Oracle Cloud Native Environment

Operator Lifecycle Manager Module for Release 1.8

F87573-02
March 2026

Oracle Cloud Native Environment Operator Lifecycle Manager Module for Release 1.8,

F87573-02

Contents

Preface

> **Warning**
>
> Oracle Cloud Native Environment 1.8 is now in Sustaining Support. See Oracle Open Source Support Policies for more information. New security patches and bug fixes are no longer provided for this software.
>
> Migrate applications and data to Oracle Cloud Native Environment Release 2.<latest> as soon as possible.

This document contains information about setting up the Operator Lifecycle Manager module in Oracle Cloud Native Environment. The Operator Lifecycle Manager module is used to install and manage the lifecycle of Kubernetes operators in a Kubernetes cluster.

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 Operator Lifecycle Manager Module

> **Warning**
>
> Oracle Cloud Native Environment 1.8 is now in Sustaining Support. See Oracle Open Source Support Policies for more information. New security patches and bug fixes are no longer provided for this software.
>
> Migrate applications and data to Oracle Cloud Native Environment Release 2.<latest> as soon as possible.

A Kubernetes operator is a design pattern for writing code to automate tasks and extend Kubernetes. An operator is a set of concepts you can use to define a service for Kubernetes and helps to automate administrative services in Kubernetes.

The Operator Lifecycle Manager module installs an instance of Operator Lifecycle Manager into a Kubernetes cluster, which you can use to manage the installation and lifecycle of operators in a Kubernetes cluster. The Operator Lifecycle Manager is a package manager that interacts with operator registries. For more information about the Operator Lifecycle Manager, see the upstream Operator Lifecycle Manager documentation.

OperatorHub is an operator registry that contains upstream Kubernetes operators that you can use to deploy operators in a cluster. The OperatorHub is at:

https://operatorhub.io/

Operator Lifecycle Manager in many ways performs the same tasks as Helm. A major extra feature that Operator Lifecycle Manager provides is that it has built-in support to validate Custom Resource Definitions (CRDs) inside Kubernetes software. Operators with CRDs can use these to ensure dependencies are met and no interfaces are duplicated. Otherwise, Operator Lifecycle Manager manages deployments in a similar way to Helm.

2

Installing the Operator Lifecycle Manager Module

> **Warning**
>
> Oracle Cloud Native Environment 1.8 is now in Sustaining Support. See Oracle Open Source Support Policies for more information. New security patches and bug fixes are no longer provided for this software.
>
> Migrate applications and data to Oracle Cloud Native Environment Release 2.<latest> as soon as possible.

This chapter discusses how to install the Operator Lifecycle Manager module in Oracle Cloud Native Environment.

Deploying the Operator Lifecycle Manager Module

This section contains information on how to install the Operator Lifecycle Manager module. You must have a Kubernetes module installed before you install Operator Lifecycle Manager.

For the syntax to use to create a Operator Lifecycle Manager module, see the `operator-lifecycle-manager` option of the `olcnectl module create` command in Platform Command-Line Interface.

To deploy the Operator Lifecycle Manager module:

1. Create and install a Kubernetes module. The name of the Kubernetes module in this example is `mycluster`.
2. Create an Operator Lifecycle Manager module and associate it with the Kubernetes module named `mycluster` using the `--olm-kubernetes-module` option. In this example, the Operator Lifecycle Manager module is named `myolm`.

   ```
   olcnectl module create \
   --environment-name myenvironment \
   --module operator-lifecycle-manager \
   --name myolm \
   --olm-kubernetes-module mycluster
   ```

   The `--module` option sets the module type to create, which is `operator-lifecycle-manager`. You define the name of the Operator Lifecycle Manager module using the `--name` option, which in this case is `myolm`.

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

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

3. Use the `olcnectl module install` command to install the Operator Lifecycle Manager module. For example:

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

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 Operator Lifecycle Manager module is deployed into the Kubernetes cluster and the required containers are running in the `operator-lifecycle-manager` namespace.

Verifying the Operator Lifecycle Manager Module Deployment

You can verify the Operator Lifecycle Manager 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
myolm                    operator-lifecycle-manager      installed
mycluster                kubernetes                      installed
...
```

Note the entry for `operator-lifecycle-manager` 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 Operator Lifecycle Manager module named `myolm` in `myenvironment`:

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

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

To verify the Operator Lifecycle Manager containers are deployed, use the `kubectl` command on a control plane node to list the deployments running in the `operator-lifecycle-manager` namespace.

```
kubectl get deployments --namespace operator-lifecycle-manager
```

The output looks similar to:

```
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
catalog-operator   1/1     1            1           2m36s
olm-operator       1/1     1            1           2m36s
packageserver      2/2     2            2           2m30s
```

3

Using Operator Lifecycle Manager

> **Warning**
>
> Oracle Cloud Native Environment 1.8 is now in Sustaining Support. See Oracle Open Source Support Policies for more information. New security patches and bug fixes are no longer provided for this software.
>
> Migrate applications and data to Oracle Cloud Native Environment Release 2.<latest> as soon as possible.

This section contains basic tests to verify you can use Operator Lifecycle Manager.

Listing Operator Registries

You can show the available operator registries using the `kubectl` command on a control plane node:

```
kubectl get catalogsource --namespace operator-lifecycle-manager
```

The output looks similar to:

```
NAME                    DISPLAY               TYPE   PUBLISHER        AGE
operatorhubio-catalog   Community Operators   grpc   OperatorHub.io   3m35s
```

The OperatorHub registry is shown in the output. This is the default operator registry.

Installing Operators

To see all the operators that can be installed, use the `kubectl` command on a control plane node:

```
kubectl get packagemanifest
```

A list of the operators available on OperatorHub is displayed. These are all available to be installed by the Operator Lifecycle Manager. The following example shows you how to create an operator which is pulled from the OperatorHub.

To create an operator:

1. In a web browser, go to the OperatorHub and find the name of the operator you want to install. The OperatorHub is at:

   https://operatorhub.io/

   This example uses the `cert-manager` operator at:

https://operatorhub.io/operator/cert-manager

Click **Install**.

A dialog is displayed that shows the `kubectl create` command to deploy the operator. For example:

```
kubectl create -f https://operatorhub.io/install/cert-manager.yaml
```

Copy the URL in this command that contains the operator manifest YAML file.

2. On a control plane node, download the `cert-manager` operator manifest YAML file from the OperatorHub:

```
curl --remote-name https://operatorhub.io/install/cert-manager.yaml
```

3. Edit this manifest YAML file as needed.

> **Important**
>
> If an operator includes the following line in the `Subscription` section:
>
> `sourceNamespace: olm`
>
> Change this to:
>
> `sourceNamespace: operator-lifecycle-manager`
>
> Operator Lifecycle Manager runs in the `operator-lifecycle-manager` namespace, which is different to the upstream namespace.

Edit the file to change `sourceNamespace` to `operator-lifecycle-manager`.

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: my-cert-manager
  namespace: operators
spec:
  channel: stable
  name: cert-manager
  source: operatorhubio-catalog
  sourceNamespace: operator-lifecycle-manager
```

4. Use the `kubectl apply` command to deploy the `cert-manager` operator.

```
kubectl apply -f cert-manager.yaml
```

The output looks similar to:

```
subscription.operators.coreos.com/my-cert-manager created
```

The operator is deployed into the namespace set in the operator manifest file, which in this example is `operators`.

5. You can see the operator's `ClusterServiceVersion` information using:

   ```
   kubectl get csv --namespace operators
   ```

   The output looks similar to:

   ```
   NAME                    DISPLAY        VERSION   REPLACES
   PHASE
   cert-manager.v1.12.2    cert-manager   1.12.2    cert-manager.v1.11.4
   Succeeded
   ```

6. You can see the operator deployments using:

   ```
   kubectl get deployments --namespace operators
   ```

   The output looks similar to:

   ```
   NAME                      READY   UP-TO-DATE   AVAILABLE   AGE
   cert-manager              1/1     1            1           6m
   cert-manager-cainjector   1/1     1            1           6m
   cert-manager-webhook      1/1     1            1           6m
   ```

Removing Operators

To remove an operator and uninstall it, you need to remove the `Subscription` and `ClusterServiceVersion` resources.

The example in this document doesn't include a `Subscription` resource, but if the operator you want to delete includes one, delete it using:

```
kubectl delete subscription subscription-name --namespace namespace
```

You also need to delete the Kubernetes `ClusterServiceVersion` resource using:

```
kubectl delete csv csv-name --namespace namespace
```

To delete the `ClusterServiceVersion` for the `cert-manager` operator, on a control plane node, run:

```
kubectl delete csv --namespace operators cert-manager.v1.12.2
```

The output looks similar to:

```
clusterserviceversion.operators.coreos.com "cert-manager.v1.12.2" deleted
```

4

Removing the Operator Lifecycle Manager Module

> **Warning**
>
> Oracle Cloud Native Environment 1.8 is now in Sustaining Support. See Oracle Open Source Support Policies for more information. New security patches and bug fixes are no longer provided for this software.
>
> Migrate applications and data to Oracle Cloud Native Environment Release 2.<latest> as soon as possible.

You can remove the Operator Lifecycle Manager module and leave the Kubernetes cluster in place. To do this, you remove the Operator Lifecycle Manager module from the environment.

Use the `olcnectl module uninstall` command to remove the Operator Lifecycle Manager module. For example, to uninstall the Operator Lifecycle Manager module named `myolm` in the environment named `myenvironment`:

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

The Operator Lifecycle Manager module is removed from the environment.