Oracle® Database

Administering Oracle Blockchain Platform

F20800-26
April 2026

Oracle Database Administering Oracle Blockchain Platform,

F20800-26

Contents

Preface

Administering Oracle Blockchain Platform explains how to provision and maintain Oracle Blockchain Platform instances.

Audience

This guide is intended for service administrators responsible for provisioning and maintaining Oracle Blockchain Platform.

Documentation Accessibility

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

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Related Resources

See these Oracle resources:

- *Using Oracle Blockchain Platform*

Conventions

The following text conventions are used in this document.

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

1

A Service Administrator's Roadmap to Oracle Blockchain Platform

This section walks you through an overview of Oracle Blockchain Platform Enterprise Edition, and provides a basic workflow for administrators.

Oracle Blockchain Platform Enterprise Edition Overview

Oracle Blockchain Platform provides a platform for building and running smart contracts and maintaining a tamper-proof distributed ledger.

Oracle Blockchain Platform is a network consisting of validating nodes (peers) that update the ledger and respond to queries by executing smart contract code—the business logic that runs on the blockchain. External applications invoke transactions or run queries through specialized inbuilt REST API calls or through their own custom client SDKs, which prompts selected peers to run the smart contracts. Multiple peers endorse (digitally sign) the results, which are then verified and sent to the ordering service. After consensus is reached on the transaction order, transaction results are grouped into cryptographically secured, tamper-proof data blocks and sent to peer nodes to be validated and appended to the ledger. Platform administrators can use the Blockchain Platform Manager to create and manage platform instances, while network administrators can use the Oracle Blockchain Platform console to configure the blockchain and monitor its operation.

Oracle Blockchain Platform Enterprise Edition provides a version of Oracle Blockchain Platform built on Kubernetes clusters and delivered as a set of pre-built container images for multiple Kubernetes distributions including Oracle Cloud Infrastructure Container Engine for Kubernetes (OKE), Red Hat OpenShift, and minikube. It can also be deployed to an existing provision of Microsoft Azure.

The Oracle Blockchain Platform Enterprise Edition downloadable artifact provides a distribution package that includes all the required container images, Helm charts and with executable scripts to help setup Oracle Blockchain Platform services onto a given Kubernetes cluster. Once Oracle Blockchain Platform Enterprise Edition has successfully been installed, Blockchain Platform Manager can be used for configuring and provisioning multiple Blockchain Platform instances, which will run across the available Kubernetes worker nodes. Similar to the cloud offering, this edition enables customers to create new complete blockchain instances in minutes.

The enterprise edition enables users to scale as required to handle the evolving workloads by increasing the replicas for various nodes. Unlike typical applications, Oracle Blockchain Platform's distributed ledger and the distributed metadata database handle data replication out-of-the-box.

Feature parity with the cloud version ensures that customers can deploy chaincode and use the same chaincode APIs and extensive REST APIs across both versions. Oracle innovations in using Berkeley DB for world state with SQL-based queries, built-in transaction synchronization to off-chain rich history database, intuitive and comprehensive console with powerful operations and monitoring tools, and all the other unique enterprise-grade features are shared across the cloud and on-premises versions.

Security, Authentication, and Authorization

Introduction to Oracle Blockchain Platform Enterprise Edition Security

Oracle Blockchain Platform Enterprise Edition deals with security on several levels. At the top level is the security related to the Oracle Blockchain Platform nodes. Next is the security associated with Blockchain Platform Manager that is used to manage the life cycle on Oracle Blockchain Platform instances. Users of Blockchain Platform Manager (the control plane) are able to create, scale out, scale in, and complete other life cycle operations on instances. For each instance there are users authorized for managing, monitoring, and administering an instance. Finally there are users of the instance that access an instance either via the Fabric SDKs or the Oracle Blockchain Platform REST Proxy. All user information including roles and passwords are stored in the built-in LDAP authentication server.

All sensitive data related to Oracle Blockchain Platform services (passwords, certificates and private keys) are stored using Kubernetes Secrets. It's important to ensure Kubernetes Secrets are secured by following their guidelines: Good practices for Kubernetes Secrets.

Managing Security

Securing Data at Rest

You may want to enable disk encryption in Kubernetes to protect data at rest. All Kubernetes APIs that let you write persistent API resource data support at-rest encryption.

See Encrypting Confidential Data at Rest.

Additionally, follow typical Kubernetes best practices for securing access to the components in your cluster, especially for Kubernetes secrets, because Oracle Blockchain Platform stores confidential information there.

Ports Exposed

Oracle Blockchain Platform makes use of Istio as the ingress gateway service to accept external traffic into Oracle Blockchain Platform services. Oracle Blockchain Platform Enterprise Edition uses the `https` port of the `istio-ingressgateway` service, as a single point of entry to listen to all external traffic. However, based on the configured service type, the public port number may vary.

ServiceType	Port Name	Exposed Port Number	Configurable During Installation?
LoadBalancer (default)	https	443 (default)	Yes
NodePort	https	3xxxx (nodePort value corresponding to https port)	Yes

Configuring Authentication and Authorization

Authentication in Oracle Blockchain Platform is performed using the included LDAP server. Users must have an account in the authentication server in order to be able to use the service.

Users associated with certain authentication groups are granted specific privileges as defined in User Groups and Roles.

Workflow for Administering Oracle Blockchain Platform

To start using Oracle Blockchain Platform, refer to the following tasks as a guide.

Task	Description	More Information
Prepare your environment	Read through the prerequisites for supported Kubernetes platforms and decide which is appropriate for your configuration. Ensure your hardware meets the required prerequisites.	Supported Topologies Prerequisites
Deploy Oracle Blockchain Platform Enterprise Edition	Deploy the Oracle Blockchain Platform Enterprise Edition on the platform of your choice. Access Blockchain Platform Manager	Select the topic for your platform: • Deploy Oracle Blockchain Platform Enterprise Edition on Oracle Kubernetes Engine • Deploy Oracle Blockchain Platform Enterprise Edition on minikube • Deploy Oracle Blockchain Platform Enterprise Edition on Red Hat OpenShift Local • Deploy Oracle Blockchain Platform Enterprise Edition on Azure Red Hat OpenShift • Deploy Oracle Blockchain Platform Enterprise Edition on a Red Hat OpenShift On-Premises Cluster Log on to Oracle Blockchain Platform for the First Time
Add and manage users and roles	A default LDAP server is provided with Oracle Blockchain Platform Enterprise Edition.	Configure the Built-In LDAP Server User Groups and Roles
Provision a service instance	Use the Create Instance wizard in Blockchain Platform Manager to create a service instance.	Provision an Instance using the Blockchain Platform Manager
Configure your blockchain network	Once your instance is created, you can use the Blockchain Platform Console to configure the network.	What's the Console?

After you've created your instance and any required users, you can begin to use Oracle Blockchain Platform as described in What's the Console?

2

Prepare to Install Oracle Blockchain Platform

This section lists the prerequisites and supported topologies of Oracle Blockchain Platform Enterprise Edition.

Prerequisites

This topic contains the hardware and software prerequisites for installing Oracle Blockchain Platform Enterprise Edition.

Kubernetes Platforms

The following platforms are supported:

- Oracle Cloud Infrastructure Container Engine for Kubernetes (OKE) v1.29.1 or later
- Red Hat OpenShift Local - Embedded OpenShift version 4.15.3 or later compatible versions, Code Ready Container version 2.34.1 or later compatible versions
- Red Hat OpenShift on Microsoft Azure v4.17.27 or later compatible versions
- Red Hat OpenShift on premises cluster v4.17 or later compatible versions
- minikube v1.33.1 or later - test environment only, not for production

You must have a valid domain name that can be resolved by a DNS server. For testing or non-production environments only, you can configure local name resolution by adding entries in the `/etc/hosts` file. The specific host names and IP addresses to use in the `/etc/hosts` file are generated after deployment. For more information, see the *Access Blockchain Platform Manager* section of the installation topic for your version of Kubernetes.

Other Prerequisite Software

Additionally you'll need the following tools to assist with managing your Kubernetes platform and installing your Oracle Blockchain Platform container:

- kubectl version 1.29.3 or later - Kubernetes' command line tool
- Helm version 3.12.3 or later - a Kubernetes package manager
- Tools to manage your containers and pods - choose one of the following:
  - Podman version 4.9.4-rhel or later
  - Docker version 24.0.6 or later
- yq version 4.42.1 - a command line YAML processor
- jq v1.7.1 or later - a command line JSON processor
- Istio - security and traffic management tool for deployments
  - v1.20.2 or later for OKE or minikube running Oracle Blockchain Platform Enterprise Edition patch 4 or earlier
  - v1.22.1 or later for OKE or minikube running Oracle Blockchain Platform Enterprise Edition patch 5 or later
  - v1.22.1 for Red Hat OpenShift Local

  - v1.26.0 for Red Hat OpenShift on Microsoft Azure
  - v1.26.0 for Red Hat OpenShift On Premises
- istioctl - the command line tool for Istio

Web Browsers

All administrative tools that are included with Oracle Blockchain Platform can be accessed by using the following browsers:

- Mozilla Firefox
- Microsoft Edge
- Google Chrome
- Apple Safari

Install Prerequisite Software

This section provides an example walkthrough of installing the tested versions of the prerequisites. Refer to each product's documentation for additional information and any required modifications to the install instructions.

Oracle Linux

Install kubectl

The kubectl version should always be within one minor version difference of your cluster. For example if your Kubernetes cluster version is v1.29.1, you can use kubectl v1.29.3. For additional information, see Kubernetes Install Tools.

```
# Download:
    curl -LO https://dl.k8s.io/release/v1.29.3/bin/linux/amd64/kubectl

# Setup:
    # Make binary file executable:
    chmod +x ./kubectl
    # Move the downloaded binary to /usr/local/bin or /usr/bin and make
"root" as the owner of this binary
    sudo mv ./kubectl /usr/bin/kubectl
    sudo chown root: /usr/bin/kubectl

# Verify:
    $ kubectl version --client --output=yaml
    clientVersion:
        buildDate: "2024-03-15T00:08:19Z"
        compiler: gc
        gitCommit: 6813625b7cd706db5bc7388921be03071e1a492d
        gitTreeState: clean
        gitVersion: v1.29.3
        goVersion: go1.21.8
```

```
        major: "1"
        minor: "29"
        platform: linux/amd64
    kustomizeVersion: v5.0.4-0.20230601165947-6ce0bf390ce3
```

Install Helm

Oracle Blockchain Platform Enterprise Edition was tested with Helm v3.12.3. For additional information, see Installing Helm.

```
#Download install script:
    curl -fsSL -o get_helm.sh https://raw.githubusercontent.com/helm/helm/
master/scripts/get-helm-3

# Provide execute permission and execute:
    chmod +x get_helm.sh
    ./get_helm.sh

# Verify:"
    $ helm version
    version.BuildInfo{Version:"v3.12.3",
GitCommit:"3a31588ad33fe3b89af5a2a54ee1d25bfe6eaa5e", GitTreeState:"clean",
GoVersion:"go1.20.7"}
```

Install Podman

Oracle Blockchain Platform Enterprise Edition was tested with Podman v4.9.4-rhel. For additional information, see Podman Installation Instructions.

```
# Update the package list:
    sudo yum update

# Install podman:
    sudo yum install podman -y

# create docker alias to run podman commands
    sudo yum install -y podman-docker

#Verify:
    $ podman --version
        podman version 4.9.4-rhel
    $ docker --version
        podman version 4.9.4-rhel
```

Install yq

Oracle Blockchain Platform Enterprise Edition was tested with yq v4.42.1. For additional information, see yq README.

```
# Download:
    wget https://github.com/mikefarah/yq/releases/download/v4.42.1/
yq_linux_amd64.tar.gz -O - | tar xz

# Setup:
    sudo mv yq_linux_amd64 /usr/bin/yq
```

```
    sudo chmod +x /usr/bin/yq

# Verify:
yq --version
```

Install jq

Oracle Blockchain Platform Enterprise Edition was tested with jq v1.7.1. For additional information, see Download jq.

```
# Install:
    sudo dnf install jq

# Verify:
   jq --version
   jq-1.7.1
```

macOS

Install kubectl

The kubectl version should always be within one minor version difference of your cluster. For example if your Oracle Kubernetes Engine cluster version is v1.29.1, you can use kubectl v1.29.3. For additional information, see Kubernetes Install Tools.

```
curl -LO https://dl.k8s.io/release/v1.29.3/bin/darwin/amd64/kubectl

# Setup:
    # Make binary file executable:
    chmod +x ./kubectl
    # Move the downloaded binary to /usr/local/bin and make "root" as the
owner of this binary
    sudo mv ./kubectl /usr/local/bin/kubectl
    sudo chown root: /usr/local/bin/kubectl

# Verify:
    $ kubectl version --client --output=yaml

    clientVersion:
        buildDate: "2024-03-15T00:08:19Z"
        compiler: gc
        gitCommit: 6813625b7cd706db5bc7388921be03071e1a492d
        gitTreeState: clean
        gitVersion: v1.29.3
        goVersion: go1.21.8
        major: "1"
        minor: "29"
        platform: darwin/amd64
    kustomizeVersion: v5.0.4-0.20230601165947-6ce0bf390ce3
```

Install Helm

Oracle Blockchain Platform Enterprise Edition was tested with Helm v3.12.3. For additional information, see Installing Helm.

```
#Download install script:
    curl -fsSL -o get_helm.sh https://raw.githubusercontent.com/helm/helm/
master/scripts/get-helm-3

# Provide execute permission and execute:
    chmod +x get_helm.sh
    ./get_helm.sh

# Verify:"
    $ helm version
    version.BuildInfo{Version:"v3.12.3",
GitCommit:"3a31588ad33fe3b89af5a2a54ee1d25bfe6eaa5e", GitTreeState:"clean",
GoVersion:"go1.20.7"}
```

Install Podman

Oracle Blockchain Platform Enterprise Edition was tested with Podman v4.9.4-rhel. For additional information, see Podman Installation Instructions.

```
# Install:
     brew install podman

# After installing, you need to create and start your first Podman machine:

     podman machine init
     podman machine start

# Verify:
     podman info
     host: arch: amd64 buildahVersion: 1.36.0 cgroupControllers: - cpu - io -
memory - pids cgroupManager: systemd

# Make docker commands available via podman by creating a symlink:
     sudo ln -s /usr/local/bin/podman /usr/local/bin/docker
```

Install yq

Oracle Blockchain Platform Enterprise Edition was tested with yq v4.44.1. For additional information, see yq README.

```
# Install:
    brew install yq

# Verify:
   yq--version
   yq (https://github.com/mikefarah/yq/) version v4.44.1
```

Install jq

Oracle Blockchain Platform Enterprise Edition was tested with jq v1.7.1. For additional information, see Download jq.

```
# Install:
    brew install jq

# Verify:
   jq --version
   jq-1.7.1
```

Install Istio

Istio extends Kubernetes to provide traffic management and security to complex deployments. Oracle Blockchain Platform Enterprise Edition uses Istio as the ingress gateway service to accept incoming traffic into various services.

- Oracle Linux
- macOS

Oracle Linux

We recommend installing Istio and istioctl (Istio configuration command line utility).

To download Istio:

```
# Download istioctl tool
   curl -L https://istio.io/downloadIstio | sh -

# (optional) You can move the downloaded "istio-1.22.1" directory
   mv ./istio-1.22.1 $HOME/istioctl

# Add the istioctl client to your path
export PATH=$HOME/istioctl/bin:$PATH
```

macOS

We recommend installing Istio and istioctl (Istio configuration command line utility).

To download Istio:

```
# Download istioctl tool
   curl -L https://istio.io/downloadIstio | sh -

# (optional) You can move the downloaded "istio-1.22.1" directory
   mv ./istio-1.22.1 $HOME/istioctl

# Add the istioctl client to your path
export PATH=$HOME/istioctl/bin:$PATH
```

> **Note**
>
> Istio installation will be completed after your Kubernetes cluster has been created. It has a dependency on the `.kube/config` file.

Istio's `ingressgateway` service can be configured in Kubernetes to run with either `LoadBalancer` or `NodePort` service types. See the Istio documentation for details: Istio Ingress Gateways

Load Balancer
If your Kubernetes cluster supports an external load balancer, it's recommended to configure the Istio ingress gateway service as a load balancer using the Oracle Blockchain Platform Enterprise Edition `runme` script that you'll use during deployment. Oracle Blockchain Platform Enterprise Edition will use Istio ingress gateway's https port (443) as the public port to accept the incoming traffic. This port value can be optionally customized during installation of Oracle Blockchain Platform Enterprise Edition using the `runme` script.

Node Port
In cases where the load balancer service type cannot be used, the Istio ingress gateway service can be configured with node port service type. Oracle Blockchain Platform Enterprise Edition will use the `nodePort` value of the https port in the Istio ingress gateway to route external traffic to Oracle Blockchain Platform Enterprise Edition inside the Kubernetes cluster. The value of the `nodePort` for the https port can be optionally customized (based on the allowed `nodePort` range) during installation of Oracle Blockchain Platform Enterprise Edition using the `runme` script. By default, the allowed `nodePort` range in Kubernetes cluster is 30000-32767.

> **Note**
>
> Do not update the value of the https `port` or `nodePort` after installation of Oracle Blockchain Platform Enterprise Edition as it will affect its function.

Hostname Resolution for Oracle Blockchain Platform Enterprise Edition Services

Oracle Blockchain Platform Enterprise Edition services use uniquely generated hostnames that are configured in the Istio gateways/virtualservices. To communicate with the Oracle Blockchain Platform Enterprise Edition services inside the Kubernetes cluster, you are required to use these unique hostnames from your browsers or applications. Based on the chosen service type for `istio-ingressgateway`, these hostnames are required to be resolved to an IPv4 address in the following way:

- If `LoadBalancer`, they resolve to the external IP address generated for the `istio-ingressgateway` service
- If `NodePort`, they resolve to the worker nodes' IP addresses

Supported Topologies

In addition to creating a topology in which both the founder and participant are on Oracle Blockchain Platform Enterprise Edition, the following interoperability scenarios are supported:

- Oracle Blockchain Platform Enterprise Edition founder, Oracle Blockchain Platform Cloud participant

- Oracle Blockchain Platform Cloud founder, Oracle Blockchain Platform Enterprise Edition participant
- Oracle Blockchain Platform Enterprise Edition on Microsoft Azure founder and Oracle Blockchain Platform Cloud participant

> **Note**
>
> Interoperability between Hyperledger Fabric 1.4 and Hyperledger Fabric 2.5 is not supported.

3

Install Your Oracle Blockchain Platform Instance

This section describes how to deploy Oracle Blockchain Platform Enterprise Edition, and log on to Oracle Blockchain Platform for the first time.

Deploy Oracle Blockchain Platform Enterprise Edition on Oracle Kubernetes Engine

Before deploying Oracle Blockchain Platform Enterprise Edition, you must have a Kubernetes cluster running and you must install several prerequisites.

For detailed information about Oracle Kubernetes Engine, see Oracle Cloud Infrastructure Container Engine for Kubernetes

Create an Oracle Kubernetes Engine Cluster on OCI

Recommended minimum specifications for your Oracle Kubernetes Engine Cluster:

	Development	Production with High Availability
Minimum Version	v1.29.1	v1.29.1
Node Type	Managed	Managed
Node Image	Oracle Linux 8	Oracle Linux 8
Node CPU	2 OCPUs or higher	4 OCPUs or higher
Node Memory	24 GB or higher	32 GB or higher
Node Count	1 or higher	3 or higher
Boot volume size	100 GB or higher. The default boot volume of 50 GB may not be sufficient to hold Oracle Blockchain Platform Enterprise Edition container images and temporary data for chaincode pods because of limited ephemeral storage.	100 GB or higher. The default boot volume of 50 GB may not be sufficient to hold Oracle Blockchain Platform Enterprise Edition container images and temporary data for chaincode pods because of limited ephemeral storage.

- For added security, select `Private workers` for the Kubernetes worker nodes.
- Ensure that the worker nodes have access to the internet, which is required to install chaincodes on your Oracle Blockchain Platform instances.

This section walks through creating an example Oracle Kubernetes Engine on OCI. For additional options and details, see Creating Kubernetes Clusters Using Console Workflows

1. Log in to your OCI tenancy, selecting your region and compartment.
2. Open the navigation menu and click **Developer Services**. Under **Containers & Artifacts**, click **Kubernetes Clusters (OKE)**.

3. On the **Cluster List** page, click **Create cluster**.
4. In the **Create cluster** dialog, select **Quick create** and click **Submit**.
5. On the **Create cluster** page, either accept the default configuration details for the new cluster, or specify alternatives as follows:
   - **Name:** The name of the new cluster. Either accept the default name or enter a name of your choice.
   - **Compartment:** The compartment in which to create the new cluster and the associated network resources.
   - **Kubernetes version:** The version of Kubernetes to run on the control plane nodes and worker nodes of the cluster. v1.29.1 was tested with Oracle Blockchain Platform Enterprise Edition.
   - **Kubernetes API endpoint:** The type of access to the cluster's Kubernetes API endpoint. Select **Public** (accessible directly from internet). A public regional subnet is created and the Kubernetes API endpoint is hosted in that subnet. The Kubernetes API endpoint is assigned a public IP address as well as a private IP address.
   - **Node type:** Specify the type of worker nodes in the first node pool in the cluster. Select **Managed**. You have responsibility for managing the worker nodes in the node pool. Managed nodes run on compute instances (either bare metal or virtual machine) in your tenancy. As you are responsible for managing managed nodes, you have the flexibility to configure them to meet your specific requirements. You are responsible for upgrading Kubernetes on managed nodes, and for managing cluster capacity.
   - **Kubernetes worker nodes:** The type of access to the cluster's worker nodes. Select **Private** (accessible through other VCN subnets). A private regional subnet is created to host worker nodes. The worker nodes are assigned a private IP address.
   - **Node shape:** The shape to use for each node in the node pool. The shape determines the number of CPUs and the amount of memory allocated to each node. The list shows only those shapes available in your tenancy that are supported by Container Engine for Kubernetes. Oracle Blockchain Platform Enterprise Edition was tested with VM.Standard.E3.Flex and VM.Standard.E4.Flex shapes.
   - **Image:** The image to use on worker nodes in the managed node pool. An image is a template of a virtual hard drive that determines the operating system and other software for the managed node pool. Oracle Blockchain Platform Enterprise Edition was tested with Oracle Linux 8.
   - **Node count:** The number of worker nodes to create in the node pool, placed in the regional subnet created for the cluster. Select three or more.

   Select the following Advanced Options:

   - **Boot volume**: Configure the size and encryption options for the worker node's boot volume. The default boot volume of 50 GB may not be sufficient to hold Oracle Blockchain Platform Enterprise Edition images and temporary data for chaincode pods because of limited ephemeral storage. If you plan to deploy several chaincodes (more than five), increase the boot volume to approximately 100 GB.
6. Review the options you've selected, and click **Create Cluster**.
7. Ensure that your worker nodes and node pools are running:
   - Under **Resources**, select **Nodes**. For each worker node, ensure that the node is ready, active, and matches the Kubernetes cluster version.
   - Under **Resources**, select **Node pools**. For your node pool, ensure that the pool is active and matches the Kubernetes cluster version.

Install the OCI Command Line Interface

This section provides an example walkthrough of installing the OCI Command Line Interface. Oracle Blockchain Platform Enterprise Edition was tested with v3.42.0. For additional information, refer to OCI Command Line Interface.

- Oracle Linux
- macOS

Oracle Linux

```
# Install:
    sudo dnf -y install oraclelinux-developer-release-el8
    sudo dnf -y install python36-oci-cli

# Verify:
    $ oci --version
    3.42.0
```

macOS

```
# Install:

    brew update && brew install oci-cli

## If this fails with "Error: python@3.12: the bottle needs the Apple Command
Line Tools to be installed.", run below command:

    xcode-select --install

# Verify:
   oci --version
   3.43.1
```

Create Install Initiator System

Set up local access for the cluster

See the following for additional information: Setting Up Local Access to Clusters.

1. Copy your RSA key to the Oracle Linux or macOS system on which you installed the prerequisites. Your keys can be found in the OCI console: **Identity**, and then **Domains**, and then **OracleIdentityCloudService domain**, and then **Users**, and then **User name**, and then **API keys**. Secure the key: `chmod 400 your_rsa.key`
   You can create a key if needed. See How to Generate an API Signing Key
2. In the OCI console, go to your cluster and open the **Cluster Details** page.
3. Select **Access Cluster**, and then **Local Access**.
   a. Create a directory to contain the `kubeconfig` file: `mkdir -p $HOME/.kube`

b. Copy the **To access the kubeconfig for your cluster via the VCN-Native public endpoint** command.

c. Run the command on your Linux or macOS system. Because the config file doesn't exist yet, you'll be prompted for the following:

- Do you want to create a new config file? [Y/n]: y
- Do you want to create your config file by logging in through a browser? [Y/n]: n
- Enter a location for your config [/home/opc/.oci/config]: select a location
- Enter a user OCID: can be found in the OCI console
- Enter a tenancy OCID: can be found in the OCI console
- Enter a region by index or name: Enter the number corresponding to your Tenancy's Region, for example 12
- Do you want to generate a new API Signing RSA key pair? If you decline you will be asked to supply the path to an existing key. [Y/n]: n
- Enter the location of your API Signing private key file: location of RSA key file created above

This creates a config file that gives the Kubernetes control plane VM access to the cluster hosted on OCI.

d. When the OCI config file is created, you must re-run the copied **To access the kubeconfig for your cluster via the VCN-Native public endpoint** command. It will used the config file that you just created.

4. Verify that you can reach the Oracle Kubernetes Engine cluster: `kubectl get nodes`. If the setup is correct, the command will show all the worker nodes in your cluster.
5. Restrict access to the config file: `chmod 600 $HOME/.kube/config`
6. Set your `KUBECONFIG` environment variable to the file for this cluster: `export KUBECONFIG=$HOME/.kube/config`

Note

Check whether your OCI config file has multiple profiles similar to the following text:

```
[OCI_PROFILE_A]
fingerprint = .............
key_file = .............
tenancy = .............
region = .............
user = .............

[OCI_PROFILE_B]
fingerprint = .............
key_file = .............
tenancy = .............
region = .............
user = .............
```

If so, you'll need to customize the `kubeconfig` file or you'll get an authorization error when you try to install Oracle Blockchain Platform Enterprise Edition. In the users section of the `kubeconfig` file, add a line to specify which user to use in your OCI config file. For example

```
users:
- name: user-c3xxxxxq
  user:
    exec:
      apiVersion: client.authentication.k8s.io/v1beta1
      args:
      - ce
      - cluster
      - generate-token
      - --cluster-id
      - ocid1.cluster.oc1.eu-frankfurt-1.aaaaaaxxxxxxxxxyyyyyy
      - --region
      - eu-frankfurt-1
      - --profile
      - <OCI_PROFILE_NAME>
      command: oci
      env: []
      interactiveMode: IfAvailable
      provideClusterInfo: false
```

where `<OCI_PROFILE_NAME>` would be `OCI_PROFILE_A`

Complete the Istio Install

Oracle Blockchain Platform Enterprise Edition supports version 1.20.2 and later. You must have completed the steps in Install Istio before running the following commands.

```
# Install
    istioctl install --set profile=default --set
```

```
values.pilot.env.ENABLE_TLS_ON_SIDECAR_INGRESS=true --set
components.cni.enabled=true --set values.cni.repair.deletePods="true"
    ## Enter "y" when prompted for "Proceed? (y/N)"

# Verify:
    $ istioctl version
    client version: 1.22.1
    control plane version: 1.22.1
    data plane version: 1.22.1 (1 proxies)
```

Set Up an Auth Token for Your User

Create an Auth Token for your administrative user so that you can pull images from the OCI Registry: Generating an Auth Token to Enable Login to Oracle Cloud Infrastructure Registry.

Install Oracle Blockchain Platform Enterprise Edition

1. On the Oracle Blockchain Platform Enterprise Edition page, click **Download** and follow the steps to download the Oracle Blockchain Platform Enterprise Edition package. Extract the package from the `.zip` file, and then extract the package from the downloaded archive file.

   ```
   tar -xzf <distribution-package-file>
   # example tar -xzf obpee_package_24.1.3-20240723083137.tgz
   ```

2. Update the `runme-input.yaml` file with the required values. The following example `runme-input.yaml` file can be used as reference:

   ```
   imageRegistryConfiguration:
     registry: <container_registry_name>
     imageTagPrefix: <container-image-repository-prefix>
     username: <container-registry-username>

   imageReleaseVersion: 24.1.3-20240723083137

   # storageClassName should be set to create a dynamic persistent volume. If
   empty, default storageClass is used.

   controlPlaneStorage:
     storageClassName:
     # Example 500Mi, 5Gi
     size: 4Gi

   parentDomainName: example.com
   #imagePullTimeout: Use this field to customize the wait time (in seconds)
   for pulling the required container images from the repository. Default is
   1800 seconds.
       imagePullTimeout: 1800
   ```

   In the previous example, the variables are defined as shown in the following list:

   - **`imageRegistryConfiguration.registry`**: Container registry server to use. Example: `iad.ocir.io`

- **`imageRegistryConfiguration.imageTagPrefix`**: Container base repository path with the registry, where the images will be pushed to and pulled from. Example: `iad.ocir.io/obpee/bcs`
- **`imageRegistryConfiguration.username`**: Container registry login user name
- **`imageReleaseVersion`** - Oracle Blockchain Platform Enterprise Edition release version
- **`controlPlaneStorage.storageClassName`**: Kubernetes storage class to use for PVC (PersistentVolumeClaim). If empty, the default `storageClass` is used
- **`controlPlaneStorage.size`**: PVC size for Blockchain Platform Manager (control plane) services
- **`parentDomainName`**: Domain name to use for Blockchain Platform Manager services. Example: `example.com`
- **`imagePullTimeout`**: Image pull wait timeout in seconds during Oracle Blockchain Platform Enterprise Edition installation. Default is 1800 seconds.

3. Run the `runme_oke.sh [--publish-images]` script, following the prompts.

> **Note**
>
> The optional `--publish-images` command uploads the containers to a container image registry such as Oracle Cloud Infrastructure Registry using the details specified in the `runme-input.yaml` file.

- Enter the default LDAP admin password (the password will not be displayed): sets the administrator's password for the built-in LDAP authentication server.
- Enter the default control plane admin user password (the password will not be displayed): sets the Blockchain Platform Manager administrator's password.
- If `StorageClass` wasn't provided in the `runme-input.yaml` file, the system checks if the default storage class is set and ask if you want to use it.
- Confirm the Istio ingress gateway service type: `LoadBalancer` is the default. `NodePort` is also supported. Note that accessing `NodePorts` requires that the Kubernetes cluster was created with public worker nodes. See Install Istio.
- Confirm the Istio ingress gateway service https port: the default is 443 for the `LoadBalancer` service type.
- Enter registry *<registry name>* password: this password is used to connect to your container image registry (as specified in the `runme-input.yaml` file) for downloading images.

4. The script lists the `Istio-ingressgateway` URL as part of the output. Record the IP address listed.
5. The script installs the following services under the `obp-cp` namespace:
   - `control-plane`
   - `openldap`
   - `obp-auth-server`
   - `obp-operator`
   - `hlf-operator`

Access Blockchain Platform Manager

After installation, configure the host name resolution for the generated Blockchain Platform Manager host names.

1. Run the following command to get the list of configured host names:

   ```
   kubectl get virtualservice -n obp-cp -o json | jq -r
   '.items[].spec.hosts[0]'
   ```

2. Based on the chosen service type for `istio-ingressgateway`, these generated host names must be resolved to an IPv4 address according to the following requirements:

   - `LoadBalancer`: Resolve to the external IP address generated for the `istio-ingressgateway` service. Run the following command to get the IP address.

     ```
     kubectl get svc/istio-ingressgateway -n istio-system
     ```

     Add the following line to the `/etc/hosts` file (Linux, macOS) or `C:\Windows\system32\drivers\etc\hosts` file (Microsoft Windows) on the host used to connect to the Blockchain Platform Manager console.

     ```
     <public_svc_ip> controlplane.<parentDomainName>
     openldap.<parentDomainName> auth.<parentDomainName>
     ```

   - `NodePort`: Resolve to the worker nodes' IP addresses.

Deploy Oracle Blockchain Platform Enterprise Edition on minikube

You can use minikube for testing and internal development purposes. Do not use minikube for production environments.

Prerequisites:

- CPUs: 8 or higher
- Memory: 16 GB
- Free disk space: 50 GB or higher
- A non-root user with superuser privileges
- Ensure that the minikube node has access to the internet, which is required to install chaincodes on your Oracle Blockchain Platform instances.
- minikube v1.33.1 or later

Install minikube

This section provides an example walkthrough of installing the tested version of the minikube. For additional information, see Kubernetes Install Tools.

- Oracle Linux

- macOS

Oracle Linux

```
#Download the latest minikube binary
curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-
linux-amd64

#Install
sudo install minikube-linux-amd64 /usr/local/bin/minikube

#Verify Installation
minikube version

#Start Minikube
minikube start

#Verify Minikube Installation
minikube status
```

macOS

```
#Install Homebrew (if not already installed)
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/
HEAD/install.sh)"

#Install minikube using Homebrew
brew install minikube

#Start minikube
minikube start

#Verify minikube Installation
minikube status
```

Complete the Istio Install

Oracle Blockchain Platform Enterprise Edition supports version 1.20.2 and later. You must complete the steps in Install Istio before running the following commands.

```
# Install
    istioctl install --set profile=default --set
values.pilot.env.ENABLE_TLS_ON_SIDECAR_INGRESS=true --set
components.cni.enabled=true --set values.cni.repair.deletePods="true"
    ## Enter "y" when prompted for "Proceed? (y/N)"

# Verify:
    $ istioctl version
    client version: 1.22.1
    control plane version: 1.22.1
    data plane version: 1.22.1 (1 proxies)
```

Install Oracle Blockchain Platform Enterprise Edition

Download and install Oracle Blockchain Platform Enterprise Edition on your minikube instance:

1. On the Oracle Blockchain Platform Enterprise Edition page, click **Download** and follow the steps to download the Oracle Blockchain Platform Enterprise Edition package. Extract the package from the `.zip` file, and then extract the package from the downloaded archive file.

   ```
   tar -xzf <distribution-package-file>
   # example tar -xzf obpee_package_24.1.3-20240723083137.tgz
   ```

2. Update the `runme-input.yaml` file with the required values. The following example `runme-input.yaml` file can be used as reference:

   ```
   imageRegistryConfiguration:
     registry: <container_registry_name>
     imageTagPrefix: <container-image-repository-prefix>
     username: <container-registry-username>

   imageReleaseVersion: 24.1.3-20240723083137

   # storageClassName should be set to create a dynamic persistent volume. If
   empty, default storageClass is used.

   controlPlaneStorage:
     storageClassName:
     # Example 500Mi, 5Gi
     size: 4Gi

   parentDomainName: example.com
   #imagePullTimeout: Use this field to customize the wait time (in seconds)
   for pulling the required container images from the repository. Default is
   1800 seconds.
       imagePullTimeout: 1800
   ```

   In the previous example, the variables are defined as shown in the following list:

   - **`imageRegistryConfiguration.registry`**: Container registry server to use. Example: `iad.ocir.io`
   - **`imageRegistryConfiguration.imageTagPrefix`**: Container base repository path with the registry, where the images will be pushed to and pulled from. Example: `iad.ocir.io/obpee/bcs`
   - **`imageRegistryConfiguration.username`**: Container registry login user name
   - **`imageReleaseVersion`**: Oracle Blockchain Platform Enterprise Edition release version
   - **`controlPlaneStorage.storageClassName`**: Kubernetes storage class to use for PVC (PersistentVolumeClaim). If empty, the default `storageClass` is used
   - **`controlPlaneStorage.size`**: PVC size for Blockchain Platform Manager (control plane) services
   - **`parentDomainName`**: Domain name to use for Blockchain Platform Manager services. Example: `example.com`

- **`imagePullTimeout`**: Image pull wait timeout in seconds during Oracle Blockchain Platform Enterprise Edition installation. Default is 1800 seconds.

3. Ensure that minikube is running.
4. Open a new terminal window and go to the distribution package directory. Run the `runme_minikube.sh` script and follow the steps as prompted by the script output:

   ```
   ./runme_minikube.sh [--publish-images]
   ```

   > **Note**
   >
   > The optional `--publish-images` command uploads the containers to a container image registry such as Oracle Cloud Infrastructure Registry.

   - Enter the default LDAP admin password (the password will not be displayed): sets the administrator's password for the built-in LDAP authentication server.
   - Enter the default control plane admin user password (the password will not be displayed): sets the Blockchain Platform Manager administrator's password.
   - If `StorageClass` wasn't provided in the `runme-input.yaml` file, the system checks if the default storage class is set and asks if you want to use it.
   - Confirm the Istio ingress gateway service https port: the default is 443 for the `LoadBalancer` service type.
   - Enter registry <*registry name*> password: this password is used to connect to your container image registry (as specified in the `runme-input.yaml` file) for downloading images.
5. In a different terminal window, run the following command:

   ```
   export KUBECONFIG=/<path_to>/.kube/minikube
   ```

6. Ensure that the minikube tunnel is active for accessing Blockchain Platform Manager and instances:

   ```
   minikube tunnel --bind-address 0.0.0.0
   ```

7. The script installs the following services under the `obp-cp` namespace:
   - `control-plane`
   - `openldap`
   - `obp-auth-server`
   - `obp-operator`
   - `hlf-operator`
8. The script displays the Blockchain Platform Manager URL, which you can use to access the control plane UI.

Access Blockchain Platform Manager

After installation, configure the host name resolution for the generated Blockchain Platform Manager host names.

1. Run the following command to get the list of configured host names:

```
kubectl get virtualservice -n obp-cp -o json | jq -r
'.items[].spec.hosts[0]'
```

2. Use the IPv4 address of the minikube host as the mapping IP address for the generated host names.
3. Ensure the minikube tunnel is active to access Blockchain Platform Manager and Oracle Blockchain Platform instances.

Deploy Oracle Blockchain Platform Enterprise Edition on Red Hat OpenShift Local

You can install Oracle Blockchain Platform Enterprise Edition on Red Hat OpenShift Local for testing and internal development purposes. It is not supported for production environments.

Red Hat OpenShift Local is designed to run on a local computer to simplify setup and testing, and to emulate the cloud development environment locally with all of the tools needed to develop container-based applications. It was previously known as Red Hat CodeReady Containers.
For detailed information about Red Hat OpenShift Local, see Red Hat OpenShift Local.

Prerequisites:

- CPUs: 12 or more
- Memory: 30GB or higher
- Disk size: 150GB or higher
- Operating system: CentOS 8

This section provides an example walkthrough of installing the tested versions of the prerequisites. Refer to each product's documentation for additional information and any required modifications to the installation instructions. The following walkthrough was tested using CentOS 8 as the operating system. Other distributions of Linux, such as Oracle Linux or Red Hat Enterprise Linux, can also be used.

> **Note**
>
> When you install Istio as part of the Install Prerequisite Software tasks, install version 1.22.1, not the latest version.

Install Red Hat OpenShift Local

Complete the following steps to download and install Red Hat OpenShift Local. For additional information, see Installing CodeReady Containers.

1. Navigate to Red Hat OpenShift Local and click **Install OpenShift on your laptop**. You are redirected to a login page where you can enter your Red Hat credentials or create an account if you do not already have one.
2. After you log in, click **Clusters** and then click the **Local** tab.
3. On the Local page, click **Download OpenShift Local** and then **Download pull secret**.

4. Copy the `crc` package and the pull secret that you downloaded in the previous step to your VM. For example:

   ```
   scp -r -i ~/.ssh/id_rsa ~/Downloads/crc-linux-amd64.tar.xz <username>@<ip-address>:/tmp
   scp -r -i ~/.ssh/id_rsa ~/Downloads/pull-secret.txt <username>@<ip-address>:/tmp
   cd
   ```

5. On your VM, create a directory called `crc` and copy the `crc` package and the pull secret to that directory.
6. Install OpenShift by running the following command:

   ```
   sudo dnf install NetworkManager
   ```

   If an error starting with `Failed loading plugin "osmsplugin": No module named 'librepo'` is displayed, run the following commands and then run the installation command again.

   ```
   sudo sed -i 's/mirrorlist/#mirrorlist/g' /etc/yum.repos.d/CentOS-*
   sudo sed -i 's|#baseurl=http://mirror.centos.org|baseurl=http://vault.centos.org|g' /etc/yum.repos.d/CentOS-*
   ```

7. Run the following commands to extract the archive file:

   ```
   cd ~/crc
   tar xvf crc-linux-amd64.tar.xz
   ```

8. Run the following commands to move the binary file to the `/bin` directory and to update the path:

   ```
   mkdir -p ~/bin
   cp ~/crc/crc-linux-*-amd64/crc ~/bin
   export PATH=$PATH:$HOME/bin
   echo 'export PATH=$PATH:$HOME/bin' >> ~/.bashrc
   ```

Increase Disk Space in the Root Partition

If the VM root partition space is less than 150 GB, complete the following steps.

1. Ensure that the boot disk, root file system, or logical volume manager (LVM) has at least 150 GB free space.
2. Use `fdisk` to create a Linux file system partition of at least 100GB (in this example, the new partition is `/dev/sda4`) .

   ```
   sudo fdisk /dev/sda
   ```

3. Use the following command to add the physical volume to a volume group.

   ```
   sudo vgextend centosvolume /dev/sda4
   ```

4. Use the following command to increase the size of the logical volume.

   ```
   sudo lvextend -L+99G /dev/mapper/centosvolume-root
   ```

5. Use the following command to increase the size of the filesystem.

   ```
   sudo xfs_growfs /dev/centosvolume/root
   ```

   The equivalent command on Oracle Enterprise Linux is `oci_growfs`.

Configure OpenShift Local Parameters

Run the following commands to configure OpenShift Local to use 12 CPU cores, 30 GB memory, and 100 GB disk space.

```
crc config set cpus 12
crc config set memory 30720
crc config set disk-size 100
```

Download and Install the OpenShift Client

Run the following commands to download and install `oc`, the OpenShift client software.

```
wget -O ~/crc/openshift-client-linux.tar.gz https://mirror.openshift.com/pub/
openshift-v4/x86_64/clients/ocp/4.15.3/openshift-client-linux-4.15.3.tar.gz
tar xvzf openshift-client-linux.tar.gz
sudo mv oc /usr/local/bin
```

Start the Cluster

Run the following commands to set up, start, and check the status of the cluster. Use the pull secret that you downloaded when you downloaded Red Hat OpenShift Local.

```
crc setup
crc start -p ~/crc/pull-secret.txt
crc status
```

Once the installation is complete, information similar to the following will display:

```
INFO Adding crc-admin and crc-developer contexts to kubeconfig...
Started the OpenShift cluster.

The server is accessible via web console at:
  https://console-openshift-console.apps-crc.testing

Log in as administrator:
  Username: kubeadmin
  Password: password  (note this password)

Log in as user:
  Username: developer
  Password: developer

Use the 'oc' command line interface:
```

```
$ eval $(crc oc-env)
$ oc login -u developer https://api.crc.testing:6443
```

Install Oracle Blockchain Platform Enterprise Edition

1. Complete the following steps to ensure Red Hat OpenShift Local is up and running.
   a. Log in to the console.

      ```
      oc login -u kubeadmin -p <password> https://api.crc.testing:6443
      ```

   b. Verify that Red Hat OpenShift Local is reachable.

      ```
      oc get nodes
      ```

2. Run the following command to set the Istio profile. You must have completed the steps in Install Istio before running the following commands.

   ```
   istioctl install --set profile=openshift --set
   values.pilot.env.ENABLE_TLS_ON_SIDECAR_INGRESS=true --set
   components.cni.enabled=true --set values.cni.repair.deletePods="true"
   ```

3. On the Oracle Blockchain Platform Enterprise Edition page, click **Download** and then follow the steps to download the Oracle Blockchain Platform Enterprise Edition package, which is approximately 6.5 GB.
4. Extract the downloaded archive file.

   The extracted folder structure includes `runme` scripts for various platforms, including OpenShift.
5. Update the `runme-input.yaml` file with the required values. Also, make the `runme-input.yaml` and `runme.sh` files executable. Ensure that you can log in to the registry from the user account that you specify in the `runme-input.yaml` file. The following example `runme-input.yaml` file can be used as a reference:

   ```
   imageRegistryConfiguration:
     registry: <container_registry_name>
     imageTagPrefix: <container-image-repository-prefix>
     username: <container-registry-username>

   imageReleaseVersion: <obpee-release-version>

   # Set storageClassName to create a dynamic persistent volume. If empty,
   default storageClass is used.

   controlPlaneStorage:
     storageClassName:
     # Example 500Mi, 5Gi
     size: 1Gi
   parentDomainName: example.com
   #imagePullTimeout: Use this field to customize the wait time (in seconds)
   for pulling the required container images from the repository. Default is
   1800 seconds.
   imagePullTimeout: 1800
   ```

   The variables in the example have the following values:

- `imageRegistryConfiguration.registry` is the container registry server to use. Example: `iad.ocir.io`
- `imageRegistryConfiguration.imageTagPrefix` is the container base repository path with the registry, where the images will be pushed to or pulled from. Example: `iad.ocir.io/obpee/bcs`
- `imageRegistryConfiguration.username` is the container registry login user name.
- `imageReleaseVersion` is the Oracle Blockchain Platform Enterprise Edition release version.
- `controlPlaneStorage.storageClassName` is the Kubernetes storage class to use for PVC (PersistentVolumeClaim). If empty, the default `storageClass` is used.
- `controlPlaneStorage.size` is the PVC size for Blockchain Platform Manager (control plane) services.
- `parentDomainName` is the domain name to use for Blockchain Platform Manager services. Example: `example.com`.
- `imagePullTimeout` is the image pull wait timeout in seconds during Oracle Blockchain Platform Enterprise Edition installation. The default is 1800 seconds.

6. Open a new terminal window and go to the distribution package directory. Follow the steps as prompted by the script output.
   a. Run the following command to make the script executable.

      ```
      chmod +x runme_openshift.sh
      ```

   b. Run the following script to push the containers in the archive file to the specified repository and then install the product.

      ```
      ./runme_openshift.sh --publish-images
      ```

      If the container images are already uploaded to a repository, you can pull them from the repository and install by using the following command.

      ```
      ./runme_openshift.sh
      ```

   - Enter the default LDAP admin password (the password will not be displayed): this is used to set the admin user's password for the built-in LDAP authentication server.
   - Enter the default control plane admin user password (the password will not be displayed): this is used to set the Blockchain Platform Manager admin user's password.
   - Enter registry <*registry name*> password: This is used to connect to your container image registry (as specified in `runme-input.yaml`) for downloading images.

   The script installs the following services under the `obp-cp` namespace:

   - `control-plane`
   - `openldap`
   - `obp-auth-server`
   - `obp-operator`
   - `hlf-operator`

7. Add the following line to the `/etc/hosts` file on the `crc` VM.

   ```
   <CRC_IP_address> controlplane.<parentDomainName>
   openldap.<parentDomainName> auth.<parentDomainName>
   ```

   In the example, *<CRC_IP_address>* is the output of the `crc ip` command.

8. Add the following line to the `/etc/hosts` file on the computer that you use to connect to the Blockchain Platform Manager console.

   ```
   <public_vm_ip> controlplane.<parentDomainName> openldap.<parentDomainName>
   auth.<parentDomainName>
   ```

   In the example, *<public_vm_ip>* is the public IP address of the VM.

9. Complete the following steps to access Blockchain Platform Manager from a client computer. The following steps use `firewalld` to allow inbound traffic to the server and `HAProxy` to forward the traffic to the OpenShift Local instance.

   a. Ensure that the following entry is in the `/etc/NetworkManager/conf.d/crc-nm-dnsmasq.conf` configuration file.

      ```
      [main]
              dns=dnsmasq
      ```

      The `dnsmasq` instance configuration file, `/etc/NetworkManager/dnsmasq.d/crc.conf`, might look similar to the following:

      ```
      server=/crc.testing/198.x.x.x
      server=/apps-crc.testing/198.x.x.x
      ```

      198.x.x.x is the output of the `crc ip` command. The `dnsmasq` instance of `NetworkManager` forwards requests for `crc.testing` and `apps-crc.testing` to the *198.x.x.x* DNS server.

   b. Enter the following command to install required dependencies.

      ```
      sudo dnf -y install haproxy policycoreutils-python-utils
      ```

   c. Enter the following commands to configure the firewall.

      ```
      sudo systemctl start firewalld
      sudo firewall-cmd --add-port=80/tcp --permanent
      sudo firewall-cmd --add-port=6443/tcp --permanent
      sudo firewall-cmd --add-port=443/tcp --permanent
      sudo systemctl restart firewalld
      sudo semanage port -a -t http_port_t -p tcp 6443
      sudo semanage port -a -t http_port_t -p tcp 6443
      ```

d. Before you can configure HAProxy, you must know the IP address of the server and the IP address of the Red Hat OpenShift Local VM. Run the following commands.

```
export SERVER_IP=$(hostname --ip-address)
export CRC_IP=$(crc ip)
cd /etc/haproxy;sudo cp haproxy.cfg haproxy.cfg.orig
```

e. Replace the contents of the `haproxy.cfg` file with the following text.

```
global
debug

defaults
log global
mode http
timeout connect 0
timeout client 0
timeout server 0

frontend apps
bind SERVER_IP:80
bind SERVER_IP:443
option tcplog
mode tcp
default_backend apps

backend apps
mode tcp
balance roundrobin
option ssl-hello-chk
server webserver1 CRC_IP:80 check
server webserver2 CRC_IP:443 check

frontend api
bind SERVER_IP:6443
option tcplog
mode tcp
default_backend api

backend api
mode tcp
balance roundrobin
option ssl-hello-chk
server webserver1 CRC_IP:6443 check
```

f. Run the following commands to replace the IP addresses in the `HAproxy` configuration, and then start `HAproxy`.

```
export CRC_IP=$(crc ip)
export SERVER_IP=$(hostname --ip-address)
sudo sed -i "s/CRC_IP/$CRC_IP/g" haproxy.cfg
sudo sed -i "s/SERVER_IP/$SERVER_IP/g" haproxy.cfg
sudo systemctl start haproxy
```

Access Blockchain Platform Manager

After installation, configure the host name resolution for the generated Blockchain Platform Manager host names.

1. Run the following command to get the list of configured hostnames:

   ```
   kubectl get virtualservice -n obp-cp -o json | jq -r .items[].spec.hosts[0]
   ```

2. Configure host name resolution for these generated host names to the IP address of the running Red Hat OpenShift cluster by adding the following entry to the `/etc/hosts` file on the VM.

   `<IP_Address>` `controlplane.<parentDomainName>` `openldap.<parentDomainName>` `auth.<parentDomainName>`

   In the example, `<IP_address>` is the output of the `crc ip` command.

3. Ensure that the `crc` VM security list allows ports 443, 6443, and 80 from the client host computer.

You can now log on to Blockchain Platform Manager (the control plane management tool) to create an instance.

Once you've created your instance, you must configure DNS forwarding as described in: Provisioning Postrequisites.

Deploy Oracle Blockchain Platform Enterprise Edition on Azure Red Hat OpenShift

You can install Oracle Blockchain Platform Enterprise Edition on Azure Red Hat OpenShift. OpenShift is Red Hat's enterprise Kubernetes platform that enhances standard Kubernetes with a suite of tools for building, deploying, and managing containerized applications at scale. Azure Red Hat OpenShift is a fully managed OpenShift service on Microsoft Azure.

The instructions in this topic are a guideline suggesting how you might deploy Oracle Blockchain Platform Enterprise Edition on Azure Red Hat OpenShift. Before actually attempting this, particularly for production environments, you should familiarize yourself with the Azure Red Hat OpenShift documentation which supersedes any information in this topic.

- Azure Red Hat OpenShift: About Azure Red Hat OpenShift
- Azure Red Hat OpenShift documentation: Azure Red Hat OpenShift documentation

Create a Red Hat OpenShift Cluster on Microsoft Azure

Recommended minimum specifications for your Azure Red Hat OpenShift Cluster.

Create a Azure Red Hat OpenShift cluster using the Azure documentation as a guide: Create an Azure Red Hat OpenShift cluster

You'll need to create the following:

- Control plane VMs: three Standard_D8s_v3
- Worker node VMs: four Standard_D4s_v3
- Generate a Red Hat pull secret.

Install Oracle Blockchain Platform Enterprise Edition

The images needed for Oracle Blockchain Platform deployment are stored in a Docker repository before being pushed to the Microsoft Azure VMs. We recommend installing from a Cloud Linux VM that's as geographically as close as possible to the region where your software will be installed. Local storage for the Docker images should be approximately 100GB.

1. Open your OpenShift cluster console.
2. Select `kube:admin` then `Copy login command`. You'll be prompted for your user ID and password.
   Click **Display Token**. Copy the **Log in with this token** command and paste it in the OpenShift command-line interface.

   ```
   $ oc login --token=<token> --server=<openshift_cluster_url>
   ```

3. Verfiy that you're connected to OpenShift.

   ```
   oc get nodes
   ```

   This command should return the list of master and worker nodes and their status.
4. Set the istio profile. You must have completed the steps in Install Istio before running the following commands.

   ```
   istioctl install --set profile=openshift --set
   values.pilot.env.ENABLE_TLS_ON_SIDECAR_INGRESS=true --set
   components.cni.enabled=true --set
   values.cni.repair.deletePods="true"
   ```

5. Download Oracle Blockchain Platform: Oracle Blockchain Platform.
6. Extract the zip package and find the `runme` script for OpenShift.
7. Update the `runme-input.yaml` file with the required values. The following example `runme-input.yaml` file can be used as reference:

   ```
   imageRegistryConfiguration:
     registry: <container_registry_name.azurecr.io>
     imageTagPrefix: <container-image-repository-prefix.azurecr.io/bcs-k8s-
   dev  >
     username: <container-registry-username>

   imageReleaseVersion: 24.1.3-20240723083137

   # storageClassName should be set to create a dynamic persistent volume. If
   empty, default storageClass is used.

   controlPlaneStorage:
     storageClassName:
     # Example 500Mi, 5Gi
     size: 10Gi

   parentDomainName: example.com
   #imagePullTimeout: Use this field to customize the wait time (in seconds)
   ```

```
for pulling the required container images from the repository. Default is
1800 seconds.
  imagePullTimeout: 1800
```

In the previous example, the variables are defined as shown in the following list:

- **`imageRegistryConfiguration.registry`**: Container registry server to use.
- **`imageRegistryConfiguration.imageTagPrefix`**: Container base repository path with the registry, where the images will be pushed to and pulled from. Example: `registryname.azurecr.io/obpee/bcs`
- **`imageRegistryConfiguration.username`**: Container registry login user name. For Azure it is the container registry token.
- **`imageReleaseVersion`** - Oracle Blockchain Platform Enterprise Edition release version
- **`controlPlaneStorage.storageClassName`**: Kubernetes storage class to use for PVC (PersistentVolumeClaim). If empty, the default `storageClass` is used
- **`controlPlaneStorage.size`**: PVC size for Blockchain Platform Manager (control plane) services
- **`parentDomainName`**: Domain name to use for Blockchain Platform Manager services. Example: `example.com`
- **`imagePullTimeout`**: Image pull wait timeout in seconds during Oracle Blockchain Platform Enterprise Edition installation. Default is 1800 seconds.

8. Open `runme_openshift.sh` in an editor, and comment out the following two lines:
   - `openshift_status_check`
   - `check_software_exists "crc"`
9. Run `runme_openshift.sh` and complete the steps as prompted:

```
chmod +x runme_openshift.sh

./runme_openshift.sh --publish-images
# The publish-images option uploads the containers to the container image
registry specified in runme-input.yaml
# Running the script without this option pulls images you've already pre-
uploaded from the repository specified in runme-input.yaml
```

The prompts you'll encounter:

- Enter OpenLDAP admin password - This will be used by Blockchain Platform Manager and users will be created on this OpenLDAP server.
- Enter Control Plane Admin password - This password will be for the Oracle Blockchain Platform admin user and will be used for first time log in into the Blockhain Platform Manager console.
- Enter the registry login password for the user specified in `run-input.yaml` - This is used to connect to the container repository.
- The script installs the following services under the `obp-cp` namespace:
  - `control-plane`
  - `openldap`
  - `obp-auth-server`

- `obp-operator`
- `hlf-operator`

10. Get the Istio ingress gateway service's external IP address:

```
kubectl get svc/istio-ingressgateway -n istio-system
```

11. Add the following line as required to one of these files on the host used to connect to the Blockchain Platform Manager console:
    - Linux or macOS: `/etc/hosts`
    - Windows: `C:\Windows\system32\drivers\etc\hosts`

```
<public_svc_ip> controlplane.<parentDomainName>
openldap.<parentDomainName> auth.<parentDomainName>
```

where `<public_svc_ip>` is the public, external IP address from the previous step.

Postrequisites

You're now ready to log on to the Oracle Blockchain Platform for the first time and provision an instance.

Deploy Oracle Blockchain Platform Enterprise Edition on a Red Hat OpenShift On-Premises Cluster

Before deploying Oracle Blockchain Platform Enterprise Edition, you must have a Red Hat OpenShift Enterprise cluster running and you must install several prerequisites.

The instructions in this topic are a guideline suggesting how you might deploy Oracle Blockchain Platform Enterprise Edition on Red Hat OpenShift on premises. Before actually attempting this, particularly for production environments, you should familiarize yourself with the Red Hat OpenShift documentation which supersedes any information in this topic.

- Red Hat OpenShift: OpenShift Container Platform 4.17 Documentation
- Red Hat OpenShift on-premises installation documentation: Installing an on-premise cluster

Create a Red Hat OpenShift Cluster

Your cluster must meet the requirements outlined in the Red Hat OpenShift documentation outlined here: OpenShift Container Platform Prerequisites

Recommended minimum specifications for your cluster worker nodes:

	Production with High Availability
Node CPU	4 CPUs or higher
Node Memory	32 GB or higher
Node Count	3 or higher
Boot volume size	100 GB or higher

Install the OpenShift CLI (oc) so that you can access your cluster. See OpenShift CLI.

Verfiy that you're connected to OpenShift.

```
oc get nodes
```

This command should return the list of master and worker nodes and their status.

Install Oracle Blockchain Platform Enterprise Edition

1. Set the istio profile. You must have completed the steps in Install Istio before running the following commands.

   ```
   istioctl install --set profile=openshift --set
   values.pilot.env.ENABLE_TLS_ON_SIDECAR_INGRESS=true --set
   components.cni.enabled=true --set
   values.cni.repair.deletePods="true"
   ```

2. Download Oracle Blockchain Platform: Oracle Blockchain Platform.
3. Extract the zip package and find the `runme` script for OpenShift.
4. Update the `runme-input.yaml` file with the required values. The following example `runme-input.yaml` file can be used as reference:

   ```
   imageRegistryConfiguration:
     registry: <container_registry_name>
     imageTagPrefix: <container-image-repository-prefix>
     username: <container-registry-username>

   imageReleaseVersion: 24.1.3-20240723083137

   controlPlaneStorage:
     storageClassName:
     size: 10Gi

   parentDomainName: example.com
   #imagePullTimeout: Use this field to customize the wait time (in seconds)
   for pulling the required container images from the repository. Default is
   1800 seconds.
     imagePullTimeout: 1800
   ```

   > **Note**
   >
   > We recommend setting up the container registry in your environment so that the image pulls are faster. If you are using your own container registry, use the `imageReleaseVersion` value of `24.1.3-20240723083137`.

   In the previous example, the variables are defined as shown in the following list:

   - **`imageRegistryConfiguration.registry`**: Container registry server to use.
   - **`imageRegistryConfiguration.imageTagPrefix`**: Container base repository path with the registry, where the images will be pushed to and pulled from. Example: `registryname.rhoscr.io/obpee/bcs`
   - **`imageRegistryConfiguration.username`**: Container registry login user name.

- **`imageReleaseVersion`** - Oracle Blockchain Platform Enterprise Edition release version
- **`controlPlaneStorage.storageClassName`**: Kubernetes storage class to use for PVC (PersistentVolumeClaim). If empty, the default `storageClass` is used
- **`controlPlaneStorage.size`**: PVC size for Blockchain Platform Manager (control plane) services
- **`parentDomainName`**: Domain name to use for Blockchain Platform Manager services. Example: `example.com`. Blockchain Platform Manager and service console URLs will contain this domain name.
- **`imagePullTimeout`**: Image pull wait timeout in seconds during Oracle Blockchain Platform Enterprise Edition installation. Default is 1800 seconds.

5. Open `runme_openshift.sh` in an editor, and comment out the following two lines:
   - `openshift_status_check`
   - `check_software_exists "crc"`
6. Run `runme_openshift.sh` and complete the steps as prompted:

   ```
   chmod +x runme_openshift.sh

   ./runme_openshift.sh --publish-images
   # The publish-images option uploads the containers to the container image
   registry specified in runme-input.yaml
   # Running the script without this option pulls images you've already pre-
   uploaded from the repository specified in runme-input.yaml
   ```

   The prompts you'll encounter:
   - Enter OpenLDAP admin password - This will be used by Blockchain Platform Manager and users will be created on this OpenLDAP server.
   - Enter Control Plane Admin password - This password will be for the Oracle Blockchain Platform admin user and will be used for first time log in into the Blockhain Platform Manager console.
   - Enter the registry login password for the user specified in `run-input.yaml` - This is used to connect to the container repository.
   - The script installs the following services under the `obp-cp` namespace:
     - `control-plane`
     - `openldap`
     - `obp-auth-server`
     - `obp-operator`
     - `hlf-operator`
7. Get the Istio ingress gateway service's external IP address:

   ```
   kubectl get svc/istio-ingressgateway -n istio-system
   ```

8. Add the following line as required to one of these files on the host used to connect to the Blockchain Platform Manager console:
   - Linux or macOS: `/etc/hosts`

- Windows: `C:\Windows\system32\drivers\etc\hosts`

```
<public_svc_ip> controlplane.<parentDomainName>
openldap.<parentDomainName> auth.<parentDomainName>
```

where *<public_svc_ip>* is the public, external IP address from the previous step.

Postrequisites

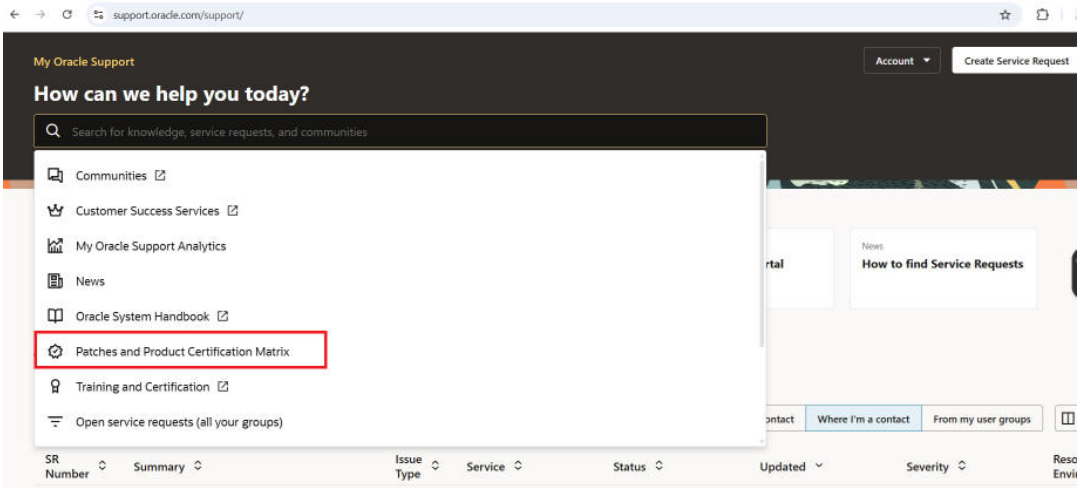
You're now ready to log on to the Oracle Blockchain Platform for the first time and provision an instance.

Patch Oracle Blockchain Platform

After you've installed Oracle Blockchain Platform Enterprise Edition you can find patches for it on the Oracle Support site.

Patch set 5 is available for Oracle Blockchain Platform Enterprise Edition. The change log for the patch as well as prerequisites and instructions are included in the readme file that comes with the patch.

1. Go to the My Oracle Support portal https://support.oracle.com/portal/, and log in.
2. Click the Search bar and select **Patches and Product Certification Matrix** from the drop down list.



3. Search for the patch by name or number (38964498).
4. Click the patch name link to see the patch details. Download and extract the patch.
5. Follow the instructions in the readme file to install the patch.
6. After the patch is applied, restart any existing Oracle Blockchain Platform instances. Because the container registry images are updated by the patch, any instances created after patching will use the updated images.

Roll Back Patches

You can also roll back the patch using the same patch script and selecting the rollback option.

The rollback can be performed only if no lifecycle operations have been done from Blockchain Platform Manager after the patch application. If you have completed lifecycle operations after installing the patch and need to roll back, please contact Oracle Support.

Critical Patches

Critical patches and security updates are listed on the Oracle Critical Patch Updates, Security Alerts and Bulletins website: https://www.oracle.com/security-alerts/

Log on to Oracle Blockchain Platform for the First Time

After you've deployed and started Oracle Blockchain Platform Enterprise Edition , you can log on to Blockchain Platform Manager (the control plane management tool) to create an instance.

You can directly log on to the Platform Manager by using the URL `https://controlplane.<parentDomainName>.com/console/index.html`. The URL for your instance is displayed at the end of successfully running the `runme` script for your install. The initial user name is `obpadmin` and the password configured during installation. This user is only meant for performing initial configuration and does not have instance creation privileges.

Enable the LDAP server

1. Log into Blockchain Platform Manager.
2. Go to **Configuration**, then **Authentication Servers**.
3. Click **Save and Set Active** to activate the default LDAP configuration.
4. Click **Test Configuration** to test your connectivity with the server.

Set the Blockchain Platform Manager Name

On the **Configuration** page **Platform Settings** tab of Blockchain Platform Manager, you can set a name for the Platform Manager.

> **Note**
>
> Once the name for the Platform Manager has been set, any users added to the LDAP server will be associated with this name. If you change the name after adding users, those users will lose access to Blockchain Platform Manager and any Oracle Blockchain Platform instances.

Set the Notification and Console Idle Timeouts

On the **Configuration** page **Platform Settings** tab of Blockchain Platform Manager, you can set the timeouts for notifications and the console.

- **Console Idle Timeout**: in minutes, how long the console can be idle before it logs out the current user.
- **Notification Timeout**: in seconds, how long notifications will remain visible on the browser. Select `-1` if you want notifications to remain visible until you close them.

4

User Management

This section describes how to configure the included default LDAP server or an external authentication server. It also describes the user groups and roles used by Oracle Blockchain Platform.

Configure the Built-In LDAP Server

The built-in LDAP server has a default configuration already set up when you log in. You can use it as-is, or modify the configuration to meet your needs.

1. Open the **Configuration** tab.
2. Edit the configuration information for the LDAP server as needed:
   a. Configuration Name:

      Not editable

   b. Authentication Server Type:

      Not editable

   c. Host:

      Not editable

   d. Port:

      Not editable

   e. TLS Enabled:

      Not editable

   f. Connect Timeout:

      In milliseconds.

   g. Base DN:

      Enter the base distinguished name of the directory you want to connect to. It should be in the form: `ou=organizationunit,dc=mycompany,dc=com`

   h. Bind User DN:

      The distinguished name of your administrative user account.

   i. Bind User Password:

      The password for the account.

   j. UserName Attribute:

      This is the filter used when searching to convert a login user name to a distinguished name.

   k. User Class Name:

      The attribute value to a user object in the directory.

   l. GroupName Attribute:

This is the filter used when searching to convert a group name to a distinguished name.

m. Group Membership Attribute:

The membership attribute name of the group.

n. Group Class Name:

The ObjectClass attribute value for a group object in the directory.

3. Click **Test Configuration** to ensure your settings work. The test results show if the configuration was successful.
4. Click **Save**. Your configuration is now available to be used by any instances you provision.

Add Users to Your LDAP Server Using Blockchain Platform Manager

Once you've configured your LDAP server in Blockchain Platform Manager, you need to add users to the LDAP server, and then log back into Blockchain Platform Manager with one of these users to create an instance.

Add your initial user to the LDAP server. On the **Authentication Servers** tab of the **Configuration** page of Blockchain Platform Manager, click **Add User**. Once you've entered the user name and password, this user will be added to the LDAP server as an administrative user. You can now log out of Blockchain Platform Manager with your default admin user, and log in with this newly added user to create an instance.

Once you've successfully logged into Blockchain Platform Manager with this user and provisioned an instance, you may want to disable the default admin user (`obpadmin`) for security reasons. This can be done from the **Configuration** page **Platform Settings** tab.

User Groups and Roles

This overview describes the groups and roles that are relevant to Oracle Blockchain Platform. Anyone who uses or administers Oracle Blockchain Platform must be added to the authentication server and granted the correct group.

Groups

Below are the group roles that are available for Oracle Blockchain Platform.

User Role	LDAP Group Name in LDAP	Description
Application	OBP_<platform-name>_<instance-name>	Security identifier for an individual instance.
Control Plane Management	OBP_<platform-name>_CP_ADMIN	User can provision a new Oracle Blockchain Platform instance, configure existing instances, set the LDAP configuration, and perform life cycle operations on Oracle Blockchain Platform instances. A user must be a member of this group to be able to log in to the Blockchain Platform Manager or create an instance.

User Role	LDAP Group Name in LDAP	Description
CA Administrator	OBP_<platform-name>_<instance-name>_CA_ADMIN	The CA Admin group is the bootstrap and overall administrator for the Oracle Blockchain Platform application. Users must be part of this group to create an instance.
Instance Administrator	OBP_<platform-name>_<instance-name>_ADMIN	Users in this group can manage instances via the console UI or REST. Users must be part of this group to create an instance. See the table in Access Control List for Console Function by User Roles for a complete list of console functions available for this user role.
Instance User	OBP_<platform-name>_<instance-name>_USER	Users in this group can view instance via console UI or REST See the table in Access Control List for Console Function by User Roles for a complete list of console functions available for this user role.
REST Proxy Client	OBP_<platform-name>_<instance-name>_REST	Users in this group can call REST proxy to execute transactions using the default enrollment.

Access Control List for Console Function by User Roles

The following table lists which console features are available to the Instance Administrator and Instance User roles.

Feature	Instance Administrator	Instance User
Dashboard	Yes	Yes
Network: list orgs	Yes	Yes
Network: add orgs	Yes	No
Network: Ordering service setting	Yes	No
Network: Export certificates	Yes	No
Network: Export orderer settings	Yes	Yes
Node: list	Yes	Yes
Node: start/stop/restart	Yes	No
Node: view attributes	Yes	Yes
Node: edit attributes	Yes	No
Node: view metrics	Yes	Yes
Node: Export/Import Peers	Yes	No
Peer Node: list channels	Yes	Yes
Peer Node: join channel	Yes	No
Peer Node: list chaincode	Yes	Yes
Channel: list	Yes	Yes
Channel: create	Yes	No
Channel: add org to channel	Yes	No

Feature	Instance Administrator	Instance User
Channel: Update ordering service settings	Yes	No
Channel: view/query ledger	Yes	Yes
Channel: list instantiated chaincode	Yes	Yes
Channel: list joined peers	Yes	Yes
Channel: set anchor peer	Yes	No
Channel: upgrade chaincode	Yes	No
Chaincode: list	Yes	Yes
Chaincode: install	Yes	No
Chaincode: instantiate	Yes	No
Sample chaincode: install	Yes	No
Sample chaincode: instantiate	Yes	No
Sample chaincode: invoke	Yes	Yes
CRL	Yes	No

5

Provision an Instance

This section describes how to provision your Oracle Blockchain Platform instance using Blockchain Platform Manager.

Before You Create an Oracle Blockchain Platform Instance

Before you provision Oracle Blockchain Platform, decide if a developer or enterprise instance meets your needs.

Deciding Which Provisioning Shape to Use

When provisioning an instance, you choose between two configurations. Migration between these options isn't supported currently.

Configuration	Features
Developer Recommended use for this starter shape is development and evaluation.	• 1 Fabric-CA node • 3-node Fabric Ordering Service Network • 1-node repository for instance metadata • Dynamically managed chaincode execution containers • Console service for operations web user interface • REST proxy service for RESTful API • LDAP server integration for authentication and role management
Enterprise Highly available instance configuration, with higher replica count for each service.	• 3 Fabric-CA nodes • 3-node Fabric Ordering Service Network • 3-node cluster repository for high availability of instance metadata • Dynamically managed chaincode execution containers • Console service for operations web user interface • Multiple replicas for REST proxy service for RESTful API • LDAP server integration for authentication and role management

Provision an Instance using the Blockchain Platform Manager

To create a blockchain founder or participant instance in Blockchain Platform Manager, use the Create New Instance wizard.

There are two types of Oracle Blockchain Platform instances you can provision:

- **Founder organization**: a complete blockchain environment, including a new network to which participants can join later on.
- **Participant instance**: if there is already a founder organization you want to join, you can create a participant instance if your credentials provide you with access to the network. Note that a participant cannot function on its own.

1. In Blockchain Platform Manager, open the **Instances** page.
2. Select **Create Instance**.
3. Complete the following fields:

Field	Description
Instance Name	Enter a name for your Oracle Blockchain Platform instance. The service instance name: • Must contain one or more characters. • Must not exceed 15 characters. • Must start with an ASCII letter: a to z. • Must contain only ASCII letters or numbers. • Must not contain a hyphen. • Must not contain any other special characters. • Must be unique within the identity domain.
Description	Optional. Enter a short description of the Oracle Blockchain Platform instance.
Domain Name	Enter the domain name for the cluster. The hostnames generated for the Blockchain Instance services make use of the domain name and the instance name as parent domain and sub domain respectively.
Role	Select **Founder** to create a complete blockchain environment. This instance becomes the founder organization and you can onboard new participants in the network later. Select **Participant** to create an instance that will join an existing blockchain network created elsewhere before this instance can be used.
Configuration	Select a provisioning shape which meets the needs of your deployment: • **Developer** • **Enterprise**
Peers	Specify the number of peer nodes to be initially created in this service instance. You can create additional peer nodes in the Oracle Blockchain Platform console at a later time.

4. Click **Create Instance**.

Provisioning Postrequisites

Oracle Linux

Before accessing the Oracle Blockchain Platform service console, configure the hostname resolution for the blockchain instance services, similar to what you did earlier for the Blockchain Platform Manager hostnames. Use the following command to get the list of hostnames for the created blockchain instance:

```
kubectl get virtualservice -n <instance-namespace> -o json | jq -r
      .items[].spec.hosts[0]
```

macOS

Before accessing the Oracle Blockchain Platform service console, configure the hostname resolution for the blockchain instance services, similar to what you did earlier for the Blockchain Platform Manager hostnames. Use the following command to get the list of hostnames for the created blockchain instance:

```
kubectl get virtualservice -n <instance-namespace> -o json | jq -r
      .items[].spec.hosts[0]
```

OpenShift Local

After you create an Oracle Blockchain Platform Enterprise Edition instance, you must configure DNS so that public host names of the components are resolvable from the OpenShift Local network.

After you create an instance, the instance name and parent domain are used as the subdomain for the Oracle Blockchain Platform Enterprise Edition components. You must set up DNS forwarding on the default DNS configuration for OpenShift, so that DNS requests are forwarded to the custom DNS pod where the exposed services are resolved. Complete the following steps to configure DNS forwarding.

1. Run the following command.

   ```
   oc new-project obp-coredns
   ```

2. Deploy a custom `coredns` server in the `obp-coredns` namespace by running the following command.

   ```
   kubectl apply -f <coredns-deployment-yaml-file>
   ```

   Use the following manifest file with the command.

   ```
   apiVersion: v1
   kind: ConfigMap
   metadata:
     name: coredns
     namespace: obp-coredns
   data:
     Corefile: |-
       .:1053 {
         errors
         health {
           lameduck 5s
         }
         ready
         kubernetes cluster.local in-addr.arpa ip6.arpa {
           pods insecure
           fallthrough in-addr.arpa ip6.arpa
         }
         prometheus :9153
         forward . /etc/resolv.conf
         cache 30
         loop
         reload
         loadbalance
   ```

```
    }
    import custom/*.server

---
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: obp-coredns
  name: obp-coredns
  namespace: obp-coredns
spec:
  progressDeadlineSeconds: 600
  replicas: 1
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      app: obp-coredns
  strategy:
    type: RollingUpdate
  template:
    metadata:
      labels:
        app: obp-coredns
    spec:
      containers:
      - args:
        - -conf
        - /etc/coredns/Corefile
        image: docker.io/coredns/coredns:latest
        imagePullPolicy: IfNotPresent
        livenessProbe:
          failureThreshold: 5
          httpGet:
            path: /health
            port: 8080
            scheme: HTTP
          initialDelaySeconds: 60
          periodSeconds: 10
          successThreshold: 1
          timeoutSeconds: 5
        name: coredns
        ports:
        - containerPort: 1053
          name: dns
          protocol: UDP
        - containerPort: 1053
          name: dns-tcp
          protocol: TCP
        - containerPort: 9153
          name: metrics
          protocol: TCP
        readinessProbe:
          failureThreshold: 3
          httpGet:
            path: /ready
```

```
        port: 8181
        scheme: HTTP
      periodSeconds: 10
      successThreshold: 1
      timeoutSeconds: 1
    resources:
      limits:
        memory: 170Mi
      requests:
        cpu: 100m
        memory: 70Mi
    securityContext:
      allowPrivilegeEscalation: false
      capabilities:
        add:
        - NET_BIND_SERVICE
        drop:
        - all
      readOnlyRootFilesystem: true
    terminationMessagePath: /dev/termination-log
    terminationMessagePolicy: File
    volumeMounts:
    - mountPath: /etc/coredns
      name: config-volume
      readOnly: true
    - mountPath: /etc/coredns/custom
      name: custom-config-volume
      readOnly: true
  dnsPolicy: Default
  restartPolicy: Always
  schedulerName: default-scheduler
  securityContext: {}
  serviceAccount: obp-coredns
  serviceAccountName: obp-coredns
  terminationGracePeriodSeconds: 30
  volumes:
  - configMap:
      defaultMode: 420
      items:
      - key: Corefile
        path: Corefile
      name: coredns
    name: config-volume
  - configMap:
      defaultMode: 420
      name: coredns-custom
      optional: true
    name: custom-config-volume

---
apiVersion: v1
kind: Service
metadata:
  labels:
    app: obp-coredns
  name: obp-coredns
```

```
    namespace: obp-coredns
spec:
  ports:
  - name: dns
    port: 53
    protocol: UDP
    targetPort: 1053
  - name: dns-tcp
    port: 53
    protocol: TCP
    targetPort: 1053
  - name: metrics
    port: 9153
    protocol: TCP
    targetPort: 9153
  selector:
    app: obp-coredns
  sessionAffinity: None
  type: ClusterIP
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: obp-coredns
  namespace: obp-coredns
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: obp-coredns
rules:
- apiGroups:
  - ""
  resources:
  - endpoints
  - services
  - pods
  - namespaces
  verbs:
  - list
  - watch
- apiGroups:
  - ""
  resources:
  - nodes
  verbs:
  - get
- apiGroups:
  - discovery.k8s.io
  resources:
  - endpointslices
  verbs:
  - list
  - watch
---
apiVersion: rbac.authorization.k8s.io/v1
```

```
kind: ClusterRoleBinding
metadata:
  name: obp-coredns
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: obp-coredns
subjects:
- kind: ServiceAccount
  name: obp-coredns
  namespace: obp-coredns
```

3. Create a `coredns-custom-configmap` file in YAML format for your blockchain instance.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: coredns-custom
  namespace: obp-coredns
data:
  obp.server: |2

    <instanceName>.<domain>:1053 {
        log
        kubernetes cluster.local in-addr.arpa ip6.arpa {
            pods insecure
            fallthrough in-addr.arpa ip6.arpa
        }
        rewrite stop {
            name regex (.*)\.<instanceName>\.<domain> istio-
ingressgateway.istio-system.svc.cluster.local answer auto
        }
        forward . /etc/resolv.conf
    }
```

   In the previous example, <*instanceName*> is the name of the instance and <*domain*> is the domain passed when creating the instance.

4. Run the following command to apply the custom `ConfigMap` object.

```
kubectl apply -f <coredns-custom-configmap-yaml-file>
```

5. Run the following command to get the cluster IP address. Record the IP address.

```
kubectl get svc -n obp-coredns
```

6. Run the following command to edit the OpenShift DNS custom resource.

```
kubectl edit dnses.operator/default
```

7. Update the zones section of the DNS custom resource to use your instance and domain names, as shown in the following example.

```
## Add the following section to the dns custom resource under spec
  servers:
```

```
  - forwardPlugin:
      policy: Random
      upstreams:
      - 192.0.2.233
    name: obp-server
    zones:
    - <instanceName>.<domain>
```

In the previous example, <*instanceName*> is the name of the instance and <*domain*> is the parent domain.

8. To add new instances, add entries to the `coredns-custom-configmap` file and update the OpenShift DNS custom resource for the domain of the new instance as shown in the following example:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: coredns-custom
  namespace: obp-coredns
data:
  obp.server: |2

    myobp.example.com:1053 {
        log
        kubernetes cluster.local in-addr.arpa ip6.arpa {
            pods insecure
            fallthrough in-addr.arpa ip6.arpa
        }
        rewrite stop {
            name regex (.*)\.myobp\.example\.com istio-
ingressgateway.istio-system.svc.cluster.local answer auto
        }
        forward . /etc/resolv.conf
    }

    otherobp.example.org:1053 {
        log
        kubernetes cluster.local in-addr.arpa ip6.arpa {
            pods insecure
            fallthrough in-addr.arpa ip6.arpa
        }
        rewrite stop {
            name regex (.*)\.otherobp\.example\.org istio-
ingressgateway.istio-system.svc.cluster.local answer auto
        }
        forward . /etc/resolv.conf
    }



## Edit the Openshift's DNS custom resource
kubectl edit dnses.operator/default

## Add the new instance domain to the existing .spec.servers.zones
  servers:
```

```
- forwardPlugin:
    policy: Random
    upstreams:
    - 192.0.2.233
  name: obp-server
  zones:
  - myobp.example.com
  - otherobp.example.org
```

You must stop and restart the Blockchain Platform instance using Blockchain Platform Manager for DNS changes to take effect.

OpenShift on Microsoft Azure

After you create an Oracle Blockchain Platform Enterprise Edition instance, you must configure DNS so that public host names of the components are resolvable from the OpenShift Local network.

After you create an instance, the instance name and parent domain are used as the subdomain for the Oracle Blockchain Platform Enterprise Edition components. You must set up DNS forwarding on the default DNS configuration for OpenShift, so that DNS requests are forwarded to the custom DNS pod where the exposed services are resolved. Complete the following steps to configure DNS forwarding.

1. Run the following command.

   ```
   oc new-project obp-coredns
   ```

2. Deploy a custom `coredns` server in the `obp-coredns` namespace by running the following command.

   ```
   kubectl apply -f <coredns-deployment-yaml-file>
   ```

   Use the following manifest file with the command.

   ```
   apiVersion: v1
   kind: ConfigMap
   metadata:
     name: coredns
     namespace: obp-coredns
   data:
     Corefile: |-
       .:1053 {
         errors
         health {
           lameduck 5s
         }
         ready
         kubernetes cluster.local in-addr.arpa ip6.arpa {
           pods insecure
           fallthrough in-addr.arpa ip6.arpa
         }
         prometheus :9153
         forward . /etc/resolv.conf
         cache 30
         loop
   ```

```
      reload
      loadbalance
    }
    import custom/*.server

---
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: obp-coredns
  name: obp-coredns
  namespace: obp-coredns
spec:
  progressDeadlineSeconds: 600
  replicas: 1
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      app: obp-coredns
  strategy:
    type: RollingUpdate
  template:
    metadata:
      labels:
        app: obp-coredns
    spec:
      containers:
      - args:
        - -conf
        - /etc/coredns/Corefile
        image: docker.io/coredns/coredns:latest
        imagePullPolicy: IfNotPresent
        livenessProbe:
          failureThreshold: 5
          httpGet:
            path: /health
            port: 8080
            scheme: HTTP
          initialDelaySeconds: 60
          periodSeconds: 10
          successThreshold: 1
          timeoutSeconds: 5
        name: coredns
        ports:
        - containerPort: 1053
          name: dns
          protocol: UDP
        - containerPort: 1053
          name: dns-tcp
          protocol: TCP
        - containerPort: 9153
          name: metrics
          protocol: TCP
        readinessProbe:
          failureThreshold: 3
```

```
      httpGet:
        path: /ready
        port: 8181
        scheme: HTTP
      periodSeconds: 10
      successThreshold: 1
      timeoutSeconds: 1
    resources:
      limits:
        memory: 170Mi
      requests:
        cpu: 100m
        memory: 70Mi
    securityContext:
      allowPrivilegeEscalation: false
      capabilities:
        add:
        - NET_BIND_SERVICE
        drop:
        - all
      readOnlyRootFilesystem: true
    terminationMessagePath: /dev/termination-log
    terminationMessagePolicy: File
    volumeMounts:
    - mountPath: /etc/coredns
      name: config-volume
      readOnly: true
    - mountPath: /etc/coredns/custom
      name: custom-config-volume
      readOnly: true
  dnsPolicy: Default
  restartPolicy: Always
  schedulerName: default-scheduler
  securityContext: {}
  serviceAccount: obp-coredns
  serviceAccountName: obp-coredns
  terminationGracePeriodSeconds: 30
  volumes:
  - configMap:
      defaultMode: 420
      items:
      - key: Corefile
        path: Corefile
      name: coredns
    name: config-volume
  - configMap:
      defaultMode: 420
      name: coredns-custom
      optional: true
    name: custom-config-volume

---
apiVersion: v1
kind: Service
metadata:
  labels:
```

```
    app: obp-coredns
  name: obp-coredns
  namespace: obp-coredns
spec:
  ports:
  - name: dns
    port: 53
    protocol: UDP
    targetPort: 1053
  - name: dns-tcp
    port: 53
    protocol: TCP
    targetPort: 1053
  - name: metrics
    port: 9153
    protocol: TCP
    targetPort: 9153
  selector:
    app: obp-coredns
  sessionAffinity: None
  type: ClusterIP
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: obp-coredns
  namespace: obp-coredns
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: obp-coredns
rules:
- apiGroups:
  - ""
  resources:
  - endpoints
  - services
  - pods
  - namespaces
  verbs:
  - list
  - watch
- apiGroups:
  - ""
  resources:
  - nodes
  verbs:
  - get
- apiGroups:
  - discovery.k8s.io
  resources:
  - endpointslices
  verbs:
  - list
  - watch
```

```
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: obp-coredns
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: obp-coredns
subjects:
- kind: ServiceAccount
  name: obp-coredns
  namespace: obp-coredns
```

3. Create a `coredns-custom-configmap` file in YAML format for your blockchain instance.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: coredns-custom
  namespace: obp-coredns
data:
  obp.server: |2

    <instanceName>.<domain>:1053 {
        log
        kubernetes cluster.local in-addr.arpa ip6.arpa {
            pods insecure
            fallthrough in-addr.arpa ip6.arpa
        }
        rewrite stop {
            name regex (.*)\.<instanceName>\.<domain>\ istio-
ingressgateway.istio-system.svc.cluster.local answer auto
        }
        forward . /etc/resolv.conf
    }
```

   In the previous example, <*instanceName*> is the name of the instance and <*domain*> is the parent domain.

4. Run the following command to apply the custom `ConfigMap` object.

```
kubectl apply -f <coredns-custom-configmap-yaml-file>
```

5. Run the following command to get the cluster IP address. Record the IP address.

```
kubectl get svc -n obp-coredns
```

6. Run the following command to edit the OpenShift DNS custom resource.

```
kubectl edit dnses.operator/default
```

7. Update the zones section of the DNS custom resource to use your instance and domain names, as shown in the following example:

```
## Add the following section to the dns custom resource under spec
  servers:
  - forwardPlugin:
      policy: Random
      upstreams:
      - 192.0.2.233
    name: obp-server
    zones:
    - <instanceName>.<domain>
```

   In the previous example, <*instanceName*> is the name of the instance and <*domain*> is the parent domain.

8. To add new instances, add entries to the `coredns-custom-configmap` file and update the OpenShift DNS custom resource for the domain of the new instance as shown in the following example.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: coredns-custom
  namespace: obp-coredns
data:
  obp.server: |2

    myobp.example.com:1053 {
        log
        kubernetes cluster.local in-addr.arpa ip6.arpa {
            pods insecure
            fallthrough in-addr.arpa ip6.arpa
        }
        rewrite stop {
            name regex (.*)\.myobp\.example\.com istio-
ingressgateway.istio-system.svc.cluster.local answer auto
        }
        forward . /etc/resolv.conf
    }

    otherobp.example.org:1053 {
        log
        kubernetes cluster.local in-addr.arpa ip6.arpa {
            pods insecure
            fallthrough in-addr.arpa ip6.arpa
        }
        rewrite stop {
            name regex (.*)\.otherobp\.example\.org istio-
ingressgateway.istio-system.svc.cluster.local answer auto
        }
        forward . /etc/resolv.conf
    }
```

```
## Edit the Openshift's DNS custom resource
kubectl edit dnses.operator/default

## Add the new instance domain to the existing .spec.servers.zones
  servers:
  - forwardPlugin:
      policy: Random
      upstreams:
      - 192.0.2.233
    name: obp-server
    zones:
    - myobp.example.com
    - otherobp.example.org
```

You must stop and restart the instance for DNS changes to take effect.

Before accessing the Oracle Blockchain Platform service console, configure the hostname resolution for the blockchain instance services, similar to what you did earlier for the Blockchain Platform Manager hostnames. Use the following command to get the list of hostnames for the created blockchain instance:

```
kubectl get virtualservice -n <instance-namespace> -o json | jq -r
      .items[].spec.hosts[0]
```

Add Additional Nodes to Cluster

It's possible for instance creation to fail due to insufficient nodes in the cluster. If this occurs, you may need to add more nodes to the cluster.

1. Run the following command to check for pods in the `Pending` state:

   ```
   kubectl get pods -n <instancename> | grep Pending
   ```

   Additionally you can check for no pods being available:

   ```
   kubectl get pods -n instancename
   ```

2. Next, check the available worker nodes:

   ```
   kubectl get nodes | grep worker
   ```

3. To check if there are nodes available to take new pods, run the following command against each worker node:

   ```
   kubectl describe node $<worker_node>
   ```

   where *<worker_node>* is the name of a worker node. Ensure that the worker node does not exceed 100% capacity.

4. To add additional nodes, first get the number of MachineSets in the cluster:

   ```
   oc get machinesets -n openshift-machine-api
   ```

5. For any MachineSets with fewer than 2 nodes, try upscaling them.

   ```
   oc scale --replicas=2 machineset <obpee00-qtthx-worker-eastus2> -n
   openshift-machine-api
   ```

   where <*obpee00-qtthx-worker-eastus2*> is an example name of a MachineSet you want to upscale to 2 nodes.

6. Query the MachineSets again until the list of ready and available nodes reaches the number of nodes you've selected.
7. You can now redeploy your failed instances.

OpenShift On Premises

After you create an Oracle Blockchain Platform Enterprise Edition instance, you must configure DNS so that public host names of the components are resolvable from the OpenShift Local network.

After you create an instance, the instance name and parent domain are used as the subdomain for the Oracle Blockchain Platform Enterprise Edition components. You must set up DNS forwarding on the default DNS configuration for OpenShift, so that DNS requests are forwarded to the custom DNS pod where the exposed services are resolved. Complete the following steps to configure DNS forwarding.

1. Run the following command.

   ```
   oc new-project obp-coredns
   ```

2. Deploy a custom `coredns` server in the `obp-coredns` namespace by running the following command.

   ```
   kubectl apply -f <coredns-deployment-yaml-file>
   ```

   Use the following manifest file with the command.

   ```
   apiVersion: v1
   kind: ConfigMap
   metadata:
     name: coredns
     namespace: obp-coredns
   data:
     Corefile: |-
       .:1053 {
         errors
         health {
           lameduck 5s
         }
         ready
         kubernetes cluster.local in-addr.arpa ip6.arpa {
           pods insecure
           fallthrough in-addr.arpa ip6.arpa
         }
         prometheus :9153
         forward . /etc/resolv.conf
         cache 30
   ```

```
      loop
      reload
      loadbalance
    }
    import custom/*.server

---
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: obp-coredns
  name: obp-coredns
  namespace: obp-coredns
spec:
  progressDeadlineSeconds: 600
  replicas: 1
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      app: obp-coredns
  strategy:
    type: RollingUpdate
  template:
    metadata:
      labels:
        app: obp-coredns
    spec:
      containers:
      - args:
        - -conf
        - /etc/coredns/Corefile
        image: docker.io/coredns/coredns:latest
        imagePullPolicy: IfNotPresent
        livenessProbe:
          failureThreshold: 5
          httpGet:
            path: /health
            port: 8080
            scheme: HTTP
          initialDelaySeconds: 60
          periodSeconds: 10
          successThreshold: 1
          timeoutSeconds: 5
        name: coredns
        ports:
        - containerPort: 1053
          name: dns
          protocol: UDP
        - containerPort: 1053
          name: dns-tcp
          protocol: TCP
        - containerPort: 9153
          name: metrics
          protocol: TCP
        readinessProbe:
```

```
        failureThreshold: 3
        httpGet:
          path: /ready
          port: 8181
          scheme: HTTP
        periodSeconds: 10
        successThreshold: 1
        timeoutSeconds: 1
      resources:
        limits:
          memory: 170Mi
        requests:
          cpu: 100m
          memory: 70Mi
      securityContext:
        allowPrivilegeEscalation: false
        capabilities:
          add:
          - NET_BIND_SERVICE
          drop:
          - all
        readOnlyRootFilesystem: true
      terminationMessagePath: /dev/termination-log
      terminationMessagePolicy: File
      volumeMounts:
      - mountPath: /etc/coredns
        name: config-volume
        readOnly: true
      - mountPath: /etc/coredns/custom
        name: custom-config-volume
        readOnly: true
    dnsPolicy: Default
    restartPolicy: Always
    schedulerName: default-scheduler
    securityContext: {}
    serviceAccount: obp-coredns
    serviceAccountName: obp-coredns
    terminationGracePeriodSeconds: 30
    volumes:
    - configMap:
        defaultMode: 420
        items:
        - key: Corefile
          path: Corefile
        name: coredns
      name: config-volume
    - configMap:
        defaultMode: 420
        name: coredns-custom
        optional: true
      name: custom-config-volume

---
apiVersion: v1
kind: Service
metadata:
```

```
    labels:
      app: obp-coredns
    name: obp-coredns
    namespace: obp-coredns
spec:
    ports:
    - name: dns
      port: 53
      protocol: UDP
      targetPort: 1053
    - name: dns-tcp
      port: 53
      protocol: TCP
      targetPort: 1053
    - name: metrics
      port: 9153
      protocol: TCP
      targetPort: 9153
    selector:
      app: obp-coredns
    sessionAffinity: None
    type: ClusterIP
---
apiVersion: v1
kind: ServiceAccount
metadata:
    name: obp-coredns
    namespace: obp-coredns
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
    name: obp-coredns
rules:
- apiGroups:
    - ""
    resources:
    - endpoints
    - services
    - pods
    - namespaces
    verbs:
    - list
    - watch
- apiGroups:
    - ""
    resources:
    - nodes
    verbs:
    - get
- apiGroups:
    - discovery.k8s.io
    resources:
    - endpointslices
    verbs:
    - list
```

```
  - watch
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: obp-coredns
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: obp-coredns
subjects:
- kind: ServiceAccount
  name: obp-coredns
  namespace: obp-coredns
```

3. Create a `coredns-custom-configmap` file in YAML format for your blockchain instance.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: coredns-custom
  namespace: obp-coredns
data:
  obp.server: |2

    <instanceName>.<domain>:1053 {
        log
        kubernetes cluster.local in-addr.arpa ip6.arpa {
            pods insecure
            fallthrough in-addr.arpa ip6.arpa
        }
        rewrite stop {
            name regex (.*)\.<instanceName>\.<domain> istio-
ingressgateway.istio-system.svc.cluster.local answer auto
        }
        forward . /etc/resolv.conf
    }
```

   In the previous example, <*instanceName*> is the name of the instance and <*domain*> is the domain passed when creating the instance.

4. Run the following command to apply the custom `ConfigMap` object.

```
kubectl apply -f <coredns-custom-configmap-yaml-file>
```

5. Run the following command to get the cluster IP address. Record the IP address.

```
kubectl get svc -n obp-coredns
```

6. Run the following command to edit the OpenShift DNS custom resource.

```
kubectl edit dnses.operator/default
```

7. Update the zones section of the DNS custom resource to use your instance and domain names, as shown in the following example.

```
## Add the following section to the dns custom resource under spec
  servers:
  - forwardPlugin:
      policy: Random
      upstreams:
      - 192.0.2.233
    name: obp-server
    zones:
    - <instanceName>.<domain>
```

   In the previous example, <*instanceName*> is the name of the instance and <*domain*> is the parent domain.

8. To add new instances, add entries to the `coredns-custom-configmap` file and update the OpenShift DNS custom resource for the domain of the new instance as shown in the following example:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: coredns-custom
  namespace: obp-coredns
data:
  obp.server: |2

    myobp.example.com:1053 {
        log
        kubernetes cluster.local in-addr.arpa ip6.arpa {
            pods insecure
            fallthrough in-addr.arpa ip6.arpa
        }
        rewrite stop {
            name regex (.*)\.myobp\.example\.com istio-
ingressgateway.istio-system.svc.cluster.local answer auto
        }
        forward . /etc/resolv.conf
    }

    otherobp.example.org:1053 {
        log
        kubernetes cluster.local in-addr.arpa ip6.arpa {
            pods insecure
            fallthrough in-addr.arpa ip6.arpa
        }
        rewrite stop {
            name regex (.*)\.otherobp\.example\.org istio-
ingressgateway.istio-system.svc.cluster.local answer auto
        }
        forward . /etc/resolv.conf
    }
```

```
## Edit the Openshift's DNS custom resource
kubectl edit dnses.operator/default

## Add the new instance domain to the existing .spec.servers.zones
  servers:
  - forwardPlugin:
      policy: Random
      upstreams:
      - 192.0.2.233
    name: obp-server
    zones:
    - myobp.example.com
    - otherobp.example.org
```

You must stop and restart the Blockchain Platform instance using Blockchain Platform Manager for DNS changes to take effect.

Add the following entry to the local hosts file:

```
<ISTIO-EXTERNAL-IP> console.myobp.founder.com auth.myobp.founder.com
```

Where <*ISTIO-EXTERNAL-IP*> is the ClusterIP value of the istio-ingressgateway Kubernetes Service. For each additional instance, update the same hosts file entry by appending the new hostnames to the existing line; do not create a new line per instance.

Add Additional Nodes to Cluster

It's possible for instance creation to fail due to insufficient nodes in the cluster. If this occurs, you may need to add more nodes to the cluster.

1. Run the following command to check for pods in the `Pending` state:

   ```
   kubectl get pods -n <instancename> | grep Pending
   ```

   Additionally you can check for no pods being available:

   ```
   kubectl get pods -n instancename
   ```

2. Next, check the available worker nodes:

   ```
   kubectl get nodes | grep worker
   ```

3. To check if there are nodes available to take new pods, run the following command against each worker node:

   ```
   kubectl describe node $<worker_node>
   ```

   where <*worker_node*> is the name of a worker node. Ensure that the worker node does not exceed 100% capacity.

4. To add additional nodes, first get the number of MachineSets in the cluster:

   ```
   oc get machinesets -n openshift-machine-api
   ```

5. For any MachineSets with fewer than 2 nodes, try upscaling them.

   ```
   oc scale --replicas=2 machineset <obpee00-qtthx-worker-eastus2> -n
   openshift-machine-api
   ```

   where <*obpee00-qtthx-worker-eastus2*> is an example name of a MachineSet you want to upscale to 2 nodes.

6. Query the MachineSets again until the list of ready and available nodes reaches the number of nodes you've selected.
7. You can now redeploy your failed instances.

Once your instance has been created and is listed in the Instances list, you can launch the service console from the menu next to the instance name. Use the console to configure your network as described in What's the Console? in *Using Oracle Blockchain Platform*.

6

Manage Oracle Blockchain Platform

Once you've provisioned your instance, you can manage its lifecycle operations in Blockchain Platform Manager. You can also manage your ephemeral storage, worker nodes, and configure a proxy so that your network can run in an offline environment.

Manage Your Instance Lifecycle Operations

Once you've provisioned your instance, you can manage it in Blockchain Platform Manager.

View Instance Details

Clicking on your instance name in Blockchain Platform Manager opens the **Instances** tab displaying details about the instance.

The **Instance Details** page lists information such as the health of the instance, namespace and domain name.

You can manage the instance from the **Actions** menu, or launch the Oracle Blockchain Platform Service Console to manage your blockchain network.

View Instance Activity

The Activity pages shows the status of operations that have been performed on your instances.

To see the activity of an instance, select your instance name and on the **Instances** page click **Activity**.

This tab lists any operations that have been performed on your instance such as starting, stopping, and updating, as well as whether or not it was successful, the time of the operation, and the user ID who initiated the operation.

You can see and filter the activity of all instances managed by Blockchain Platform Manager on the **Activities** page. The **Activities** page can be used to see the history of operations performed on any instances, including deleted instances. These activities can be filtered by different search criteria such as instance name, operation types, and date range.

Start or Stop an Instance

You can start or stop an instance in the Blockchain Platform Manager.

To start or stop an instance:

1. In Blockchain Platform Manager, find your instance and select the menu beside it.
2. Select **Start** or **Stop**. You'll be prompted to confirm your selection.

Delete an Instance

You can delete your instance in Blockchain Platform Manager.

To delete your instance:

1. Open Blockchain Platform Manager and find your instance.
2. From the menu beside your instance, select **Terminate**.
3. You'll be prompted to confirm your action. Click **Confirm**.

Scale an Instance In or Out

You can scale an instance in or out in Blockchain Platform Manager.

Scale Out

You can scale out your instance by increasing replicas or creating peers or orderers:

1. In Blockchain Platform Manager open the **Actions** menu in the instance details page and click **Scale Out**.
2. You can scale out using any of these methods:
   - **New Replicas**: adds additional nodes; REST proxy or CA.
   - **New Peers**: adds one additional peer at a time.
   - **New Orderers**: adds additional orderers.

Scale In

You can scale in your instance by deleting peers.

Before scaling in an instance, you must transfer all this peer's responsibilities to other running peers, and then remove all the responsibilities this peer has.

- Check all other peers' gossip bootstrap address lists, remove the peer address, and add another running peer's address if needed. After the peer configuration change, restart the peer.
- Check all channels' anchor peer lists, remove the peer from the anchor peer lists, and add another running peer to the anchor peer list if needed.
- If a channel or chaincode is only joined or installed on this peer, consider using another running peer to join the same channel and install the same chaincode.

1. In Blockchain Platform Manager open the **Actions** menu in the instance details page and click **Scale In**.
2. Select the peer that you want to delete.

Manage Ephemeral Storage on Kubernetes

Kubernetes pods require ephemeral (temporary) local storage.

Kubernetes pods use ephemeral storage for scratch space, caching, and logs. This storage is temporary and specific to the life cycle of the pod. Ephemeral storage is not shared across pods and it goes away when the pod is deleted.

For more general information about Kubernetes ephemeral storage, see Local ephemeral storage in the Kubernetes documentation.

The following steps apply to Oracle Kubernetes Engine, but the concepts are similar in other Kubernetes environments.

In a node pool, the nodes use their boot volumes for pod storage. Because images are stored in the `/var` directory, most of the ephemeral storage is occupied by images in the root partition. The space needed in the boot volume increases every time that you install an instance of Oracle Blockchain Platform Enterprise Edition and create chaincodes on that node.

You can use the following kubelet API call to check the total ephemeral storage and the amount to assigned to each pod in a given node:

```
kubectl get --raw "/api/v1/nodes/<node IP>/proxy/stats/summary"
```

The `rootfs` and `fs` sections of the JSON result show the capacity and the available bytes.

You can update the amount of ephemeral storage by resizing the boot volume while a node is running. For more information, see Updating a Node Pool for Oracle Kubernetes Engine.

Complete the following steps to resize the ephemeral storage on an Oracle Kubernetes Engine cluster with running nodes.

1. On your Oracle Kubernetes Engine cluster, under **Resources**, select **Node pools**.
2. Click **Edit**. On the Edit node pool page, select **Specify a custom boot volume size** and then enter a **Boot volume size** value in GB. Any nodes that are created will use this value for ephemeral storage.
3. For each worker node in the node pool, complete the following steps to resize the boot volume.
    a. Click the down arrow beside the node to see detailed information about the node.
    b. Navigate to the **Boot volume** for the node and then click **Edit**.
    c. On the Edit volume page under **Volume size and performance**, specify a **Volume size** value in GB and then click **Save changes**.
4. Complete the following steps to set up a Bastion session and then use it to connect to the private worker nodes.
    a. On the instance details page, click the **Oracle Cloud Agent** tab, and then enable the **Bastion** plugin.
    b. Search for `bastion` in the search bar, and then click **Bastion Identity & Security** under Services in the results.
    c. Click **Create bastion**.
    d. On the Create bastion page, for the **Target virtual cloud network** (VCN) specify the Oracle Kubernetes Engine VCN followed by the cluster name. For **Target subnet**, specify the Kubernetes API endpoint. For **CIDR block allowlist**, enter `0.0.0.0/0`, and then click **Create bastion**.
    e. Click the bastion to open it, and then click **Create session**.
    f. Enter `opc` for the **Username** value and select your node from the **Compute instance** list.
    g. Paste your SSH key under **Add SSH Key**.
    h. Click **Show advanced options** and then select the IP address of the node or instance from the **Target compute instance IP address** list. This is the private IPv4 address of the node or instance, which is available in the information section for the instance.
    i. Click **Create session**.
    j. From the context menu for the session, click **Copy SSH command**.

k. You can now log in to the node via SSH by providing your private key with the `-i` parameter in the SSH command.

l. Repeat the previous steps for each worker node in the cluster.

5. For each node, log in to the node via SSH and then run the following commands, which scan for new block storage devices added to instances or nodes, and then expand the file system when storage is available.

```
sudo dd iflag=direct if=/dev/oracleoci/oraclevda of=/dev/null count=1
echo "1" | sudo tee /sys/class/block/`readlink /dev/oracleoci/oraclevda |
cut -d'/' -f 2`/device/rescan
sudo /usr/libexec/oci-growfs -y
```

Determining Ephemeral Storage Usage

You can run the following script to see the ephemeral storage usage of an instance running on Oracle Kubernetes Engine. The script uses the Kubernetes API to retrieve ephemeral storage usage for each pod that is running on each node in the cluster.

```
#!/usr/bin/env bash

kubectl proxy --append-server-path &

set -eo pipefail

{
    echo "NODE NAMESPACE POD EPHEMRAL_USED"
    for node in $(kubectl get nodes -o=jsonpath='{range .items[*]}
{.metadata.name}{"\n"}{end}'); do
        curl -fsSL "http://127.0.0.1:8001/api/v1/nodes/$node/proxy/stats/
summary" |
            yq '.pods[] | [.podRef.namespace, .podRef.name, .ephemeral-
storage.usedBytes] | join(" ")' |
            while read -r namespace name usedBytes; do
                # A pod might have no running containers and consequently no
ephemeral-storage usage.
                echo "$node" "$namespace" "$name" "$(numfmt --to iec "$
{usedBytes:-0}")"
            done
    done | sort -k4,4rh
} | column -t
```

Additional Steps When Replacing Worker Nodes

When you replace an existing worker node (where peers/orderers are running) with a new worker node, you must also complete the following additional steps:

1. Ensure that the Persistent Volumes that are mounted on the existing node can be migrated to and accessed from the new node. To do this on Oracle Kubernetes Engine, create a node in the same Availability Domain as the existing node.
2. Stop all instances that use the older node.
3. Cordon and drain the older node. This might affect Blockchain Platform Manager services, if those services are running on the older node. Wait for the running pods to move into the new node.

4. Run the following commands to get the list of all of the peers and orderers that were running on the cordoned node.

```
kubectl get peer -A -o=custom-
columns='NAMESPACE:.metadata.namespace,NAME:.metadata.name,NODESELECTOR:.sp
ec.nodeSelector'
kubectl get orderernode -A -o=custom-
columns='NAMESPACE:.metadata.namespace,NAME:.metadata.name,NODESELECTOR:.sp
ec.nodeSelector'
```

5. For the peers and orderers that were configured with `nodeSelector` for the older node, run the following commands to update the custom resource `.spec.nodeSelector` to select the new node.

```
kubectl patch peer <PEER> -n <NAMESPACE> -p '{"spec":{"nodeSelector":
{"kubernetes.io/hostname":"<NEW_NODE_HOSTNAME>"}}}' --type='merge'
kubectl patch orderernode <ORDERER> -n <NAMESPACE> -p '{"spec":
{"nodeSelector":{"kubernetes.io/hostname":"<NEW_NODE_HOSTNAME>"}}}' --
type='merge'
```

6. Verify the updated `nodeSelector` value by running the commands from Step 4 again.
7. Start all instances that were previously stopped.

Configure a Proxy

If your instance runs in a private network without internet connectivity, you must configure a proxy for the blockchain services.

Complete the following tasks to set up a proxy for your blockchain instances.

Create a Service Entry

Use the following configuration to create an Istio `ServiceEntry` object as an external proxy in your instance namespace. You must create a TCP (not HTTP) `ServiceEntry` object to enable Istio-controlled traffic to the external proxy. For more information, see Configure traffic to external HTTPS proxy in the Istio documentation.

```
apiVersion: networking.istio.io/v1alpha3
kind: ServiceEntry
metadata:
  name: obpee-ext-proxy
  namespace: <INSTANCE_NAMESPACE>
spec:
  hosts:
  - <PROXY-HOST-FQDN>
  addresses:
  - <PROXY-IP-ADDRESS>
  exportTo:
  - "."
  location: MESH_EXTERNAL
  ports:
  - number: <PROXY-PORT-NUMBER>
    name: tcp
    protocol: TCP
```

Configure the Proxy Environment

The Oracle Blockchain Platform Enterprise Edition distribution package includes the `setProxy.sh` script, which you can use to configure the proxy environment for all services of the blockchain instance. Run the following commands from the command line. When the `setProxy.sh` script runs, it restarts the required blockchain services in your Kubernetes cluster.

```
# Configure environment variables before running the script
export mspId="<INSTANCE_NAME>"
export httpProxy="<HTTP_PROXY>"
export httpsProxy="<HTTPS_PROXY>"
export noProxy="<NO_PROXY>"

# Go to the distribution package dir
cd <distribution-package-dir>

# Run the setProxy.sh script
./setProxy.sh
```

7

Logging

You can use Oracle Cloud Infrastructure (OCI) or external tools to configure persistent logging for Oracle Blockchain Platform Enterprise Edition.

Overview

Oracle Blockchain Platform Enterprise Edition is based on Kubernetes, where logs are stored locally on each pod. To prevent logs from being deleted when a pod is deleted, you must set up persistent logging, where logs are stored in a central location. There are two methods you can use for persistent logging. You can use an external logging tool such as Fluentd and Elastic Stack. Alternately, if you are running on Oracle Kubernetes Engine, you can use the centralized logging solution supported by Oracle Cloud Infrastructure (OCI).

Persistent Logging with OCI

To store logs centrally using OCI, you define log groups and configure agents to parse the logs. The logs are stored in the Object Storage service. Before you configure persistent logging with OCI, your deployment must meet the following requirements.

- A dynamic group in the Kubernetes compartment. To create a dynamic group, click **Identity & Security** in the navigation menu. Under **Identity**, click **Domains** and then click **Create dynamic group**. Add the following to your dynamic group in the **Matching rules** section, substituting the Oracle Cloud ID for your compartment.

  ```
  instance.compartment.id = '<compartment_ocid>'
  ```

  For example:

  ```
  instance.compartment.id =
  'ocid1.compartment.oc1..aaaaaaaa4ws3242343243244nyb423432rwqsxigt2sia'
  ```

- A policy that allows the dynamic group to interact with the logging service. To create a policy, click **Identity & Security** in the navigation menu. Under **Identity**, click **Policies** and then click **Create Policy**.

  ```
  Allow dynamic-group <my-group> to use log-content in compartment
  <target_compartment_name>
  ```

  For example:

  ```
  Allow dynamic-group okelogging to use log-content in compartment
  BlockchainTeam
  ```

After you have satisfied the prerequisites, complete the following steps to store logs centrally using OCI.

1. Click the menu icon in the upper left corner, search for `log`, and then select **Logs**.
2. Create a log group. Under Logging, select **Log Groups** and then click **Create Log Group**.
3. Create a custom log. Under Logging, select **Logs** and then click **Create custom log** to open the Create custom log wizard. Select the log group that you created previously.
4. On the second page of the Create custom log wizard, create an agent configuration for the custom log, specifying the Kubernetes compartment and the dynamic group.
5. In the **Configure log inputs** section of the Agent configuration page, configure the log input for the agent to use the following file path, which is the default for application containers. Select **Log path** from the **Input type** list. Enter the following file path for **File paths**. This path includes all container logs, including system and service containers.

   ```
   /var/log/pods/*/*/*.log
   ```

6. Wait until logs are ingested. Typically, logs are ingested in 3-5 minutes.
7. Select **Logs** and then navigate to the custom log and click **Explore Log**. You can analyze, parse, and filter the logs.
8. You can also use OCI to store the logs in the Object Storage service.
   a. Create a connector. Under Logging, select **Connectors** and then click **Create Connector**. Select **Logging** as the **Source** and **Object Storage** as the **Target**.
   b. Configure the source and target as needed.
   c. Under the **Enable logs** section, set **Enable Log** to **Enabled** for the connector that you created. The Create Log panel is displayed, with a default value for log retention time.
   d. Wait until logs are ingested. Typically, logs are ingested in 3-5 minutes. You can then see read and write operations in the connector logs. Logs are now being written to the Object Storage service.

For more information, see Monitor Kubernetes and OKE clusters with OCI Logging Analytics.

Persistent Logging with External Tools

You can store logs centrally using Fluentd and Elastic Stack. The following steps have been tested with Fluentd v1.16.2 and Elasticsearch 7.9.1. Use these versions or later when you complete these steps.

1. Create a Kubernetes namespace called `fluentd`.
2. Use the following command to create a role-based access control resource.

   ```
   kubectl create -f fluentd-rbac.yaml -n fluentd
   ```

   Use the following `fluentd-rbac.yaml` file with the command.

   ```
   apiVersion: v1
   kind: ServiceAccount
   metadata:
     name: fluentd
     namespace: fluentd
   ---
   apiVersion: rbac.authorization.k8s.io/v1
   ```

```
kind: ClusterRole
metadata:
  name: fluentd
  namespace: fluentd
rules:
- apiGroups:
  - ""
  resources:
  - pods
  - namespaces
  verbs:
  - get
  - list
  - watch
---
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: fluentd
roleRef:
  kind: ClusterRole
  name: fluentd
  apiGroup: rbac.authorization.k8s.io
subjects:
- kind: ServiceAccount
  name: fluentd
  namespace: fluentd
```

3. Use the following command to create a `ConfigMap` object for Fluentd or Elastic Stack.

   ```
   kubectl create -f fluentd-configmap_removefilter_ascii.yaml -n fluentd
   ```

   Use the following `fluentd-configmap_removefilter_ascii.yaml` file with the command.
   In the following file, remove the number sign (#) to uncomment only one of the following lines.

   - Uncomment `@include file-fluent.conf` if you are writing to a file in the `/tmp/obp.log` path.
   - Uncomment `@include elastic-fluent.conf` if you are writing to Elasticsearch.

   The following file shows an example of writing to the `/tmp/obp.log` path.

   ```
   apiVersion: v1
   kind: ConfigMap
   metadata:
     name: fluentd-config
     namespace: fluentd
   data:
     fluent.conf: |-
       ##################################################################
       # This source gets all logs from local docker host
       #@include pods-kind-fluent.conf
       #@include pods-fluent.conf
       @include pods-nofilter.conf
       @include file-fluent.conf
   ```

```
    #@include elastic-fluent.conf
  pods-nofilter.conf: |-
    <source>
      @type tail
      path /var/log/containers/*.log
      format //^(?<time>.+) (?<stream>stdout|stderr) (?<logtag>.)? (?
<log>.*)$/ /
      pos_file /var/log/fluentd-containers.log.pos
      tag kubernetes.*
      read_from_head true
    </source>
    <filter kubernetes.**>
     @type kubernetes_metadata
    </filter>
  file-fluent.conf: |-
     <match kubernetes.var.log.containers.**fluentd**.log>
      @type null
     </match>
     <match kubernetes.var.log.containers.**kube-system**.log>
       @type null
     </match>
     <match kubernetes.**>
       @type file
       path /tmp/obp.log
     </match>
  elastic-fluent.conf: |-
    <match kubernetes.var.log.containers.**fluentd**.log>
      @type null
    </match>
    <match kubernetes.var.log.containers.**kube-system**.log>
      @type null
    </match>
    <match kubernetes.**>
      @type elasticsearch
      host "#{ENV['FLUENT_ELASTICSEARCH_HOST'] || 'elasticsearch.elastic-
kibana'}"
      port "#{ENV['FLUENT_ELASTICSEARCH_PORT'] || '9200'}"
      index_name fluentd-k8s-3
      type_name fluentd
      include_timestamp true
    </match>
```

4. Use the following command to create a `DaemonSet` object for Fluentd. This command creates a Fluentd pod on each node.

   ```
   kubectl create -f fluentd.yaml -n fluentd
   ```

   Use the following `fluentd.yaml` file with the command.

   ```
   apiVersion: apps/v1
   kind: DaemonSet
   metadata:
     name: fluentd
     namespace: fluentd
     labels:
   ```

```
    k8s-app: fluentd-logging
    version: v1
spec:
  selector:
    matchLabels:
      k8s-app: fluentd-logging
      version: v1
  template:
    metadata:
      labels:
        k8s-app: fluentd-logging
        version: v1
    spec:
      serviceAccount: fluentd
      serviceAccountName: fluentd
      tolerations:
      - key: node-role.kubernetes.io/master
        effect: NoSchedule
      - key: node-role.kubernetes.io/control-plane
        effect: NoSchedule
      containers:
      - name: fluentd1
        imagePullPolicy: "Always"
        image: fluent/fluentd-kubernetes-daemonset:v1.16.2-debian-
elasticsearch7-1.1
        env:
          - name:  FLUENT_ELASTICSEARCH_HOST
            value: "elasticsearch.elastic-kibana"
          - name:  FLUENT_ELASTICSEARCH_PORT
            value: "9200"
        resources:
          limits:
            memory: 200Mi
          requests:
            cpu: 100m
            memory: 200Mi
        volumeMounts:
        - name: fluentd-config
          mountPath: /fluentd/etc
        - name: logs
          mountPath: /tmp
        - name: varlog
          mountPath: /var/log
        - name: varlibdockercontainers
          mountPath: /var/lib/docker/containers
          readOnly: true
      terminationGracePeriodSeconds: 30
      volumes:
      - name: fluentd-config
        configMap:
          name: fluentd-config
      - name: varlog
        hostPath:
          path: /var/log
      - name: varlibdockercontainers
        hostPath:
```

```
          path: /var/lib/docker/containers
      - name: logs
        hostPath:
          path: /tmp
```

The Oracle Blockchain Platform logs are available in the `/tmp` directory of the Fluentd pod or the Kubernetes node.

5. To send the logs to Elastic Stack, create a Kubernetes namespace called `elastic-kibana`.
6. Use the following command to create a deployment for Elastic Stack and to expose it as a service.

```
kubectl create -f elastic.yaml -n elastic-kibana
```

Use the following `elastic.yaml` file with the command.

```
apiVersion: v1
kind: Namespace
metadata:
  name: elastic-kibana

---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: elasticsearch
  namespace: elastic-kibana
  labels:
    app: elasticsearch
spec:
  selector:
    matchLabels:
      app: elasticsearch
  replicas: 1
  template:
    metadata:
      labels:
        app: elasticsearch
    spec:
      initContainers:
      - name: vm-max-fix
        image: busybox
        command: ["sysctl", "-w", "vm.max_map_count=262144"]
        securityContext:
          privileged: true
      containers:
      - name: elasticsearch
        image: elasticsearch:7.9.1
        imagePullPolicy: IfNotPresent
        ports:
        - containerPort: 9200
        env:
        - name: node.name
          value: "elasticsearch"
        - name: cluster.initial_master_nodes
```

```
          value: "elasticsearch"
        - name: bootstrap.memory_lock
          value: "false"
        - name: ES_JAVA_OPTS
          value: "-Xms512m -Xmx512m"
---
apiVersion: v1
kind: Service
metadata:
  name: elasticsearch
  namespace: elastic-kibana
  labels:
    app: elasticsearch
spec:
  type: ClusterIP
  selector:
    app: elasticsearch
  ports:
    - protocol: TCP
      name: http
      port: 9200
      targetPort: 9200
```

7. You can then use the following commands to examine the log data in the Elasticsearch index.

```
curl -X GET "localhost:9200/_cat/indices/fluentd-k8s-*?
v=true&s=index&pretty"
curl -X GET "localhost:9200/fluentd-k8s-3/_search?pretty=true"
```

8. You can also use Fluentd to store the log on the local block volume on each node.
   a. Create a block volume for each node, attach the volume, and create a directory called `/u01`.
   b. Format the attached block volume for the ext4 file system.
   c. Mount the `/u01` directory on the device path.
   d. Change the Fluentd deployment file (`fluentd.yaml`) so that the logs volume is `/u01`, not `/tmp`, as shown in the following snippet.

```
- name: logs
        hostPath:
          path: /u01
```

   e. Run the following command to apply the Fluentd deployment.

```
kubectl apply -f fluentd.yaml -n fluentd
```

   f. The logs are now visible in the /u01 directory on each node.

Set the Log Level for Operator Pods

You can set the log level for the `hlf-operator` and `obp-operator` pods. The steps to set the log level are different depending on whether Oracle Blockchain Platform is installed. If Oracle Blockchain Platform Enterprise Edition is not yet installed, complete the following steps.

1. Open the corresponding `deployment.yaml` file for editing. The file for the `hlf-operator` pod is in the following location:

   distribution_package_location`/distribution-package/operators/helmcharts/hlf-operator/templates/deployment.yaml`

   The file for the `obp-operator` pod is in the following location:

   distribution_package_location`/distribution-package/operators/helmcharts/obp-operator/templates/deployment.yaml`

2. Add the following line to the file. As shown in the comment, you can set the log level to `debug`, `info`, or `error`. In the following example the log level is set to `info`.

   ```
   --zap-log-level=info # debug, info, error
   ```

After you edit the file, that section of the file might look similar to the following text:

```
containers:
    - args:
    - --enable-leader-election
    - --zap-log-level=info # debug, info, error
```

If Oracle Blockchain Platform is already installed, complete the following step.

- Use the following commands to edit the deployment definitions for the `hlf-operator` and `obp-operator` pods. Add or update the argument that configures the log level for the `manager` container under the pod template specification.

  ```
  kubectl edit deployment -n obp-cp obp-operator
  kubectl edit deployment -n obp-cp hlf-operator-controller-manager
  ```

After you update the container arguments, the deployment definition might look similar to the following text:

```
containers:
    - args:
    - --enable-leader-election
    - --zap-log-level=info # debug, info, error
```

A

Accessibility Features and Tips for Oracle Blockchain Platform

This topic describes accessibility features and information for Oracle Blockchain Platform.

Table A-1 Keyboard Shortcuts for Blockchain Platform Manager

Task	Keyboard Shortcut
Create an Oracle Blockchain Platform instance.	Windows: Alt+Shift+C Mac: Shift+Option+C
Refresh the Instance Summary page.	Windows: Alt+Shift+R Mac: Shift+Option+R