Oracle Cloud Native Environment

Quick Start for Release 2.0

F96192-02
September 2024

Oracle Cloud Native Environment Quick Start for Release 2.0,

F96192-02

Contents

Preface

The document provides the steps you need to create a Kubernetes cluster using the Oracle Cloud Native Environment CLI. The cluster is created using the `libvirt` provider. The `libvirt` provider creates a virtualized cluster on the local host using the Oracle KVM stack. This is the fastest method to create a cluster with Oracle Cloud Native Environment. Steps are also provided to expose the Oracle Cloud Native Environment UI and install applications from the application catalog.

Documentation License

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.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at https://www.oracle.com/corporate/accessibility/.

Access to Oracle Support for Accessibility

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit https://www.oracle.com/corporate/accessibility/learning-support.html#support-tab.

Diversity and Inclusion

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

1 Introduction

Describes how to create a Kubernetes cluster using the Oracle Cloud Native Environment CLI. The cluster is created using the `libvirt` provider.

The `libvirt` provider creates a virtualized cluster on the local host using the Oracle KVM stack. This is the fastest method to create a cluster with Oracle Cloud Native Environment.

The `libvirt` provider is the default cluster provider, and can be used to provision Kubernetes clusters using Kernel-based Virtual Machines (KVM). The Oracle KVM stack is used to install libvirt as this KVM version offers many more features for Oracle Linux systems.

The `libvirt` provider provisions Kubernetes clusters using libvirt on a single host, and is useful for creating and destroying Kubernetes clusters for testing and development. While the `libvirt` provider can be used for test and development clusters, it does deploy a production worthy cluster configuration.

> **Important:**
>
> As all libvirt cluster nodes are running on a single host, be aware that if the host running the cluster goes down, so do all the cluster nodes.

You can also create clusters using other providers, and information on those providers is available in Oracle Cloud Native Environment: Kubernetes Clusters.

Steps are also provided to expose the Oracle Cloud Native Environment UI, and install applications from the default application catalog.

You're guided through:

1. Installing the CLI on an Oracle Linux host. The host can be Oracle Linux 8 or 9.
2. Setting up the Oracle KVM stack on the host.
3. Using the CLI to create a Kubernetes cluster using the `libvirt` provider. This is a virtualized cluster, running on the localhost. Use the `ocne cluster start` command to create a default, single node, cluster. You can also use `ocne cluster start` command options, or a cluster configuration file, to customize the cluster to add more nodes.
4. Creating a Kubernetes access token to access the UI. The UI is installed by default in the cluster.
5. Exposing the UI on the localhost using port forwarding.
6. Installing applications from the default application catalog available in the UI. You can use the CLI or the UI to search for, and install applications. Both CLI and UI options are provided.

2
Installing the CLI

Install the Oracle Cloud Native Environment CLI on an Oracle Linux host, using the Oracle Linux Yum Server, or the Unbreakable Linux Network (ULN).

The Oracle Cloud Native Environment CLI (CLI) is the command line tool to create and manage Kubernetes clusters in Oracle Cloud Native Environment. The CLI (`ocne` command) includes a help system to show all command options, and a set of configuration files at various levels to configure the environment and clusters.

1. Set up the Oracle Linux Yum Server Repository.

   If the system uses the Oracle Linux Yum Server, set it up to install the CLI:

   Oracle Linux 9:

   ```
   sudo dnf install oracle-ocne-release-el9
   sudo dnf config-manager --enable ol9_ocne
   ```

   Oracle Linux 8:

   ```
   sudo dnf install oracle-ocne-release-el8
   sudo dnf config-manager --enable ol8_ocne
   ```

2. Set up ULN.

   If the system is registered to use ULN, use the ULN web interface to subscribe the system to the appropriate channel.

   For Oracle Linux 9, subscribe to `ol9_x86_64_ocne` or `ol9_aarch64_ocne`.

   For Oracle Linux 8, subscribe to `ol8_x86_64_ocne` or `ol8_aarch64_ocne`.

3. Install the CLI.

   ```
   sudo dnf install ocne
   ```

3
Setting Up the libvirt Provider

Set up an Oracle Linux host to create Kubernetes clusters using the `libvirt` provider.

Clusters can be created on the localhost, or on a remote system. Perform these steps on the system to be used to create the cluster, whether that's the localhost for local clusters, or on a remote host if you're creating clusters on a remote system.

1. Install the Oracle KVM stack.
   - Oracle Linux 9:

     If you have an existing installation of the default KVM stack, remove it:

     ```
     sudo dnf remove libvirt qemu-kvm edk2
     ```

     Install the Oracle KVM stack:

     ```
     sudo dnf config-manager --enable ol9_kvm_utils
     sudo dnf group install "Virtualization Host"
     sudo dnf install virt-install virt-viewer
     ```

     Start the virtualization daemons.

     ```
     for drv in qemu network nodedev nwfilter secret storage interface
     proxy
       do
        sudo systemctl enable virt${drv}d.service
        sudo systemctl enable virt${drv}d{,-ro,-admin}.socket
        sudo systemctl start virt${drv}d{,-ro,-admin}.socket
       done
     ```

   - Oracle Linux 8:

     If you have an existing installation of the default KVM stack, remove it:

     ```
     sudo dnf module remove virt --all
     sudo dnf module reset virt
     ```

     Install the Oracle KVM stack:

     ```
     sudo dnf config-manager --enable ol8_kvm_appstream
     sudo dnf module enable virt:kvm_utils3
     sudo dnf --allowerasing distro-sync
     sudo dnf module install virt:kvm_utils3
     ```

     Enable and start the `libvirtd.service`:

     ```
     sudo systemctl enable --now libvirtd.service
     ```

For more information on installing and configuring KVM, see the Oracle Linux: KVM User's Guide.

2. Validate the host.

   Validate the host is set up for hardware virtualization, and can be used as a KVM host:

   ```
   virt-host-validate qemu
   ```

3. Configure the user.

   Configure the user to have privileged access to libvirt, add the user to the `libvirt` and `qemu` groups.

   ```
   sudo usermod -a -G libvirt,qemu $USER
   ```

   To enable the change to the user, log out, and log back into the host or terminal session.

4. (Optional) Open a range of ports in the firewall.

   If you're installing libvirt on a remote host, open a series of firewall ports so you can access nodes in the cluster from the localhost. You don't need to do this if you're installing libvirt on the localhost. Use the format:

   ```
   sudo firewall-cmd --add-port 6443-endrange/tcp
   sudo firewall-cmd --add-port 6443-endrange/tcp --permanent
   ```

   Replace *endrange* with the highest port number you want to open. For example, to open 20 ports, use:

   ```
   sudo firewall-cmd --add-port 6443-6463/tcp
   sudo firewall-cmd --add-port 6443-6463/tcp --permanent
   ```

   Restart `firewalld.service`

   ```
   sudo systemctl restart firewalld.service
   ```

5. (Optional) Set up proxy configuration.

   If you use a proxy server, configure the libvirt host so container images can be pulled from the Oracle Container Registry using HTTPS. For example:

   ```
   export HTTPS_PROXY=https://proxy.example.com:3128
   ```

4
Creating a libvirt Cluster

Create a Kubernetes cluster using the `libvirt` provider.

1. (Optional) Set up a cluster configuration file.

   A cluster configuration contains the cluster specific information to use when creating the cluster. Use a cluster configuration file to override cluster defaults, or you can use `ocne cluster start` command options to configure a cluster. A cluster configuration file might include:

   ```
   provider: libvirt
   name: mycluster
   workerNodes: 3
   controlPlaneNodes: 3
   providers:
     libvirt:
       controlPlaneNode:
         cpu: 2
         memory: 16Gi
         storage: 8Gi
       workerNode:
         cpu: 2
         memory: 16Gi
         storage: 8Gi
   ```

   For information about cluster configuration files, see Oracle Cloud Native Environment: Kubernetes Clusters.

2. Create a libvirt cluster.

   Use the `ocne cluster start` command to create a cluster. The syntax is:

   ```
   ocne cluster start
   [{-u|--auto-start-ui} {true|false}]
   [{-o|--boot-volume-container-image} URI]
   [{-C|--cluster-name} name]
   [{-c|--config} path]
   [{-n|--control-plane-nodes} integer]
   [{-i|--key} path]
   [--load-balancer address]
   [{-P|--provider} provider]
   [{-s|--session} URI]
   [{-v|--version} version]
   [--virtual-ip IP]
   [{-w|--worker-nodes} integer]
   ```

   For more information on the syntax options, see Oracle Cloud Native Environment: CLI.

When you create a cluster on a remote system, include the session information in the format `qemu+ssh://user@host/system` where *host* is the name or IP address of the remote system. For example:

```
--session qemu+ssh://myuser@myhost.example.com/system
```

Example 4-1 Create a default cluster using the `libvirt` provider

To create a libvirt cluster, using all default settings:

```
ocne cluster start
```

Example 4-2 Create a `libvirt` cluster using a configuration file

To create a `libvirt` cluster using a configuration file:

```
ocne cluster start --config myconfig.yaml
```

Example 4-3 Create a `libvirt` cluster with specified nodes and virtual IP

To create a cluster with a specified number of worker and control plane nodes, and a virtual IP address:

```
ocne cluster start --control-plane-nodes 3 --worker-nodes 5 --virtual-ip
192.168.0.100
```

Example 4-4 Create a remote `libvirt` cluster using a configuration file

To create a cluster on a remote host using a configuration file:

```
ocne cluster start --session qemu+ssh://myuser@myhost.example.com/system --
config myconfig.yaml
```

5
Connecting to a Cluster

Use the `kubectl` package to connect to a Kubernetes cluster created with the `libvirt` provider.

After creating a Kubernetes cluster, two Kubernetes configuration (`kubeconfig`) files are created to connect to the cluster using the `kubectl` command.

One file provides direct access to the true Kubernetes API server endpoint. This `kubeconfig` file is saved as `$HOME/.kube/kubeconfig.`*`cluster_name`*`.local`. You can use this file to access a cluster created on the localhost, or to access a cluster created on a remote libvirt host.

The second `kubeconfig` file provides access to a dedicated tunnel implemented with SLiRP to access the cluster on remote systems. This file is saved as `$HOME/.kube/kubeconfig.`*`cluster_name`*`.vm`. If the cluster is started on a remote system, and you want to log into that remote system to access the cluster, you need to copy this second `kubeconfig` to the remote system.

Install `kubectl` on the host on which you want to access the cluster, either the localhost, or the remote libvirt system.

These steps, unless explicitly mentioned, are to be performed on the host on which you want to access the cluster (either the localhost or a remote libvirt system).

1. (Optional) Copy the `kubeconfig` file.

   If you created the cluster on a remote system, and you want to log in to the remote system to access the cluster, copy the `kubeconfig` file ending in `.vm` from the localhost to the remote system. The file is available on the localhost at:

   ```
   $HOME/.kube/kubeconfig.cluster_name.vm
   ```

   Replace *cluster_name* with the name you used to create the cluster. The default is `ocne`.

   > **Tip:**
   >
   > If you copy the file to the remote system as `$HOME/.kube/config`, you don't need to set the `$KUBECONFIG` environment variable to access the cluster on the remote host.

2. Install the `kubectl` package.

   ```
   sudo dnf install kubectl
   ```

3. Set the `kubeconfig` file location using an environment variable.

   For a cluster running on the localhost, use:

   ```
   export KUBECONFIG=$HOME/.kube/kubeconfig.cluster_name.local
   ```

4. (Optional) Persist the environment variable.

   Add the environment variable to the `.bashrc` file. For example:

   ```
   echo 'export KUBECONFIG=$HOME/.kube/kubeconfig.cluster_name'.local
   >> $HOME/.bashrc
   ```

5. Verify that you can use `kubectl` to connect to the cluster.

   For example:

   ```
   kubectl get deployments --all-namespaces
   ```

6
Creating an Access Token

Create an access token to authenticate a connection to the UI.

The UI is deployed into the cluster as a Kubernetes deployment named `ui`, running in the `ocne-system` namespace. A Kubernetes service to access this deployment is also created. The service is also named `ui` and running in the `ocne-system` namespace. To connect to the service, generate an access token.

1. Create an access token.

   Create an access token for the `ui` service:

   ```
   kubectl --namespace ocne-system create token ui
   ```

   The token is displayed.

2. Save the token.

   Save the token in a secure location so you can use it to authenticate a connection when you access the UI.

7

Exposing the UI Using Port Forwarding

Expose the UI service using port forwarding.

Port-forwarding is a convenient way of exposing the UI service on the localhost for debugging and troubleshooting in a development environment.

> **Important:**
>
> We don't recommend you use port-forwarding to expose the UI in a production environment.

1. Set up port forwarding.

   Set up port forwarding by running the following command:

   ```
   kubectl port-forward --namespace ocne-system service/ui 8443:443
   ```

   > **Note:**
   >
   > You must let the `kubectl port-forward` command continue to run for the time you need access to the UI.

2. Access to the UI.

   You can access the UI on the localhost using a web browser. Open a browser session and enter the following address:

   ```
   https://127.0.0.1:8443
   ```

   The **Authentication** page is displayed.

   > **Tip:**
   >
   > You can access the UI on the localhost from a remote machine by using local port forwarding. For example, to enable incoming connections to localhost on port `9898`, enter the following command on the remote machine:
   >
   > ```
   > ssh -L 9898:127.0.0.1:8443 myuser@myhost.example.com
   > ```

3. Enter the access token.

   Enter the access token into the **ID token** field on the Security page and click **AUTHENTICATE**.

The UI application uses the token to authenticate the connection and the **Overview** page appears.

8 Searching a Catalog

Search for applications in an application catalog.

Search a catalog to find the list of applications that can be installed or updated.

- CLI
- UI

CLI

Use the `ocne catalog search` command to search for applications in an application catalog. The syntax is:

```
ocne catalog search
[{-N|--name} name]
[{-p|--pattern} pattern]
```

For more information on the syntax options, see Oracle Cloud Native Environment: CLI.

Examples:

To search for all applications in the default catalog:

```
ocne catalog search
```

To search the default application catalog for a specific string:

```
ocne catalog search --pattern 'ingress-*'
```

UI

To search for applications in the default catalog:

1. In the navigation menu, click **Apps**.

   The **Applications** page is displayed, listing all the applications available in the default catalog.

2. Enter characters in the **Search** field to display only applications whose name contains the entered characters.

9
Installing an Application

Install an application from a catalog.

- CLI
- UI

CLI

Use the `ocne application install` command to install an application from a catalog. The syntax is:

```
ocne application install
{-b|--built-in-catalog}
{-c|--catalog} name
{-N|--name} name
[{-n|--namespace} namespace]
{-r|--release} name
[{-u|--values} URI]
[{-v|--version} version]
```

For more information on the syntax options, see Oracle Cloud Native Environment: CLI.

Examples:

To install an application from the default catalog:

```
ocne application install --release ingress-nginx --namespace ingress-nginx --
name ingress-nginx
```

UI

To install an application from the default catalog:

1. In the navigation menu, click **Apps**.

   The **Applications** page is displayed listing all the applications available in the default catalog.

2. Enter characters in the **Search** field to display only applications whose name contain the entered characters.

3. Find the application you need to install and click **Install**.

   A dialog box appears with the following fields:

   - An edit box with YAML configuration for the application that's to be installed.

     You can edit the YAML configuration according to the system requirements.

- The following mandatory fields: **Release Name**, **Namespaces**, **Versions** and **Release Description**.

4. Click **Install**.

   The dialog is dismissed, and a message is displayed onscreen to confirm the installation request has been accepted.

   The installation completes and a message is displayed confirming the release has been created successfully.