Oracle Cloud Native Environment

Platform Command-Line Interface for Release 1.6

F75585-07
November 2023

Oracle Cloud Native Environment Platform Command-Line Interface for Release 1.6,

F75585-07

Contents

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.

Conventions

The following text conventions are used in this document:

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

Documentation Accessibility

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

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 that positively impacts our employees, customers, and partners, we are working to remove insensitive terms from our products and documentation. We are also mindful of the necessity to maintain compatibility with our customers' existing technologies and the need to ensure continuity of service as Oracle's offerings and industry standards evolve. Because of these technical constraints, our effort to remove insensitive terms is ongoing and will take time and external cooperation.

1

Introduction to the 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 Platform CLI Commands.

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

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:
  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
      --olcne-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 writen 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 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 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 
"1.25.11")
...
```

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

```
olcnectl module update --help
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 --environment-name myenvironment --name mycluster 
--help
Update a module

Usage:
  olcnectl module update [flags]

Flags:
      --compact string                                 Allow workloads to be 
deployed on control-plane nodes (default "false")
```

```
  -r, --container-registry string                    Container Registry that holds
the kubernetes images

  -c, --control-plane-nodes string       A comma separated list of control plane nodes
  -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 "1.25.11")
  -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.

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
```

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-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 `$HOME/.olcne/olcne.conf` file.
For more information on setting the Platform API Server see Setting the Platform API Server.
This option maps to the `$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.

`--update-config`
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.
The global arguments configuration information is saved to `$HOME/.olcne/olcne.conf` on the local host.
If you use Vault to generate certificates for nodes, the certificate is saved to `$HOME/.olcne/certificates/`*`environment_name`*`/` on the local host.

`--olcne-ca-path` ***`ca_path`***
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 `/etc/olcne/certificates/ca.cert`, or gathered from the local configuration if the `--update-config` option is used.
This option maps to the `$OLCNE_SM_CA_PATH` environment variable. If this environment variable is set it takes precedence over and overrides the Platform CLI setting.

`--olcne-node-cert-path` ***`node_cert_path`***
The path to a predefined certificate, or the a destination if using a secrets manager to download the certificate. The default is `/etc/olcne/certificates/node.cert`, or gathered from the local configuration if the `--update-config` option is used.
This option maps to the `$OLCNE_SM_CERT_PATH` environment variable. If this environment variable is set it takes precedence over and overrides the Platform CLI setting.

`--olcne-node-key-path` ***`node_key_path`***
The path to a predefined key, or the destination of the key if using a secrets manager to download the key. The default is `/etc/olcne/certificates/node.key`, or gathered from the local configuration if the `--update-config` option is used.
This option maps to the `$OLCNE_SM_KEY_PATH` environment variable. If this environment variable is set it takes precedence over and overrides the Platform CLI setting.

`--olcne-tls-cipher-suites` ***`ciphers`***
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:

- `TLS_AES_128_GCM_SHA256`
- `TLS_AES_256_GCM_SHA384`
- `TLS_CHACHA20_POLY1305_SHA256`
- `TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA`
- `TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256`
- `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_CHACHA20_POLY1305_SHA256`
- `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_CHACHA20_POLY1305_SHA256`
- `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_SHA3
84
```

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.

`--olcne-tls-min-version` ***`version`***
The TLS minimum version for Oracle Cloud Native Environment components. The default is `VersionTLS12`. Options are:

- `VersionTLS10`
- `VersionTLS11`

- `VersionTLS12`
- `VersionTLS13`

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

`--vault-address` ***`vault_address`***
The IP address of the Vault instance. The default is `https://127.0.0.1:8200`, or gathered from the local configuration if the `--update-config` option is used.

`--vault-cert-sans` ***`vault_cert_sans`***
Subject Alternative Names (SANs) to pass to Vault to generate the Oracle Cloud Native Environment certificate. The default is `127.0.0.1`, or gathered from the local configuration if the `--update-config` option is used.

`--vault-token` ***`vault_token`***
The Vault authentication token.

Setting up Command Line Completion

You can set up command line completion for the `olcnectl` command. Use the `olcnectl completion` command to generate a command line completion script for the shell. For example, to generate a command line completion script for the Bash shell on Oracle Linux, run:

```
olcnectl completion bash
```

The generated command line completion script must be saved at `/etc/bash_completion.d/olcnectl`.

You can generate the script and save it to the correct location using:

```
olcnectl completion bash | sudo tee /etc/bash_completion.d/olcnectl
```

You need to start a new shell session for this setup to take effect. This setup also requires the `bash-completion` package to be installed on the system.

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.

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 \
--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
```

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.3QKNuRoTqLbjXaGBOmO6Psjh
      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: 1.25.11
          container-registry: container-registry.oracle.com/olcne
          load-balancer: lb.example.com:6443
          control-plane-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.

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 Operator Lifecycle Manager module, the Istio module, and finally, the Oracle Cloud Infrastructure Cloud Controller Manager module. 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
          control-plane-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
          control-plane-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
          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: operator-lifecycle-manager
        name: myolm
        args:
          olm-kubernetes-module: mycluster2
      - module: istio
        name: myistio
        args:
          istio-kubernetes-module: mycluster2
      - module: oci-ccm
        name: myoci
        args:
          oci-ccm-kubernetes-module: mycluster2
          oci-region: us-ashburn-1
          oci-tenancy: ocid1.tenancy.oc1..unique_ID
          oci-compartment: ocid1.compartment.oc1..unique_ID
          oci-user: ocid1.user.oc1..unique_ID
          oci-fingerprint: b5:52:...
          oci-private-key-file: /home/opc/.oci/oci_api_key.pem
          oci-vcn: ocid1.vcn.oc1..unique_ID
          oci-lb-subnet1: ocid1.subnet.oc1..unique_ID
```

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
          control-plane-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
```

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
```

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 Installing Using a Configuration File, to create an environment and deploy Kubernetes, with the addition of the Operator Lifecycle Manager module.

```
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
          control-plane-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: operator-lifecycle-manager
        name: myolm
        args:
          olm-kubernetes-module: mycluster
```

Install 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 Operator Lifecycle Manager module is installed into the existing Kubernetes cluster in the environment.

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 Operator Lifecycle Manager module and not the Kubernetes module in an environment, create a file similar to the following. This file is the same as used in 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: operator-lifecycle-manager
        name: myolm
        args:
          olm-kubernetes-module: mycluster
```

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

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

The Operator Lifecycle Manager module is uninstalled from the environment, while leaving the Kubernetes module untouched.

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
          control-plane-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
...
```

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
          control-plane-nodes:
            - control1.example.com:8090
            - control2.example.com:8090
            - control3.example.com:8090
            - control4.example.com:8090
            - control5.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.

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: 1.25.11`, and for the Istio module, add `istio-version: 1.16.7`:

```
...
    modules:
      - module: kubernetes
        name: mycluster
        args:
          container-registry: container-registry.oracle.com/olcne
          load-balancer: lb.example.com:6443
          kube-version: 1.25.11
          control-plane-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: istio
        name: myistio
        args:
          istio-kubernetes-module: mycluster
          istio-version: 1.16.7
```

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

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

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.

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.

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.

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

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"
```

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.

4 Platform CLI Commands

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

Certificates Copy

Copies the generated CA Certificates and private keys to the Kubernetes nodes.

Copies and installs the CA Certificates and private keys for a set of nodes from a pre-generated set. The files must be located within a specific directory structure.

The certificate authority bundle must be located at: `<cert-dir>/ca.cert`.

The certificate for each node must be located at `<cert-dir>/<node-address>/node.cert`.

The node key must be located at `<cert-dir>/<node-address>/node.key`.

Syntax

```
olcnectl certificates copy
[--cert-dir certificate-directory]
[{-h|--help}]
{-n|--nodes} nodes
[{-R|--remote-command} remote-command]
[{-i|--ssh-identity-file} file_location]
[{-l|--ssh-login-name} username]
[--timeout minutes]
```

Where:

`--cert-dir` ***`certificate-directory`***
The directory to read or write key material generated by this utility. The default is `<CURRENT_DIR>/certificates`.

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

`{-n|--nodes}` ***`nodes`***
A comma separated list of the hostnames or IP addresses of nodes.
Sets the node(s) on which to perform an action. Any nodes that are not the local node use the command indicated by `--remote-command` to connect to the host (by default, `ssh`). If a node address resolves to the local system, all commands are executed locally without using the remote command.

`{-R|--remote-command}` ***`remote-command`***
Opens a connection to a remote host. The address of the host and the command are appended to the end of this command. For example:

```
ssh -i ~/.ssh/myidfile -l myuser
```

The default remote command is `ssh`.

`{-i|--ssh-identity-file}` ***`file_location`***
The location of the SSH identity file. If no value is specified, the operating system defaults are used.

`{-l|--ssh-login-name}` ***`username`***
The username to log in using SSH. The default is `opc`.

`--timeout` ***`minutes`***
The number of minutes to set for command timeouts. The default is `40` minutes.

Examples

Example 4-1 Copy certificates to nodes

This example copies the certificates to the nodes listed.

```
olcnectl certificates copy \
--nodes
operator.example.com,control1.example.com,worker1.example.com,worker2.e
xample.com
```

Certificates Distribute

Generates and distributes CA Certificates and keys to the nodes.

Syntax

```
olcnectl certificates distribute
[--byo-ca-cert certificate-path]
[--byo-ca-key key-path]
[--cert-dir certificate-directory]
[--cert-request-common-name common_name]
[--cert-request-country country]
[--cert-request-locality locality]
[--cert-request-organization organization]
[--cert-request-organization-unit organization-unit]
[--cert-request-state state]
[{-h|--help}]
[{-n|--nodes} nodes]
[--one-cert]
[{-R|--remote-command} remote-command]
[{-i|--ssh-identity-file} file_location]
[{-l|--ssh-login-name} username]
[--timeout minutes]
[{-y|--yes}]
```

Where:

`--byo-ca-cert` ***`certificate-path`***
The path to an existing public CA Certificate.

`--byo-ca-key` ***`key-path`***
The path to an existing private key.

`--cert-dir` ***`certificate-directory`***
The directory to read or write key material generated by this utility. The default is `<CURRENT_DIR>/certificates`.

`--cert-request-common-name` ***`common_name`***
The Certificate Common Name suffix. The default is `example.com`.

`--cert-request-country` ***`country`***
The two letter country code where your company is located, for example, `US` for the United States, `GB` for the United Kingdom and `CN` for China. The default is `US`.

`--cert-request-locality` ***`locality`***
The name of the city where your company is located. The default is `Redwood City`.

`--cert-request-organization` ***`organization`***
The name of the your company. The default is `OLCNE`.

`--cert-request-organization-unit` ***`organization-unit`***
The name of the department within your company. The default is `OLCNE`.

`--cert-request-state` ***`state`***
The name of the state or province where your company is located. The default is `California`.

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

`{-n|--nodes}` ***`nodes`***
A comma separated list of the hostnames or IP addresses of nodes.
Sets the node(s) on which to perform an action. Any nodes that are not the local node use the command indicated by `--remote-command` to connect to the host (by default, `ssh`). If a node address resolves to the local system, all commands are executed locally without using the remote command.

`--one-cert`
Sets whether to generate a single certificate that can be used to authenticate all the hosts. By default this option is not set.

`{-R|--remote-command}` ***`remote-command`***
Opens a connection to a remote host. The address of the host and the command are appended to the end of this command. For example:

```
ssh -i ~/.ssh/myidfile -l myuser
```

The default remote command is `ssh`.

`{-i|--ssh-identity-file}` ***`file_location`***
The location of the SSH identity file. If no value is specified, the operating system defaults are used.

`{-l|--ssh-login-name}` ***`username`***
The username to log in using SSH. The default is `opc`.

`--timeout` *`minutes`*
The number of minutes to set for command timeouts. The default is `40` minutes.

Examples

Example 4-2 Distribute certificates to nodes

This example distributes the certificates to the nodes listed.

```
olcnectl certificates distribute \
--nodes
operator.example.com,control1.example.com,worker1.example.com,worker2.e
xample.com
```

Example 4-3 Distribute certificates for nodes with Certificate and SSH login information

This example distributes the certificates for the nodes listed, sets the Certificate and SSH login information. This also accepts all prompts.

```
olcnectl certificates distribute \
--nodes
operator.example.com,control1.example.com,worker1.example.com,worker2.e
xample.com \
--cert-request-common-name cloud.example.com \
--cert-request-country US \
--cert-request-locality "My Town" \
--cert-request-organization "My Company" \
--cert-request-organization-unit "My Company Unit" \
--cert-request-state "My State" \
--ssh-identity-file ~/.ssh/id_rsa \
--ssh-login-name oracle \
--yes
```

Example 4-4 Distribute certificates for nodes using an existing CA Certificate and private key

This example distributes the certificate for the nodes listed using an existing CA Certificate and private key. This is useful for generating and copying the certificate information to nodes you want to add to an existing Kubernetes cluster.

```
olcnectl certificates distribute \
--nodes worker3.example.com,worker4.example.com \
--byo-ca-cert /etc/olcne/configs/certificates/production/ca.cert \
--byo-ca-key /etc/olcne/configs/certificates/production/ca.key
```

Example 4-5 Distribute certificates for nodes using an existing CA Certificate and private key

This example distributes the certificate for the nodes listed using an existing CA Certificate and private key. This is useful for generating and copying the certificate information to nodes you want to add to an existing Kubernetes cluster.

```
olcnectl certificates distribute \
--nodes worker3.example.com,worker4.example.com \
--byo-ca-cert $HOME/certificates/ca/ca.cert \
--byo-ca-key $HOME/certificates/ca/ca.key
```

Example 4-6 Distribute certificates for nodes using an existing CA Certificate and private key and SSH login information

This example distributes the certificate for the nodes listed using an existing CA Certificate and private key.

```
olcnectl certificates distribute \
--nodes
operator.example.com,control1.example.com,worker1.example.com,worker2.example
.com \
--byo-ca-cert /etc/olcne/certificates/ca.cert \
--byo-ca-key /etc/olcne/certificates/ca.key
```

Example 4-7 Distribute certificates for nodes using an existing CA Certificate and private key and SSH login information

This example distributes the certificate for the nodes listed using an existing CA Certificate and private key.

```
olcnectl certificates distribute \
--nodes
operator.example.com,control1.example.com,worker1.example.com,worker2.example
.com \
--byo-ca-cert $HOME/certificates/ca/ca.cert \
--byo-ca-key $HOME/certificates/ca/ca.key
```

Certificates Generate

Generates CA Certificates for the nodes.

Creates the CA Certificates and keys required to authenticate the Platform CLI, Platform API Server, and Platform Agent for a set of hosts.

If a Certificate Authority is created, its key material is placed in the directory specified with the `--cert-dir` option. The CA Certificate is written to the file `ca.cert` in that directory, and the private key is written to `ca.key`.

Syntax

```
olcnectl certificates generate
[--byo-ca-cert certificate-path]
```

```
[--byo-ca-key key-path]
[--cert-dir certificate-directory]
[--cert-request-common-name common_name]
[--cert-request-country country]
[--cert-request-locality locality]
[--cert-request-organization organization]
[--cert-request-organization-unit organization-unit]
[--cert-request-state state]
[{-h|--help}]
{-n|--nodes} nodes
[--one-cert]
```

Where:

`--byo-ca-cert` *`certificate-path`*
The path to an existing public CA Certificate.

`--byo-ca-key` *`key-path`*
The path to an existing private key.

`--cert-dir` *`certificate-directory`*
The directory to read or write key material generated by this utility. The default is `<CURRENT_DIR>/certificates`.

`--cert-request-common-name` *`common_name`*
The Certificate Common Name suffix. The default is `example.com`.

`--cert-request-country` *`country`*
The two letter country code where your company is located, for example, `US` for the United States, `GB` for the United Kingdom and `CN` for China. The default is `US`.

`--cert-request-locality` *`locality`*
The name of the city where your company is located. The default is `Redwood City`.

`--cert-request-organization` *`organization`*
The name of the your company. The default is `OLCNE`.

`--cert-request-organization-unit` *`organization-unit`*
The name of the department within your company. The default is `OLCNE`.

`--cert-request-state` *`state`*
The name of the state or province where your company is located. The default is `California`.

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

`{-n|--nodes}` *`nodes`*
A comma separated list of the hostnames or IP addresses of nodes.
Sets the node(s) on which to perform an action. Any nodes that are not the local node use the command indicated by `--remote-command` to connect to the host (by default, `ssh`). If a node address resolves to the local system, all commands are executed locally without using the remote command.

`--one-cert`
Sets whether to generate a single certificate that can be used to authenticate all the hosts. By default this option is not set.

Examples

Example 4-8 Generate certificates for nodes

This example generates the certificates for the nodes listed.

```
olcnectl certificates generate \
--nodes
operator.example.com,control1.example.com,worker1.example.com,worker2.example
.com
```

Example 4-9 Generate certificates for nodes with certificate information

This example generates the certificates for the nodes listed, and sets the certificate information.

```
olcnectl certificates generate \
--nodes
operator.example.com,control1.example.com,worker1.example.com,worker2.example
.com \
--cert-request-common-name cloud.example.com \
--cert-request-country US \
--cert-request-locality "My Town" \
--cert-request-organization "My Company" \
--cert-request-organization-unit "My Company Unit" \
--cert-request-state "My State"
```

Example 4-10 Generate certificates for the Kubernetes ExternalIPs service with existing key information

This example generates the certificates for the Kubernetes ExternalIPs service using existing CA certificate and private key.

```
olcnectl certificates generate \
--nodes externalip-validation-webhook-service.externalip-validation-
system.svc,\
externalip-validation-webhook-service.externalip-validation-
system.svc.cluster.local \
--cert-dir /etc/olcne/certificates/restrict_external_ip/ \
--byo-ca-cert /etc/olcne/configs/certificates/production/ca.cert \
--byo-ca-key /etc/olcne/configs/certificates/production/ca.key \
--one-cert
```

Example 4-11 Generate certificates for the Kubernetes ExternalIPs service with existing key information

This example generates the certificates for the Kubernetes ExternalIPs service using existing CA certificate and private key.

```
olcnectl certificates generate \
--nodes externalip-validation-webhook-service.externalip-validation-
system.svc,\
externalip-validation-webhook-service.externalip-validation-
system.svc.cluster.local \
--cert-dir $HOME/certificates/restrict_external_ip/ \
--byo-ca-cert $HOME/certificates/ca/ca.cert \
--byo-ca-key $HOME/certificates/ca/ca.key \
--one-cert
```

Example 4-12 Generate certificates for the Kubernetes ExternalIPs service

This example generates the certificates for the Kubernetes ExternalIPs service using a private CA certificate and private key that is generated at the time.

```
olcnectl certificates generate \
--nodes externalip-validation-webhook-service.externalip-validation-
system.svc,\
externalip-validation-webhook-service.externalip-validation-
system.svc.cluster.local \
--cert-dir /etc/olcne/certificates/restrict_external_ip/ \
--cert-request-organization-unit "My Company Unit" \
--cert-request-organization "My Company" \
--cert-request-locality "My Town" \
--cert-request-state "My State" \
--cert-request-country US \
--cert-request-common-name cloud.example.com \
--one-cert
```

Completion

Generates a command line completion (also known as tab completion) script.

Syntax

```
olcnectl completion
shell
[--no-descriptions]
[{-h|--help}]
```

Where:

shell
The shell type. The options are:

- `bash`: Generates the command line completion script for the Bash shell.
- `fish`: Generates the command line completion script for the fish shell.
- `powershell`: Generates the command line completion script for PowerShell shell.
- `zsh`: Generates the command line completion script for the Zsh shell.

`--no-descriptions`
Disables completion descriptions in the generated script.

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

Examples

Example 4-13 Generating a command line completion script for Bash

This example generates a command line completion script for a Bash shell.

```
olcnectl completion bash
```

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 `$HOME/.olcne/olcne.conf` file.
For more information on setting the Platform API Server see Setting the Platform API Server.
This option maps to the `$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.

`--update-config`
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.
The global arguments configuration information is saved to `$HOME/.olcne/olcne.conf` on the local host.
If you use Vault to generate certificates for nodes, the certificate is saved to `$HOME/.olcne/certificates/`*`environment_name`*`/` on the local host.

`--olcne-ca-path` ***`ca_path`***
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 `/etc/olcne/certificates/ca.cert`, or gathered from the local configuration if the `--update-config` option is used.
This option maps to the `$OLCNE_SM_CA_PATH` environment variable. If this environment variable is set it takes precedence over and overrides the Platform CLI setting.

`--olcne-node-cert-path` ***`node_cert_path`***
The path to a predefined certificate, or the a destination if using a secrets manager to download the certificate. The default is `/etc/olcne/certificates/node.cert`, or gathered from the local configuration if the `--update-config` option is used.
This option maps to the `$OLCNE_SM_CERT_PATH` environment variable. If this environment variable is set it takes precedence over and overrides the Platform CLI setting.

`--olcne-node-key-path` ***`node_key_path`***
The path to a predefined key, or the destination of the key if using a secrets manager to download the key. The default is `/etc/olcne/certificates/node.key`, or gathered from the local configuration if the `--update-config` option is used.
This option maps to the `$OLCNE_SM_KEY_PATH` environment variable. If this environment variable is set it takes precedence over and overrides the Platform CLI setting.

`--olcne-tls-cipher-suites` *`ciphers`*
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:

- `TLS_AES_128_GCM_SHA256`
- `TLS_AES_256_GCM_SHA384`
- `TLS_CHACHA20_POLY1305_SHA256`
- `TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA`
- `TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256`
- `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_CHACHA20_POLY1305_SHA256`
- `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_CHACHA20_POLY1305_SHA256`
- `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.

`--olcne-tls-min-version` ***`version`***
The TLS minimum version for Oracle Cloud Native Environment components. The default is `VersionTLS12`. Options are:

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

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

`--vault-address` ***`vault_address`***
The IP address of the Vault instance. The default is `https://127.0.0.1:8200`, or gathered from the local configuration if the `--update-config` option is used.

`--vault-cert-sans` ***`vault_cert_sans`***
Subject Alternative Names (SANs) to pass to Vault to generate the Oracle Cloud Native Environment certificate. The default is `127.0.0.1`, or gathered from the local configuration if the `--update-config` option is used.

`--vault-token` ***`vault_token`***
The Vault authentication token.

Examples

Example 4-14 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.3QKNuRoTqLbjXaGBOmO6Psjh \
--vault-address https://192.0.2.20:8200 \
--update-config
```

Example 4-15   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
```

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}` ***`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 Using Global Flags.

Examples

Example 4-16   Deleting an environment

To delete an environment named `myenvironment`:

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

Environment Update

Updates or upgrades the Platform Agent on nodes in an existing environment.

Syntax

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

Where:

`olcne`
Specifies to update the Platform Agent on each node in an environment.

`{-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.
The Platform Agent is updated on only the nodes used in this module.

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

Where *globals* is one or more of the global options as described in Using Global Flags.

Examples

Example 4-17 Updating Platform Agents

To update the Platform Agent on each node in an environment named `myenvironment`:

```
olcnectl environment update olcne \
--environment-name myenvironment
```

Example 4-18 Updating Platform Agents in a module

To update the Platform Agent on each node in a Kubernetes module named `mycluster` in an environment named `myenvironment`:

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

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}` ***`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.

`--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 Using Global Flags.

Examples

Example 4-19 Reporting summary details about an environment

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

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

Example 4-20 Reporting details about an environment

To report details about the environment named `myenvironment`:

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

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 Using Global Flags.

Examples

Example 4-21 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
```

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:

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

`{-M|--module}` ***`module`***
The module type to create in an environment. The value of *module* is the name of a module type. The available module types are:

- kubernetes
- calico
- multus
- oci-ccm
- metallb
- gluster (Deprecated)
- operator-lifecycle-manager
- prometheus
- grafana
- istio
- helm (Deprecated)
- oci-csi (Deprecated)

`{-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 *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:**

`{-o|--apiserver-advertise-address}` ***`IP_address`***
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 `--control-plane-nodes` 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 `--control-plane-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.

`--compact {true|false}`
Sets whether to allow non-system Kubernetes workloads to run on control plane nodes. The default is `false`.
If you set this to `true`, the Platform API Server does not taint the control plane node(s). This allows non-system Kubernetes workloads to be scheduled and run on control plane nodes.

> **Important:**
>
> For production environments, this option must be set to `false` (the default).

`{-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.

`{-c|--control-plane-nodes}` ***`nodes ...`***
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, `control1.example.com:8090,control2.example.com:8090`.
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_AES_128_GCM_SHA256`
- `TLS_AES_256_GCM_SHA384`
- `TLS_CHACHA20_POLY1305_SHA256`
- `TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA`
- `TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256`
- `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_CHACHA20_POLY1305_SHA256`
- `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_CHACHA20_POLY1305_SHA256`
- `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_SHA3
84
```

`--kube-tls-min-version` ***`version`***
The TLS minimum version for Kubernetes components. The default is `VersionTLS12`.
Options are:

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

`{-l|--load-balancer}` ***`load_balancer`***
The Kubernetes API server load balancer hostname or IP address, and port. The default port is `6443`. For example, `192.0.2.100:6443`.

`{-m|--master-nodes}` ***`nodes ...`***
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, `control1.example.com:8090,control2.example.com:8090`.
If you do not provide this value, you are prompted for it by the Platform CLI.

Important:

This option is deprecated. Use the `--control-plane-nodes` option instead.

`{-g|--nginx-image}` ***`container_location`***
The location for an NGINX container image to use in a highly available (HA) cluster with multiple control plane nodes. This is optional.
You can use this option if you do not provide your own load balancer using the `--load-balancer` option. This option may be useful if you are using a mirrored container registry. For example:
```
--nginx-image mirror.example.com:5000/olcne/nginx:1.17.7
```
By default, `podman` is used to pull the NGINX image that is configured in `/usr/libexec/pull_olcne_nginx`. If you set the `--nginx-image` option to use another NGINX container image, the location of the image is written to `/etc/olcne-nginx/image`, and overrides the default image.

`--node-labels` ***`label`***

> **Important:**
>
> This option, and the Oracle Cloud Infrastructure Container Storage Interface module (`oci-csi`) that required this option, is deprecated in Release 1.5.

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 (`oci-csi`). The label should be in the format:
`failure-domain.beta.kubernetes.io/zone=`*`region-identifier`*`-AD-`*`availability-domain-number`*
For example:
`--node-labels failure-domain.beta.kubernetes.io/zone=US-ASHBURN-AD-1`
For a list of the Availability Domains, see the Oracle Cloud Infrastructure documentation.

`--node-ocids` ***`OCIDs`***

> **Important:**
>
> This option, and the Oracle Cloud Infrastructure Container Storage Interface module (`oci-csi`) that required this option, is deprecated in Release 1.5.

A comma separated list of Kubernetes nodes (both control plane and worker nodes) with their Oracle Cloud Identifiers (OCIDs). This option is 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=ocid1.instance...,wo
rker2.example.com=ocid1.instance...
```
For information about OCIDs, see the Oracle Cloud Infrastructure documentation.

`{-p|--pod-cidr}` ***`pod_CIDR`***
The Kubernetes pod CIDR. The default is `10.244.0.0/16`. This is the range from which each Kubernetes pod network interface is assigned an IP address.

`{-n|--pod-network} {flannel|calico|none}`
The network fabric for the Kubernetes cluster. The options are:

- `flannel`: Installs Flannel. This is the default.
- `calico`: Installs the Calico module. You must disable `firewalld` on all Kubernetes nodes to use this option.
- `none`: Does not install any network fabric.

`{-P|--pod-network-iface}` ***`network_interface`***
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:

`--pod-network-iface "ens[1-5]|eth5"`

`--selinux {enforcing|permissive}`
Whether to use SELinux `enforcing` or `permissive` mode. `permissive` is the default. You should use this option if SELinux is set to `enforcing` on the control plane and worker nodes. SELinux is set to `enforcing` mode by default on the operating system and is the recommended mode.

`{-s|--service-cidr}` ***`service_CIDR`***
The Kubernetes service CIDR. The default is `10.96.0.0/12`. This is the range from which each Kubernetes service is assigned an IP address.

`{-i|--virtual-ip}` ***`virtual_ip`***
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 `--load-balancer` option. When you specify a virtual IP address, it is used as the primary IP address for control plane nodes.

`{-w|--worker-nodes}` ***`nodes ...`***
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 an 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 `8090`. For example, `worker1.example.com:8090,worker2.example.com:8090`.
If you do not provide this value, you are prompted for it by the Platform CLI.

`--restrict-service-externalip {true|false}`
Sets whether to restrict access to external IP addresses for Kubernetes services. The default is `true`, which restricts access to external IP addresses.
This option deploys a Kubernetes service named `externalip-validation-webhook-service` to validate `externalIPs` set in Kubernetes service configuration files. Access to any external IP addresses is set in a Kubernetes service configuration file using the `externalIPs` option in the `spec` section.

`--restrict-service-externalip-ca-cert` ***`path`***
The path to a CA certificate file for the `externalip-validation-webhook-service` application that is deployed when the `--restrict-service-externalip` option is set to `true`. For example, `/etc/olcne/certificates/restrict_external_ip/ca.cert`.

`--restrict-service-externalip-tls-cert` ***`path`***
The path to a CA certificate file for the `externalip-validation-webhook-service` application that is deployed when the `--restrict-service-externalip` option is set to `true`. For example, `/etc/olcne/certificates/restrict_external_ip/node.cert`.

`--restrict-service-externalip-tls-key` ***`path`***
The path to the private key for the `externalip-validation-webhook-service` application that is deployed when the `--restrict-service-externalip` option is set to `true`. For example, `/etc/olcne/certificates/restrict_external_ip/node.key`.

`--restrict-service-externalip-cidrs` ***`allowed_cidrs`***
Enter one or more comma separated CIDR blocks if you want to allow only IP addresses from the specified CIDR blocks. For example, `192.0.2.0/24,198.51.100.0/24`.

module_args **for the `calico` module:**

`--calico-container-registry` ***`container_registry`***
The container image registry to use when deploying the Calico container image. The default is `container-registry.oracle.com/olcne`.

`--calico-helm-module` ***`helm_module`***
The name of the `helm` module that the Calico module should be associated with.

Important:

This option is deprecated. Use the `--calico-kubernetes-module` option instead.

`--calico-kubernetes-module` ***`kubernetes_module`***
The name of the `kubernetes` module that the Calico module should be associated with.

`--calico-installation-config` ***`config-file-path`***
The path and location of a Calico configuration file.

`--calico-namespace` ***`namespace`***
The Kubernetes namespace in which Calico components are deployed. The default is `tigera-operator`.

`--calico-version` ***`version`***
The version of Calico to install. The default is the latest version. For information on the latest version number, see Release Notes.

module_args **for the `multus` module:**

`--multus-config` ***`config-file-path`***
The path and location of a Multus configuration file.

`--multus-container-registry` ***`container_registry`***
The container image registry to use when deploying the Multus container image. The default is `container-registry.oracle.com/olcne`.

`--multus-helm-module` ***`helm_module`***
The name of the `helm` module that the Multus module should be associated with.

Important:

This option is deprecated. Use the `--multus-kubernetes-module` option instead.

`--multus-kubernetes-module` ***`kubernetes_module`***
The name of the `kubernetes` module that the Multus module should be associated with.

`--multus-namespace` *`namespace`*
The Kubernetes namespace in which Multus components are deployed. The default is `kube-system`.

`--multus-version` *`version`*
The version of Multus to install. The default is the latest version. For information on the latest version number, see Release Notes.

***module_args* for the `oci-ccm` module:**

`--oci-ccm-helm-module` *`helm_module`*
The name of the `helm` module that the Oracle Cloud Infrastructure Cloud Controller Manager module should be associated with.

Important:

This option is deprecated. Use the `--oci-ccm-kubernetes-module` option instead.

`--oci-ccm-kubernetes-module` *`kubernetes_module`*
The name of the `kubernetes` module that the Oracle Cloud Infrastructure Cloud Controller Manager module should be associated with.

`--oci-tenancy` *`OCID`*
The OCID for the Oracle Cloud Infrastructure tenancy.

`--oci-region` *`region_identifier`*
The Oracle Cloud Infrastructure region identifier. The default is `us-ashburn-1`.
For a list of the region identifiers, see the Oracle Cloud Infrastructure documentation.

`--oci-compartment` *`OCID`*
The OCID for the Oracle Cloud Infrastructure compartment.

`--oci-user` *`OCID`*
The OCID for the Oracle Cloud Infrastructure user.

`--oci-private-key` *`path`*
The location of the private key for the Oracle Cloud Infrastructure API signing key. The private key must be located on the primary control plane node. The default is `/root/.ssh/id_rsa`.

Important:

This option is deprecated in Release 1.6.1. From Release 1.6.1 onwards, use the `--oci-private-key-file` option instead.

`--oci-private-key-file` *`path`*
The location of the private key for the Oracle Cloud Infrastructure API signing key.

Important:

The private key must be available on the **operator** node.

`--oci-fingerprint` ***`fingerprint`***
The fingerprint of the public key for the Oracle Cloud Infrastructure API signing key.

`--oci-passphrase` ***`passphrase`***
The passphrase for the private key for the Oracle Cloud Infrastructure API signing key, if one is set.

`--oci-vcn` ***`OCID`***
The OCID for the Oracle Cloud Infrastructure Virtual Cloud Network on which the Kubernetes cluster is available.

`--oci-lb-subnet1` ***`OCID`***
The OCID of the regional subnet for the Oracle Cloud Infrastructure load balancer. 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.

`--oci-lb-subnet2` ***`OCID`***
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.

`--oci-lb-security-mode {All|Frontend|None}`
This option sets whether the Oracle Cloud Infrastructure Cloud Controller Manager module 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 `None`.
For information on the security modes, see the Kubernetes Cloud Controller Manager implementation for Oracle Cloud Infrastructure documentation.

`--oci-use-instance-principals {true|false}`
Sets whether to enable an instance to make API calls in Oracle Cloud Infrastructure services. The default is `false`.

`--oci-container-registry` ***`container_registry`***
The container image registry to use when deploying the Oracle Cloud Infrastructure cloud provisioner image. The default is an empty string. The Platform API Server determines the correct repository for the version of the Oracle Cloud Infrastructure Cloud Controller Manager module that is to be installed. Alternatively, you can use a private registry.

`--ccm-container-registry` ***`container_registry`***
The container image registry to use when deploying the Oracle Cloud Infrastructure Cloud Controller Manager component images. The default is an empty string. The Platform API Server determines the correct repository for the version of the Oracle Cloud Infrastructure Cloud Controller Manager module that is to be installed. Alternatively, you can use a private registry.

`--oci-ccm-version` ***`version`***
The version of Oracle Cloud Infrastructure Cloud Controller Manager to install. The default is the latest version. For information on the latest version number, see Release Notes.

module_args **for the `metallb` module:**

`--metallb-helm-module` ***`helm_module`***
The name of the `helm` module that MetalLB should be associated with.

> **Important:**
>
> This option is deprecated. Use the `--metallb-kubernetes-module` option instead.

`--metallb-kubernetes-module` ***`kubernetes_module`***
The name of the `kubernetes` module that the MetalLB module should be associated with.

`--metallb-config` ***`path`***
The location of the file that contains the configuration information for MetalLB. This file must be located on operator node.

`--metallb-namespace` ***`namespace`***
The Kubernetes namespace in which to install MetalLB. The default namespace is `metallb-system`.

`--metallb-version` ***`version`***
The version of MetalLB to install. The default is the latest version. For information on the latest version number, see Release Notes.

`--metallb-container-registry` ***`container_registry`***
The container image registry and optional tag to use when installing MetalLB. The default is `container-registry.oracle.com/olcne`.

module_args **for the `gluster` module:**

> **Important:**
>
> The Gluster Container Storage Interface module, used to install Gluster and set up Glusterfs, is deprecated. The Gluster Container Storage Interface module may be removed in a future release.

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

> **Important:**
>
> This option is deprecated. Use the `--gluster-kubernetes-module` option instead.

`--gluster-kubernetes-module` ***`kubernetes_module`***
The name of the `kubernetes` module that the Gluster 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`.

`--gluster-server-user` ***`user`***
The username of the Heketi server admin user. The default is `admin`.

`--gluster-existing-secret-name` ***`secret`***
The name of the existing secret containing the admin password. The default is `heketi-admin`.

`--gluster-secret-key` ***`secret`***
The secret containing the admin password. The default is `secret`.

`--gluster-namespace` ***`namespace`***
The Kubernetes namespace in which to install the Gluster Container Storage Interface module. The default is `default`.

`--gluster-sc-name` ***`class_name`***
The StorageClass name for the Glusterfs StorageClass. The default is `hyperconverged`.

`--gluster-server-rest-auth {true|false}`
Whether the Heketi API server accepts REST authorization. The default is `true`.

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.

> **Important:**
>
> This option is deprecated. Use the `--olm-kubernetes-module` option instead.

`--olm-kubernetes-module` ***`kubernetes_module`***
The name of the `kubernetes` module that the Operator Lifecycle Manager module 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 `prometheus` module:**

`--prometheus-helm-module` ***`helm_module`***
The name of the `helm` module that Prometheus should be associated with.

> **Important:**
>
> This option is deprecated. Use the `--prometheus-kubernetes-module` option instead.

`--prometheus-kubernetes-module` ***`kubernetes_module`***
The name of the `kubernetes` module that the Prometheus module should be associated with.

`--prometheus-version` ***`version`***
The version of Prometheus to install. The default is the latest version. For information on the latest version number, see Release Notes.

`--prometheus-image` ***`container_registry`***
The container image registry and tag to use when installing Prometheus. The default is `container-registry.oracle.com/olcne/prometheus`.

`--prometheus-namespace` ***`namespace`***
The Kubernetes namespace in which to install Prometheus. The default namespace is `default`.

`--prometheus-persistent-storage {true|false}`
If this value is `false`, Prometheus writes its data into an `emptydir` on the host where the pod is running. If the pod migrates, metric data is lost.
If this value is `true`, Prometheus requisitions a Kubernetes `PersistentVolumeClaim` so that its data persists, despite destruction or migration of the pod.
The default is `false`.

`--prometheus-alerting-rules` ***`path`***
The path to a configuration file for Prometheus alerts.

`--prometheus-recording-rules` ***`path`***
The path to a configuration file for Prometheus recording rules.

`--prometheus-scrape-configuration` ***`path`***
The path to a configuration file for Prometheus metrics scraping.

module_args **for the `grafana` module:**

`--grafana-helm-module` ***`helm_module`***
The name of the `helm` module that Grafana should be associated with.

> **Important:**
>
> This option is deprecated. Use the `--grafana-kubernetes-module` option instead.

`--grafana-kubernetes-module` ***`kubernetes_module`***
The name of the `kubernetes` module that the Grafana module should be associated with.

`--grafana-version` ***`version`***
The version of Grafana to install. The default is the latest version. For information on the latest version number, see Release Notes.

`--grafana-container-registry` ***`container_registry`***
The container image registry and tag to use when installing Grafana. The default is `container-registry.oracle.com/olcne`.

`--grafana-namespace` ***`namespace`***
The Kubernetes namespace in which to install Grafana. The default namespace is `default`.

`--grafana-dashboard-configmaps` ***`configmap`***
The name of the ConfigMap reference that contains the Grafana dashboards.

`--grafana-dashboard-providers` ***`path`***
The location of the file that contains the configuration for the Grafana dashboard providers.

`--grafana-datasources` ***`path`***
The location of the file that contains the configuration for the Grafana data sources.

`--grafana-existing-sercret-name` ***`secret`***
The name of the existing secret containing the Grafana admin password.

`--grafana-notifiers` ***`path`***
The location of the file that contains the configuration for the Grafana notifiers.

`--grafana-pod-annotations` ***`annotations`***
A comma separated list of annotations to be added to the Grafana pods.

`--grafana-pod-env` ***`env_vars`***
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.

> **Important:**
>
> This option is deprecated. Use the `--istio-kubernetes-module` option instead.

`--istio-kubernetes-module` ***`kubernetes_module`***
The name of the `kubernetes` module that the Istio module 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-enable-grafana {true|false}`
Sets whether to deploy the Grafana module to visualize the metrics stored in Prometheus for Istio. The default is `true`.

`--istio-enable-prometheus {true|false}`
Sets whether to deploy the Prometheus module to store the metrics for Istio. The default is `true`.

`--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.

`--istio-parent` ***`name`***
The name of the `istio` module to use with a custom profile. When used with the `--istio-profile` option, allows multiple instances of the `istio` module to attach Istio platform components to a single Istio control plane. When this option is set, the default Istio profile is replaced with the a mostly empty profile. The only contents of the profile are the container image hub location, and tags that correspond to the currently installed version of the `istio` module.

`--istio-profile` ***`path`***
The path to the file that contains the `spec` section of an `IstioOperator` resource from the `install.istio.io/v1alpha1` Kubernetes API. The values in this resource are laid over top of, and override, the default profile for Istio.
For information on the `IstioOperator` resource file, see the upstream Istio documentation.

module_args **for the `helm` module:**

> **Important:**
>
> The `helm` module is deprecated in Release 1.6. Helm is now automatically installed with the Kubernetes module to perform optional module installations.

`--helm-kubernetes-module` ***`kubernetes_module`***
The name of the `kubernetes` module that Helm should be associated with. Each instance of Kubernetes can have one instance of Helm associated with it.

`--helm-version` *`version`*
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 `oci-csi` module:**

> **Important:**
>
> The `oci-csi` module is deprecated in Release 1.5. You should instead use the `oci-ccm` module from Release 1.5 onwards. If you have upgraded from Release 1.4 to 1.5, the `oci-csi` module is automatically renamed to `oci-ccm`. You must also perform another step after the upgrade to make sure the module is correctly configured. For information on upgrading the `oci-csi` module to the `oci-ccm` module, see Updates and Upgrades.

`--oci-csi-helm-module` *`helm_module`*
The name of the `helm` module that the Oracle Cloud Infrastructure Container Storage Interface module should be associated with.

> **Important:**
>
> This option is deprecated. Use the `--oci-ccm-kubernetes-module` option instead.

`--oci-tenancy` *`OCID`*
The OCID for the Oracle Cloud Infrastructure tenancy.

`--oci-region` *`region_identifier`*
The Oracle Cloud Infrastructure region identifier. The default is `us-ashburn-1`.
For a list of the region identifiers, see the Oracle Cloud Infrastructure documentation.

`--oci-compartment` *`OCID`*
The OCID for the Oracle Cloud Infrastructure compartment.

`--oci-user` *`OCID`*
The OCID for the Oracle Cloud Infrastructure user.

`--oci-private-key` *`path`*
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 `/root/.ssh/id_rsa`.

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

`--oci-fingerprint` *`fingerprint`*
The fingerprint of the public key for the Oracle Cloud Infrastructure API signing key.

`--oci-passphrase` ***`passphrase`***
The passphrase for the private key for the Oracle Cloud Infrastructure API signing key, if one is set.

`--oci-vcn` ***`OCID`***
The OCID for the Oracle Cloud Infrastructure Virtual Cloud Network on which the Kubernetes cluster is available.

`--oci-lb-subnet1` ***`OCID`***
The OCID of the regional subnet for the Oracle Cloud Infrastructure load balancer. 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.

`--oci-lb-subnet2` ***`OCID`***
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.

`--oci-lb-security-mode {All|Frontend|None}`
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 `None`.
For information on the security modes, see the Kubernetes Cloud Controller Manager implementation for Oracle Cloud Infrastructure documentation.

`--oci-container-registry` ***`container_registry`***
The container image registry to use when deploying the Oracle Cloud Infrastructure cloud provisioner image. The default is `iad.ocir.io/oracle`.

`--csi-container-registry` ***`container_registry`***
The container image registry to use when deploying the CSI component images. The default is `quay.io/k8scsi`.

Where *globals* is one or more of the global options as described in Using Global Flags.

Examples

Example 4-22 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 \
--control-plane-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 4-23 Creating a module for an HA cluster with an internal load balancer

This example 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.

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 \
--control-plane-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 4-24 Creating a module for a cluster to allow access to service IP address ranges

This example 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 \
--control-plane-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 4-25 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 \
--control-plane-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 4-26 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.

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 \
--control-plane-nodes control1.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 4-27 Creating a Calico module

This example creates a Calico module to set Calico as the pod networking CNI for a Kubernetes cluster. This example uses a Kubernetes module named `mycluster` and a Calico module named `mycalico`. It uses an existing configuration file named `calico-config.yaml` to configure Calico.

```
olcnectl module create \
--environment-name myenvironment \
--module calico \
--name mycalico \
--calico-kubernetes-module mycluster \
--calico-installation-config calico-config.yaml
```

Example 4-28 Creating a Multus module

This example creates a Multus module to enable bridged networking to pods. The `--module` option is set to `multus` to create a Multus module. This example uses a Kubernetes module named `mycluster` and a Multus module named `mymultus`. It uses an existing configuration file named `multus-config.conf` to configure Multus, which contains a Kubernetes `NetworkAttachmentDefinition` Custom Resource Definition.

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

Example 4-29 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` and an Istio module named `myistio`.

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-kubernetes-module mycluster
```

Example 4-30 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` and an Operator Lifecycle Manager module named `myolm`.

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-kubernetes-module mycluster
```

Example 4-31 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` and a Gluster Container Storage Interface module named `mygluster`.

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-kubernetes-module mycluster
```

Example 4-32 Creating a module for Oracle Cloud Infrastructure

This example creates a module that creates a Kubernetes StorageClass provisioner to access Oracle Cloud Infrastructure storage, and enables the provision of highly available application load balancers. The `--module` option is set to `oci-ccm` to create an Oracle Cloud Infrastructure Cloud Controller Manager module. This example uses a Kubernetes module named `mycluster` and an Oracle Cloud Infrastructure Cloud Controller Manager module named `myoci`.

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-file`
- `--oci-vcn`
- `--oci-lb-subnet1`
- `--oci-lb-subnet2`

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-ccm \
--name myoci \
--oci-ccm-kubernetes-module mycluster \
--oci-tenancy ocid1.tenancy.oc1..unique_ID \
--oci-compartment ocid1.compartment.oc1..unique_ID \
--oci-user ocid1.user.oc1..unique_ID \
--oci-fingerprint b5:52:... \
--oci-private-key-file /home/opc/.oci/oci_api_key.pem \
--oci-vcn ocid1.vcn.oc1..unique_ID \
--oci-lb-subnet1 ocid1.subnet.oc1..unique_ID \
--oci-lb-subnet2 ocid1.subnet.oc1..unique_ID
```

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]
```

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.

`{-g|--generate-scripts}`
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 *`hostname`*`:8090.sh`.

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

Where *globals* is one or more of the global options as described in Using Global Flags.

Examples

Example 4-33 Installing a module

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

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

Module Instances

Lists the installed modules in an environment.

Syntax

```
olcnectl module instances
{-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 one or more of the global options as described in Using Global Flags.

Examples

Example 4-34 Listing the deployed modules in an environment

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

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

Module List

Lists the available modules for an environment.

Syntax

```
olcnectl module list
{-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 one or more of the global options as described in Using Global Flags.

Examples

Example 4-35 Listing available modules in an environment

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

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

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:

`{-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.

`{-P|--property}` ***`property_name`***
The name of the property. You can get a list of the available properties using the `olcnectl module property list` command.

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

Where *globals* is one or more of the global options as described in Using Global Flags.

Examples

Example 4-36 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
```

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 Using Global Flags.

Examples

Example 4-37 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
```

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]
[--exclude pattern]
[--include pattern]
[--format {yaml|table}]
[{-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. When no name is specified, the output of the command contains information about all modules deployed in the selected 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.

`--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 Using Global Flags.

Examples

Example 4-38   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 4-39   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 4-40   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
```

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:

`{-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.

`{-g|--generate-scripts}`
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 `hostname:8090.sh`.

`{-F|--force}`
Skips the confirmation prompt.

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

Where *globals* is one or more of the global options as described in Using Global Flags.

Examples

Example 4-41 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
```

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
{-N|--name} name
[{-F|--force}]
[{-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.

{-F|--force}
Skips the confirmation prompt.

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

Where *globals* is one or more of the global options as described in Using Global Flags.

Examples

Example 4-42 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.

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
{-E|--environment-name} environment_name
{-N|--name} name
[{-g|--generate-scripts}]
[{-F|--force}]
[{-h|--help}]
[module_args ...]
[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.

`{-g|--generate-scripts}`
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 *`hostname`*`:8090.sh`.

`{-F|--force}`
Skips the confirmation prompt.

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

Where *module_args* is:

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

***module_args* for the `kubernetes` module:**

`{-k|--kube-version}` *version*
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.

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

`{-m|--master-nodes}` ***`nodes ...`***
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, `control1.example.com:8090,control2.example.com:8090`.

> **Important:**
>
> This option is deprecated. Use the `--control-plane-nodes` option instead.

`{-c|--control-plane-nodes}` ***`nodes ...`***
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, `control1.example.com:8090,control2.example.com:8090`.

`{-w|--worker-nodes}` ***`nodes ...`***
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.
The default port number for the Platform Agent is `8090`. For example, `worker1.example.com:8090,worker2.example.com:8090`.

`--compact {true|false}`
Sets whether to allow non-system Kubernetes workloads to run on control plane nodes. The default is `false`.
If you set this to `true`, the Platform API Server untaints control plane nodes if they are tainted. This allows non-system Kubernetes workloads to be scheduled and run on control plane nodes.
If you set this to `false`, the Platform API Server taints the control plane nodes if they are untainted. This prevents non-system Kubernetes workloads to be scheduled and run on control plane nodes.

> **Important:**
>
> For production environments, this option must be set to `false` (the default).

`--nginx-image` ***`container_location`***
The location of the NGINX container image to update. This is optional.
This option pulls the NGINX container image from the container registry location you specify to update NGINX on the control plane nodes. For example:
`--nginx-image container-registry.oracle.com/olcne/nginx:1.17.7`

`--helm-version` ***`version`***
Sets the Helm version for the upgrade. The default is the latest version. For information on the latest version number, see Release Notes.

`--restrict-service-externalip {true|false}`
Sets whether to restrict access to external IP addresses for Kubernetes services. The default is `true`, which restricts access to external IP addresses.
This option deploys a Kubernetes service named `externalip-validation-webhook-service` to validate `externalIPs` set in Kubernetes service configuration files. Access to any external IP addresses is set in a Kubernetes service configuration file using the `externalIPs` option in the `spec` section.

`--restrict-service-externalip-ca-cert` ***`path`***
The path to a CA certificate file for the `externalip-validation-webhook-service` application that is deployed when the `--restrict-service-externalip` option is set to `true`. For example, `/etc/olcne/certificates/restrict_external_ip/ca.cert`.

`--restrict-service-externalip-tls-cert` ***`path`***
The path to a CA certificate file for the `externalip-validation-webhook-service` application that is deployed when the `--restrict-service-externalip` option is set to `true`. For example, `/etc/olcne/certificates/restrict_external_ip/node.cert`.

`--restrict-service-externalip-tls-key` ***`path`***
The path to the private key for the `externalip-validation-webhook-service` application that is deployed when the `--restrict-service-externalip` option is set to `true`. For example, `/etc/olcne/certificates/restrict_external_ip/node.key`.

`--restrict-service-externalip-cidrs` ***`allowed_cidrs`***
Enter one or more comma separated CIDR blocks if you want to allow only IP addresses from the specified CIDR blocks. For example,
`192.0.2.0/24,198.51.100.0/24`.

`--selinux {enforcing|permissive}`
Whether to use SELinux `enforcing` or `permissive` mode. `permissive` is the default.
You should use this option if SELinux is set to `enforcing` on the control plane and worker nodes. SELinux is set to `enforcing` mode by default on the operating system and is the recommended mode.

module_args **for the `calico` module**:

`--calico-version` ***`version`***
Sets the Calico version for the upgrade. The default is the latest version. For information on the latest version number, see Release Notes.

`--calico-container-registry` ***`container_registry`***
The container registry that contains the Calico images when performing an update or upgrade. Use the Oracle Container Registry (the default) or a local registry to pull the Calico images.

module_args **for the `multus` module**:

`--multus-version` ***`version`***
Sets the Multus version for the upgrade. The default is the latest version. For information on the latest version number, see Release Notes.

`--multus-container-registry` ***`container_registry`***
The container registry that contains the Multus images when performing an update or upgrade. Use the Oracle Container Registry (the default) or a local registry to pull the Multus images.

module_args **for the `oci-ccm` module**:

`--oci-ccm-version` ***`version`***
Sets the Oracle Cloud Infrastructure Cloud Controller Manager version for the upgrade. The default is the latest version. For information on the latest version number, see Release Notes.

`--oci-container-registry` ***`container_registry`***
The container image registry to use when updating the Oracle Cloud Infrastructure cloud provisioner image. The default is an empty string. The Platform API Server determines the correct repository for the version of the Oracle Cloud Infrastructure Cloud Controller Manager module that is to be upgrade. Alternatively, you can use a private registry.

`--ccm-container-registry` ***`container_registry`***
The container image registry to use when updating the Oracle Cloud Infrastructure Cloud Controller Manager component images. The default is an empty string. The Platform API Server determines the correct repository for the version of the Oracle Cloud Infrastructure Cloud Controller Manager module that is to be upgraded. Alternatively, you can use a private registry.

`--oci-lb-subnet1` ***`OCID`***
The OCID of the regional subnet for the Oracle Cloud Infrastructure load balancer. 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.

`--oci-lb-subnet2` ***`OCID`***
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.

`--oci-lb-security-mode {All|Frontend|None}`
This option sets whether the Oracle Cloud Infrastructure Cloud Controller Manager module 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 `None`.
For information on the security modes, see the Kubernetes Cloud Controller Manager implementation for Oracle Cloud Infrastructure documentation.

module_args **for the `metallb` module**:

`--metallb-version` ***`version`***
Sets the MetalLB version for the upgrade. The default is the latest version. For information on the latest version number, see Release Notes.

`--metallb-container-registry` ***`container_registry`***
The container image registry and optional tag to use when upgrading MetalLB. The default is `container-registry.oracle.com/olcne`.

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

`--olm-version` ***`version`***
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.

module_args **for the `prometheus` module**:

`--prometheus-version` ***`version`***
Sets the Prometheus version for the upgrade. The default is the latest version. For information on the latest version number, see Release Notes.

`--prometheus-container-registry` ***`container_registry`***
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.

module_args **for the `grafana` module**:

`--grafana-version` ***`version`***
Sets the Grafana version for the upgrade. The default is the latest version. For information on the latest version number, see Release Notes.

`--grafana-container-registry` ***`container_registry`***
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.

module_args **for the `istio` module**:

`--istio-version` ***`version`***
Sets the Istio version for the upgrade. The default is the latest version. For information on the latest version number, see Release Notes.

`--istio-container-registry` ***`container_registry`***
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.

Where *globals* is one or more of the global options as described in Using Global Flags.

Examples

Example 4-43 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 \
--control-plane-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
```

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 \
--control-plane-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 `--control-plane-nodes` and `--worker-nodes` options, the Platform API Server removes those nodes from the cluster.

Example 4-44   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 4-45   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 4-46   Upgrading the Kubernetes release

To upgrade a Kubernetes module named `mycluster` in an environment named `myenvironment` to Kubernetes Release 1.25.11, enter:

```
olcnectl module update \
--environment-name myenvironment \
--name mycluster \
--kube-version 1.25.11
```

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

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 1.25.11.

Example 4-47 Upgrading using a different container registry

To upgrade a Kubernetes module named `mycluster` in an environment named `myenvironment` to Kubernetes Release 1.25.11 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 1.25.11
```

The `--kube-version` option specifies the release to which you want to upgrade. This example uses release number 1.25.11. 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 1.25.11.

Example 4-48 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 4-49 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
```

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}` ***`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.

`{-g|--generate-scripts}`
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 *`hostname`*`:8090.sh`.

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

Where *globals* is one or more of the global options as described in Using Global Flags.

Examples

Example 4-50 Validating a module

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

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

Module Version

Lists the latest module versions in the Oracle Container Registry that can be installed for the Oracle Cloud Native Environment currently on your system.

You can use the command before running `olcnectl module update` to see which versions of modules and their subcomponents will be installed.

The command parameters enable you to choose whether to print version information for one specific module and its subcomponents, or for all available modules.

Syntax

```
olcnectl module version
{-E|--environment-name} environment_name
[{--all | {-M|--module} module_name}]
[--print-subcomponents]
[{-o|--output} {pretty|yaml|json}]
[{-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.

`{--all | {-M|--module}` ***`module_name}`***
Specifies whether to print version information for all available modules, or for the specified module only. Options are:

- `--all`: Prints versions for all available modules.
- `{-M|--module}` *`module_name`*: Prints only the version of the module named *module_name*.

`--print-subcomponents`
Specifies that versions of the module subcomponents are also to be included in the output. This is optional.

`{-o|--output} {pretty|yaml|json}`
Prints the output in the specified format. The format options are `pretty` (human readable), `yaml` and `json`. The default is `pretty`. This is optional.

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

Where *globals* is one or more of the global options as described in Using Global Flags.

Examples

Example 4-51 Printing latest versions for all available modules

To print the latest versions for all modules in the Oracle Container Registry that can be installed for the environment named `myenvironment`, run the following:

```
olcnectl module version \
--environment-name myenvironment \
--all
```

Example 4-52 Printing latest version of a specific module in YAML format

To print the latest version of a specific module, such as `kubernetes`, available for the environment named `myenvironment`, in `yaml` format, run the following:

```
olcnectl module version \
--environment-name myenvironment \
--module kubernetes \
--output yaml
```

Example 4-53 Printing latest versions of a specific module and its subcomponents

To print the latest versions of a specific module, such as `kubernetes`, and its subcomponents, available for the environment named `myenvironment`, run the following:

```
olcnectl module version \
--environment-name myenvironment \
--module kubernetes \
--print-subcomponents \
```

Node Install-Agent

Installs the Platform Agent software packages on Kubernetes nodes.

Syntax

```
olcnectl node install-agent
[{-h|--help}]
{-n|--nodes} nodes
[{-R|--remote-command} remote-command]
[{-i|--ssh-identity-file} file_location]
[{-l|--ssh-login-name} username]
[--timeout minutes]
```

Where:

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

`{-n|--nodes}` *`nodes`*
A comma separated list of the hostnames or IP addresses of nodes.
Sets the node(s) on which to perform an action. Any nodes that are not the local node use the command indicated by `--remote-command` to connect to the host (by default, `ssh`). If a node address resolves to the local system, all commands are executed locally without using the remote command.

`{-R|--remote-command}` *`remote-command`*
Opens a connection to a remote host. The address of the host and the command are appended to the end of this command. For example:
`ssh -i ~/.ssh/myidfile -l myuser`
The default remote command is `ssh`.

`{-i|--ssh-identity-file}` *`file_location`*
The location of the SSH identity file. If no value is specified, the operating system defaults are used.

`{-l|--ssh-login-name}` *`username`*
The username to log in using SSH. The default is `opc`.

`--timeout` *`minutes`*
The number of minutes to set for command timeouts. The default is `40` minutes.

Examples

Example 4-54 Install the Platform Agent on nodes

This example installs the Platform Agent on the nodes listed.

```
olcnectl node install-agent \
--nodes control1.example.com,worker1.example.com,worker2.example.com
```

Node Install-Api-Server

Installs the Platform API Server software packages on the operator node.

Syntax

```
olcnectl node install-api-server
[{-h|--help}]
{-n|--nodes} nodes
[{-R|--remote-command} remote-command]
[{-i|--ssh-identity-file} file_location]
[{-l|--ssh-login-name} username]
[--timeout minutes]
```

Where:

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

`{-n|--nodes}` ***`nodes`***
A comma separated list of the hostnames or IP addresses of nodes.
Sets the node(s) on which to perform an action. Any nodes that are not the local node use the command indicated by `--remote-command` to connect to the host (by default, `ssh`). If a node address resolves to the local system, all commands are executed locally without using the remote command.

`{-R|--remote-command}` ***`remote-command`***
Opens a connection to a remote host. The address of the host and the command are appended to the end of this command. For example:
`ssh -i ~/.ssh/myidfile -l myuser`
The default remote command is `ssh`.

`{-i|--ssh-identity-file}` ***`file_location`***
The location of the SSH identity file. If no value is specified, the operating system defaults are used.

`{-l|--ssh-login-name}` ***`username`***
The username to log in using SSH. The default is `opc`.

`--timeout` ***`minutes`***
The number of minutes to set for command timeouts. The default is `40` minutes.

Examples

Example 4-55 Install the Platform API Server on a node

This example installs the Platform API Server on the node listed.

```
olcnectl install-api-server \
--nodes operator.example.com
```

Node Install-Certificates

Installs the CA Certificates and key for the Platform API Server and Platform Agent to the nodes, with the appropriate file ownership.

The certificates and key:

- Are copied to `/etc/olcne/certificates` on the nodes.
- Have ownership of the certificate files (using `chown`) set to `olcne:olcne`.
- Have permissions of the certificate files (using `chmod`) set to `0440`.

Syntax

```
olcnectl node install-certificates
[{-c|--certificate} path]
[{-C|--certificate-authority-chain} path]
[{-h|--help}]
[{-K|--key} path]
{-n|--nodes} nodes
[{-R|--remote-command} remote-command]
[{-i|--ssh-identity-file} file_location]
[{-l|--ssh-login-name} username]
[--timeout minutes]
```

Where :

`{-c|--certificate}` ***`path`***
The path to the `node.cert` certificate.

`{-C|--certificate-authority-chain}` ***`path`***
The path to the `ca.cert` Certificate Authority certificate.

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

`{-K|--key}` ***`path`***
The path to the `node.key` key.

`{-n|--nodes}` ***`nodes`***
A comma separated list of the hostnames or IP addresses of nodes.
Sets the node(s) on which to perform an action. Any nodes that are not the local node use the command indicated by `--remote-command` to connect to the host (by default, `ssh`). If a node address resolves to the local system, all commands are executed locally without using the remote command.

`{-R|--remote-command}` ***`remote-command`***
Opens a connection to a remote host. The address of the host and the command are appended to the end of this command. For example:
`ssh -i ~/.ssh/myidfile -l myuser`
The default remote command is `ssh`.

`{-i|--ssh-identity-file}` ***`file_location`***
The location of the SSH identity file. If no value is specified, the operating system defaults are used.

`{-l|--ssh-login-name}` ***`username`***
The username to log in using SSH. The default is `opc`.

`--timeout` ***`minutes`***
The number of minutes to set for command timeouts. The default is `40` minutes.

Examples

Example 4-56 Install certificates on nodes

This example installs the certificates on the node listed.

```
olcnectl node install-certificates \
--nodes
operator.example.com,control1.example.com,worker1.example.com,worker2.e
xample.com
```

Node Setup-Kubernetes

Sets up nodes to prepare for an installation of the Kubernetes module.

Configures hosts so they can be used as either Kubernetes control plane or worker nodes. Performs operations such as configuring firewall rules and opening network ports. Before changes are made to the hosts, a prompt is displayed to list the changes to be made and asks for confirmation.

Syntax

```
olcnectl node setup-kubernetes
{-a|--api-server} api-server-address
[--control-plane-ha-nodes nodes ]
[{-c|--control-plane-nodes} nodes]
[{-d|--debug}]
[{-h|--help}]
[{-m|--master-nodes} nodes] (Deprecated)
[{-n|--nodes} nodes]
[{-R|--remote-command} remote-command]
[{-r|--role} role]
[--selinux {permissive|enforcing}]
[{-i|--ssh-identity-file} file_location]
[{-l|--ssh-login-name} username]
[--timeout minutes]
[{-w|--worker-nodes} nodes]
[{-y|--yes}]
```

Where:

`{-a|--api-server}` ***`api-server-address`***
The hostname or IP address of the Platform API Server host.

`{-c|--control-plane-nodes}` ***`nodes`***
A comma separated list of the hostnames or IP addresses of the Kubernetes control plane nodes. For example, `control1.example.com,control2.example.com`.

`--control-plane-ha-nodes` ***`nodes`***
A comma separated list of the hostnames or IP addresses of the Kubernetes control plane nodes in a High Availability cluster. For example, `control1.example.com,control2.example.com,control3.example.com`.

`{-d|--debug}`
Enable debug logging.

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

`{-m|--master-nodes}` ***`nodes`***
A comma separated list of the hostnames or IP addresses of the Kubernetes control plane nodes. For example, `control1.example.com,control2.example.com,control3.example.com`.

Note:

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

`{-n|--nodes}` ***`nodes`***
A comma separated list of the hostnames or IP addresses of nodes.
Sets the node(s) on which to perform an action. Any nodes that are not the local node use the command indicated by `--remote-command` to connect to the host (by default, `ssh`). If a node address resolves to the local system, all commands are executed locally without using the remote command.

`{-R|--remote-command}` ***`remote-command`***
Opens a connection to a remote host. The address of the host and the command are appended to the end of this command. For example:
```
ssh -i ~/.ssh/myidfile -l myuser
```
The default remote command is `ssh`.

`{-r|--role}` ***`role`***
The role of a host. The roles are one of:

- `api-server`: The Platform API Server.
- `control-plane`: A Kubernetes control plane node for a cluster that has only one.
- `control-plane-ha`: A Kubernetes control plane node for a cluster that has more than one.
- `worker`: A Kubernetes worker node.

`--selinux {permissive|enforcing}`
Sets whether SELinux should be set to `enforcing` or `permissive`. The default is `permissive`.

`{-i|--ssh-identity-file}` ***`file_location`***
The location of the SSH identity file. If no value is specified, the operating system defaults are used.

`{-l|--ssh-login-name}` ***`username`***
The username to log in using SSH. The default is `opc`.

`--timeout` ***`minutes`***
The number of minutes to set for command timeouts. The default is `40` minutes.

`{-w|--worker-nodes}` ***`nodes`***
A comma separated list of the hostnames or IP addresses of the Kubernetes worker nodes. For example, `worker1.example.com,worker2.example.com`.

`{-y|--yes}`
Sets whether to assume the answer to a confirmation prompt is affirmative (`yes`).

Node Setup-Package-Repositories

Sets up the software package repositories on nodes.

Syntax

```
olcnectl node setup-package-repositories
[{-d|--debug}]
[{-h|--help}]
{-n|--nodes} nodes
[{-R|--remote-command} remote-command]
[{-i|--ssh-identity-file} file_location]
[{-l|--ssh-login-name} username]
[--timeout minutes]
```

Where:

`{-d|--debug}`
Enable debug logging.

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

`{-n|--nodes}` ***`nodes`***
A comma separated list of the hostnames or IP addresses of nodes.
Sets the node(s) on which to perform an action. Any nodes that are not the local node use the command indicated by `--remote-command` to connect to the host (by default, `ssh`). If a node address resolves to the local system, all commands are executed locally without using the remote command.

`{-R|--remote-command}` ***`remote-command`***
Opens a connection to a remote host. The address of the host and the command are appended to the end of this command. For example:
`ssh -i ~/.ssh/myidfile -l myuser`
The default remote command is `ssh`.

{-i|--ssh-identity-file} ***file_location***
The location of the SSH identity file. If no value is specified, the operating system defaults are used.

{-l|--ssh-login-name} ***username***
The username to log in using SSH. The default is `opc`.

--timeout ***minutes***
The number of minutes to set for command timeouts. The default is `40` minutes.

Examples

Example 4-57 Set up software repositories

This example sets up the software package repositories for the nodes listed.

```
olcnectl node setup-package-repositories \
--nodes control1.example.com,worker1.example.com,worker2.example.com
```

Node Setup-Platform

Installs the Oracle Cloud Native Environment platform (Platform API Server and Platform Agent) and starts the platform services.

Completely configures, installs, and starts the Oracle Cloud Native Environment platform components on a set of hosts.

1. Configures the yum software package repositories.
2. Configures networking ports.
3. Installs the Platform API Server and Platform Agent.
4. Generates and installs CA Certificates.
5. Starts the platform services (`olcne-api-server.service` and `olcne-agent.service`).

Syntax

```
olcnectl node setup-platform
{-a|--api-server} api-server-address
[--byo-ca-cert certificate-path]
[--byo-ca-key key-path]
[--cert-dir certificate-directory]
[--cert-request-common-name common_name]
[--cert-request-country country]
[--cert-request-locality locality]
[--cert-request-organization organization]
[--cert-request-organization-unit organization-unit]
[--cert-request-state state]
[--control-plane-ha-nodes nodes ]
[--control-plane-nodes nodes]
[{-d|--debug}]
[{-h|--help}]
[--http-proxy proxy-server]
```

```
[--https-proxy proxy-server]
[--no-proxy no_proxy]
[{-n|--nodes} nodes]
[--one-cert]
[{-R|--remote-command} remote-command]
[{-r|--role} role]
[--selinux {permissive|enforcing}]
[{-i|--ssh-identity-file} file_location]
[{-l|--ssh-login-name} username]
[--timeout minutes]
[{-w|--worker-nodes} nodes]
[{-y|--yes}]
```

Where:

`{-a|--api-server}` ***`api-server-address`***
The hostname or IP address of the Platform API Server host.

`--byo-ca-cert` ***`certificate-path`***
The path to an existing public CA Certificate.

`--byo-ca-key` ***`key-path`***
The path to an existing private key.

`--cert-dir` ***`certificate-directory`***
The directory to read or write key material generated by this utility. The default is `<CURRENT_DIR>/certificates`.

`--cert-request-common-name` ***`common_name`***
The Certificate Common Name suffix. The default is `example.com`.

`--cert-request-country` ***`country`***
The two letter country code where your company is located, for example, `US` for the United States, `GB` for the United Kingdom and `CN` for China. The default is `US`.

`--cert-request-locality` ***`locality`***
The name of the city where your company is located. The default is `Redwood City`.

`--cert-request-organization` ***`organization`***
The name of the your company. The default is `OLCNE`.

`--cert-request-organization-unit` ***`organization-unit`***
The name of the department within your company. The default is `OLCNE`.

`--cert-request-state` ***`state`***
The name of the state or province where your company is located. The default is `California`.

`--control-plane-ha-nodes` ***`nodes`***
A comma separated list of the hostnames or IP addresses of the Kubernetes control plane nodes in a High Availability cluster. For example, `control1.example.com,control2.example.com,control3.example.com`.

{-c|--control-plane-nodes} ***nodes***
A comma separated list of the hostnames or IP addresses of the Kubernetes control plane nodes. For example, `control1.example.com,control2.example.com`.

{-d|--debug}
Enable debug logging.

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

--http-proxy ***proxy-server***
The location of the HTTP proxy server if required.

--https-proxy ***proxy-server***
The location of the HTTPS proxy server if required.

--no-proxy ***no_proxy***
The list of hosts for which to exclude from the proxy server settings.

{-n|--nodes} ***nodes***
A comma separated list of the hostnames or IP addresses of nodes.
Sets the node(s) on which to perform an action. Any nodes that are not the local node use the command indicated by `--remote-command` to connect to the host (by default, `ssh`). If a node address resolves to the local system, all commands are executed locally without using the remote command.

--one-cert
Sets whether to generate a single certificate that can be used to authenticate all the hosts. By default this option is not set.

{-R|--remote-command} ***remote-command***
Opens a connection to a remote host. The address of the host and the command are appended to the end of this command. For example:
`ssh -i ~/.ssh/myidfile -l myuser`
The default remote command is `ssh`.

{-r|--role} ***role***
The role of a host. The roles are one of:

- `api-server`: The Platform API Server.
- `control-plane`: A Kubernetes control plane node for a cluster that has only one.
- `control-plane-ha`: A Kubernetes control plane node for a cluster that has more than one.
- `worker`: A Kubernetes worker node.

--selinux {permissive|enforcing}
Sets whether SELinux should be set to `enforcing` or `permissive`. The default is `permissive`.

{-i|--ssh-identity-file} ***file_location***
The location of the SSH identity file. If no value is specified, the operating system defaults are used.

{-l|--ssh-login-name} ***username***
The username to log in using SSH. The default is `opc`.

`--timeout` ***`minutes`***
The number of minutes to set for command timeouts. The default is `40` minutes.

`{-w|--worker-nodes}` ***`nodes`***
A comma separated list of the hostnames or IP addresses of the Kubernetes worker nodes. For example, `worker1.example.com,worker2.example.com`.

`{-y|--yes}`
Sets whether to assume the answer to a confirmation prompt is affirmative (`yes`).

Examples

Example 4-58 Install the platform using default options

This example installs the Oracle Cloud Native Environment platform using default options.

```
olcnectl node setup-platform \
--api-server operator.example.com \
--control-plane-nodes
control1.example.com,control2.example.com,control3.example.com \
--worker-nodes
worker1.example.com,worker2.example.com,worker2.example.com
```

Example 4-59 Install the platform for an HA deploy using default options

This example installs the Oracle Cloud Native Environment platform using default options.

```
olcnectl node setup-platform \
--api-server operator.example.com \
--control-plane-ha-nodes
control1.example.com,control2.example.com,control3.example.com \
--worker-nodes
worker1.example.com,worker2.example.com,worker2.example.com
```

Example 4-60 Install the platform for an HA deploy using default options

This example installs the Oracle Cloud Native Environment platform using default options.

```
olcnectl node setup-platform \
--api-server operator.example.com \
--control-plane-ha-nodes
control1.example.com,control2.example.com,control3.example.com \
--worker-nodes
worker1.example.com,worker2.example.com,worker2.example.com
```

Example 4-61 Install the Platform API Server

This example installs the Platform API Server and accepts all prompts.

```
olcnectl node setup-platform \
--nodes operator.example.com \
--role api-server \
--yes
```

Example 4-62 Install the Platform Agent on HA control plane nodes

This example installs the Platform Agent on Kubernetes control plane nodes for an HA install. This accepts all prompts.

```
olcnectl node setup-platform \
--nodes control1.example.com,control2.example.com,control3.example.com \
--role control-plane-ha \
--yes
```

Example 4-63 Install the Platform Agent on worker nodes with a proxy server and SSH login information

This example installs the Platform Agent on Kubernetes worker nodes. This uses proxy server information, provides SSH login information, and accepts all prompts.

```
olcnectl node setup-platform \
--nodes worker1.example.com,worker2.example.com,worker2.example.com \
--role worker \
--ssh-identity-file ~/.ssh/id_rsa \
--ssh-login-name oracle \
--http-proxy "http://www-proxy.example.com:80" \
--https-proxy "https://www-proxy.example.com:80" \
--no-proxy ".example.com" \
--yes
```

Node Start-Platform

Starts the Platform API Server service on the operator node and the Platform Agent service on Kubernetes nodes.

Syntax

```
olcnectl node start-platform
[{-h|--help}]
{-n|--nodes} nodes
[{-R|--remote-command} remote-command]
[{-i|--ssh-identity-file} file_location]
[{-l|--ssh-login-name} username]
[--timeout minutes]
[{-y|--yes}]
```

Where:

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

`{-n|--nodes}` ***`nodes`***
A comma separated list of the hostnames or IP addresses of nodes.
Sets the node(s) on which to perform an action. Any nodes that are not the local node use the command indicated by `--remote-command` to connect to the host (by default, `ssh`). If a node address resolves to the local system, all commands are executed locally without using the remote command.

`{-R|--remote-command}` ***`remote-command`***
Opens a connection to a remote host. The address of the host and the command are appended to the end of this command. For example:
`ssh -i ~/.ssh/myidfile -l myuser`
The default remote command is `ssh`.

`{-i|--ssh-identity-file}` ***`file_location`***
The location of the SSH identity file. If no value is specified, the operating system defaults are used.

`{-l|--ssh-login-name}` ***`username`***
The username to log in using SSH. The default is `opc`.

`--timeout` ***`minutes`***
The number of minutes to set for command timeouts. The default is `40` minutes.

`{-y|--yes}`
Sets whether to assume the answer to a confirmation prompt is affirmative (`yes`).

Examples

Example 4-64 Starting Platform services

This example starts the Platform services on the nodes listed.

```
olcnectl node start-platform \
--ssh-identity-file ~/.ssh/id_rsa \
--ssh-login-name oracle \
--nodes control01.example.com,worker01.example.com,worker02.example.com
```

Provision

Sets up the nodes and installs the Oracle Cloud Native Environment platform (Platform API Server and Platform Agents), including creating and installing the Kubernetes module.

This command configures all nodes, creates and distributes certificates and keys, installs the Platform API Server and Platform Agents, starts the required system services, and creates and installs an instance of the Kubernetes module. This provides a quick installation of Oracle Cloud Native Environment with a Kubernetes cluster.

When you run this command, a prompt is displayed that lists the changes to be made to the hosts and asks for confirmation. To avoid this prompt, use the `--yes` option.

More complex deployments can be made by using a configuration file with the `--config-file` option.

Syntax

```
olcnectl provision
{-a|--api-server} api-server-address
[--byo-ca-cert certificate-path]
[--byo-ca-key key-path]
[--cert-dir certificate-directory]
[--cert-request-common-name common_name]
[--cert-request-country country]
[--cert-request-locality locality]
[--cert-request-organization organization]
[--cert-request-organization-unit organization-unit]
[--cert-request-state state]
[--config-file config-file-path]
[--container-registry registry]
{-c|--control-plane-nodes} nodes
[{-d|--debug}]
{-E|--environment-name} environment_name
[{-h|--help}]
[--http-proxy proxy-server]
[--https-proxy proxy-server]
[--load-balancer load-balancer]
{-m|--master-nodes} nodes (Deprecated)
{-N|--name} name
[--no-proxy no_proxy]
[{-n|--nodes} nodes]
[--one-cert]
[{-R|--remote-command} remote-command]
[--restrict-service-externalip-cidrs allowed_cidrs]
[--selinux {permissive|enforcing}]
[{-i|--ssh-identity-file} file_location]
[{-l|--ssh-login-name} username]
[--timeout minutes]
[--virtual-ip IP_address]
{-w|--worker-nodes} nodes
[{-y|--yes}]
```

Where:

`{-a|--api-server}` ***`api-server-address`***
The hostname or IP address of the Platform API Server host.

`--byo-ca-cert` ***`certificate-path`***
The path to an existing public CA Certificate.

`--byo-ca-key` ***`key-path`***
The path to an existing private key.

`--cert-dir` ***`certificate-directory`***
The directory to read or write key material generated by this utility. The default is `<CURRENT_DIR>/certificates`.

`--cert-request-common-name` ***`common_name`***
The Certificate Common Name suffix. The default is `example.com`.

`--cert-request-country` ***`country`***
The two letter country code where your company is located, for example, `US` for the United States, `GB` for the United Kingdom and `CN` for China. The default is `US`.

`--cert-request-locality` ***`locality`***
The name of the city where your company is located. The default is `Redwood City`.

`--cert-request-organization` ***`organization`***
The name of the your company. The default is `OLCNE`.

`--cert-request-organization-unit` ***`organization-unit`***
The name of the department within your company. The default is `OLCNE`.

`--cert-request-state` ***`state`***
The name of the state or province where your company is located. The default is `California`.

`--config-file` ***`config-file-path`***
The path and location of an Oracle Cloud Native Environment configuration file.

`--container-registry` ***`registry`***
The container registry from which to pull the Kubernetes container images. The default is `container-registry.oracle.com/olcne`.

`{-c|--control-plane-nodes}` ***`nodes`***
A comma separated list of the hostnames or IP addresses of the Kubernetes control plane nodes. For example, `control1.example.com,control2.example.com`.

`{-d|--debug}`
Enable debug logging.

`{-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.

`--http-proxy` ***`proxy-server`***
The location of the HTTP proxy server if required.

`--https-proxy` ***`proxy-server`***
The location of the HTTPS proxy server if required.

`--load-balancer` ***`load-balancer`***
The location of the external load balancer if required.

`{-m|--master-nodes}` ***`nodes`***
A comma separated list of the hostnames or IP addresses of the Kubernetes control plane nodes. For example, `control1.example.com,control2.example.com,control3.example.com`.

Note:

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

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

`--no-proxy` ***`no_proxy`***
The list of hosts for which to exclude from the proxy server settings.

`{-n|--nodes}` ***`nodes`***
A comma separated list of the hostnames or IP addresses of nodes.
Sets the node(s) on which to perform an action. Any nodes that are not the local node use the command indicated by `--remote-command` to connect to the host (by default, `ssh`). If a node address resolves to the local system, all commands are executed locally without using the remote command.

`--one-cert`
Sets whether to generate a single certificate that can be used to authenticate all the hosts. By default this option is not set.

`{-R|--remote-command}` ***`remote-command`***
Opens a connection to a remote host. The address of the host and the command are appended to the end of this command. For example:
`ssh -i ~/.ssh/myidfile -l myuser`
The default remote command is `ssh`.

`--restrict-service-externalip-cidrs` ***`allowed_cidrs`***
Enter one or more comma separated CIDR blocks if you want to allow only IP addresses from the specified CIDR blocks. For example, `192.0.2.0/24,198.51.100.0/24`.

`--selinux {permissive|enforcing}`
Sets whether SELinux should be set to `enforcing` or `permissive`. The default is `permissive`.

`{-i|--ssh-identity-file}` ***`file_location`***
The location of the SSH identity file. If no value is specified, the operating system defaults are used.

`{-l|--ssh-login-name}` ***`username`***
The username to log in using SSH. The default is `opc`.

`--timeout` ***`minutes`***
The number of minutes to set for command timeouts. The default is `40` minutes.

`--virtual-ip` ***`IP_address`***
The virtual IP address to use for the internal load balancer.

`{-w|--worker-nodes}` ***`nodes`***
A comma separated list of the hostnames or IP addresses of the Kubernetes worker nodes. For example, `worker1.example.com,worker2.example.com`.

`{-y|--yes}`
Sets whether to assume the answer to a confirmation prompt is affirmative (`yes`).

Examples

Example 4-65 Quick install

To perform a quick install:

```
olcnectl provision \
--api-server operator.example.com \
--control-plane-nodes control1.example.com \
--worker-nodes
worker1.example.com,worker2.example.com,worker3.example.com \
--environment-name myenvironment \
--name mycluster
```

Example 4-66 Quick install with SSH log in information

To perform a quick install using SSH log in information and accepting all prompts:

```
olcnectl provision \
--api-server operator.example.com \
--control-plane-nodes control1.example.com \
--worker-nodes
worker1.example.com,worker2.example.com,worker3.example.com \
--environment-name myenvironment \
--name mycluster \
--ssh-identity-file ~/.ssh/id_rsa \
--ssh-login-name oracle \
--yes
```

Example 4-67 Quick HA install with an external load balancer

To perform a quick HA install using an external load balancer and accepting all prompts:

```
olcnectl provision \
--api-server operator.example.com \
--control-plane-nodes
control1.example.com,control2.example.com,control3.example.com \
--worker-nodes
worker1.example.com,worker2.example.com,worker3.example.com \
--environment-name myenvironment \
--name mycluster \
--load-balancer lb.example.com:6443 \
--yes
```

Example 4-68 Quick install using a proxy server

To perform a quick install using SSH log in information and a proxy server, and accepting all prompts:

```
olcnectl provision  \
--api-server operator.example.com \
--control-plane-nodes control1.example.com \
--worker-nodes worker1.example.com,worker2.example.com,worker3.example.com \
--environment-name myenvironment \
--name mycluster \
--ssh-identity-file ~/.ssh/id_rsa \
--ssh-login-name oracle \
--http-proxy "http://www-proxy.example.com:80" \
--https-proxy "https://www-proxy.example.com:80" \
--no-proxy ".example.com" \
--yes
```

Example 4-69 Quick install with externalIPs service

To perform a quick install that includes the `externalIPs` Kubernetes service, specify the CIDR block(s) that should be allowed access to the service. This deploys the `externalip-validation-webhook-service` Kubernetes service:

```
olcnectl provision \
--api-server operator.example.com \
--control-plane-nodes control1.example.com \
--worker-nodes worker1.example.com,worker2.example.com,worker3.example.com \
--environment-name myenvironment \
--name mycluster \
--restrict-service-externalip-cidrs 192.0.2.0/24,198.51.100.0/24
```

Example 4-70 Quick install using a configuration file

To perform a quick install using a configuration file and SSH log in information:

```
olcnectl provision \
--config-file myenvironment.yaml \
--ssh-identity-file ~/.ssh/id_rsa \
--ssh-login-name oracle \
--yes
```

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:

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

Examples

Example 4-71 Creating a sample configuration template

To create a sample configuration template:

```
olcnectl template
```