Oracle® Cloud Native Environment

Platform Command-Line Interface for Release 1.3

F50647-02
April 2022

Oracle Legal Notices

Table of 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.

Document generated on: 2022-04-29 (revision: 1232)

Audience

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

Related Documents

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

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

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
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.

Chapter 1 Introduction to the Platform CLI

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

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

```
olcnectl command {{-h|--help}|command_options}
```

The full syntax and options for each command is provided in Chapter 2, *Platform CLI Commands*.

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

1.1 Getting Syntax Help

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

```
olcnectl --help

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

Usage:
  olcnectl [command]

Available Commands:
  environment Environment operations
  help        Help about any command
  module      Modules that can be modified in an environment

Flags:
  -a, --api-server string              Platform API Server to talk to. If this is not specified ...
  -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 [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` and `olcnectl module`.

The `Flags` section lists the available command options you can use. In this case, the options shown are also the global flags which are used by all subcommands. For more information on the global flags, see Section 1.3, “Using Global Flags”.

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
  restore     restore a module
  uninstall   Uninstall a module
  update      Update a module
  validate    Validate that an module can be installed

Flags:
  -h, --help   help for module

Global Flags:
  -a, --api-server string               Platform API Server to talk to. If this is not specified ...
  -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 `Global Flags` section lists the global flags which can be used by all subcommands. This is the same list of options as shown earlier with the `olcnectl --help` command and listed under the `Flags` section.

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

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

```
olcnectl module property get --help

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

Usage:
  olcnectl module property get [flags]

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

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

```
olcnectl module create --help

Create a module in a environment

Usage:
 olcnectl module create [flags]

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

To see the options for creating each module you must use the `--module` option and provide the module type. The module types are listed in Section 2.4, “Module Create”. For example, to get help on creating a Kubernetes module you specify the module type as `kubernetes`:

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

Create a module in a environment

Usage:
  olcnectl module create [flags]

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

```
  -h, --help                                help for create
  -x, --kube-proxy-mode string              Routing mode for the Kubernetes proxy (default     ...
  -v, --kube-version string                 Kubernetes version (default "1.17.4")
...
```

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

```
olcnectl module update --help

Update a module

Usage:
  olcnectl module update [flags]

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

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

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

Update a module

Usage:
  olcnectl module update [flags]

Flags:
  -E, --environment-name string   Name of the environment
  -F, --force                     Update without prompting
  -g, --generate-scripts          Generate a script for each node that takes all suggested actions
  -h, --help                      help for update
  -v, --kube-version string       Kubernetes version (default "1.20.11")
  -m, --master-nodes string       A comma separated list of master nodes
  -N, --name strings              Modules to update
  -w, --worker-nodes string       A comma separated list of worker nodes
...
```

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

1.2 Setting the Platform API Server

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

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

When you create an environment with the `olcnectl environment create` command you can optionally include the `--update-config` option. This option writes information about the environment to

a local configuration file at `$HOME/.olcne/olcne.conf`, and this configuration is used for future calls to the Platform API Server. If you use this option, you do not need to specify the Platform API Server in future `olcnectl` commands.

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

```
olcnectl environment create \
--api-server 127.0.0.1:8091 \
--environment-name myenvironment \
--secret-manager-type vault \
--vault-token s.3QKNuRoTqLbjXaGBOmO6Psjh \
--vault-address https://192.0.2.20:8200 \
--update-config
```

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

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

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

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

1.3 Using Global Flags

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

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

```
[{-a|--api-server} api_server_address:8091]
[--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 Section 1.2, “Setting the Platform API Server”.
	This option maps to the `$OLCNE_API_SERVER_BIN` environment variable. If this environment variable is set it takes precedence over and overrides the Platform CLI setting.
`--secret-manager-type {file|vault}`	The secrets manager type. The options are `file` or `vault`. Use `file` for certificates saved on the nodes and use `vault` for certificates managed by Vault.
`--update-config`	Writes the global arguments for an environment to a local configuration file which is used for future calls to the Platform API Server. If this option has not been used previously, global arguments must be specified for every Platform API Server call.
	The global arguments configuration information is saved to `$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 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 `$OLCNE_SM_CERT_PATH` environment variable. If this environment variable is set it takes precedence over and overrides the Platform CLI setting.
`--olcne-node-key-path` *`node_key_path`*	The path to a predefined certificate, or the a destination if using a secrets manager to download the certificate. The default is `/etc/olcne/certificates/node.cert`, or gathered from the local configuration if the `--update-config` option is used.
	This option maps to the `$OLCNE_SM_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_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_RC4_128_SHA
- TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305
- TLS_ECDHE_RSA_WITH_RC4_128_SHA
- TLS_RSA_WITH_3DES_EDE_CBC_SHA
- TLS_RSA_WITH_AES_128_CBC_SHA
- TLS_RSA_WITH_AES_128_CBC_SHA256
- TLS_RSA_WITH_AES_128_GCM_SHA256
- TLS_RSA_WITH_AES_256_CBC_SHA
- TLS_RSA_WITH_AES_256_GCM_SHA384
- TLS_RSA_WITH_RC4_128_SHA

For example:

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

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.

Chapter 2 Platform CLI Commands

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

2.1 Environment Create

Creates an empty environment.

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

Syntax

```
olcnectl environment create
{-E|--environment-name} environment_name
[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.

Where *`globals`* is:

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

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

For more information on setting the Platform API Server see Section 1.2, “Setting the Platform API Server”.

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

`--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 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_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 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_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_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_RC4_128_SHA`
- `TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA`
- `TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA`
- `TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256`
- `TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256`
- `TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA`
- `TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384`
- `TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305`
- `TLS_ECDHE_RSA_WITH_RC4_128_SHA`
- `TLS_RSA_WITH_3DES_EDE_CBC_SHA`
- `TLS_RSA_WITH_AES_128_CBC_SHA`
- `TLS_RSA_WITH_AES_128_CBC_SHA256`
- `TLS_RSA_WITH_AES_128_GCM_SHA256`
- `TLS_RSA_WITH_AES_256_CBC_SHA`
- `TLS_RSA_WITH_AES_256_GCM_SHA384`
- `TLS_RSA_WITH_RC4_128_SHA`

For example:

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

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 2.1 Creating an environment using Vault

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

```
olcnectl environment create \
--api-server 127.0.0.1:8091 \
--environment-name myenvironment \
--secret-manager-type vault \
--vault-token s.3QKNuRoTqLbjXaGBOmO6Psjh \
--vault-address https://192.0.2.20:8200 \
--update-config
```

Example 2.2 Creating an environment using certificates

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

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

2.2 Environment Delete

Deletes an existing environment.

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

Syntax

```
olcnectl environment delete
{-E|--environment-name} environment_name
[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.

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

Examples

Example 2.3 Deleting an environment

To delete an environment named `myenvironment`:

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

2.3 Module Backup

Backs up a module in an environment.

Syntax

```
olcnectl module backup
{-E|--environment-name} environment_name
{-N|--name} name
[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.

Where *`globals`* is one or more of the global options as described in Section 1.3, “Using Global Flags”.

Examples

Example 2.4 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
```

2.4 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
[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` • `helm` • `prometheus` • `istio` • `operator-lifecycle-manager`
`{-N|--name}` *`name`*	The module name. The value of *`name`* is the name to use to identify a module in an environment.

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 `--master-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 `--master-nodes` argument instead.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

The version of Kubernetes to install. The default is the latest version. For information on the latest version number, see *Release Notes*.

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

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

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

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

- `TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA`
- `TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256`
- `TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256`
- `TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA`
- `TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384`
- `TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305`
- `TLS_ECDHE_ECDSA_WITH_RC4_128_SHA`
- `TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA`
- `TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA`
- `TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256`
- `TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256`
- `TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA`
- `TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384`
- `TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305`
- `TLS_ECDHE_RSA_WITH_RC4_128_SHA`
- `TLS_RSA_WITH_3DES_EDE_CBC_SHA`
- `TLS_RSA_WITH_AES_128_CBC_SHA`
- `TLS_RSA_WITH_AES_128_CBC_SHA256`
- `TLS_RSA_WITH_AES_128_GCM_SHA256`
- `TLS_RSA_WITH_AES_256_CBC_SHA`
- `TLS_RSA_WITH_AES_256_GCM_SHA384`
- `TLS_RSA_WITH_RC4_128_SHA`

For example:

```
--kube-tls-cipher-suites
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256,TLS_ECDHE_RSA_WITH_AES_25
```

`--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.

`{-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.

`{-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}` *`network_fabric`*

The network fabric for the Kubernetes cluster. The default is `flannel`.

`{-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/configs/certificates/restrict_external_ip/production/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/configs/certificates/restrict_external_ip/production/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/configs/certificates/restrict_external_ip/production/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 `helm` module:**

`--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 `prometheus` module:**

`--prometheus-alerting-rules` *`path`*	The path to a configuration file for Prometheus alerts.
`--prometheus-helm-module` *`helm_module`*	The name of the `helm` module that Prometheus should be associated with.
`--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`. **Important** Oracle Cloud Native Environment does not yet have any modules that provide support for Kubernetes `PersistentVolumeClaim`s, so persistent storage must be manually set up.
`--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.

`--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*.

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

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

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

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

  The default is `true`.

  

  Important

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

`--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*.

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

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

`--olm-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`.

`--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*.

Where *`globals`* is one or more of the global options as described in Section 1.3, “Using Global Flags”.

Examples

Example 2.5 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.

Important

In Release 1.2.0, the options to set the options for the `externalip-validation-webhook-service` Kubernetes service are not required and cannot be used. These options are only available and required in Release 1.2.2 or later.

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

Example 2.6 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.

Important

In Release 1.2.0, the options to set the options for the `externalip-validation-webhook-service` Kubernetes service are not required and cannot be used. These options are only available and required in Release 1.2.2 or later.

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

Example 2.7 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 \
--master-nodes control1.example.com:8090,control2.example.com:8090,control3.example.com:8090 \
--worker-nodes worker1.example.com:8090,worker2.example.com:8090,worker3.example.com:8090 \
--selinux enforcing \
--restrict-service-externalip-ca-cert=/etc/olcne/configs/certificates/restrict_external_ip/production/ca.cert
--restrict-service-externalip-tls-cert=/etc/olcne/configs/certificates/restrict_external_ip/production/node.ce
--restrict-service-externalip-tls-key=/etc/olcne/configs/certificates/restrict_external_ip/production/node.key
--restrict-service-externalip-cidrs=192.0.2.0/24,198.51.100.0/24
```

Example 2.8 Creating a module for a cluster to allow access to all service IP addresses

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

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

Example 2.9 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.

Important

In Release 1.2.0, the options to set the options for the `externalip-validation-webhook-service` Kubernetes service are not required and cannot be used. These options are only available and required in Release 1.2.2 or later.

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

Example 2.10 Creating a module for a service mesh

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

You can provide all the required module options to deploy a service mesh (Istio module) in a single command. As the Istio module requires Kubernetes and Helm, you must also provide the options for those modules.

The `--helm-kubernetes-module` option sets the name of the Kubernetes module to use. If you have an existing Kubernetes module installed, you can specify the name of the module using this option. If no Kubernetes module is created or installed with the name you provide, a new Kubernetes module is configured which allows you to install Kubernetes at the same time as a service mesh.

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

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 \
--helm-kubernetes-module mycluster \
--istio-helm-module myhelm
```

Example 2.11 Creating a module for Operator Lifecycle Manager

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

You can provide all the required module options to deploy Operator Lifecycle Manager in a single command. As the Operator Lifecycle Manager module requires Kubernetes and Helm, you must also provide the options for those modules.

The `--helm-kubernetes-module` option sets the name of the Kubernetes module to use. If you have an existing Kubernetes module installed, you can specify the name of the module using this option. If no Kubernetes module is created or installed with the name you provide, a new Kubernetes module is configured which allows you to install Kubernetes at the same time as Operator Lifecycle Manager.

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

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 \
--helm-kubernetes-module mycluster \
--olm-helm-module myhelm
```

2.5 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}]
[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`.

Where *`globals`* is one or more of the global options as described in Section 1.3, “Using Global Flags”.

Examples

Example 2.12 Installing a module

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

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

2.6 Module Instances

Lists the installed modules in an environment.

Syntax

```
olcnectl module instances
{-E|--environment-name} environment_name
[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.

Where *`globals`* is one or more of the global options as described in Section 1.3, “Using Global Flags”.

Examples

Example 2.13 Listing the deployed modules in an environment

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

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

2.7 Module List

Lists the available modules for an environment.

Syntax

```
olcnectl module list
{-E|--environment-name} environment_name
[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.

Where *`globals`* is one or more of the global options as described in Section 1.3, “Using Global Flags”.

Examples

Example 2.14 Listing available modules in an environment

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

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

2.8 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
[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.

Where *`globals`* is one or more of the global options as described in Section 1.3, “Using Global Flags”.

Examples

Example 2.15 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
```

2.9 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
[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.

Where *`globals`* is one or more of the global options as described in Section 1.3, “Using Global Flags”.

Examples

Example 2.16 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
```

2.10 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}]
[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.

Where *`globals`* is one or more of the global options as described in Section 1.3, “Using Global Flags”.

Examples

Example 2.17 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
```

2.11 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}]
[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.

Where *`globals`* is one or more of the global options as described in Section 1.3, “Using Global Flags”.

Examples

Example 2.18 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.

2.12 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
[{-r|--container-registry} container_registry]
[{-k|--kube-version} version]
[{-m|--master-nodes} nodes ...]
[{-w|--worker-nodes} nodes ...]
[--nginx-image container_location]
[--istio-version version]
[--olm-version version]
--restrict-service-externalip {true|false}
--restrict-service-externalip-ca-cert path
--restrict-service-externalip-tls-cert path
--restrict-service-externalip-tls-key path
--restrict-service-externalip-cidrs allowed_cidrs
[--selinux {enforcing|permissive}]
[{-g|--generate-scripts}]
[{-F|--force}]
[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.

`{-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`.
`{-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`.
`--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`
`--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*.
`--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*.
`--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/configs/certificates/restrict_external_ip/production/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/configs/certificates/restrict_external_ip/production/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/configs/certificates/restrict_external_ip/production/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.
`{-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.

Where *`globals`* is one or more of the global options as described in Section 1.3, “Using Global Flags”.

Examples

Example 2.19 Scaling a cluster

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

```
olcnectl module update \
--environment-name myenvironment \
--name mycluster \
--master-nodes control1.example.com:8090,control2.example.com:8090,control3.example.com:8090 \
--worker-nodes worker1.example.com:8090,worker2.example.com:8090,worker3.example.com:8090
```

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

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

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

Example 2.20 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 2.21 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 2.22 Upgrading the Kubernetes release

To upgrade a Kubernetes module named `mycluster` in an environment named `myenvironment` to Kubernetes Release 1.20.11, enter:

```
olcnectl module update \
--environment-name myenvironment \
--name mycluster \
--kube-version 1.20.11
```

The `--kube-version` option specifies the release to which you want to upgrade. This example uses release number 1.20.11.

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 1.20.11.

Example 2.23 Upgrading using a different container registry

To upgrade a Kubernetes module named `mycluster` in an environment named `myenvironment` to Kubernetes Release 1.20.11 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 1.20.11
```

The `--kube-version` option specifies the release to which you want to upgrade. This example uses release number 1.20.11. 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 1.20.11.

Example 2.24 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 2.25 Modifing 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
```

2.13 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}]
[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`.

Where *`globals`* is one or more of the global options as described in Section 1.3, “Using Global Flags”.

Examples

Example 2.26 Validating a module

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

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