb.ini` is readable from the bastion:

```
/var/ocne/cluster/$OCCNE_CLUSTER/cnlb.ini
```

Mock mode

Append `--mock` to simulate output without touching the cluster.

This is especially useful for demos, documentation, and offline testing.

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="ocne-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
occne-elastic-exporter-elasticsearch-exporter	1/1	1
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
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 to 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.