

The software described in this documentation is either no longer supported or is in extended support.
Oracle recommends that you upgrade to a current supported release.

Oracle® Cloud Native Environment

Platform Command-Line Interface for Release 1.4

ORACLE®

F37063-10
March 2023

The software described in this documentation is either no longer supported or is in extended support.
Oracle recommends that you upgrade to a current supported release.

Oracle Legal Notices

Copyright © 2020, 2023, Oracle and/or its affiliates.

Table of Contents

Preface	v
1 Introduction to the Platform CLI	1
1.1 Getting Syntax Help	1
1.2 Setting the Platform API Server	4
1.3 Using Global Flags	5
2 Using a Configuration File	9
2.1 Creating a Configuration File	9
2.2 Installing Using a Configuration File	12
2.3 Adding Modules or Environments Using a Configuration File	13
2.4 Uninstalling Specific Modules or Environments Using a Configuration File	14
2.5 Scaling a Cluster Using a Configuration File	15
2.6 Updating and Upgrading Using a Configuration File	16
2.7 Uninstalling Using a Configuration File	17
3 Reporting Oracle Cloud Native Environment Details	19
3.1 Reporting Environment Information	19
3.2 Reporting Detailed Module Information	19
3.3 Filtering Report Responses	20
3.4 Changing Report Format	21
4 Platform CLI Commands	23
4.1 Environment Create	23
4.2 Environment Delete	26
4.3 Environment Report	27
4.4 Module Backup	28
4.5 Module Create	29
4.6 Module Install	41
4.7 Module Instances	42
4.8 Module List	43
4.9 Module Property Get	43
4.10 Module Property List	44
4.11 Module Report	44
4.12 Module Restore	46
4.13 Module Uninstall	46
4.14 Module Update	47
4.15 Module Validate	52
4.16 Template	53

Preface

This document contains information about the Oracle Cloud Native Environment Platform Command-Line Interface. This document provides the full syntax of the `olcnectl` command, usage and examples.

Document generated on: 2023-03-14 (revision: 1279)

Audience

This document is written for system administrators and developers who want to use the Oracle Cloud Native Environment Platform Command-Line Interface (the `olcnectl` command). It is assumed that readers have a general understanding of the Oracle Linux operating system and container concepts.

Related Documents

The latest version of this document and other documentation for this product are available at:

<https://docs.oracle.com/en/operating-systems/olcne/>

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.
<i>italic</i>	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
<code>monospace</code>	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/>.

For information about the accessibility of the Oracle Help Center, see the Oracle Accessibility Conformance Report at <https://www.oracle.com/corporate/accessibility/templates/t2-11535.html>.

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

The software described in this documentation is either no longer supported or is in extended support.
Oracle recommends that you upgrade to a current supported release.

Diversity and Inclusion

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.

Chapter 1 Introduction to the Platform CLI

The Oracle Cloud Native Environment Platform Command-Line Interface, `olcnectl`, is used to configure, deploy and manage the components of Oracle Cloud Native Environment. The `olcnectl` command is installed using the `olcnectl` package on an operator node. For information on setting up an operator node, see [Getting Started](#).

You interact with `olcnectl` by entering commands with a series of options. The Platform CLI syntax is:

```
olcnectl command [command_options|-h|--help]
```

The full syntax and options for each command is provided in [Chapter 4, Platform CLI Commands](#).

When you use the `olcnectl` command, you are prompted for any missing options.

1.1 Getting Syntax Help

You can get help on the syntax for `olcnectl` commands using the `--help` option. For example, to show the command options available for the `olcnectl` command, enter:

```
olcnectl --help

A CLI that talks to an Oracle Cloud Native Environment Platform API Server endpoint,
facilitating deployment and management of Kubernetes clusters and their resources

Usage:
  olcnectl [command]

Available Commands:
  environment Environment operations
  help         Help about any command
  module       Modules that can be modified in an environment
  template     Generate a configuration file template

Flags:
  -h, --help  help for olcnectl

Use "olcnectl [command] --help" for more information about a command.
```

The **Available Commands** section lists any available commands for the `olcnectl` command. In this case, you can use the commands `olcnectl environment`, `olcnectl help`, `olcnectl module` and `olcnectl template`.

The **Flags** section lists the available command options you can use.

The `olcnectl help` command is the equivalent of using `olcnectl --help`. That is, it prints out the help for the `olcnectl` command.

You can drill further down into the help system by providing the `--help` option to the commands listed in the **Available Commands** section. For example, to show the available commands and options for the `olcnectl module` command, enter:

```
olcnectl module --help

Modules that are used to customize your environment

Usage:
  olcnectl module [command]

Available Commands:
```

Getting Syntax Help

```
backup      backup a module
create      Create a module
get         Get a module
install     Install a module
instances  List all module instances that are defined in an environment
list       Show all modules that can be installed
property    Commands that interact with module properties
report     Display the report of the selected module instance
restore    restore a module
uninstall  Uninstall a module
update     Update a module
validate   Validate that an module can be installed

Flags:
-a, --api-server string      Platform API Server to talk to. If this is not specified ...
--config-file string        Location of configuration file that contains the ...
-h, --help                  help for olcnectl
--olcn-ca-path string       Optional path to a predefined CA or the a destination if ...
--olcne-node-cert-path string Optional path to a predefined Key or the a destination ...
--olcne-node-key-path string Optional path to a predefined Cert or the a destination ...
--olcne-tls-cipher-suites string TLS Cipher Suites, Possible value(s) (comma separated): ...
--olcne-tls-max-version string TLS Maximum Version, Default value: VersionTLS12, ...
--olcne-tls-min-version string TLS Minimum Version, Default value: VersionTLS12, ...
--secret-manager-type string Manager that will handle the secrets. Options are: file, ...
--update-config             When defined the global arguments will be written to a ...
--vault-address string      Address of Vault. Default: https://127.0.0.1:8200 or ...
--vault-cert-sans string    Sans that will passed to Vault to generate the Platform ...
--vault-token string        Token to authentic with Vault

Use "olcnectl module [command] --help" for more information about a command.
```

Again, the [Available Commands](#) section lists any sub commands available for the command. In this case, you can use commands such as `olcnectl module backup`, `olcnectl module create`, `olcnectl module get` and so on.

The [Flags](#) section lists the options that can be used by all subcommands.

Drilling further down into the help system you can see the `olcnectl module property` command has a further two options, `get` and `list`.

```
olcnectl module property --help

Commands that interact with module properties

Usage:
  olcnectl module property [command]

Available Commands:
  get      Gets the value of one or more properties
  list     Show all properties for a module

Flags:
  -h, --help  help for property

Global Flags:
  -a, --api-server string      Platform API Server to talk to. If this is not ...
  --config-file string        Location of configuration file that contains ...
  --olcne-ca-path string       Optional path to a predefined CA or the a ...
  ...
Use "olcnectl module property [command] --help" for more information about a command.
```

A new set of command options is listed under The [Global Flags](#) section. The options shown in this section are global flags, which are used by all `olcnectl` subcommands. For more information on global flags, see [Section 1.3, "Using Global Flags"](#).

To get a list of the command options, you need to include the full command with the `--help` option. In this case, the `olcnectl module property get` command has four options as shown in the [Flags](#) section.

```
olcnectl module property get --help

Given a list of properties, fetch the value of each for a specific module

Usage:
  olcnectl module property get [flags]

Flags:
  -E, --environment-name string  Name of the environment
  -h, --help                     help for get
  -N, --name string              Name of the module
  -P, --property strings         Names of properties to fetch
  ...
```

The help system for the `olcnectl module create` and the `olcnectl module update` commands behaves differently to the other uses of the `--help` option. As there are multiple modules within an environment, you must provide information about a module in order for the Platform CLI to display the appropriate help. To display the help for the `olcnectl module create` command, enter:

```
olcnectl module create --help

Create a module in a environment

Usage:
  olcnectl module create [flags]

Flags:
  -E, --environment-name string Name of the environment
  -h, --help help for create
  -M, --module strings Module to create
  -N, --name strings Name to assign the module
  ...
```

To see the options for creating each module you must use the `--module` option and provide the module type. The module types are listed in [Section 4.5, "Module Create"](#). For example, to get help on creating a Kubernetes module you specify the module type as `kubernetes`:

```
olcnectl module create --help --module kubernetes

Create a module in a environment

Usage:
  olcnectl module create [flags]

Flags:
  -o, --apiserver-advertise-address string  (DEPRECATED) Advertised address for internal ...
  -b, --apiserver-bind-port string          Kubernetes API Server bind port (default "6443")
  -B, --apiserver-bind-port-alt string      Port for the Kubernetes API Server to bind to if ...
  -e, --apiserver-cert-extra-sans string    Kubernetes API Server extra sans
  -r, --container-registry string          Container Registry that holds the kubernetes images
  -E, --environment-name string            Name of the environment
  -h, --help                               help for create
  -x, --kube-proxy-mode string              Routing mode for the Kubernetes proxy (default ...
  -v, --kube-version string                Kubernetes version (default "1.17.4")
  ...
```

Similarly, to get help on the `olcnectl module update` command use:

```
olcnectl module update --help
```

Setting the Platform API Server

```
Update a module

Usage:
  olcnectl module update [flags]

Flags:
  -E, --environment-name string  Name of the environment
  -F, --force                    Update without prompting
  -g, --generate-scripts        Generate a script for each node that takes all suggested actions
  -h, --help                    help for update
  -N, --name strings            Modules to update
  ...
```

The output shows a `--name` option. This is the option you use to specify the module. This example shows the output for the `olcnectl module update --help` command for a Kubernetes module named `mycluster`:

```
olcnectl module update --help --name mycluster

Update a module

Usage:
  olcnectl module update [flags]

Flags:
  -E, --environment-name string  Name of the environment
  -F, --force                    Update without prompting
  -g, --generate-scripts        Generate a script for each node that takes all suggested actions
  -h, --help                    help for update
  -v, --kube-version string      Kubernetes version (default "1.21.14-3")
  -m, --master-nodes string      A comma separated list of master nodes
  -N, --name strings            Modules to update
  -w, --worker-nodes string      A comma separated list of worker nodes
  ...
```

The output shows the options you can use to scale or update/upgrade the Kubernetes module.

1.2 Setting the Platform API Server

The Platform CLI connects to an Oracle Cloud Native Environment Platform API Server. You can use an operator node with the Platform CLI installed to connect to multiple Platform API Server instances. You specify the Platform API Server using the `olcnectl --api-server api_server_address:8091` option. This enables you to use a single operator node to manage multiple environments. For example, to connect to a Platform API Server on `apiserver.example.com`, you would use:

```
olcnectl module property list \
--api-server apiserver.example.com:8091 \
--environment-name myenvironment \
--name mycluster
```

When you create an environment with the `olcnectl environment create` command you can optionally include the `--update-config` option. This option writes information about the environment to a local configuration file at `$HOME/.olcne/olcne.conf`, and this configuration is used for future calls to the Platform API Server. If you use this option, you do not need to specify the Platform API Server in future `olcnectl` commands.

For example, if you create an environment using the `--update-config` option:

```
olcnectl environment create \
--api-server 127.0.0.1:8091 \
--environment-name myenvironment \
```

```
--secret-manager-type vault \  
--vault-token s.3QKNuRoTqLbjXaGBOmO6Psjh \  
--vault-address https://192.0.2.20:8200 \  
--update-config
```

When you write all future `olcnectl` commands you can omit the `--api-server` option. For example:

```
olcnectl module property list \  
--environment-name myenvironment \  
--name mycluster
```

You can also set an environment variable to set the Platform API Server. You can do this using the `$OLCNE_API_SERVER_BIN` environment variable on the operator node. For example, to set the Platform API Server to the localhost, use:

```
export OLCNE_API_SERVER_BIN=127.0.0.1:8091
```

1.3 Using Global Flags

There are a number of global flags, or command options, that can be used with all `olcnectl` commands.

These options are most often used when creating an environment using the `olcnectl environment create` command, however they can also be used with all other `olcnectl` commands. The global options are:

```
[{-a|--api-server} api_server_address:8091]  
[--config-file path]  
[--secret-manager-type {file|vault}]  
[--update-config]  
[--olcne-ca-path ca_path]  
[--olcne-node-cert-path node_cert_path]  
[--olcne-node-key-path node_key_path]  
[--olcne-tls-cipher-suites ciphers]  
[--olcne-tls-max-version version]  
[--olcne-tls-min-version version]  
[--vault-address vault_address]  
[--vault-cert-sans vault_cert_sans]  
[--vault-token vault_token]
```

Where:

`{-a|--api-server}`
`api_server_address:8091`

The Platform API Server for the environment. This is the host running the `olcne-api-server` service in an environment. The value of `api_server_address` is the IP address or hostname of the Platform API Server. The port number is the port on which the `olcne-api-server` service is available. The default port is `8091`.

If a Platform API Server is not specified, a local instance is used. If no local instance is set up, it is configured in the `$HOME/.olcne/olcne.conf` file.

For more information on setting the Platform API Server see [Section 1.2, "Setting the Platform API Server"](#).

This option maps to the `$OLCNE_API_SERVER_BIN` environment variable. If this environment variable is set it takes precedence over and overrides the Platform CLI setting.

`--config-file path`

The location of a YAML file that contains the configuration information for the environment(s) and module(s). The filename extension must be

Using Global Flags

	either <code>yaml</code> or <code>yml</code> . When you use this option, any other command line options are ignored, with the exception of the <code>--force</code> option. Only the information contained in the configuration file is used.
<code>--secret-manager-type</code> <code>{file vault}</code>	The secrets manager type. The options are <code>file</code> or <code>vault</code> . Use <code>file</code> for certificates saved on the nodes and use <code>vault</code> for certificates managed by Vault.
<code>--update-config</code>	<p>Writes the global arguments for an environment to a local configuration file which is used for future calls to the Platform API Server. If this option has not been used previously, global arguments must be specified for every Platform API Server call.</p> <p>The global arguments configuration information is saved to <code>\$HOME/.olcne/olcne.conf</code> on the local host.</p> <p>If you use Vault to generate certificates for nodes, the certificate is saved to <code>\$HOME/.olcne/certificates/environment_name/</code> on the local host.</p>
<code>--olcne-ca-path</code> <i>ca_path</i>	<p>The path to a predefined Certificate Authority certificate, or the destination of the certificate if using a secrets manager to download the certificate. The default is <code>/etc/olcne/certificates/ca.cert</code>, or gathered from the local configuration if the <code>--update-config</code> option is used.</p> <p>This option maps to the <code>\$OLCNE_SM_CA_PATH</code> environment variable. If this environment variable is set it takes precedence over and overrides the Platform CLI setting.</p>
<code>--olcne-node-cert-path</code> <i>node_cert_path</i>	<p>The path to a predefined certificate, or the a destination if using a secrets manager to download the certificate. The default is <code>/etc/olcne/certificates/node.cert</code>, or gathered from the local configuration if the <code>--update-config</code> option is used.</p> <p>This option maps to the <code>\$OLCNE_SM_CERT_PATH</code> environment variable. If this environment variable is set it takes precedence over and overrides the Platform CLI setting.</p>
<code>--olcne-node-key-path</code> <i>node_key_path</i>	<p>The path to a predefined key, or the destination of the key if using a secrets manager to download the key. The default is <code>/etc/olcne/certificates/node.key</code>, or gathered from the local configuration if the <code>--update-config</code> option is used.</p> <p>This option maps to the <code>\$OLCNE_SM_KEY_PATH</code> environment variable. If this environment variable is set it takes precedence over and overrides the Platform CLI setting.</p>
<code>--olcne-tls-cipher-suites</code> <i>ciphers</i>	<p>The TLS cipher suites to use for Oracle Cloud Native Environment services (the Platform Agent and Platform API Server). Enter one or more in a comma separated list. The options are:</p> <ul style="list-style-type: none">• <code>TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA</code>• <code>TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256</code>

Using Global Flags

- TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305
- TLS_ECDHE_ECDSA_WITH_RC4_128_SHA
- TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305
- TLS_ECDHE_RSA_WITH_RC4_128_SHA
- TLS_RSA_WITH_3DES_EDE_CBC_SHA
- TLS_RSA_WITH_AES_128_CBC_SHA
- TLS_RSA_WITH_AES_128_CBC_SHA256
- TLS_RSA_WITH_AES_128_GCM_SHA256
- TLS_RSA_WITH_AES_256_CBC_SHA
- TLS_RSA_WITH_AES_256_GCM_SHA384
- TLS_RSA_WITH_RC4_128_SHA

For example:

```
--olcne-tls-cipher-suites  
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 , TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
```

This option maps to the `$OLCNE_TLS_CIPHER_SUITES` environment variable. If this environment variable is set it takes precedence over and overrides the Platform CLI setting.

<code>--olcne-tls-max-version version</code>	<p>The TLS maximum version for Oracle Cloud Native Environment components. The default is <code>VersionTLS12</code>. Options are:</p> <ul style="list-style-type: none">• <code>VersionTLS10</code>• <code>VersionTLS11</code>• <code>VersionTLS12</code>• <code>VersionTLS13</code> <p>This option maps to the <code>\$OLCNE_TLS_MAX_VERSION</code> environment variable. If this environment variable is set it takes precedence over and overrides the Platform CLI setting.</p>
<code>--olcne-tls-min-version version</code>	<p>The TLS minimum version for Oracle Cloud Native Environment components. The default is <code>VersionTLS12</code>. Options are:</p> <ul style="list-style-type: none">• <code>VersionTLS10</code>• <code>VersionTLS11</code>• <code>VersionTLS12</code>• <code>VersionTLS13</code> <p>This option maps to the <code>\$OLCNE_TLS_MIN_VERSION</code> environment variable. If this environment variable is set it takes precedence over and overrides the Platform CLI setting.</p>
<code>--vault-address vault_address</code>	<p>The IP address of the Vault instance. The default is <code>https://127.0.0.1:8200</code>, or gathered from the local configuration if the <code>--update-config</code> option is used.</p>
<code>--vault-cert-sans vault_cert_sans</code>	<p>Subject Alternative Names (SANs) to pass to Vault to generate the Oracle Cloud Native Environment certificate. The default is <code>127.0.0.1</code>, or gathered from the local configuration if the <code>--update-config</code> option is used.</p>
<code>--vault-token vault_token</code>	<p>The Vault authentication token.</p>

Chapter 2 Using a Configuration File

To simplify creating and managing environments and modules, you can use a configuration file. The configuration file includes all information about the environments and modules you want to create. Using a configuration file saves repeated entries of Platform CLI command options.

You can use a configuration file using the `--config-file` option with any Platform CLI command as it is a global command option. When you use the `--config-file` option with a Platform CLI command, any other command line options are ignored, with the exception of the `--force` option. Only the information contained in the configuration file is used with an `olcnectl` command.

The following sections contain information on writing a configuration file and using a configuration file create and remove environments and modules. There are more uses for the configuration file than this chapter describes. The use cases described in this chapter are the most common ways to use a configuration file.

2.1 Creating a Configuration File

The configuration file must be valid YAML with a file extension of `yaml` or `yml`. The basic format of components in the configuration file is:

```
environments:
- environment-name: name
  globals:
    key: value
  modules:
    - module: name
      name: name
      args:
        key: value
    - module: name
      name: name
      args:
        key: value
- environment-name: name
  globals:
    key: value
  modules:
    - module: name
      name: name
      args:
        key: value
    - module: name
      name: name
      args:
        key: value
```

The `olcnectl template` command is useful to create a YAML file that contains some basic configuration options to start a configuration file for your environment.

```
olcnectl template
```

This command creates a file named `config-file-template.yaml` in the local directory. You can edit this file to suit your needs.

The configuration file should contain `key: value` pairs for `olcnectl` command options. For example, when creating an environment, you might use an `olcnectl` command like:

```
olcnectl environment create \
```

The software described in this documentation is either no longer supported or is in extended support.
Oracle recommends that you upgrade to a current supported release.

Creating a Configuration File

```
--api-server 127.0.0.1:8091 \  
--environment-name myenvironment \  
--secret-manager-type vault \  
--vault-token s.3QKNuRoTqLbjXaGBOm06Psjh \  
--vault-address https://192.0.2.20:8200 \  
--update-config
```

To represent this same information in YAML format in the configuration file, you would use:

```
environments:  
  - environment-name: myenvironment  
    globals:  
      api-server: 127.0.0.1:8091  
      secret-manager-type: vault  
      vault-token: s.3QKNuRoTqLbjXaGBOm06Psjh  
      vault-address: https://192.0.2.20:8200  
      update-config: true
```

Notice that the `olcnectl environment create` command options to create the environment map directly to the YAML `key: value` pairs.

When you write the `modules` section, you can use any `olcnectl module` command option that relates to modules. That is, any `olcnectl module` command option that can be used with a module can be included in the `module` section. The `args` section for a module should only contain the options available with the `olcnectl module create` command. Any other options should be under the main `module` set of options.

In this example, the `--generate-scripts` and `--force` options are not valid with the `olcnectl module create` command, but they are valid options for the `olcnectl module validate` or `olcnectl module uninstall` options. The `generate-scripts` and `force` options should not be added as module `args`, instead they should be listed under the `module: kubernetes` section.

```
...  
modules:  
  - module: kubernetes  
    name: mycluster  
    generate-scripts: true  
    force: true  
    args:  
      kube-version: 1.21.14-3  
      container-registry: container-registry.oracle.com/olcne  
      load-balancer: lb.example.com:6443  
      master-nodes:  
        - control1.example.com:8090  
        - control2.example.com:8090  
        - control3.example.com:8090  
      worker-nodes:  
        - worker1.example.com:8090  
        - worker2.example.com:8090  
        - worker3.example.com:8090  
    selinux: enforcing  
    restrict-service-externalip: true  
    restrict-service-externalip-ca-cert: /etc/olcne/certificates/restrict_external_ip/ca.cert  
    restrict-service-externalip-tls-cert: /etc/olcne/certificates/restrict_external_ip/node.cert  
    restrict-service-externalip-tls-key: /etc/olcne/certificates/restrict_external_ip/node.key
```

If you do not provide all mandatory options for a command, you are prompted for them when you use the configuration file with `olcnectl`. If you do not supply a value for a key, the default for that `olcnectl` command option is used, or if there is no default value, that key is ignored. If you add key values that are not valid, an error is displayed to help you correct the invalid option. If you add keys that are not valid, they are ignored.

Creating a Configuration File

Do not include the `--config-file` option for any `olcnectl` commands in the configuration file. This option is ignored and cannot be used in a configuration file.

The order of the components in the YAML file is important. The components should be in the same order as you would create them using the Platform CLI.

For example, this file creates two environments, the first environment includes only the Kubernetes module. The second environment includes the Kubernetes module, the Helm module, the Operator Lifecycle Manager module (which requires Helm), the Istio module (which also requires Helm) and finally, the Oracle Cloud Infrastructure Container Storage Interface module (which requires Helm as well). Both environments and all modules can be created and installed using a single set of `olcnectl` commands.

```
environments:
- environment-name: myenvironment1
  globals:
    api-server: 127.0.0.1:8091
    secret-manager-type: file
    olcne-ca-path: /etc/olcne/certificates/ca.cert
    olcne-node-cert-path: /etc/olcne/certificates/node.cert
    olcne-node-key-path: /etc/olcne/certificates/node.key
  modules:
    - module: kubernetes
      name: mycluster1
      args:
        container-registry: container-registry.oracle.com/olcne
        load-balancer: lb.example.com:6443
        master-nodes:
          - control1.example.com:8090
          - control2.example.com:8090
          - control3.example.com:8090
        worker-nodes:
          - worker1.example.com:8090
          - worker2.example.com:8090
          - worker3.example.com:8090
        selinux: enforcing
        restrict-service-externalip: true
        restrict-service-externalip-ca-cert: /etc/olcne/certificates/restrict_external_ip/ca.cert
        restrict-service-externalip-tls-cert: /etc/olcne/certificates/restrict_external_ip/node.cert
        restrict-service-externalip-tls-key: /etc/olcne/certificates/restrict_external_ip/node.key
    - environment-name: myenvironment2
      globals:
        api-server: 127.0.0.1:8091
        secret-manager-type: file
        olcne-ca-path: /etc/olcne/certificates/ca.cert
        olcne-node-cert-path: /etc/olcne/certificates/node.cert
        olcne-node-key-path: /etc/olcne/certificates/node.key
      modules:
        - module: kubernetes
          name: mycluster2
          args:
            container-registry: container-registry.oracle.com/olcne
            load-balancer: lb.example.com:6443
            master-nodes:
              - control4.example.com:8090
              - control5.example.com:8090
              - control6.example.com:8090
            worker-nodes:
              - worker4.example.com:8090
              - worker5.example.com:8090
              - worker6.example.com:8090
            node-labels: failure-domain.beta.kubernetes.io/zone=US-ASHBURN-AD-2
            node-ocids:
              - control4.example.com=ocid1.instance.oc1.iad...
```

Installing Using a Configuration File

```
- control5.example.com=ocidl.instance.oc1.iad...
- control6.example.com=ocidl.instance.oc1.iad...
- worker4.example.com=ocidl.instance.oc1.iad...
- worker5.example.com=ocidl.instance.oc1.iad...
- worker6.example.com=ocidl.instance.oc1.iad...
selinux: enforcing
restrict-service-externalip: true
restrict-service-externalip-ca-cert: /etc/olcne/certificates/restrict_external_ip/ca.cert
restrict-service-externalip-tls-cert: /etc/olcne/certificates/restrict_external_ip/node.cert
restrict-service-externalip-tls-key: /etc/olcne/certificates/restrict_external_ip/node.key
- module: helm
  name: myhelm
  args:
    helm-kubernetes-module: mycluster2
- module: operator-lifecycle-manager
  name: myolm
  args:
    olm-helm-module: myhelm
- module: istio
  name: myistio
  args:
    istio-helm-module: myhelm
- module: oci-csi
  name: myoci
  args:
    oci-csi-helm-module: myhelm
    oci-region: us-ashburn-1
    oci-tenancy: ocidl.tenancy.oc1...
    oci-compartment: ocidl.compartment.oc1...
    oci-user: ocidl.user.oc1...
    oci-fingerprint: b5:52:...
    oci-private-key: /home/opc/.oci/oci_api_key.pem
```

2.2 Installing Using a Configuration File

This section contains an example of using a configuration file to create an environment and deploy Kubernetes into it.

The configuration file for this is named `myenvironment.yaml` and contains:

```
environments:
- environment-name: myenvironment
  globals:
    api-server: 127.0.0.1:8091
    secret-manager-type: file
    olcne-ca-path: /etc/olcne/certificates/ca.cert
    olcne-node-cert-path: /etc/olcne/certificates/node.cert
    olcne-node-key-path: /etc/olcne/certificates/node.key
  modules:
- module: kubernetes
  name: mycluster
  args:
    container-registry: container-registry.oracle.com/olcne
    load-balancer: lb.example.com:6443
  master-nodes:
    - control11.example.com:8090
    - control12.example.com:8090
    - control13.example.com:8090
  worker-nodes:
    - worker1.example.com:8090
    - worker2.example.com:8090
    - worker3.example.com:8090
  selinux: enforcing
  restrict-service-externalip: true
```

```
restrict-service-externalip-ca-cert: /etc/olcne/certificates/restrict_external_ip/ca.cert
restrict-service-externalip-tls-cert: /etc/olcne/certificates/restrict_external_ip/node.cert
restrict-service-externalip-tls-key: /etc/olcne/certificates/restrict_external_ip/node.key
```

Use the same commands as you would usually use to create an environment and deploy the Kubernetes module, but instead of passing all the command options using the Platform CLI, simply provide the location of the configuration file.

To create the environment and deploy Kubernetes, on the operator node:

1. Use the `olcnectl environment create` command with the `--config-file` option:

```
olcnectl environment create \
--config-file myenvironment.yaml
```

The environment is created and ready to use to install the Kubernetes module. If you have multiple environments set up in your configuration file, they are all created using this one step.

2. Use the `olcnectl module create` command to create the Kubernetes module.

```
olcnectl module create \
--config-file myenvironment.yaml
```

If you have multiple modules set up in your configuration file, they are all created using this one step.

3. You should also validate the module is able to be installed on the nodes. Use the `olcnectl module validate` command to validate the module.

```
olcnectl module validate \
--config-file myenvironment.yaml
```

If you have multiple modules set up in your configuration file, they are all validated.

4. The last step is to install the module. Use the `olcnectl module install` command to install the module.

```
olcnectl module install \
--config-file myenvironment.yaml
```

If you have multiple modules set up in your configuration file, they are all installed.

5. You can verify the Kubernetes module is deployed and the nodes are set up using the `olcnectl module instances` command.

```
olcnectl module instances \
--config-file myenvironment.yaml
INSTANCE                                MODULE                                STATE
control1.example.com:8090               node                                  installed
control2.example.com:8090               node                                  installed
control3.example.com:8090               node                                  installed
worker1.example.com:8090                 node                                  installed
worker2.example.com:8090                 node                                  installed
worker3.example.com:8090                 node                                  installed
mycluster                                kubernetes                            installed
```

2.3 Adding Modules or Environments Using a Configuration File

If you want to add modules or environments to your deployment, add them to your configuration file, then run the `olcnectl` commands to add them to your deployment. For example, to add the Operator Lifecycle

Manager module to an existing Kubernetes deployment, create a file similar to the following. This file is the same as that used in [Section 2.2, “Installing Using a Configuration File”](#), to create an environment and deploy Kubernetes, with the addition of the Helm and Operator Lifecycle Manager modules.

```
environments:
- environment-name: myenvironment
  globals:
    api-server: 127.0.0.1:8091
    secret-manager-type: file
    olcne-ca-path: /etc/olcne/certificates/ca.cert
    olcne-node-cert-path: /etc/olcne/certificates/node.cert
    olcne-node-key-path: /etc/olcne/certificates/node.key
  modules:
    - module: kubernetes
      name: mycluster
      args:
        container-registry: container-registry.oracle.com/olcne
        load-balancer: lb.example.com:6443
        master-nodes:
          - control1.example.com:8090
          - control2.example.com:8090
          - control3.example.com:8090
        worker-nodes:
          - worker1.example.com:8090
          - worker2.example.com:8090
          - worker3.example.com:8090
        selinux: enforcing
        restrict-service-externalip: true
        restrict-service-externalip-ca-cert: /etc/olcne/certificates/restrict_external_ip/ca.cert
        restrict-service-externalip-tls-cert: /etc/olcne/certificates/restrict_external_ip/node.cert
        restrict-service-externalip-tls-key: /etc/olcne/certificates/restrict_external_ip/node.key
    - module: helm
      name: myhelm
      args:
        helm-kubernetes-module: mycluster
    - module: operator-lifecycle-manager
      name: myolm
      args:
        olm-helm-module: myhelm
```

Install the Helm module and the Operator Lifecycle Manager module using the `olcnectl module` commands.

```
olcnectl module create \
--config-file myenvironment.yaml
olcnectl module validate \
--config-file myenvironment.yaml
olcnectl module install \
--config-file myenvironment.yaml
```

The additional Helm and Operator Lifecycle Manager modules are installed into the existing Kubernetes cluster in the environment.

2.4 Uninstalling Specific Modules or Environments Using a Configuration File

As the Platform API Server acts upon all the information contained in a configuration file, if you want to remove specific components from your deployment, while leaving other components, you need to create a separate configuration file with only the components you want to remove. The new configuration file includes only the information about the environment and module(s) you want to uninstall.

For example, to remove the Helm and Operator Lifecycle Manager modules and not the Kubernetes module in an environment, create a file similar to the following. This file is the same as used in [Section 2.3, “Adding Modules or Environments Using a Configuration File”](#), without the information about the Kubernetes module. Specify the environment in which the modules are deployed, and only the modules you want to remove.

```
environments:
- environment-name: myenvironment
  globals:
    api-server: 127.0.0.1:8091
    secret-manager-type: file
    olcne-ca-path: /etc/olcne/certificates/ca.cert
    olcne-node-cert-path: /etc/olcne/certificates/node.cert
    olcne-node-key-path: /etc/olcne/certificates/node.key
  modules:
    - module: helm
      name: myhelm
      args:
        helm-kubernetes-module: mycluster
    - module: operator-lifecycle-manager
      name: myolm
      args:
        olm-helm-module: myhelm
```

The filename in this example is `myenvironment-olm.yaml`.



Important

Make sure you confirm the configuration file is correct before you use it to uninstall modules in order to maintain the integrity of your deployment.

Uninstall the Helm and Operator Lifecycle Manager modules using the `olcnectl module uninstall` command. Remember to use the `--force` option to make sure the modules are removed in the correct order by the Platform API Server.

```
olcnectl module uninstall \
--config-file myenvironment-olm.yaml \
--force
```

The Helm and Operator Lifecycle Manager modules are uninstalled from the environment, while leaving the Kubernetes module untouched.

2.5 Scaling a Cluster Using a Configuration File

The information in this section shows you how to scale a Kubernetes cluster using a configuration file. For more information about scaling a cluster and preparing nodes, see [Container Orchestration](#).

To scale a Kubernetes cluster using a configuration file, simply change the nodes listed in the Kubernetes module and use the `olcnectl module update` command to apply the changes to the module. For example, to add nodes to an existing cluster that has the following listed in the configuration file:

```
...
modules:
- module: kubernetes
  name: mycluster
  args:
    container-registry: container-registry.oracle.com/olcne
    load-balancer: lb.example.com:6443
    master-nodes:
      - controll1.example.com:8090
```

The software described in this documentation is either no longer supported or is in extended support.
Oracle recommends that you upgrade to a current supported release.

Updating and Upgrading Using a Configuration File

```
...
- control2.example.com:8090
- control3.example.com:8090
worker-nodes:
- worker1.example.com:8090
- worker2.example.com:8090
- worker3.example.com:8090
...
```

Add the new nodes to the configuration file. In this case there are two additional control plane nodes and one additional worker node.

```
...
modules:
- module: kubernetes
  name: mycluster
  args:
    container-registry: container-registry.oracle.com/olcne
    load-balancer: lb.example.com:6443
  master-nodes:
    - control11.example.com:8090
    - control12.example.com:8090
    - control13.example.com:8090
    - control14.example.com:8090
    - control15.example.com:8090
  worker-nodes:
    - worker1.example.com:8090
    - worker2.example.com:8090
    - worker3.example.com:8090
    - worker4.example.com:8090
...
```

Use the `olcnectl module update` command to scale up the cluster.

```
olcnectl module update \
--config-file myenvironment.yaml
```

The Platform API Server backs up the cluster and adds the new nodes.

To scale down a cluster, perform the same steps, except delete the information about the nodes you want to remove from the cluster from the configuration file.

2.6 Updating and Upgrading Using a Configuration File

You can use the configuration file when you update or upgrade modules. For more information about updating or upgrading modules, see [Updates and Upgrades](#).

To update all modules to the latest available errata release, use the `olcnectl module update` command.

```
olcnectl module update \
--config-file myenvironment.yaml
```

To upgrade modules to the latest available release, set the version for the module in the configuration file and use the `olcnectl module update` command. For example, to upgrade the Kubernetes module to the latest version, add `kube-version: 1.21.14-3`, and for the Istio module, add `istio-version: 1.13.9`:

```
...
modules:
- module: kubernetes
  name: mycluster
```

Uninstalling Using a Configuration File

```
args:
  container-registry: container-registry.oracle.com/olcne
  load-balancer: lb.example.com:6443
  kube-version: 1.21.14-3
  master-nodes:
    - control1.example.com:8090
    - control2.example.com:8090
    - control3.example.com:8090
  worker-nodes:
    - worker1.example.com:8090
    - worker2.example.com:8090
    - worker3.example.com:8090
  selinux: enforcing
  restrict-service-externalip: true
  restrict-service-externalip-ca-cert: /etc/olcne/certificates/restrict_external_ip/ca.cert
  restrict-service-externalip-tls-cert: /etc/olcne/certificates/restrict_external_ip/node.cert
  restrict-service-externalip-tls-key: /etc/olcne/certificates/restrict_external_ip/node.key
modules:
  - module: helm
    name: myhelm
    args:
      helm-kubernetes-module: mycluster
  - module: istio
    name: myistio
    args:
      istio-helm-module: myhelm
      istio-version: 1.13.9
```

Use the `olcnectl module update` command to upgrade the modules listed in the configuration file.

```
olcnectl module update \
--config-file myenvironment.yaml
```

2.7 Uninstalling Using a Configuration File

To use a configuration file to uninstall environments and modules, you should use the same `olcnectl` commands you would use to remove modules without using the file. That is, you should remove the modules first, then remove the environment.

Use the `--force` option of the `olcnectl module uninstall` command to make sure the module dependency order is maintained internally by the Platform API Server when you remove modules from an environment.

```
olcnectl module uninstall \
--config-file myenvironment.yaml \
--force
```

All the modules in the configuration file are removed.

Remove the environment using:

```
olcnectl environment delete \
--config-file myenvironment.yaml
```

The environment is removed.

Chapter 3 Reporting Oracle Cloud Native Environment Details

You can query the status of your environment, including such items as the installed modules, the configuration of specific hosts in a Kubernetes cluster, and so on using the Platform CLI. This information can be useful for reporting status and for troubleshooting purposes.

The following sections contain information about reporting information about your environments, installed modules, and their associated properties.

3.1 Reporting Environment Information

To obtain general information about your environment and a summary of its modules, you can use the `olcnectl module report` or `olcnectl environment report` commands along with the `--environment-name` option to specify the environment you want information about, such as the following:

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

or

```
olcnectl environment report \  
--environment-name myenvironment
```

Environment details included in the response depend on the modules you have installed. Responses include information about each installed module, their status, and other related information. For example, response might include:

- Deployed Kubernetes module, including information such as cluster name and status such as:
 - The Kubernetes cluster name,
 - The status of pod networking,
 - The health status of the cluster.
- Cluster Node IP addresses, port numbers and status information such as:
 - networking status
 - kubecfg file status
 - SELinux status
 - swap status
 - IP connectivity status
- Other Installed modules including Helm chart details relating to each module.

3.2 Reporting Detailed Module Information

To isolate the summary information about specific modules in your environment, use the `olcnectl module report` command with the `--environment-name` and `--name` options to specify a module name. Only one module can be specified at a time. For example, the following command returns information about the Kubernetes cluster:

```
olcnectl module report \  
--environment-name myenvironment \  
--name mycluster
```

To obtain detailed information about a specific module (if the module has additional detailed information) in your environment, you can add the `--children` option:

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

In this example, the results returns a table listing information such as:

- The summary module information included in the command without the `--children` option.
- Details for each node such as:
 - IP address, port number, and connectivity status
 - Container images available
 - Firewall settings such as open ports
 - Kernel versions
 - IP Interface
 - Swap status
 - Packages installed and corresponding version numbers
 - Services running on the node

3.3 Filtering Report Responses

To filter responses such that only specified properties are returned, you can use the `olcnectl module report` command along with the `--include` option. For example, the following command returns only Kubernetes package details and version numbers:

```
olcnectl module report \  
--environment-name myenvironment \  
--name mycluster \  
--children \  
--include "package"
```

To return one or more properties, use the `--include` option followed by a comma separated list of each property you want to return. For example, the following returns information about the state of a cluster and the services running on each node in the cluster:

```
olcnectl module report \  
--environment-name myenvironment \  
--name mycluster \  
--children \  
--include "service","status_check"
```

You can also exclude certain properties from the response using the `--exclude` option. For example, the following excludes only the kernel and IPS information from the response:

```
olcnectl module report \  
--environment-name myenvironment \  
--name mycluster \  
--children \  
--exclude "kernel","ips"
```

3.4 Changing Report Format

The default format for returned information is the table format. For example:

```
olcnectl module report \  
--environment-name myenvironment \  
--name mycluster
```

myenvironment	
mycluster	
Property	Current Value
status_check	healthy
kubectl	
kubecfg	file exist
podnetworking	running
externalip-webhook	running

To return responses in yaml format, use the `--format yaml` option. For example:

```
olcnectl module report \  
--environment-name myenvironment \  
--name mycluster \  
--format yaml
```

```
Environments:  
  myenvironment:  
    ModuleInstances:  
      - Name: mycluster  
        Properties:  
          - Name: kubecfg  
            Value: file exist  
          - Name: podnetworking  
            Value: running  
          - Name: externalip-webhook  
            Value: running  
          - Name: status_check  
            Value: healthy  
          - Name: kubectl
```

You can also redirect the YAML format output of your responses to a file by using a right angle bracket `>` followed by a file name. For example:

```
olcnectl module report \  
--environment-name myenvironment \  
--name mycluster \  
--format yaml \  
> cluster_summary.yaml
```

The contents of the response can be viewed in the file.

Chapter 4 Platform CLI Commands

This chapter contains the syntax for each `olcnectl` command option, including usage and examples.

4.1 Environment Create

Creates an empty environment.

The first step to deploying Oracle Cloud Native Environment is to create an empty environment. You can create an environment using certificates provided by Vault, or using existing certificates on the nodes.

Syntax

```
olcnectl environment create
{-E|--environment-name} environment_name
[{-h|--help}]
[globals]
```

Where:

`{-E|--environment-name} environment_name` The Oracle Cloud Native Environment. The value of `environment_name` is the name to use to identify an environment.

`{-h|--help}` Lists information about the command and the available options.

Where `globals` is:

`{-a|--api-server} api_server_address:8091` The Platform API Server for the environment. This is the host running the `olcne-api-server` service in an environment. The value of `api_server_address` is the IP address or hostname of the Platform API Server. The port number is the port on which the `olcne-api-server` service is available. The default port is 8091.

If a Platform API Server is not specified, a local instance is used. If no local instance is set up, it is configured in the `$HOME/.olcne/olcne.conf` file.

For more information on setting the Platform API Server see [Section 1.2, "Setting the Platform API Server"](#).

This option maps to the `$OLCNE_API_SERVER_BIN` environment variable. If this environment variable is set it takes precedence over and overrides the Platform CLI setting.

`--config-file path` The location of a YAML file that contains the configuration information for the environment(s) and module(s). The filename extension must be either `yaml` or `yml`. When you use this option, any other command line options are ignored, with the exception of the `--force` option. Only the information contained in the configuration file is used.

`--secret-manager-type {file|vault}` The secrets manager type. The options are `file` or `vault`. Use `file` for certificates saved on the nodes and use `vault` for certificates managed by Vault.

Syntax

<code>--update-config</code>	<p>Writes the global arguments for an environment to a local configuration file which is used for future calls to the Platform API Server. If this option has not been used previously, global arguments must be specified for every Platform API Server call.</p> <p>The global arguments configuration information is saved to <code>\$HOME/.olcne/olcne.conf</code> on the local host.</p> <p>If you use Vault to generate certificates for nodes, the certificate is saved to <code>\$HOME/.olcne/certificates/environment_name/</code> on the local host.</p>
<code>--olcne-ca-path ca_path</code>	<p>The path to a predefined Certificate Authority certificate, or the destination of the certificate if using a secrets manager to download the certificate. The default is <code>/etc/olcne/certificates/ca.cert</code>, or gathered from the local configuration if the <code>--update-config</code> option is used.</p> <p>This option maps to the <code>\$OLCNE_SM_CA_PATH</code> environment variable. If this environment variable is set it takes precedence over and overrides the Platform CLI setting.</p>
<code>--olcne-node-cert-path node_cert_path</code>	<p>The path to a predefined certificate, or the a destination if using a secrets manager to download the certificate. The default is <code>/etc/olcne/certificates/node.cert</code>, or gathered from the local configuration if the <code>--update-config</code> option is used.</p> <p>This option maps to the <code>\$OLCNE_SM_CERT_PATH</code> environment variable. If this environment variable is set it takes precedence over and overrides the Platform CLI setting.</p>
<code>--olcne-node-key-path node_key_path</code>	<p>The path to a predefined key, or the destination of the key if using a secrets manager to download the key. The default is <code>/etc/olcne/certificates/node.key</code>, or gathered from the local configuration if the <code>--update-config</code> option is used.</p> <p>This option maps to the <code>\$OLCNE_SM_KEY_PATH</code> environment variable. If this environment variable is set it takes precedence over and overrides the Platform CLI setting.</p>
<code>--olcne-tls-cipher-suites ciphers</code>	<p>The TLS cipher suites to use for Oracle Cloud Native Environment services (the Platform Agent and Platform API Server). Enter one or more in a comma separated list. The options are:</p> <ul style="list-style-type: none">• <code>TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA</code>• <code>TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256</code>• <code>TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256</code>• <code>TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA</code>• <code>TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384</code>• <code>TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305</code>

Syntax

- `TLS_ECDHE_ECDSA_WITH_RC4_128_SHA`
- `TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA`
- `TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA`
- `TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256`
- `TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256`
- `TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA`
- `TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384`
- `TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305`
- `TLS_ECDHE_RSA_WITH_RC4_128_SHA`
- `TLS_RSA_WITH_3DES_EDE_CBC_SHA`
- `TLS_RSA_WITH_AES_128_CBC_SHA`
- `TLS_RSA_WITH_AES_128_CBC_SHA256`
- `TLS_RSA_WITH_AES_128_GCM_SHA256`
- `TLS_RSA_WITH_AES_256_CBC_SHA`
- `TLS_RSA_WITH_AES_256_GCM_SHA384`
- `TLS_RSA_WITH_RC4_128_SHA`

For example:

```
--olcne-tls-cipher-suites  
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256,TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
```

This option maps to the `$OLCNE_TLS_CIPHER_SUITES` environment variable. If this environment variable is set it takes precedence over and overrides the Platform CLI setting.

`--olcne-tls-max-version
version`

The TLS maximum version for Oracle Cloud Native Environment components. The default is `VersionTLS12`. Options are:

- `VersionTLS10`
- `VersionTLS11`
- `VersionTLS12`
- `VersionTLS13`

This option maps to the `$OLCNE_TLS_MAX_VERSION` environment variable. If this environment variable is set it takes precedence over and overrides the Platform CLI setting.

Examples

<code>--olcne-tls-min-version version</code>	<p>The TLS minimum version for Oracle Cloud Native Environment components. The default is <code>VersionTLS12</code>. Options are:</p> <ul style="list-style-type: none">• <code>VersionTLS10</code>• <code>VersionTLS11</code>• <code>VersionTLS12</code>• <code>VersionTLS13</code> <p>This option maps to the <code>\$OLCNE_TLS_MIN_VERSION</code> environment variable. If this environment variable is set it takes precedence over and overrides the Platform CLI setting.</p>
<code>--vault-address vault_address</code>	<p>The IP address of the Vault instance. The default is <code>https://127.0.0.1:8200</code>, or gathered from the local configuration if the <code>--update-config</code> option is used.</p>
<code>--vault-cert-sans vault_cert_sans</code>	<p>Subject Alternative Names (SANs) to pass to Vault to generate the Oracle Cloud Native Environment certificate. The default is <code>127.0.0.1</code>, or gathered from the local configuration if the <code>--update-config</code> option is used.</p>
<code>--vault-token vault_token</code>	<p>The Vault authentication token.</p>

Examples

Example 4.1 Creating an environment using Vault

To create an environment named `myenvironment` using certificates generated from a Vault instance, use the `--secret-manager-type vault` option:

```
olcnectl environment create \  
--api-server 127.0.0.1:8091 \  
--environment-name myenvironment \  
--secret-manager-type vault \  
--vault-token s.3QKNuRoTqLbjXaGBOm06Ppsjh \  
--vault-address https://192.0.2.20:8200 \  
--update-config
```

Example 4.2 Creating an environment using certificates

To create an environment named `myenvironment` using certificates on the node's file system, use the `--secret-manager-type file` option:

```
olcnectl environment create \  
--api-server 127.0.0.1:8091 \  
--environment-name myenvironment \  
--secret-manager-type file \  
--olcne-node-cert-path /etc/olcne/certificates/node.cert \  
--olcne-ca-path /etc/olcne/certificates/ca.cert \  
--olcne-node-key-path /etc/olcne/certificates/node.key \  
--update-config
```

4.2 Environment Delete

Deletes an existing environment.

You must uninstall any modules from an environment before you can delete it.

Syntax

```
olcnectl environment delete
{-E|--environment-name} environment_name
[{-h|--help}]
[globals]
```

Where:

`{-E|--environment-name}` The Oracle Cloud Native Environment. The value of `environment_name` is the name to use to identify an environment.

`{-h|--help}` Lists information about the command and the available options.

Where `globals` is one or more of the global options as described in [Section 1.3, “Using Global Flags”](#).

Examples

Example 4.3 Deleting an environment

To delete an environment named `myenvironment`:

```
olcnectl environment delete \
--environment-name myenvironment
```

4.3 Environment Report

Reports summary and detailed information about environments.

Syntax

```
olcnectl environment report
[{-E|--environment-name} environment_name]
[--children]
[--exclude pattern]
[--include pattern]
[--format {yaml|table}]
[{-h|--help}]
[globals]
```

Where:

`{-E|--environment-name}` The Oracle Cloud Native Environment. The value of `environment_name` is the name to use to identify an environment.

`--children` When added to the command, this option recursively displays the properties for all children of a module instance. The default value is 'false'.

`--exclude pattern` An RE2 regular expression selecting the properties to exclude from the report. This option may specify more than one property as a comma separated lists.

`--include pattern` An RE2 regular expression selecting the properties to include in the report. This option may specify more than one property as a comma separated lists. By default, all properties are displayed. Using this option one or more times overrides this behavior.

Examples

`--format {yaml|table}` To generate reports in YAML or table format, use this option. The default format is `table`.

`{-h|--help}` Lists information about the command and the available options.

Where *globals* is one or more of the global options as described in [Section 1.3, “Using Global Flags”](#).

Examples

Example 4.4 Reporting summary details about an environment

To report a summary about the environment named `myenvironment`:

```
olcnectl environment report \  
--environment-name myenvironment
```

Example 4.5 Reporting details about an environment

To report details about the environment named `myenvironment`:

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

4.4 Module Backup

Backs up a module in an environment.

Syntax

```
olcnectl module backup  
{-E|--environment-name} environment_name  
{-N|--name} name  
[{-h|--help}]  
[globals]
```

Where:

`{-E|--environment-name} environment_name` The Oracle Cloud Native Environment. The value of *environment_name* is the name to use to identify an environment.

`{-N|--name} name` The module name. The value of *name* is the name to use to identify a module in an environment.

`{-h|--help}` Lists information about the command and the available options.

Where *globals* is one or more of the global options as described in [Section 1.3, “Using Global Flags”](#).

Examples

Example 4.6 Backing up a control plane nodes

To back up the configuration for the Kubernetes control plane nodes in a `kubernetes` module named `mycluster` in an environment named `myenvironment`:

```
olcnectl module backup \  
--environment-name myenvironment \  
--name mycluster
```

4.5 Module Create

Adds and configures a module in an environment.

Syntax

```
olcnectl module create
{-E|--environment-name} environment_name
{-M|--module} module
{-N|--name} name
[{-h|--help}]
[module_args ...]
[globals]
```

Where:

<code>{-E --environment-name} environment_name</code>	The Oracle Cloud Native Environment. The value of <code>environment_name</code> is the name to use to identify an environment.
<code>{-M --module} module</code>	The module type to create in an environment. The value of <code>module</code> is the name of a module type. The available module types are: <ul style="list-style-type: none">• <code>kubernetes</code>• <code>helm</code>• <code>prometheus</code>• <code>grafana</code>• <code>istio</code>• <code>operator-lifecycle-manager</code>• <code>gluster</code>• <code>oci-csi</code>
<code>{-N --name} name</code>	The module name. The value of <code>name</code> is the name to use to identify a module in an environment.
<code>{-h --help}</code>	Lists information about the command and the available options.

Where `module_args` is:

The value of `module_args` is one or more arguments to configure a module in an environment.

`module_args` for the `kubernetes` module:

<code>{-o --apiserver-advertise-address} IP_address</code>	The IP address on which to advertise the Kubernetes API server to members of the Kubernetes cluster. This address must be reachable by the cluster nodes. If no value is provided, the interface on the control plane node is used specified with the <code>--master-nodes</code> argument. This option is not used in a highly available (HA) cluster with multiple control plane nodes.
--	--



Important

This argument has been deprecated. Use the `--master-nodes` argument instead.

`{-b|--apiserver-bind-port} port`

The Kubernetes API server bind port. The default is `6443`.

`{-B|--apiserver-bind-port-alt} port`

The port on which the Kubernetes API server listens when you use a virtual IP address for the load balancer. The default is `6444`. This is optional.

When you use a virtual IP address, the Kubernetes API server port is changed from the default of `6443` to `6444`. The load balancer listens on port `6443` and receives the requests and passes them to the Kubernetes API server. If you want to change the Kubernetes API server port in this situation from `6444`, you can use this option to do so.

`{-e|--apiserver-cert-extra-sans} api_server_sans`

The Subject Alternative Names (SANs) to use for the Kubernetes API server serving certificate. This value can contain both IP addresses and DNS names.

`{-r|--container-registry} container_registry`

The container registry that contains the Kubernetes images. Use container-registry.oracle.com/olcne to pull the Kubernetes images from the Oracle Container Registry.

If you do not provide this value, you are prompted for it by the Platform CLI.

`{-x|--kube-proxy-mode} {userspace|iptables|ipvs}`

The routing mode for the Kubernetes proxy. The default is `iptables`. The available proxy modes are:

- `userspace`: This is an older proxy mode.
- `iptables`: This is the fastest proxy mode. This is the default mode.
- `ipvs`: This is an experimental mode.

If no value is provided, the default of `iptables` is used. If the system's kernel or `iptables` version is insufficient, the `userspace` proxy is used.

`{-v|--kube-version} version`

The version of Kubernetes to install. The default is the latest version. For information on the latest version number, see [Release Notes](#).

`{-t|--kubeadm-token} token`

The token to use for establishing bidirectional trust between Kubernetes nodes and control plane nodes. The format is `[a-z0-9]{6}\.[a-z0-9]{16}`, for example, `abcdef.0123456789abcdef`.

`--kube-tls-cipher-suites ciphers`

The TLS cipher suites to use for Kubernetes components. Enter one or more in a comma separated list. The options are:

- `TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA`
- `TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256`

Syntax

- TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305
- TLS_ECDHE_ECDSA_WITH_RC4_128_SHA
- TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305
- TLS_ECDHE_RSA_WITH_RC4_128_SHA
- TLS_RSA_WITH_3DES_EDE_CBC_SHA
- TLS_RSA_WITH_AES_128_CBC_SHA
- TLS_RSA_WITH_AES_128_CBC_SHA256
- TLS_RSA_WITH_AES_128_GCM_SHA256
- TLS_RSA_WITH_AES_256_CBC_SHA
- TLS_RSA_WITH_AES_256_GCM_SHA384
- TLS_RSA_WITH_RC4_128_SHA

For example:

```
--kube-tls-cipher-suites  
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 , TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
```

`--kube-tls-min-version`
version

The TLS minimum version for Kubernetes components. The default is VersionTLS12. Options are:

- VersionTLS10
- VersionTLS11
- VersionTLS12
- VersionTLS13

Syntax

<code>{-l --load-balancer}</code> <code>load_balancer</code>	The Kubernetes API server load balancer hostname or IP address, and port. The default port is <code>6443</code> . For example, <code>192.0.2.100:6443</code> .
<code>{-m --master-nodes}</code> <code>nodes ...</code>	<p>A comma separated list of the hostnames or IP addresses of the Kubernetes control plane nodes, including the port number for the Platform Agent. For example, <code>control1.example.com:8090,control2.example.com:8090</code>.</p> <p>If you do not provide this value, you are prompted for it by the Platform CLI.</p>
<code>{-g --nginx-image}</code> <code>container_location</code>	<p>The location for an NGINX container image to use in a highly available (HA) cluster with multiple control plane nodes. This is optional.</p> <p>You can use this option if you do not provide your own load balancer using the <code>--load-balancer</code> option. This option may be useful if you are using a mirrored container registry. For example:</p> <pre>--nginx-image mirror.example.com:5000/olcne/nginx:1.17.7</pre> <p>By default, <code>podman</code> is used to pull the NGINX image that is configured in <code>/usr/libexec/pull_olcne_nginx</code>. If you set the <code>--nginx-image</code> option to use another NGINX container image, the location of the image is written to <code>/etc/olcne-nginx/image</code>, and overrides the default image.</p>
<code>--node-labels label</code>	<p>The label to add to Kubernetes nodes on Oracle Cloud Infrastructure instances to set the Availability Domain for pods. This option is used with the Oracle Cloud Infrastructure Container Storage Interface module (<code>oci-csi</code>). The label should be in the format:</p> <pre>failure-domain.beta.kubernetes.io/zone=region-identifier-AD-availability-domain-number</pre> <p>For example:</p> <pre>--node-labels failure-domain.beta.kubernetes.io/zone=US-ASHBURN-AD-1</pre> <p>For a list of the Availability Domains, see the Oracle Cloud Infrastructure documentation.</p>
<code>--node-ocids OCIDs</code>	A comma separated list of Kubernetes nodes (both control plane and worker nodes) with their Oracle Cloud Identifiers (OCIDs). This option is

Syntax

used with the Oracle Cloud Infrastructure Container Storage Interface module (`oci-csi`). The format for the list is:

```
FQDN=OCID, ...
```

For example:

```
--node-ocids  
control1.example.com=ocid1.instance..., worker1.example.com=
```

For information about OCIDs, see the [Oracle Cloud Infrastructure documentation](#).

<code>{-p --pod-cidr} pod_CIDR</code>	The Kubernetes pod CIDR. The default is <code>10.244.0.0/16</code> . This is the range from which each Kubernetes pod network interface is assigned an IP address.
<code>{-n --pod-network} network_fabric</code>	The network fabric for the Kubernetes cluster. The default is <code>flannel</code> .
<code>{-P --pod-network-iface} network_interface</code>	The name of the network interface on the nodes to use for the Kubernetes data plane network communication. The data plane network is used by the pods running on Kubernetes. If you use regex to set the interface name, the first matching interface returned by the kernel is used. For example: <pre>--pod-network-iface "ens[1-5] eth5"</pre>
<code>--selinux {enforcing permissive}</code>	Whether to use SELinux <code>enforcing</code> or <code>permissive</code> mode. <code>permissive</code> is the default. You should use this option if SELinux is set to <code>enforcing</code> on the control plane and worker nodes. SELinux is set to <code>enforcing</code> mode by default on the operating system and is the recommended mode.
<code>{-s --service-cidr} service_CIDR</code>	The Kubernetes service CIDR. The default is <code>10.96.0.0/12</code> . This is the range from which each Kubernetes service is assigned an IP address.
<code>{-i --virtual-ip} virtual_ip</code>	The virtual IP address for the load balancer. This is optional. You should use this option if you do not specify your own load balancer using the <code>--load-balancer</code> option. When you specify a virtual IP address, it is used as the primary IP address for control plane nodes.
<code>{-w --worker-nodes} nodes ...</code>	A comma separated list of the hostnames or IP addresses of the Kubernetes worker nodes, including the port number for the Platform Agent. If a worker node is behind a NAT gateway, use the public IP address for the node. The worker node's interface behind the NAT gateway must have a public IP address using the /32 subnet mask that is reachable by the Kubernetes cluster. The /32 subnet restricts the subnet to one IP address, so that all traffic from the Kubernetes cluster flows through this public IP address (for more information about configuring NAT, see Getting Started). The default port number is <code>8090</code> . For example, <code>worker1.example.com:8090,worker2.example.com:8090</code> .

	If you do not provide this value, you are prompted for it by the Platform CLI.
<code>--restrict-service-externalip {true false}</code>	Sets whether to restrict access to external IP addresses for Kubernetes services. The default is <code>true</code> , which restricts access to external IP addresses. This option deploys a Kubernetes service named <code>externalip-validation-webhook-service</code> to validate <code>externalIPs</code> set in Kubernetes service configuration files. Access to any external IP addresses is set in a Kubernetes service configuration file using the <code>externalIPs</code> option in the <code>spec</code> section.
<code>--restrict-service-externalip-ca-cert path</code>	The path to a CA certificate file for the <code>externalip-validation-webhook-service</code> application that is deployed when the <code>--restrict-service-externalip</code> option is set to <code>true</code> . For example, <code>/etc/olcne/certificates/restrict_external_ip/ca.cert</code> .
<code>--restrict-service-externalip-tls-cert path</code>	The path to a CA certificate file for the <code>externalip-validation-webhook-service</code> application that is deployed when the <code>--restrict-service-externalip</code> option is set to <code>true</code> . For example, <code>/etc/olcne/certificates/restrict_external_ip/node.cert</code> .
<code>--restrict-service-externalip-tls-key path</code>	The path to the private key for the <code>externalip-validation-webhook-service</code> application that is deployed when the <code>--restrict-service-externalip</code> option is set to <code>true</code> . For example, <code>/etc/olcne/certificates/restrict_external_ip/node.key</code> .
<code>--restrict-service-externalip-cidrs allowed_cidrs</code>	Enter one or more comma separated CIDR blocks if you want to allow only IP addresses from the specified CIDR blocks. For example, <code>192.0.2.0/24,198.51.100.0/24</code> .

module_args for the helm module:

<code>--helm-kubernetes-module kubernetes_module</code>	The name of the <code>kubernetes</code> module that Helm should be associated with. Each instance of Kubernetes can have one instance of Helm associated with it.
<code>--helm-version version</code>	The version of Helm to install. The default is the latest version. For information on the latest version number, see Release Notes .

module_args for the prometheus module:

<code>--prometheus-helm-module helm_module</code>	The name of the <code>helm</code> module that Prometheus should be associated with.
<code>--prometheus-version version</code>	The version of Prometheus to install. The default is the latest version. For information on the latest version number, see Release Notes .
<code>--prometheus-image container_registry</code>	The container image registry and tag to use when installing Prometheus. The default is <code>container-registry.oracle.com/olcne/prometheus</code> .

Syntax

<code>--prometheus-namespace namespace</code>	The Kubernetes namespace in which to install Prometheus. The default namespace is <code>default</code> .
<code>--prometheus-persistent-storage {true false}</code>	<p>If this value is <code>false</code>, Prometheus writes its data into an <code>emptydir</code> on the host where the pod is running. If the pod migrates, metric data is lost.</p> <p>If this value is <code>true</code>, Prometheus requisitions a Kubernetes <code>PersistentVolumeClaim</code> so that its data persists, despite destruction or migration of the pod.</p> <p>The default is <code>false</code>.</p>
<code>--prometheus-alerting-rules path</code>	The path to a configuration file for Prometheus alerts.
<code>--prometheus-recording-rules path</code>	The path to a configuration file for Prometheus recording rules.
<code>--prometheus-scrape-configuration path</code>	The path to a configuration file for Prometheus metrics scraping.
module_args for the grafana module:	
<code>--grafana-helm-module helm_module</code>	The name of the <code>helm</code> module that Grafana should be associated with.
<code>--grafana-version version</code>	The version of Grafana to install. The default is the latest version. For information on the latest version number, see Release Notes .
<code>--grafana-container-registry container_registry</code>	The container image registry and tag to use when installing Grafana. The default is <code>container-registry.oracle.com/olcne</code> .
<code>--grafana-namespace namespace</code>	The Kubernetes namespace in which to install Grafana. The default namespace is <code>default</code> .
<code>--grafana-dashboard-configmaps configmap</code>	The name of the ConfigMap reference that contains the Grafana dashboards.
<code>--grafana-dashboard-providers path</code>	The location of the file that contains the configuration for the Grafana dashboard providers.
<code>--grafana-datasources path</code>	The location of the file that contains the configuration for the Grafana data sources.
<code>--grafana-existing-secret-name secret</code>	The name of the existing secret containing the Grafana admin password.
<code>--grafana-notifiers path</code>	The location of the file that contains the configuration for the Grafana notifiers.
<code>--grafana-pod-annotations annotations</code>	A comma separated list of annotations to be added to the Grafana pods.
<code>--grafana-pod-env env_vars</code>	A comma separated list of environment variables to be passed to Grafana deployment pods.

`--grafana-service-port port` The port number for the Grafana service. The default is `3000`.

`--grafana-service-type service` The service type to access Grafana. The default is `ClusterIP`.

`module_args` for the `istio` module:

`--istio-helm-module helm_module` The name of the `helm` module that Istio should be associated with.

`--istio-version version` The version of Istio to install. The default is the latest version. For information on the latest version number, see [Release Notes](#).

`--istio-container-registry container_registry` The container image registry to use when deploying Istio. The default is `container-registry.oracle.com/olcne`.

`--istio-mutual-tls {true|false}` Sets whether to enable Mutual Transport Layer Security (mTLS) for communication between the control plane pods for Istio, and for any pods deployed into the Istio service mesh.

The default is `true`.



Important

It is strongly recommended that this value is not set to `false`, especially in production environments.

`module_args` for the `operator-lifecycle-manager` module:

`--olm-helm-module helm_module` The name of the `helm` module that Operator Lifecycle Manager should be associated with.

`--olm-version version` The version of Operator Lifecycle Manager to install. The default is the latest version. For information on the latest version number, see [Release Notes](#).

`--olm-container-registry container_registry` The container image registry to use when deploying the Operator Lifecycle Manager. The default is `container-registry.oracle.com/olcne`.

`--olm-enable-operatorhub {true|false}` Sets whether to enable the Operator Lifecycle Manager to use the OperatorHub registry as a catalog source.

The default is `true`.

`module_args` for the `gluster` module:

`--gluster-helm-module helm_module` The name of the `helm` module that the Gluster Container Storage Interface module should be associated with.

`--gluster-server-url URL` The URL of the Heketi API server endpoint. The default is `http://127.0.0.1:8080`.

Syntax

<code>--gluster-server-user</code> <i>user</i>	The username of the Heketi API server admin user. The default is <code>admin</code> .
<code>--gluster-existing-secret-name</code> <i>secret</i>	The name of the existing secret containing the admin password. The default is <code>heketi-admin</code> .
<code>--gluster-secret-key</code> <i>secret</i>	The secret containing the admin password. The default is <code>secret</code> .
<code>--gluster-namespace</code> <i>namespace</i>	The Kubernetes namespace in which to install the Gluster Container Storage Interface module. The default is <code>default</code> .
<code>--gluster-sc-name</code> <i>class_name</i>	The StorageClass name for the Glusterfs StorageClass. The default is <code>hyperconverged</code> .
<code>--gluster-server-rest-auth</code> {true false}	Whether the Heketi API server accepts REST authorization. The default is <code>true</code> .

module_args for the ***oci-csi*** module:

<code>--oci-csi-helm-module</code> <i>helm_module</i>	The name of the <code>helm</code> module that the Oracle Cloud Infrastructure Container Storage Interface module should be associated with.
<code>--oci-tenancy</code> <i>OCID</i>	The OCID for the Oracle Cloud Infrastructure tenancy.
<code>--oci-region</code> <i>region_identifier</i>	The Oracle Cloud Infrastructure region identifier. The default is <code>us-ashburn-1</code> . For a list of the region identifiers, see the Oracle Cloud Infrastructure documentation .
<code>--oci-compartment</code> <i>OCID</i>	The OCID for the Oracle Cloud Infrastructure compartment.
<code>--oci-user</code> <i>OCID</i>	The OCID for the Oracle Cloud Infrastructure user.
<code>--oci-private-key</code> <i>path</i>	The location of the private key for the Oracle Cloud Infrastructure API signing key. This must be located on the primary control plane node. The default is <code>/root/.ssh/id_rsa</code> .



Important

The private key must be available on the primary control plane node. This is the first control plane node listed in the `--master-nodes` option when you create the Kubernetes module.

<code>--oci-fingerprint</code> <i>fingerprint</i>	The fingerprint of the public key for the Oracle Cloud Infrastructure API signing key.
<code>--oci-passphrase</code> <i>passphrase</i>	The passphrase for the private key for the Oracle Cloud Infrastructure API signing key, if one is set.
<code>--oci-vcn</code> <i>OCID</i>	The OCID for the Oracle Cloud Infrastructure Virtual Cloud Network on which the Kubernetes cluster is available.
<code>--oci-lb-subnet1</code> <i>OCID</i>	The OCID of the regional subnet for the Oracle Cloud Infrastructure load balancer.

Examples

	Alternatively, the OCID of the first subnet of the two required availability domain specific subnets for the Oracle Cloud Infrastructure load balancer. The subnets must be in separate availability domains.
<code>--oci-lb-subnet2 <i>OCID</i></code>	The OCID of the second subnet of the two subnets for the Oracle Cloud Infrastructure load balancer. The subnets must be in separate availability domains.
<code>--oci-lb-security-mode {All Frontend None}</code>	This option sets whether the Oracle Cloud Infrastructure CSI plug-in should manage security lists for load balancer services. This option sets the configuration mode to use for security lists managed by the Kubernetes Cloud Controller Manager. The default is <code>None</code> .
	For information on the security modes, see the Kubernetes Cloud Controller Manager implementation for Oracle Cloud Infrastructure documentation .
<code>--oci-container-registry <i>container_registry</i></code>	The container image registry to use when deploying the Oracle Cloud Infrastructure cloud provisioner image. The default is <code>iad.ocir.io/oracle</code> .
<code>--csi-container-registry <i>container_registry</i></code>	The container image registry to use when deploying the CSI component images. The default is <code>quay.io/k8scsi</code> .

Where *globals* is one or more of the global options as described in [Section 1.3, “Using Global Flags”](#).

Examples

Example 4.7 Creating a module for an HA cluster with an external load balancer

This example creates an HA cluster with an external load balancer, available on the host `lb.example.com` and running on port `6443`.

You must also include the location of the certificates for the `externalip-validation-webhook-service` Kubernetes service.

```
olcnectl module create \  
--environment-name myenvironment \  
--module kubernetes \  
--name mycluster \  
--container-registry container-registry.oracle.com/olcne \  
--load-balancer lb.example.com:6443 \  
--master-nodes control1.example.com:8090,control2.example.com:8090,control3.example.com:8090 \  
--worker-nodes worker1.example.com:8090,worker2.example.com:8090,worker3.example.com:8090 \  
--selinux enforcing \  
--restrict-service-externalip-ca-cert /etc/olcne/certificates/restrict_external_ip/ca.cert \  
--restrict-service-externalip-tls-cert /etc/olcne/certificates/restrict_external_ip/node.cert \  
--restrict-service-externalip-tls-key /etc/olcne/certificates/restrict_external_ip/node.key
```

Example 4.8 Creating a module for an HA cluster with an internal load balancer

This example creates an HA Kubernetes cluster using the load balancer deployed by the Platform CLI. The `--virtual-ip` option sets the virtual IP address to `192.0.2.100`, which is the IP address of the primary control plane node. The primary control plane node is the first node in the list of control plane nodes. This cluster contains three control plane nodes and three worker nodes.

Examples

You must also include the location of the certificates for the [externalip-validation-webhook-service](#) Kubernetes service.

```
olcnectl module create \  
--environment-name myenvironment \  
--module kubernetes \  
--name mycluster \  
--container-registry container-registry.oracle.com/olcne \  
--virtual-ip 192.0.2.100 \  
--master-nodes control1.example.com:8090,control2.example.com:8090,control3.example.com:8090 \  
--worker-nodes worker1.example.com:8090,worker2.example.com:8090,worker3.example.com:8090 \  
--selinux enforcing \  
--restrict-service-externalip-ca-cert /etc/olcne/certificates/restrict_external_ip/ca.cert \  
--restrict-service-externalip-tls-cert /etc/olcne/certificates/restrict_external_ip/node.cert \  
--restrict-service-externalip-tls-key /etc/olcne/certificates/restrict_external_ip/node.key
```

Example 4.9 Creating a module for a cluster to allow access to service IP address ranges

This example creates a Kubernetes cluster that sets the external IP addresses that can be accessed by Kubernetes services. The IP ranges that are allowed are within the [192.0.2.0/24](#) and [198.51.100.0/24](#) CIDR blocks.

You must also include the location of the certificates for the [externalip-validation-webhook-service](#) Kubernetes service.

```
olcnectl module create \  
--environment-name myenvironment \  
--module kubernetes \  
--name mycluster \  
--container-registry container-registry.oracle.com/olcne \  
--virtual-ip 192.0.2.100 \  
--master-nodes control1.example.com:8090,control2.example.com:8090,control3.example.com:8090 \  
--worker-nodes worker1.example.com:8090,worker2.example.com:8090,worker3.example.com:8090 \  
--selinux enforcing \  
--restrict-service-externalip-ca-cert /etc/olcne/certificates/restrict_external_ip/ca.cert \  
--restrict-service-externalip-tls-cert /etc/olcne/certificates/restrict_external_ip/node.cert \  
--restrict-service-externalip-tls-key /etc/olcne/certificates/restrict_external_ip/node.key \  
--restrict-service-externalip-cidrs 192.0.2.0/24,198.51.100.0/24
```

Example 4.10 Creating a module for a cluster to allow access to all service IP addresses

This example creates a Kubernetes cluster that allows access to all external IP addresses for Kubernetes services. This disables the deployment of the [externalip-validation-webhook-service](#) Kubernetes service, which means no validation of external IP addresses is performed for Kubernetes services, and access is allowed for all CIDR blocks.

```
olcnectl module create \  
--environment-name myenvironment \  
--module kubernetes \  
--name mycluster \  
--container-registry container-registry.oracle.com/olcne \  
--virtual-ip 192.0.2.100 \  
--master-nodes control1.example.com:8090,control2.example.com:8090,control3.example.com:8090 \  
--worker-nodes worker1.example.com:8090,worker2.example.com:8090,worker3.example.com:8090 \  
--selinux enforcing \  
--restrict-service-externalip false
```

Example 4.11 Creating module for a cluster with a single control plane node

This example creates a Kubernetes module to deploy a Kubernetes cluster with a single control plane node. The `--module` option is set to `kubernetes` to create a Kubernetes module. This cluster contains one control plane and two worker nodes.

Examples

You must also include the location of the certificates for the `externalip-validation-webhook-service` Kubernetes service.

```
olcnectl module create \  
--environment-name myenvironment \  
--module kubernetes \  
--name mycluster \  
--container-registry container-registry.oracle.com/olcne \  
--master-nodes controll1.example.com:8090 \  
--worker-nodes worker1.example.com:8090,worker2.example.com:8090 \  
--selinux enforcing \  
--restrict-service-externalip-ca-cert /etc/olcne/certificates/restrict_external_ip/ca.cert \  
--restrict-service-externalip-tls-cert /etc/olcne/certificates/restrict_external_ip/node.cert \  
--restrict-service-externalip-tls-key /etc/olcne/certificates/restrict_external_ip/node.key
```

Example 4.12 Creating a module for a service mesh

This example creates a service mesh using the Istio module. The `--module` option is set to `istio` to create an Istio module. This example uses a Kubernetes module named `mycluster`, a Helm module named `myhelm`, and an Istio module named `myistio`.

The `--istio-helm-module` option sets the name of the Helm module to use.

If you do not include all the required options when adding the modules you are prompted to provide them.

```
olcnectl module create \  
--environment-name myenvironment \  
--module istio \  
--name myistio \  
--istio-helm-module myhelm
```

Example 4.13 Creating a module for Operator Lifecycle Manager

This example creates a module that can be used to install Operator Lifecycle Manager. The `--module` option is set to `operator-lifecycle-manager` to create an Operator Lifecycle Manager module. This example uses a Kubernetes module named `mycluster`, a Helm module named `myhelm`, and an Operator Lifecycle Manager module named `myolm`.

The `--olm-helm-module` option sets the name of the Helm module to use.

If you do not include all the required options when adding the modules you are prompted to provide them.

```
olcnectl module create \  
--environment-name myenvironment \  
--module operator-lifecycle-manager \  
--name myolm \  
--olm-helm-module myhelm
```

Example 4.14 Creating a module for Gluster Storage

This example creates a module that creates a Kubernetes StorageClass provisioner to access Gluster storage. The `--module` option is set to `gluster` to create a Gluster Container Storage Interface module. This example uses a Kubernetes module named `mycluster`, a Helm module named `myhelm`, and a Gluster Container Storage Interface module named `mygluster`.

The `--gluster-helm-module` option sets the name of the Helm module to use.

If you do not include all the required options when adding the modules you are prompted to provide them.

```
olcnectl module create \  

```

```
--environment-name myenvironment \  
--module gluster \  
--name mygluster \  
--gluster-helm-module myhelm
```

Example 4.15 Creating a module for Oracle Cloud Infrastructure Storage

This example creates a module that creates a Kubernetes StorageClass provisioner to access Oracle Cloud Infrastructure storage. The `--module` option is set to `oci-csi` to create an Oracle Cloud Infrastructure Container Storage Interface module. This example uses a Kubernetes module named `mycluster`, a Helm module named `myhelm`, and an Oracle Cloud Infrastructure Container Storage Interface module named `myoci`.

The `--oci-csi-helm-module` option sets the name of the Helm module to use.

You should also provide the information required to access Oracle Cloud Infrastructure using the options as shown in this example, such as:

- `--oci-tenancy`
- `--oci-compartment`
- `--oci-user`
- `--oci-fingerprint`
- `--oci-private-key`

You may need to provide more options to access Oracle Cloud Infrastructure, depending on your environment.

If you do not include all the required options when adding the modules you are prompted to provide them.

```
olcnectl module create \  
--environment-name myenvironment \  
--module oci-csi \  
--name myoci \  
--oci-csi-helm-module myhelm \  
--oci-tenancy ocid1.tenancy.oc1... \  
--oci-compartment ocid1.compartment.oc1... \  
--oci-user ocid1.user.oc1... \  
--oci-fingerprint b5:52:... \  
--oci-private-key /home/opc/.oci/oci_api_key.pem
```

4.6 Module Install

Installs a module in an environment. When you install a module, the nodes are checked to make sure they are set up correctly to run the module. If the nodes are not set up correctly, the commands required to fix each node are shown in the output and optionally saved to files.

Syntax

```
olcnectl module install  
{-E|--environment-name} environment_name  
{-N|--name} name  
[{-g|--generate-scripts}]  
[{-h|--help}]  
[globals]
```

Examples

Where:

<code>{-E --environment-name} <i>environment_name</i></code>	The Oracle Cloud Native Environment. The value of <i>environment_name</i> is the name to use to identify an environment.
<code>{-N --name} <i>name</i></code>	The module name. The value of <i>name</i> is the name to use to identify a module in an environment.
<code>{-g --generate-scripts}</code>	Generates a set of scripts which contain the commands required to fix any set up errors for the nodes in a module. A script is created for each node in the module, saved to the local directory, and named <i>hostname:8090.sh</i> .
<code>{-h --help}</code>	Lists information about the command and the available options.

Where *globals* is one or more of the global options as described in [Section 1.3, “Using Global Flags”](#).

Examples

Example 4.16 Installing a module

To install a Kubernetes module named `mycluster` in an environment named `myenvironment`:

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

4.7 Module Instances

Lists the installed modules in an environment.

Syntax

```
olcnectl module instances \  
{-E|--environment-name} environment_name \  
[{-h|--help}] \  
[globals]
```

Where:

<code>{-E --environment-name} <i>environment_name</i></code>	The Oracle Cloud Native Environment. The value of <i>environment_name</i> is the name to use to identify an environment.
<code>{-h --help}</code>	Lists information about the command and the available options.

Where *globals* is one or more of the global options as described in [Section 1.3, “Using Global Flags”](#).

Examples

Example 4.17 Listing the deployed modules in an environment

To list the deployed modules for an environment named `myenvironment`:

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


4.8 Module List

Lists the available modules for an environment.

Syntax

```
olcnectl module list
{-E|--environment-name} environment_name
[{-h|--help}]
[globals]
```

Where:

<code>{-E --environment-name}</code>	The Oracle Cloud Native Environment. The value of <code><i>environment_name</i></code> is the name to use to identify an environment.
<code>{-h --help}</code>	Lists information about the command and the available options.

Where `globals` is one or more of the global options as described in [Section 1.3, “Using Global Flags”](#).

Examples

Example 4.18 Listing available modules in an environment

To list the modules for an environment named `myenvironment`:

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

4.9 Module Property Get

Lists the value of a module property.

Syntax

```
olcnectl module property get
{-E|--environment-name} environment_name
{-N|--name} name
{-P|--property} property_name
[{-h|--help}]
[globals]
```

Where:

<code>{-E --environment-name}</code>	The Oracle Cloud Native Environment. The value of <code><i>environment_name</i></code> is the name to use to identify an environment.
<code>{-N --name} <i>name</i></code>	The module name. The value of <code><i>name</i></code> is the name to use to identify a module in an environment.
<code>{-P --property} <i>property_name</i></code>	The name of the property. You can get a list of the available properties using the <code>olcnectl module property list</code> command.
<code>{-h --help}</code>	Lists information about the command and the available options.

Where `globals` is one or more of the global options as described in [Section 1.3, “Using Global Flags”](#).

Examples

Example 4.19 Listing module properties

To list the value of the `kubecfg` property for a Kubernetes module named `mycluster` in an environment named `myenvironment`:

```
olcnectl module property get \  
--environment-name myenvironment \  
--name mycluster \  
--property kubecfg
```

4.10 Module Property List

Lists the available properties for a module in an environment.

Syntax

```
olcnectl module property list  
{-E|--environment-name} environment_name  
{-N|--name} name  
[{-h|--help}]  
[globals]
```

Where:

`{-E|--environment-name} environment_name` The Oracle Cloud Native Environment. The value of `environment_name` is the name to use to identify an environment.

`{-N|--name} name` The module name. The value of `name` is the name to use to identify a module in an environment.

`{-h|--help}` Lists information about the command and the available options.

Where `globals` is one or more of the global options as described in [Section 1.3, “Using Global Flags”](#).

Examples

Example 4.20 Listing module properties

To list the properties for a Kubernetes module named `mycluster` in an environment named `myenvironment`:

```
olcnectl module property list \  
--environment-name myenvironment \  
--name mycluster
```

4.11 Module Report

Reports summary and detailed information about module and properties in an environment.

Syntax

```
olcnectl module report  
{-E|--environment-name} environment_name  
[{-N|--name} name]  
[--children]
```

Examples

```
[--exclude pattern]  
[--include pattern]  
[--format {yaml|table}]  
[{-h|--help}]  
[globals]
```

Where:

<code>{-E --environment-name} <i>environment_name</i></code>	The Oracle Cloud Native Environment. The value of <i>environment_name</i> is the name to use to identify an environment.
<code>{-N --name} <i>name</i></code>	The module name. The value of <i>name</i> is the name to use to identify a module in an environment. When no name is specified, the output of the command contains information about all modules deployed in the selected environment.
<code>--children</code>	When added to the command, this option recursively displays the properties for all children of a module instance. The default value is 'false'.
<code>--exclude <i>pattern</i></code>	An RE2 regular expression selecting the properties to exclude from the report. This option may specify more than one property as a comma separated lists.
<code>--include <i>pattern</i></code>	An RE2 regular expression selecting the properties to include in the report. This option may specify more than one property as a comma separated lists. By default, all properties are displayed. Using this option one or more times overrides this behavior.
<code>--format {yaml table}</code>	To generate reports in YAML or table format, use this option. The default format is <code>table</code> .
<code>{-h --help}</code>	Lists information about the command and the available options.

Where *globals* is one or more of the global options as described in [Section 1.3, "Using Global Flags"](#).

Examples

Example 4.21 Reporting summary details about an environment

To report a summary of all modules deployed in the environment named `myenvironment`:

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

```

Example 4.22 Reporting summary details about a Kubernetes module

To report summary details about a Kubernetes module named `mycluster`:

```
olcnectl module report \  
--environment-name myenvironment \  
--name mycluster
```

Example 4.23 Reporting comprehensive details about a Kubernetes module

To report comprehensive details about a Kubernetes module named `mycluster`:

```
olcnectl module report \  

```

```
--environment-name myenvironment \  
--name mycluster \  
--children
```

4.12 Module Restore

Restores a module from a back in an environment.

Syntax

```
olcnectl module restore  
{-E|--environment-name} environment_name  
{-N|--name} name  
[{-g|--generate-scripts}]  
[{-F|--force}]  
[{-h|--help}]  
[globals]
```

Where:

<code>{-E --environment-name} <i>environment_name</i></code>	The Oracle Cloud Native Environment. The value of <i>environment_name</i> is the name to use to identify an environment.
<code>{-N --name} <i>name</i></code>	The module name. The value of <i>name</i> is the name to use to identify a module in an environment.
<code>{-g --generate-scripts}</code>	Generates a set of scripts which contain the commands required to fix any set up errors for the nodes in a module. A script is created for each node in the module, saved to the local directory, and named <i>hostname:8090.sh</i> .
<code>{-F --force}</code>	Skips the confirmation prompt.
<code>{-h --help}</code>	Lists information about the command and the available options.

Where *globals* is one or more of the global options as described in [Section 1.3, “Using Global Flags”](#).

Examples

Example 4.24 Restoring control plane nodes from a back up

To restore the Kubernetes control plane nodes from a back up in a Kubernetes module named `mycluster` in an environment named `myenvironment`:

```
olcnectl module restore \  
--environment-name myenvironment \  
--name mycluster
```

4.13 Module Uninstall

Uninstalls a module from an environment. Uninstalling the module also removes the module configuration from the Platform API Server.

Syntax

```
olcnectl module uninstall  
{-E|--environment-name} environment_name
```

Examples

```
{-N|--name} name  
[{-F|--force}]  
[{-h|--help}]  
[globals]
```

Where:

<code>{-E --environment-name}</code> <i>environment_name</i>	The Oracle Cloud Native Environment. The value of <i>environment_name</i> is the name to use to identify an environment.
<code>{-N --name}</code> <i>name</i>	The module name. The value of <i>name</i> is the name to use to identify a module in an environment.
<code>{-F --force}</code>	Skips the confirmation prompt.
<code>{-h --help}</code>	Lists information about the command and the available options.

Where *globals* is one or more of the global options as described in [Section 1.3, “Using Global Flags”](#).

Examples

Example 4.25 Uninstalling a module

To uninstall a Kubernetes module named `mycluster` from an environment named `myenvironment`:

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

In this example, the Kubernetes containers are stopped and deleted on each node, and the Kubernetes cluster is removed.

4.14 Module Update

Updates a module in an environment. The module configuration is automatically retrieved from the Platform API Server. This command can be used to:

- Update the Kubernetes release on nodes to the latest errata release
- Upgrade the Kubernetes release on nodes to the latest release
- Update or upgrade other modules and components
- Scale up a Kubernetes cluster (add control plane and/or worker nodes)
- Scale down a Kubernetes cluster (remove control plane and/or worker nodes)



Important

Before you update or upgrade the Kubernetes cluster, make sure you have updated or upgraded Oracle Cloud Native Environment to the latest release. For information on updating or upgrading Oracle Cloud Native Environment, see [Updates and Upgrades](#).

Syntax

```
olcnectl module update
```

Syntax

```
{-E|--environment-name} environment_name
{-N|--name} name
[{-r|--container-registry} container_registry]
[{-k|--kubernetes-version} version]
[{-m|--master-nodes} nodes ...]
[{-w|--worker-nodes} nodes ...]
[--nginx-image container_location]
[--helm-version version]
[--prometheus-version version]
[--prometheus-container-registry container_registry]
[--grafana-version version]
[--grafana-container-registry container_registry]
[--istio-version version]
[--istio-container-registry container_registry]
[--olm-version version]
--restrict-service-externalip {true|false}
--restrict-service-externalip-ca-cert path
--restrict-service-externalip-tls-cert path
--restrict-service-externalip-tls-key path
--restrict-service-externalip-cidrs allowed_cidrs
[--selinux {enforcing|permissive}]
[{-g|--generate-scripts}]
[{-F|--force}]
[{-h|--help}]
[globals]
```

Where:

<code>{-E --environment-name} <i>environment_name</i></code>	The Oracle Cloud Native Environment. The value of <i>environment_name</i> is the name to use to identify an environment.
<code>{-N --name} <i>name</i></code>	The module name. The value of <i>name</i> is the name to use to identify a module in an environment.
<code>{-k --kubernetes-version} <i>version</i></code>	Sets the Kubernetes version for the upgrade. The default is the latest version. For information on the latest version number, see Release Notes . If this option is not provided any Kubernetes errata updates are installed.
<code>{-r --container-registry} <i>container_registry</i></code>	The container registry that contains the Kubernetes images when performing an update or upgrade. Use the Oracle Container Registry or a local registry to pull the Kubernetes images. This option allows you to update or upgrade using a different container registry. This option sets the default container registry during all subsequent updates or upgrades and need only be used when changing the default container registry.
<code>{-m --master-nodes} <i>nodes ...</i></code>	A comma-separated list of the hostnames or IP addresses of the Kubernetes control plane nodes that should remain in or be added to the Kubernetes cluster, including the port number for the Platform Agent. Any control plane nodes not included in this list are removed from the cluster. The nodes in this list are the nodes that are to be included in the cluster. The default port number for the Platform Agent is 8090. For example, <code>control1.example.com:8090,control2.example.com:8090</code> .

<code>{-w --worker-nodes}</code> <code>nodes ...</code>	<p>A comma-separated list of the hostnames or IP addresses of the Kubernetes worker nodes that should remain in or be added to the Kubernetes cluster, including the port number for the Platform Agent. Any worker nodes not included in this list are removed from the cluster. The nodes in this list are the nodes that are to be included in the cluster.</p> <p>The default port number for the Platform Agent is <code>8090</code>. For example, <code>worker1.example.com:8090,worker2.example.com:8090</code>.</p>
<code>--nginx-image</code> <code>container_location</code>	<p>The location of the NGINX container image to update. This is optional.</p> <p>This option pulls the NGINX container image from the container registry location you specify to update NGINX on the control plane nodes. For example:</p> <pre>--nginx-image container-registry.oracle.com/olcne/nginx:1.17.7</pre>
<code>--helm-version</code> <i>version</i>	<p>Sets the Helm version for the upgrade. The default is the latest version. For information on the latest version number, see Release Notes.</p>
<code>--prometheus-version</code> <i>version</i>	<p>Sets the Prometheus version for the upgrade. The default is the latest version. For information on the latest version number, see Release Notes.</p>
<code>--prometheus-</code> <code>container-registry</code> <i>container_registry</i>	<p>The container registry that contains the Prometheus images when performing an update or upgrade. Use the Oracle Container Registry (the default) or a local registry to pull the Prometheus images.</p>
<code>--grafana-version</code> <i>version</i>	<p>Sets the Grafana version for the upgrade. The default is the latest version. For information on the latest version number, see Release Notes.</p>
<code>--grafana-</code> <code>container-registry</code> <i>container_registry</i>	<p>The container registry that contains the Grafana images when performing an update or upgrade. Use the Oracle Container Registry (the default) or a local registry to pull the Grafana images.</p>
<code>--istio-version</code> <i>version</i>	<p>Sets the Istio version for the upgrade. The default is the latest version. For information on the latest version number, see Release Notes.</p>
<code>--istio-container-</code> <code>registry</code> <i>container_registry</i>	<p>The container registry that contains the Istio images when performing an update or upgrade. Use the Oracle Container Registry (the default) or a local registry to pull the Istio images.</p>
<code>--olm-version</code> <i>version</i>	<p>Sets the Operator Lifecycle Manager version for the upgrade. The default is the latest version. For information on the latest version number, see Release Notes.</p>
<code>--restrict-service-</code> <code>externalip</code> { <i>true false</i> }	<p>Sets whether to restrict access to external IP addresses for Kubernetes services. The default is <code>true</code>, which restricts access to external IP addresses.</p> <p>This option deploys a Kubernetes service named <code>externalip-validation-webhook-service</code> to validate <code>externalIPs</code> set in Kubernetes service configuration files. Access to any external IP</p>

Examples

	addresses is set in a Kubernetes service configuration file using the <code>externalIPs</code> option in the <code>spec</code> section.
<code>--restrict-service-externalip-ca-cert path</code>	The path to a CA certificate file for the <code>externalip-validation-webhook-service</code> application that is deployed when the <code>--restrict-service-externalip</code> option is set to <code>true</code> . For example, <code>/etc/olcne/certificates/restrict_external_ip/ca.cert</code> .
<code>--restrict-service-externalip-tls-cert path</code>	The path to a CA certificate file for the <code>externalip-validation-webhook-service</code> application that is deployed when the <code>--restrict-service-externalip</code> option is set to <code>true</code> . For example, <code>/etc/olcne/certificates/restrict_external_ip/node.cert</code> .
<code>--restrict-service-externalip-tls-key path</code>	The path to the private key for the <code>externalip-validation-webhook-service</code> application that is deployed when the <code>--restrict-service-externalip</code> option is set to <code>true</code> . For example, <code>/etc/olcne/certificates/restrict_external_ip/node.key</code> .
<code>--restrict-service-externalip-cidrs allowed_cidrs</code>	Enter one or more comma separated CIDR blocks if you want to allow only IP addresses from the specified CIDR blocks. For example, <code>192.0.2.0/24,198.51.100.0/24</code> .
<code>--selinux {enforcing permissive}</code>	Whether to use SELinux <code>enforcing</code> or <code>permissive</code> mode. <code>permissive</code> is the default. You should use this option if SELinux is set to <code>enforcing</code> on the control plane and worker nodes. SELinux is set to <code>enforcing</code> mode by default on the operating system and is the recommended mode.
<code>{-g --generate-scripts}</code>	Generates a set of scripts which contain the commands required to fix any set up errors for the nodes in a module. A script is created for each node in the module, saved to the local directory, and named <code>hostname:8090.sh</code> .
<code>{-F --force}</code>	Skips the confirmation prompt.
<code>{-h --help}</code>	Lists information about the command and the available options.

Where `globals` is one or more of the global options as described in [Section 1.3, “Using Global Flags”](#).

Examples

Example 4.26 Scaling a cluster

To scale up a cluster, list all nodes to be included in the cluster. If an existing cluster includes two control plane and two worker nodes, and you want to add a new control plane and a new worker, list all the nodes to include. For example, to add a `control3.example.com` control plane node, and a `worker3.example.com` worker node to a Kubernetes module named `mycluster`:

```
olcnectl module update \  
--environment-name myenvironment \  
--name mycluster \  
--master-nodes control1.example.com:8090,control2.example.com:8090,control3.example.com:8090 \  

```


The software described in this documentation is either no longer supported or is in extended support.
Oracle recommends that you upgrade to a current supported release.

Examples

```
--worker-nodes worker1.example.com:8090,worker2.example.com:8090,worker3.example.com:8090
```

To scale down a cluster, list all the nodes to be included in the cluster. To remove the `control3.example.com` control plane node, and `worker3.example.com` worker node from the `kubernetes` module named `mycluster`:

```
olcnectl module update \  
--environment-name myenvironment \  
--name mycluster \  
--master-nodes control1.example.com:8090,control2.example.com:8090 \  
--worker-nodes worker1.example.com:8090,worker2.example.com:8090
```

As the `control3.example.com` control plane node and `worker3.example.com` worker node are not listed in the `--master-nodes` and `--worker-nodes` options, the Platform API Server removes those nodes from the cluster.

Example 4.27 Updating the Kubernetes release for errata updates

To update a Kubernetes module named `mycluster` in an environment named `myenvironment` to the latest Kubernetes errata release, enter:

```
olcnectl module update \  
--environment-name myenvironment \  
--name mycluster
```

The nodes in the environment are updated to the latest Kubernetes errata release.

Example 4.28 Updating using a different container registry

To update a Kubernetes module named `mycluster` in an environment named `myenvironment` to the latest Kubernetes errata release using a different container registry than the default specified when creating the Kubernetes module, enter:

```
olcnectl module update \  
--environment-name myenvironment \  
--name mycluster \  
--container-registry container-registry-austin-mirror.oracle.com/olcne/
```

The nodes in the environment are updated to the latest Kubernetes errata release contained on the mirror container registry.

Example 4.29 Upgrading the Kubernetes release

To upgrade a Kubernetes module named `mycluster` in an environment named `myenvironment` to Kubernetes Release 1.21.14-3, enter:

```
olcnectl module update \  
--environment-name myenvironment \  
--name mycluster \  
--kube-version 1.21.14-3
```

The `--kube-version` option specifies the release to which you want to upgrade. This example uses release number 1.21.14-3.

Make sure you upgrade to the latest Kubernetes release. To get the version number of the latest Kubernetes release, see [Release Notes](#).

The nodes in the environment are updated to Kubernetes Release 1.21.14-3.

Example 4.30 Upgrading using a different container registry

To upgrade a Kubernetes module named `mycluster` in an environment named `myenvironment` to Kubernetes Release 1.21.14-3 using a different container registry than the current default container registry, enter:

```
olcnectl module update \  
--environment-name myenvironment \  
--name mycluster \  
--container-registry container-registry-austin-mirror.oracle.com/olcne/ \  
--kube-version 1.21.14-3
```

The `--kube-version` option specifies the release to which you want to upgrade. This example uses release number 1.21.14-3. The specified container registry becomes the new default container registry for all subsequent updates or upgrades.

Make sure you upgrade to the latest Kubernetes release. To get the version number of the latest Kubernetes release, see [Release Notes](#).

The nodes in the environment are updated to Kubernetes 1.21.14-3.

Example 4.31 Setting access to external IP addresses for Kubernetes services

This example sets the range of external IP addresses that Kubernetes services can access.

```
olcnectl module update \  
--environment-name myenvironment \  
--name mycluster \  
--restrict-service-externalip-cidrs 192.0.2.0/24,198.51.100.0/24
```

Example 4.32 Modifying host SELinux settings

This example updates the configuration with the Platform API Server that nodes in the Kubernetes cluster have SELinux `enforcing` mode enabled.

```
olcnectl module update \  
--environment-name myenvironment \  
--name mycluster \  
--selinux enforcing
```

4.15 Module Validate

Validates a module for an environment. When you validate a module, the nodes are checked to make sure they are set up correctly to run the module. If the nodes are not set up correctly, the commands required to fix each node are shown in the output and optionally saved to files.

Syntax

```
olcnectl module validate  
{-E|--environment-name} environment_name  
{-N|--name} name  
[{-g|--generate-scripts}]  
[{-h|--help}]  
[globals]
```

Where:

`{-E|--environment-name}` The Oracle Cloud Native Environment. The value of `environment_name` is the name to use to identify an environment.

Examples

<code>{-N --name} name</code>	The module name. The value of <code>name</code> is the name to use to identify a module in an environment.
<code>{-g --generate-scripts}</code>	Generates a set of scripts which contain the commands required to fix any set up errors for the nodes in a module. A script is created for each node in the module, saved to the local directory, and named <code>hostname:8090.sh</code> .
<code>{-h --help}</code>	Lists information about the command and the available options.

Where `globals` is one or more of the global options as described in [Section 1.3, “Using Global Flags”](#).

Examples

Example 4.33 Validating a module

To validate a Kubernetes module named `mycluster` in an environment named `myenvironment`:

```
olcnectl module validate \  
--environment-name myenvironment \  
--name mycluster
```

4.16 Template

Generates a simple configuration file template. The template file is named `config-file-template.yaml` and created in the local directory.

Syntax

```
olcnectl template  
[{-h|--help}]
```

Where:

<code>{-h --help}</code>	Lists information about the command and the available options.
--------------------------	--

Examples

Example 4.34 Creating a sample configuration template

To create a sample configuration template:

```
olcnectl template
```

