Oracle Cloud Native Environment

Multus Module for Release 1.9

F93859-01
May 2024

Oracle Cloud Native Environment Multus Module for Release 1.9,

F93859-01

Contents

Preface

This document contains information about installing and using Multus in Oracle Cloud Native Environment. It describes the Multus module provided with Oracle Cloud Native Environment. It provides basic examples on how to test that Multus is installed and working.

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 to the Multus Module

Multus is a Container Network Interface (CNI) plugin for Kubernetes. Unlike other CNI plugins, Multus doesn't implement pod networking. Multus is a multiplexer for other CNI plugins. By using Multus, it's possible to attach many network interfaces to a single pod.

Multus creates a networking bridge to either Flannel or Calico. For more information on Multus, see the upstream Multus documentation.

You can create NetworkAttachmentDefinitions using a Multus configuration file during the Multus module installation, or you can create NetworkAttachmentDefinitions after the module is deployed using the `kubectl` command.

2
Installing the Multus Module

This chapter discusses how to install the Multus module in Oracle Cloud Native Environment.

Prerequisites

This section contains the prerequisite information you might need to set up the Multus module.

Updating Proxy Configuration

If you're using a proxy server in the environment, edit the CRI-O proxy configuration file and add the Kubernetes service IP (the default is `10.96.0.1`) to the `NO_PROXY` variable. For example, on each Kubernetes node, edit the `/etc/systemd/system/crio.service.d/proxy.conf` file:

```
[Service]
Environment="HTTP_PROXY=http://proxy.example.com:3128"
Environment="HTTPS_PROXY=https://proxy.example.com:3128"
Environment="NO_PROXY=mydomain.example.com,10.96.0.1"
```

Reload the configuration file and restart the `crio` service:

```
sudo systemctl daemon-reload
sudo systemctl restart crio.service
```

> **Note:**
>
> You don't need to perform this step if you're using the `olcnectl provision` command to perform a quick installation. This is set up for you automatically when using that installation method and you provide any proxy information.

Creating a Multus Configuration File

You can optionally provide a Multus configuration file to set up NetworkAttachmentDefinitions when you deploy the Multus module.

If you deploy the Multus module without a configuration file, the Multus module is created with no NetworkAttachmentDefinitions set up. You can then create NetworkAttachmentDefinitions using the `kubectl` command after the module is installed.

The configuration file must contain one or more Kubernetes NetworkAttachmentDefinition Custom Resource Definitions (CRDs). These definitions set up the network attachments,

which configure the secondary interfaces for pods. You provide a Multus configuration file on the operator node in YAML format. For example:

```
---
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: bridge-conf
spec:
  config: '{
      ...
      }
    }'
---
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: macvlan-conf
spec:
  config: '{
      ...
      }
    }'
```

The Platform API Server uses the information contained in the configuration file when creating the Multus module to create any NetworkAttachmentDefinitions.

For information on creating the Multus configuration file, see the upstream Multus documentation.

Deploying the Multus Module

This section contains the information on how to install the Multus module. You must have a Kubernetes module installed before you install Multus. The Kubernetes module uses Flannel as the default Kubernetes pod networking CNI.

For the syntax to use to create a Multus module, see the `multus` option of the `olcnectl module create` command in Platform Command-Line Interface.

To deploy the Multus module:

1. Create and install a Kubernetes module. The name of the Kubernetes module in this example is `mycluster`.
2. Create a Multus module and associate it with the Kubernetes module named `mycluster` using the `--multus-kubernetes-module` option. In this example, the Multus module is named `mymultus`.

   ```
   olcnectl module create \
   --environment-name myenvironment \
   --module multus \
   --name mymultus \
   --multus-kubernetes-module mycluster
   ```

The `--module` option sets the module type to create, which is `multus`. You define the name of the Multus module using the `--name` option, which in this case is `mymultus`.

The `--multus-kubernetes-module` option sets the name of the Kubernetes module.

You can optionally provide the `--multus-config` option to set the location for a Multus configuration file. This file must be available on the operator node under the provided path. For information on creating this configuration file, see Prerequisites.

If you don't include all the required options when adding the module, you're prompted to provide them.

> **Important:**
>
> Multus is deployed to the Kubernetes `kube-system` namespace by default. If you're creating NetworkAttachmentDefinitions when you deploy the module using a configuration file, specify the namespace in which to create these with the `--multus-namespace` option. Both the Multus pods and the NetworkAttachmentDefinitions are created in this namespace when you use a configuration file. Any Kubernetes applications that use these NetworkAttachmentDefinitions must also be created in the namespace you specify with this option. For example:
>
> ```
> --multus-namespace default
> ```

3. Use the `olcnectl module install` command to install the Multus module. For example:

```
olcnectl module install \
--environment-name myenvironment \
--name mymultus
```

You can optionally use the `--log-level` option to set the level of logging displayed in the command output. By default, error messages are displayed. For example, you can set the logging level to show all messages when you include:

```
--log-level debug
```

The log messages are also saved as an operation log. You can view operation logs as commands are running, or when they've completed. For more information using operation logs, see Platform Command-Line Interface.

The Multus module is deployed into the Kubernetes cluster.

Verifying the Multus Module Deployment

You can verify the Multus module is deployed using the `olcnectl module instances` command on the operator node. For example:

```
olcnectl module instances \
--environment-name myenvironment
```

The output looks similar to:

```
INSTANCE                  MODULE        STATE
mymultus                  multus        installed
mycluster                 kubernetes    installed
...
```

Note the entry for `multus` in the `MODULE` column is in the `installed` state.

In addition, use the `olcnectl module report` command to review information about the module. For example, use the following command to review the Multus module named `mymultus` in `myenvironment`:

```
olcnectl module report \
--environment-name myenvironment \
--name mymultus \
--children
```

For more information on the syntax for the `olcnectl module report` command, see Platform Command-Line Interface.

3
Using Multus

This section contains a basic test to verify you can create a Kubernetes application that uses Multus. This example verifies you can use Multus to create a second network interface on a pod, which uses a network created with a NetworkAttachmentDefinitions CRD.

To create a second network interface:

1. On a control plane node, create a YAML file named `nad-bridge.yaml` that contains a NetworkAttachmentDefinition CRD for a bridged connection. Create a YAML file with the following contents:

   ```
   apiVersion: "k8s.cni.cncf.io/v1"
   kind: NetworkAttachmentDefinition
   metadata:
     name: bridge-conf
   spec:
     config: '{
         "cniVersion": "0.3.1",
         "type": "bridge",
         "bridge": "mybr0",
         "ipam": {
             "type": "host-local",
             "subnet": "192.168.12.0/24",
             "rangeStart": "192.168.12.10",
             "rangeEnd": "192.168.12.200"
         }
       }'
   ```

2. Use the `kubectl apply` command to create the NetworkAttachmentDefinition with the file:

   ```
   kubectl apply -f nad-bridge.yaml
   ```

   The NetworkAttachmentDefinition is created.

3. You can verify the NetworkAttachmentDefinition is created using:

   ```
   kubectl get network-attachment-definitions
   ```

   The output looks similar to:

   ```
   NAME          AGE
   bridge-conf   2m
   ```

To display more information about the NetworkAttachmentDefinition, use the `kubectl describe` command, for example:

```
kubectl describe network-attachment-definitions bridge-conf
```

4. Create a YAML file named `multus-pod.yaml` to create a pod running Oracle Linux 9 that uses the `bridge-conf` network, with the following contents:

```
apiVersion: v1
kind: Pod
metadata:
  name: samplepod
  annotations:
    k8s.v1.cni.cncf.io/networks: bridge-conf
spec:
  containers:
  - name: samplepod
    command: ["/bin/sh", "-c", "trap : TERM INT; sleep infinity &
wait"]
    image: container-registry.oracle.com/os/oraclelinux:9-slim
```

5. Start the pod using the `kubectl` command:

```
kubectl apply -f multus-pod.yaml
```

6. You can see the pod is created and running using:

```
kubectl get pods
```

The output looks similar to:

```
NAME                    READY   STATUS    RESTARTS   AGE
samplepod               1/1     Running   0          1m
```

7. Use the `kubectl describe` command to show the IP address assigned to the network interfaces of the `multus-nginx-pod`:

```
kubectl describe pod samplepod
```

The output looks similar to:

```
Name:             samplepod
Namespace:        default
...
Annotations:      k8s.v1.cni.cncf.io/network-status:
                    [{
                        "name": "cbr0",
                        "interface": "eth0",
                        "ips": [
                            "10.244.3.4"
                        ],
                        "mac": "86:e7:82:28:58:59",
```

```
                "default": true,
                "dns": {},
                "gateway": [
                    "10.244.3.1"
                ]
            },{
                "name": "default/bridge-conf",
                "interface": "net1",
                "ips": [
                    "192.168.12.10"
                ],
                "mac": "ca:bb:74:ca:9c:10",
                "dns": {}
            }]
        k8s.v1.cni.cncf.io/networks: bridge-conf
...
```

In this example, you can see the `net1` network interface is using the `bridge-conf` network, and has the IP address of `192.168.12.10`.

8. You can delete the resources creating in this example using:

```
kubectl delete pod samplepod
kubectl delete network-attachment-definitions bridge-conf
```

4

Removing the Multus Module

You can remove a deployment of the Multus module and leave the Kubernetes cluster in place. To do this, you remove the Multus module from the environment.

Use the `olcnectl module uninstall` command to remove the Multus module. For example, to uninstall the Multus module named `mymultus` in the environment named `myenvironment`:

```
olcnectl module uninstall \
--environment-name myenvironment \
--name mymultus
```

The Multus module is removed from the environment.