Oracle Cloud Native Environment

CLI for Release 2

F96194-14
May 2025

Oracle Cloud Native Environment CLI for Release 2,

F96194-14

Contents

Preface

This document contains information about the Oracle Cloud Native Environment (Oracle CNE) Command Line Interface (CLI). This is the `ocne` command. This document includes information on using the CLI, and on using configuration files to customize the environment. References to the contents of configuration files is also included. A command reference of CLI commands is included in this document.

Documentation License

Conventions

The following text conventions are used in this document:

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

Documentation Accessibility

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

Access to Oracle Support for Accessibility

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit https://www.oracle.com/corporate/accessibility/learning-support.html#support-tab.

Diversity and Inclusion

Oracle is fully committed to diversity and inclusion. Oracle respects and values having a diverse workforce that increases thought leadership and innovation. As part of our initiative to build a more inclusive culture that positively impacts our employees, customers, and partners, we are working to remove insensitive terms from our products and documentation. We are also mindful of the necessity to maintain compatibility with our customers' existing technologies and the need to ensure continuity of service as Oracle's offerings and industry standards evolve. Because of these technical constraints, our effort to remove insensitive terms is ongoing and will take time and external cooperation.

1

Installing the CLI

Install the Oracle Cloud Native Environment (Oracle CNE) Command Line Interface (CLI) on an Oracle Linux host, using the Oracle Linux Yum Server, or the Unbreakable Linux Network (ULN).

The Oracle CNE CLI is the command line tool to create and manage Kubernetes clusters in Oracle CNE. The CLI (`ocne` command) includes a help system to show all command options, and a set of configuration files at various levels to configure the environment and clusters.

1. Set up the Oracle Linux Yum Server Repository.

   If the system uses the Oracle Linux Yum Server, set it up to install the CLI:

   Oracle Linux 9:

   ```
   sudo dnf install -y oracle-ocne-release-el9
   sudo dnf config-manager --enable ol9_ocne
   ```

   Oracle Linux 8:

   ```
   sudo dnf install -y oracle-ocne-release-el8
   sudo dnf config-manager --enable ol8_ocne
   ```

2. Set up ULN.

   If the system is registered to use ULN, use the ULN web interface to subscribe the system to the appropriate channel.

   For Oracle Linux 9, subscribe to `ol9_x86_64_ocne` or `ol9_aarch64_ocne`.

   For Oracle Linux 8, subscribe to `ol8_x86_64_ocne` or `ol8_aarch64_ocne`.

3. Install the CLI.

   ```
   sudo dnf install -y ocne
   ```

2
Using the CLI

Introduces the Oracle CNE CLI (`ocne` command), which is used to create and manage Kubernetes clusters.

This chapter contains information on using the CLI.

Getting Syntax Help

Learn how to get help with CLI syntax.

All `ocne` commands include the option to display help on the syntax. If you enter the `ocne` command without any options, the help is displayed:

```
ocne
```

The output is similar to:

```
The ocne tool manages an ocne environment

Usage:
  ocne [command]

Available Commands:
  application Manage ocne applications
  catalog     Manage ocne catalogs
  cluster     Manage ocne clusters
  completion  Generate the autocompletion script for the specified shell
  help        Help about any command
  image       Manage ocne images
  info        Display CLI version information and environment variables
  node        Manage ocne nodes

Flags:
  -h, --help               help for ocne
  -l, --log-level string   Sets the log level.  Valid values are "error",
"info", "debug", and "trace". (default "info")

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

Use the `--help` or `-h` option with each command to display the help about a command. For example:

```
ocne cluster --help
```

Or, use the `ocne help` command. For example:

```
ocne help cluster
```

The output is similar to:

```
Manage the lifecycle of ocne clusters and application deployment.

Usage:
  ocne cluster [flags]
  ocne cluster [command]

Examples:

ocne cluster <subcommand>


Available Commands:
  analyze     Analyze the cluster and report problems
  backup      Backup the etcd database
  console     Launch a console on a node
  delete      Destroy a cluster
  dump        Dump the cluster
  info        Get cluster information
  join        Join a node to a cluster, or generate the materials required to
do so
  list        List clusters
  show        Show cluster configuration
  stage       Stage a cluster update to a specified k8s version
  start       Start an OCNE cluster
  template    Outputs a cluster configuration template
  update      Updates the version of a cluster

Flags:
  -h, --help                help for cluster
  -k, --kubeconfig string   the kubeconfig filepath

Global Flags:
  -l, --log-level string   Sets the log level.  Valid values are "error",
"info", "debug", and "trace". (default "info")

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

Prefix Matching

Learn how to use prefix matching in the CLI.

You can use prefix matching for any unambiguous prefix for an `ocne` command. For example, you can use the following instead of the full `ocne cluster start` command:

```
ocne cl s
ocne clu star
```

Instead of typing the full `ocne application list` command, you could use:

```
ocne ap l
```

Another example would be for the `ocne catalog list` command, you could use:

```
ocne ca l
```

Command Line Completion

Learn how to use command line completion with the CLI.

You can set up command line completion for the `ocne` command. If command line completion isn't set up by default, use the `ocne 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:

```
ocne completion bash
```

The generated command line completion script must be saved to `/etc/bash_completion.d/ocne`.

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

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

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

Environment Variables

Learn how to use environment variables in the CLI.

You can use environment variables to set options that are used in some `ocne` commands. The environment variables used are:

- `KUBECONFIG`: Sets the location of the `kubeconfig` file. This behaves the same way as the `--kubeconfig` option for most `ocne` commands. For example:

  ```
  export KUBECONFIG=$HOME/.kube/kubeconfig.ocne.local
  ```

  Or:

  ```
  export KUBECONFIG=$(ocne cluster show --cluster-name cluster-name)
  ```

  where *cluster-name* is the name of the Kubernetes cluster.

- `EDITOR`: Sets the default document editor. For example:

  ```
  export EDITOR=/usr/bin/vim
  ```

- `OCNE_DEFAULTS`: Sets the location of the default configuration file. This is used to override the default value of `$HOME/.ocne/defaults.yaml`. For example:

  ```
  export OCNE_DEFAULTS=$HOME/.ocne/mydefaults.yaml
  ```

You can inspect the values of the environment variables the CLI uses by running the `ocne info` command:

```
ocne info
```

3
Configuration Files

Describes the configuration files that can be used to customize the CLI command.

Kubernetes clusters and applications can be configured through a set of YAML configuration files and `ocne` command line arguments. Configuration is layered, with each layer of configuration taking precedence over the previous layer. The layered structure provides convenient reuse of parameters that would otherwise be duplicated into every deployment.

You can configure `ocne` subcommands using three hierarchical methods. The methods are (in hierarchical order):

- Global defaults in the default configuration file, set in the `$HOME/.ocne/defaults.yaml` file.
- Kubernetes cluster configuration files. These files set the options for individual clusters and can be any name.
- Options provided with the `ocne` command.

Global defaults can be overridden by a global configuration file. Those values can in turn be overridden by a cluster or application specific configuration file. Finally, that entire stack of configuration can be overridden by `ocne` command line options.

For information on cluster configuration files, see Oracle Cloud Native Environment: Kubernetes Clusters.

Default Configuration File

Describes the default configuration file.

The `$HOME/.ocne/defaults.yaml` file is used to set global defaults for `ocne` commands. The configuration file is used to set common, environment-specific values. The file must be in YAML format. The default configuration file isn't created when you install the CLI, so you must create it to use it.

The contents of the `$HOME/.ocne/defaults.yaml` file overrides any default settings. The values in this file are used for all `ocne` commands, unless they're overridden by a layer with higher precedence.

Default Configuration File Options

Lists the options that can be included in an Oracle CNE CLI default configuration file.

The default configuration file must be in YAML format and saved to `$HOME/.ocne/defaults.yaml`. The file can contain any of the following options:

autoStartUI
Sets whether a tunnel to the Oracle CNE UI service is created when a cluster is instantiated, and starts the default browser to load the UI. For example:

```
autoStartUI: true
```

bootVolumeContainerImage
The container image registry and tag that contains an Oracle Container Host for Kubernetes (OCK) bootable image. The default is container-registry.oracle.com/olcne/ock:1.31. For example:

```
bootVolumeContainerImage: container-registry.oracle.com/olcne/ock:1.31
```

clusterDefinition
The path to a cluster configuration file. Provide this extra layer of configuration for clusters that use complex configuration that isn't provided by this default configuration file. For example:

```
clusterDefinition: mycluster.yaml
```

clusterDefinitionInline
Specifies in-line configuration options. Provide this extra layer of configuration for clusters that use complex configuration that isn't provided by this default configuration file. This option can't be used with `clusterDefinition`. For example:

```
clusterDefinitionInline: |
  key1: value1
  key2: value2
```

cni
The Container Networking Interface (CNI) provider to install when the cluster is instantiated. The value can be any CNI available with Oracle CNE, or `none` if another CNI is to be deployed either manually or using an application catalog.

> **Note:**
>
> Multus can't be used as the primary CNI. Multus is available as an application in the default application catalog. If you install the Multus application, set this option to `none`.

For example:

```
cni: flannel
```

```
cni: none
```

communityCatalog
Sets whether the Artifact Hub application catalog is installed. If this is set to `true`, the catalog is installed. If this is set to `false`, the catalog isn't installed. The default is `false`. For example:

```
communityCatalog: true
```

ephemeralCluster
Allows customization of any short-lived clusters that might be spawned to perform tasks that can't be completed on the host system. This is often used for changing boot OCK images or deploying Kubernetes Cluster API resources. The options you can use are:

name
The name of the cluster. For example:

```
ephemeralCluster:
  name: mycluster
```

preserve
Sets whether the ephemeral cluster is automatically deleted after the work is complete. The default is `false`, so ephemeral clusters are deleted after they're used. For example:

```
ephemeralCluster:
  preserve: true
```

node
Sets the configuration for the VMs. For example:

```
ephemeralCluster:
  node:
    cpus: 2
    memory: 4GB
    storage: 15GB
```

extraIgnition
The path to an Ignition file that includes extra Ignition information to include when creating a cluster, or joining nodes to a cluster. The Ignition information must comply with the Ignition specification v3.4.0, as listed in the upstream Ignition documentation, and written in YAML using the Butane Fedora CoreOS Specification v1.5.0, as described in the upstream Butane documentation. For example:

```
extraIgnition: /home/username/.ocne/ignition.ign
```

extraIgnitionInline
Extra Ignition information to include when creating a cluster, or joining nodes to a cluster. The Ignition information must comply with the Ignition specification v3.4.0, as listed in the upstream Ignition documentation, and written in YAML using the Butane Fedora CoreOS Specification v1.5.0, as described in the upstream Butane documentation. The format must be:

```
extraIgnitionInline: |
  key1: value1
```

```
  key2: value2
  ...
```

headless
Sets whether the Oracle CNE UI is installed. If this is set to `true`, the UI isn't installed. The default is `false`. For example:

```
headless: true
```

kubeApiServerBindPort
Sets the port on which the Kubernetes API Server is exposed. The default is `6443`. For example:

```
kubeApiServerBindPort: 6443
```

kubeApiServerBindPortAlt
Sets the port on which the Kubernetes API Server listens when deploying a Highly Available cluster using the Keepalived and NGINX load balancer. The default is `6444`. For example:

```
kubeApiServerBindPortAlt: 6444
```

kubeconfig
The path to the `kubeconfig` file to use for operations that require a running cluster. For example:

```
kubeconfig: /home/username/.kube/kubeconfig.utilitycluster
```

kubeProxyMode
The mode for `kube-proxy`. This can be set to either `iptables` or `ipvs`. The default is `iptables`. For example:

```
kubeProxyMode: ipvs
```

For more information on the `kube-proxy` modes, see the upstream Kubernetes documentation.

kubernetesVersion
This defines the Kubernetes version. The default is the latest version. For example:

```
kubernetesVersion: 1.31
```

osRegistry
Combined with `osTag`, this identifies an OSTree image in a container registry. It specifies the OSTree transport and the container registry URI.
Possible prefixes for the transport are:

```
ostree-image-signed
ostree-remote-image
```

```
ostree-unverified-image
ostree-unverified-registry
```

The default value is:

```
osRegistry: ostree-unverified-registry:container-registry.oracle.com/olcne/
ock-ostree
```

osTag
Combined with `osRegistry`, this identifies an OSTree image in a container registry. It specifies the tag for the image. For example:

```
osTag: 1.31
```

password
A hashed password for the OCK image user (`ocne`) to authenticate with cluster nodes. For example:

```
password: $6$jfkldjfsd$n1YMnpdxlGXO...
```

Surrounding the password with quotes is optional.
You can use the `openssl` utility to create a hashed password. For example, to generate a hashed password with the SHA512 algorithm and an automatic salt:

```
openssl passwd -6 -salt password
```

To generate a SHA512 hashed password using the provided salt phrase:

```
openssl passwd -6 -salt saltphrase password
```

podSubnet
The subnet to use for the pod network. The CNI is automatically configured to use this subnet. For example:

```
podSubnet: 10.244.0.0/16
```

providers
Specifies provider configuration options. For example:

```
providers:
  libvirt:
    options
  oci:
    options
  byo:
    options
  none:
    options
```

The options for each provider are listed in:

- libvirt Provider Options
- OCI Provider Options
- Bring Your Own Provider Options

proxy
The proxy server information. This information is configured on the Kubernetes nodes. For example:

```
proxy:
  httpsProxy: http://myproxy.example.com:2138
  httpProxy: http://myproxy.example.com:2138

noProxy: .example.com,127.0.0.1,localhost,169.254.169.254,10.96.0.0/12,10.244.
0.0/16
```

quiet
Sets whether to reduce the messages printed by the `ocne` command. If this is set to `true`, the messages are reduced. If set to `false`, the messages aren't reduced. The default is `false`. For example:

```
quiet: true
```

registry
Sets the registry from which to provision container images. The default is `container-registry.oracle.com`. For example:

```
registry: myregistry.example.com
```

serviceSubnet
The subnet to use for the service network. The default is `10.96.0.0/12`. For example:

```
serviceSubnet: 10.96.0.0/12
```

sshPublicKey
The public key of an RSA key pair for the OCK image user (`ocne`). Paste the contents of the public key file.
For example:

```
sshPublicKey: |
    ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAA...
```

sshPublicKeyPath
The path to the public key of an RSA key pair for the OCK image user (`ocne`) to authenticate with cluster nodes.

```
sshPublicKeyPath: /home/username/.ssh/id_rsa.ocne
```

libvirt Provider Options

The options for the `libvirt` provider are:

`controlPlaneNode`
Sets the configuration options for control plane nodes. To specify sizes, use:

- `M`: megabytes
- `G`: gigabytes
- `Mi`: mebibytes
- `Gi`: gibibytes

For example:

```
providers:
  libvirt:
    controlPlaneNode:
      cpu: 2
      memory: 16Gi
      storage: 8Gi
```

`network`
The name of the virtual network to use for domains. For example:

```
providers:
  libvirt:
    network: bridge-1
```

`sshKey`
The path to an SSH key to use for SSH connections. For example:

```
providers:
  libvirt:
    sshKey: /home/username/.ssh/id_rsa.ocne
```

`storagePool`
The name of the storage pool to use for OCK images. For example:

```
providers:
  libvirt:
    storagePool: mypool
```

`uri`
The default value for the libvirt connection URI. For example, for a local connection:

```
providers:
  libvirt:
    uri: qemu:///system
```

And for a remote connection:

```
providers:
  libvirt:
    uri: qemu+ssh://user@host/system
```

workerNode
Sets the configuration options for worker nodes. To specify sizes, use:

- `M`: megabytes
- `G`: gigabytes
- `Mi`: mebibytes
- `Gi`: gibibytes

For example:

```
providers:
  libvirt:
    workerNode:
      cpu: 2
      memory: 16Gi
      storage: 8Gi
```

OCI Provider Options

The options for the `oci` provider are:

compartment
The OCI compartment in which to deploy resources. This can be either the path to a compartment (for example, `mytenancy/mycompartment`), or the OCID of a compartment. For example:

```
providers:
  oci:
    compartment: OCID
```

controlPlaneShape
The name of the shape to use for compute instances when creating control plane nodes. For example:

```
providers:
  oci:
    controlPlaneShape:
      shape: VM.Standard.E4.Flex
      ocpus: 2
```

imageBucket
The OCID or name of a bucket to use to store OCK boot images when they're uploaded to OCI object storage. The default name is `ocne-images`. For example:

```
providers:
  oci:
    imageBucket: ocne-images
```

images
The OCI OCIDs of the OCK images to use as the initial disk image for any compute resources. Sets the options for the `amd64` and `arm64` architectures. For example:

```
providers:
  oci:
    images:
      amd64: OCID
      arm64: OCID
```

kubeconfig
The path to the `kubeconfig` file to use for the target management cluster. For example:

```
providers:
  oci:
    kubeconfig: /home/username/.kube/kubeconfig.mgmtcluster
```

loadBalancer
The OCIDs for subnets to use when provisioning OCI load balancers for default deployments. For example:

```
providers:
  oci:
    loadBalancer:
      subnet1: OCID
      subnet2: OCID
```

namespace
The Kubernetes namespace where the Kubernetes Cluster API resources are to be deployed.

```
providers:
  oci:
    namespace: mynamespace
```

selfManaged
Sets whether a cluster is self-managing. If set to `true`, the cluster contains the necessary controllers and resources to manage its own life cycle. If set to `false`, or not set, those resources remain in the initial administration cluster. For example:

```
providers:
  oci:
    selfManaged: true
```

profile
Sets the OCI CLI profile to use. This is the name of the profile in the OCI CLI configuration file. The default profile is `DEFAULT`. For example:

```
providers:
  oci:
    profile: MYTENANCY
```

vcn
The OCID of the Virtual Cloud Network to use when creating load balancers for default deployments.

```
providers:
  oci:
    vcn: OCID
```

workerShape
The name of the shape to use for compute instances when creating worker nodes. For example:

```
providers:
  oci:
    workerShape:
      shape: VM.Standard.E4.Flex
      ocpus: 2
```

Bring Your Own Provider Options

The options for the `byo` provider are:

automaticTokenCreation
If set to `true`, any time a join token is required it's created automatically as part of the command. If it's set to `false`, the token must be created manually. For example:

```
providers:
  byo:
    automaticTokenCreation: false
```

networkInterface
Sets the network interface to which the CNI and other Kubernetes services bind. This option is required. For example:

```
providers:
  byo:
    networkInterface: enp1s0
```

Configuration File Examples

Provides example configuration files.

Example 3-1 Set some default options to create remote libvirt clusters

This example uses a specific SSH key to connect to a remote system to create clusters, includes the proxy server configuration, and doesn't install the UI.

```
providers:
  libvirt:
    uri: qemu+ssh://myuser@host.example.com/system
    sshKey: /home/username/.ssh/id_rsa.ocne
proxy:
  httpsProxy: http://myproxy.example.com:2138
```

```
  httpProxy: http://myproxy.example.com:2138

noProxy: .example.com,127.0.0.1,localhost,169.254.169.254,10.96.0.0/12,10.244.
0.0/16
headless: true
```

Example 3-2 Set some default options to create local libvirt clusters

This example sets several options, including the sizes for the Kubernetes nodes in a libvirt cluster.

```
provider: libvirt
name: mycluster
workerNodes: 2
controlPlaneNodes: 1
providers:
  libvirt:
    controlPlaneNode:
      cpu: 2
      memory: 8Gi
      storage: 20Gi
    workerNode:
      cpu: 2
      memory: 8Gi
      storage: 20Gi
```

4
CLI Command Reference

This chapter contains the syntax for the Oracle CNE CLI, the `ocne` command, including usage, and examples. The command options are:

```
ocne command
[{-h|--help}]
[{-l|--log-level} {debug|error|info|trace}]
```

Where *command* is one of:

- application
- catalog
- cluster
- completion
- help
- image
- info
- node

Where:

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

{-l|--log-level} {debug|error|info|trace}
A global flag that sets the level of logging to be displayed. The default is `info`.

ocne application

Manages the life cycle of applications in a Kubernetes cluster. This includes adding and removing applications.

```
ocne application command
[{-h|--help}]
[{-k|--kubeconfig} path]
[{-l|--log-level} {debug|error|info|trace}]
```

Where *command* is one of:

- install
- {list|ls}
- show
- template

- uninstall
- update

Where:

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

{-k|--kubeconfig} *path*
A Kubernetes client configuration file that describes the target cluster and how to access it. If this option is specified, all operations that work against an existing Kubernetes cluster use this cluster. This option takes precedence over the `KUBECONFIG` environment variable.

{-l|--log-level} {debug|error|info|trace}
A global flag that sets the level of logging to be displayed. The default is `info`.

ocne application install

Installs an application from an application catalog.

```
ocne application install
{-b|--built-in-catalog}
[{-c|--catalog} name]
{-N|--name} name
[{-n|--namespace} namespace]
{-r|--release} name
[{-u|--values} URI]
[{-v|--version} version]
```

Where:

{-b|--built-in-catalog}
Installs the Oracle application catalog (`ocne-catalog`) into the `ocne-system` namespace, in environments where this isn't installed by default. This option is mutually exclusive with the `--catalog` option.

{-c|--catalog} *name*
The name of the catalog that contains the application. The default is `Oracle Cloud Native Environment Application Catalog`. This option is mutually exclusive with the `--built-in-catalog` option.

{-N|--name} *name*
The name of the application to install.

{-n|--namespace} *namespace*
The Kubernetes namespace in which to install the application. The namespace is created if it doesn't already exist. If this value isn't provided, the namespace from the current context of the `kubeconfig` is used.

{-r|--release} *name*
The name of the release of the application. The same application can be installed many times, differentiated by a release name.

{-u|--values} *URI*
The URI of an application configuration. The format of the configuration depends on the style of application served by the target catalog. In general, it's a set of Helm values.

{-v|--version} *version*
The version of an application to install. By default, the version is the latest stable version of the application.

Example 4-1 Install an application

To install an application from the default catalog:

```
ocne application install --release ingress-nginx --namespace ingress-nginx --
name ingress-nginx
```

Example 4-2 Install an application with configuration information

To install an application from a catalog, and include installation configuration information, use a values configuration file. For example, to include the Prometheus `node-exporter` when deploying Prometheus:

```
ocne application install --release prometheus --namespace prometheus --name
prometheus --values - << EOF
serviceAccounts:
  nodeExporter:
    create: true
    name:
    annotations: {}
nodeExporter:
  enabled: true
  image:
    repository: container-registry.oracle.com/verrazzano/node-exporter
    tag: v1.3.1
    pullPolicy: IfNotPresent
EOF
```

Example 4-3 Install an application from an application template

To install an application from a catalog using configuration information from an application template:

```
ocne application install --release prometheus --namespace prometheus --name
prometheus --values mytemplate.yaml
```

ocne application {list|ls}

Lists the applications installed in a Kubernetes cluster.

```
ocne application {list|ls}
[{-A|--all}]
[{-n|--namespace} namespace]
```

Where:

{-A|--all}
Lists the installed applications in all namespaces.

{-n|--namespace} ***namespace***
The Kubernetes namespace in which to list the installed applications. If this value isn't provided, the namespace from the current context of the `kubeconfig` is used.

Example 4-4 List applications

To list the applications in all namespaces in a cluster:

```
ocne application list --all
```

Example 4-5 List applications

To list the applications in a cluster:

```
ocne application ls
```

ocne application show

Shows details about an application installed into a Kubernetes cluster.

```
ocne application show
[{-c|--computed}]
[{-d|--difference}]
[{-n|--namespace} namespace]
{-r|--release} name
```

Where:

{-c|--computed}
Shows the complete configuration for the application. The displayed configuration includes both the custom values and the default values.

{-d|--difference}
Shows the computed values and the default values for an application.

{-n|--namespace} ***namespace***
The Kubernetes namespace in which the application is installed. If this value isn't provided, the namespace from the current context of the `kubeconfig` is used.

{-r|--release} ***name***
The name of the release of the application.

Example 4-6 Show details about an application

To show details about an application:

```
ocne application show --namespace kube-flannel --release flannel --computed
```

ocne application template

Generates a documented template containing all the configuration options available for an application. The format of the template depends on the style of application served by the target catalog. In general, it's a set of Helm values.

```
ocne application template
{-c|--catalog} name
[{-i|--interactive}]
{-N|--name} name
[{-v|--version} version]
```

Where:

{-c|--catalog} *name*
The name of the catalog that contains the application. The default is `Oracle Cloud Native Environment Application Catalog`.

{-i|--interactive}
Opens the application defined by the `EDITOR` environment variable and populates it with the template.

{-N|--name} *name*
The name of the application for which to create a template.

{-v|--version} *version*
The version of the application for which to create a template.

Example 4-7 Display a template for an application

To display a template for an application named `prometheus` in the default catalog:

```
ocne application template --name prometheus
```

Example 4-8 Save a template for an application

To save a template for an application to a file, pipe the output to a file:

```
ocne application template --name prometheus > mytemplate.yaml
```

Example 4-9 Edit a template for an application

To save a template for an application and edit it interactively:

```
ocne application template --name prometheus --interactive
```

The template is saved to a local file using the naming convention *`application_name`*`-values.yaml`.

ocne application uninstall

Uninstalls an application.

```
ocne application uninstall
[{-n|--namespace} namespace]
{-r|--release} name
```

Where:

{-n|--namespace} *namespace*
The Kubernetes namespace in which the application is installed. If this value isn't provided, the namespace from the current context of the `kubeconfig` is used.

{-r|--release} *name*
The name of the release of the application to uninstall.

Example 4-10 Uninstall an application

To uninstall an application:

```
ocne application uninstall --release prometheus --namespace prometheus
```

ocne application update

Updates an application that was deployed from an application catalog.

```
ocne application update
{-b|--built-in-catalog}
[{-c|--catalog} name]
[{-n|--namespace} namespace]
{-r|--release} name
[--reset-values]
[{-u|--values} URI]
[{-v|--version} version]
```

Where:

{-b|--built-in-catalog}
Updates the built in catalog in the `ocne-system` namespace, when this was installed using `ocne application install` with the `--built-in-catalog` option. This option is mutually exclusive with the `--catalog` option.

{-c|--catalog} *name*
The name of the catalog that contains the application. The default is `Oracle Cloud Native Environment Application Catalog`. This option is mutually exclusive with the `--built-in-catalog` option.

{-n|--namespace} *namespace*
The Kubernetes namespace in which the application is installed. If this value isn't provided, the namespace from the current context of the `kubeconfig` is used.

{-r|--release} *name*
The name of the release of the application.

--reset-values
Reset the values to the ones built into the chart. If the `--values` option is also used, it's treated as a new set of overrides.

{-u|--values} *URI*
The URI of an application configuration. The format of the configuration depends on the style of application served by the target catalog. In general, it's a set of Helm values.

{-v|--version} *version*
The application version number. This sets the version of the application to be used when updating the application. By default, the version is the latest stable version of the application.

Example 4-11 Update an application

To update an application with extra configuration information, use a values configuration file. For example, to update the `prometheus` application to include the Prometheus `node-exporter`:

```
ocne application update --release prometheus --namespace prometheus --values
- << EOF
serviceAccounts:
  nodeExporter:
    create: true
    name:
    annotations: {}
nodeExporter:
  enabled: true
  image:
    repository: container-registry.oracle.com/verrazzano/node-exporter
    tag: v1.3.1
    pullPolicy: IfNotPresent
EOF
```

ocne catalog

Manages the life cycle of application catalogs in a Kubernetes cluster. This includes adding and removing catalogs.

```
ocne catalog command
[{-h|--help}]
[{-k|--kubeconfig} path]
[{-l|--log-level} {debug|error|info|trace}]
```

Where *command* is one of:

- add
- copy
- get
- {list|ls}
- mirror

- remove
- search

Where:

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

{-k|--kubeconfig} *path*
A Kubernetes client configuration file that describes the target cluster and how to access it. If this option is specified, all operations that work against an existing Kubernetes cluster use this cluster. This option takes precedence over the `KUBECONFIG` environment variable.

{-l|--log-level} {debug|error|info|trace}
A global flag that sets the level of logging to be displayed. The default is `info`.

ocne catalog add

Adds an application catalog to a Kubernetes cluster.

```
ocne catalog add
{-N|--name} name
[{-n|--namespace} namespace]
[{-p|--protocol} protocol]
{-u|--uri} URI
```

Where:

{-N|--name} *name*
The name of the catalog to add.

{-n|--namespace} *namespace*
The namespace in which to add the catalog. The default is `ocne-system`.

{-p|--protocol} *protocol*
The protocol for the application to add. The options are `helm` and `artifacthub`. The default is `helm`.

{-u|--uri} *URI*
The URI of the application catalog to add.

Example 4-12 Add a catalog

To add the Artifact Hub application catalog:

```
ocne catalog add --protocol artifacthub --name artifacthub --uri https://
artifacthub.io
```

ocne catalog copy

Copies a list of container images from a local file to a container registry. The container images can then be used by applications in an application catalog. The images can also be copied to another local file.

```
ocne catalog copy
{-d|--destination} URI
{-e|--destinationfilepath} path
{-f|--filepath} path
```

Where:

{-d|--destination} *URI*
The URI of the destination container registry. For example, `myregistry.example.com`.

{-e|--destinationfilepath} *path*
The file path and name of the file to copy the container images.

{-f|--filepath} *path*
The file path and name of the file that contains the container images to be copied. The list of images must be delimited by the new line character.

Example 4-13 Copy container images from a file to a container registry

To copy the container images from a local file to a container registry:

```
ocne catalog copy --filepath catalog.txt --destination myregistry.example.com
```

ocne catalog get

Prints a YAML document that contains the complete description of an application catalog.

```
ocne catalog get
{-N|--name} name
```

Where:

{-N|--name} *name*
The name of the catalog to search. The default is the `Oracle Cloud Native Environment Application Catalog`.

Example 4-14 Display information about the default application catalog

To display information about the default application catalog:

```
ocne catalog get
```

Example 4-15 Display information about an application catalog

To display information about an application catalog:

```
ocne catalog get --name artifacthub
```

ocne catalog {list|ls}

Lists the application catalogs configured for a Kubernetes cluster.

```
ocne catalog {list|ls}
```

Example 4-16 List catalogs

To list the application catalogs in a cluster:

```
ocne catalog list
```

Example 4-17 List catalogs

To list the application catalogs in a cluster:

```
ocne catalog ls
```

ocne catalog mirror

Clones the container images used by applications in an application catalog and pushes them to a private registry. When no other options are provided, it lists the applications available in the catalog.

```
ocne catalog mirror
[{-a|--archive} path]
[{-c|--config} path]
{-d|--destination} URI
[{-o|--download}]
{-N|--name} name
[{-p|--push}]
[{-s|--source} registry]
[{-q|--quiet}]
```

Where:

`{-a|--archive}` *`path`*
Specifies the path to the `.tgz` archive file to generate when used with the `--download` option.

`{-c|--config}` *`path`*
The path to an Oracle CNE configuration file. If a configuration file is provided, only the applications listed in that file are mirrored.

{-d|--destination} *URI*
The URI of the destination container registry. The images from the application catalog are tagged so they belong to this registry, and are optionally pushed to that registry with the `--push` option.

{-o|--download}
Downloads the requested images to `$HOME/.ocne/downloaded-images.tgz`, or to another location specified by the `--archive` option.

{-N|--name} *name*
The name of the catalog to mirror. The default is the `Oracle Cloud Native Environment Application Catalog`. If the Oracle catalog isn't running as a container, the `embedded` catalog, built into the CLI, is used.

{-p|--push}
Pushes the images to the destination container registry.

{-s|--source} *registry*
The source registry to use for container images without a registry. The default is `container-registry.oracle.com`. For example, `olcne/ui` is translated as `container-registry.oracle.com/olcne/ui`.

{-q|--quiet}
Output only the image names, and omit all other output.

Example 4-18 List applications in the Oracle catalog

To list all the applications in the Oracle catalog:

```
ocne catalog mirror
```

Example 4-19 Mirror the Oracle catalog

To mirror the Oracle catalog to a private container registry:

```
ocne catalog mirror --destination myregistry.example.io --push
```

Example 4-20 Mirror a catalog

To mirror the a catalog to a private container registry:

```
ocne catalog mirror --name mycatalog --destination myregistry.example.io --
push
```

Example 4-21 Mirror the embedded Oracle catalog

To mirror the Oracle catalog embedded in the CLI to a private container registry:

```
ocne catalog mirror --name embedded --destination myregistry.example.io --push
```

Example 4-22 Mirror specific applications

To mirror only those images that are used by the applications listed in a cluster configuration file to a private container registry:

```
ocne catalog mirror --destination myregistry.example.io --config
mycluster.yaml --push
```

Example 4-23 Download all images from a catalog to an archive file

To download all images to the default location (`$HOME/.ocne/downloaded-images.tgz`):

```
ocne catalog mirror --download
```

Example 4-24 Download specific applications to a named archive file

To download the images listed in a cluster configuration file to a specified local archive file:

```
ocne catalog mirror --config mycluster.yaml --download --archive $HOME/
myimages.tgz
```

ocne catalog remove

Removes a catalog from a Kubernetes cluster.

```
ocne catalog remove
{-N|--name} name
{{-n|--namespace} namespace}
```

Where:

{-N|--name} *name*
The name of the catalog to remove.

{-n|--namespace} *namespace*
The namespace for the catalog. The default is `ocne-system`.

Example 4-25 Remove a catalog

To remove an application catalog named `artifacthub`:

```
ocne catalog remove --name artifacthub
```

ocne catalog search

Print the applications in a catalog that follow a specific pattern.

```
ocne catalog search
[{-N|--name} name]
[{-p|--pattern} pattern]
```

Where:

{-N|--name} *name*
The name of the catalog to search. The default is the `Oracle Cloud Native Environment Application Catalog`.

{-p|--pattern} *pattern*
The search terms. Must be a valid RE2 regular expression. You don't need to include this option when searching the default catalog, but you must include this option for any catalogs you add.

Example 4-26 Search the default catalog

To show all applications in the Oracle catalog:

```
ocne catalog search
```

Example 4-27 List all applications in the embedded catalog

To show all applications in the embedded Oracle catalog, built into the CLI:

```
ocne catalog search --name embedded
```

Example 4-28 Search the default catalog for a string

To search the default catalog for a specific string:

```
ocne catalog search --pattern 'ingress-*'
```

Example 4-29 Search a catalog for a string

To search the specified application catalog for a specific string:

```
ocne catalog search --name artifacthub --pattern 'ingress-*'
```

ocne cluster

Creates and manages Kubernetes clusters.

```
ocne cluster command
[{-h|--help}]
[{-k|--kubeconfig} path]
[{-l|--log-level} {debug|error|info|trace}]
```

Where *command* is one of:

- analyze
- backup
- console
- delete
- dump
- info

- join
- {list|ls}
- show
- stage
- start
- template

Where:

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

{-k|--kubeconfig} ***path***
A Kubernetes client configuration file that describes the target cluster and how to access it. If this option is specified, all operations that work against an existing Kubernetes cluster use this cluster. This option takes precedence over the `KUBECONFIG` environment variable.

{-l|--log-level} {debug|error|info|trace}
A global flag that sets the level of logging to be displayed. The default is `info`.

ocne cluster analyze

Starts a set of tools that collect data from a live Kubernetes cluster, or a cluster dump file, to analyze, and diagnose problems in the cluster.

If no options are specified, the live cluster is analyzed, not a cluster dump file.

```
ocne cluster analyze
[{-d|--dump-directory} path]
[{-s|--skip-nodes}]
[{-p|--skip-pod-logs}]
[{-v|--verbose}]
```

Where:

{-d|--dump-directory} ***path***
The directory that includes the cluster dump created using the `ocne cluster dump` command. This option is mutually exclusive with the `--archive` and `--skip-nodes` options.

{-s|--skip-nodes}
Skips collecting and analyzing the data from nodes. This is only valid for analyzing a live cluster, not a cluster dump.

{-p|--skip-pod-logs}
Skips collecting and analyzing the data from pods. This is only valid for analyzing a live cluster, not a cluster dump.

-v|--verbose
Displays more detailed information related to the analysis.

Example 4-30 Analyze a cluster dump in a directory

To start the tools available to analyze a cluster:

```
ocne cluster analyze --dump-directory $HOME/dump/
```

Example 4-31 Analyze a cluster dump, displaying more detailed information

To analyze cluster dump files, displaying more detailed output:

```
ocne cluster analyze --dump-directory $HOME/dump/ --verbose
```

Example 4-32 Analyze a live cluster

To analyze a running cluster:

```
ocne cluster analyze
```

Example 4-33 Analyze a live cluster, skipping node data

To analyze a running cluster without including information from the nodes:

```
ocne cluster analyze --skip-nodes
```

ocne cluster backup

Saves a snapshot of the key Kubernetes control plane containers to the `etcd` database.

```
ocne cluster backup
{-o|--out} path
```

Where:

`{-o|--out}` *`path`*
Sets the location to write the backup information. The backup is saved as `etcd` database format. This option is mandatory.

Example 4-34 Back up the `etcd` database for a cluster

To back up the `etcd` database for a cluster to the current directory:

```
ocne cluster backup --out mybackup.db
```

ocne cluster console

Starts an administration console on a node in a Kubernetes cluster.

```
ocne cluster console
[{-d|--direct}]
{-N|--node} nodename
[{-t|--toolbox}]
[-- command]
```

Where:

{-d|--direct}
Starts the console `chroot`ed to the root of the target node's file system.

{-N|--node} *nodename*
The name of the Kubernetes cluster node where the console is to be started. The name must be one of the nodes listed in the output of a `kubectl get nodes` command.

{-t|--toolbox}
Creates the console using a container image that contains tools useful to diagnose an Oracle Linux system. The container includes tools such as `strace`, `tcpdump`, `traceroute`, and `sos`.

-- *command*
The command to run in the console. When this option is used, the console connection is closed after the command completes.

Example 4-35 Start an administration console on a node

To start an administration console on a Kubernetes node:

```
ocne cluster console --node mynode
```

Example 4-36 Start an administration console `chroot`ed to the target node's root directory

To start an administration console `chroot`ed to the root of the target node's file system:

```
ocne cluster console --direct --node mynode
```

Example 4-37 Start an administration console on a node and include the toolbox

To start an administration console on a Kubernetes node and include the debugging toolbox:

```
ocne cluster console --node mynode --toolbox
```

Example 4-38 Run a command in an administration console `chroot`ed to the target node's root directory

To run a command, such as `ls /proc`, in an administration console `chroot`ed to the target node's root directory:

```
ocne cluster console --direct --node mynode -- ls /proc
```

ocne cluster delete

Destroys a cluster that has been deployed using `ocne cluster start`. This command only applies to clusters that have been created using local virtualization. It can't be used to destroy clusters deployed to bare metal systems, compute resources, or deployed using infrastructure automation APIs.

```
ocne cluster delete
[{-C|--cluster-name} name]
[{-c|--config} URI]
```

```
[{-P|--provider} provider]
[{-s|--session} URI]
```

Where:

{-C|--cluster-name} *name*
The name of the cluster to delete. The default is `ocne`.

{-c|--config} *URI*
The path to a configuration file that contains the definition of the cluster to delete.

{-P|--provider} *provider*
The provider to use when interacting with the cluster. Options are:

- `libvirt` (the default)
- `oci`
- `byo`
- `none`

{-s|--session} *URI*
The session URI for the `libvirt` provider.

Example 4-39 Delete the default `libvirt` cluster

To delete the default `libvirt` cluster:

```
ocne cluster delete
```

Example 4-40 Delete a cluster using the cluster name

To delete a cluster named `mycluster`:

```
ocne cluster delete --cluster-name mycluster
```

Example 4-41 Delete an ephemeral cluster

To delete an ephemeral cluster named `ocne-ephemeral`:

```
ocne cluster delete --cluster-name ocne-ephemeral
```

Example 4-42 Delete a cluster using a configuration file

To delete a cluster created using a configuration file:

```
ocne cluster delete --config myconfig.yaml
```

ocne cluster dump

Dumps information about the Kubernetes cluster in a local directory. By default, all cluster resources are included, except Secrets, and ConfigMaps.

```
ocne cluster dump
[{-c|--curated-resources}]
```

```
[{-z|--generate-archive} path]
[{-m|--include-configmaps}]
[--json]
[--managed]
[{-n|--namespaces} namespace,...]
[{-N|--nodes} nodename, ...]
[{-d|--output-directory} path]
[{-r|--skip-cluster}]
[{-s|--skip-nodes}]
[{-p|--skip-pod-logs}]
[{-t|--skip-redaction}]
```

Where:

{-c|--curated-resources}
Sets whether to dump manifests from a curated subset of cluster resources. By default, all cluster resources are dumped, except Secrets, and ConfigMaps.

{-z|--generate-archive} *path*
Sets the path and file name for a `.tgz` or `.tar.gz` file in which to generate and save the dump file. This option is mutually exclusive with the `--output-directory` option.

{-m|--include-configmaps}
Sets whether to include ConfigMaps in the cluster dump. This option not valid when the `curated-resources` option is used.

--json
Dumps cluster resources in JSON format, instead of the default YAML.

--managed
Dumps cluster resources as managed fields, so that Kubernetes can track changes to those resources.

{-n|--namespaces} *namespace,...*
A comma separated list of namespaces. The default is to include all namespaces.

{-N|--nodes} *nodename, ...*
A comma separated list of the nodes to include. The default is all nodes.

{-d|--output-directory} *path*
The path on the local system in which to save the output. This option is mutually exclusive with the `--generate-archive` option.

{-r|--skip-cluster}
Don't include data from the cluster.

{-s|--skip-nodes}
Don't include data from the nodes.

{-p|--skip-pod-logs}
Don't include data from the pods.

{-t|--skip-redaction}
Don't redact sensitive data.

Example 4-43 Dump cluster information to a local directory

To dump information about a cluster to a local directory:

```
ocne cluster dump --output-directory $HOME/dump
```

Example 4-44 Dump node information to a local directory

To dump information about specific nodes, and not include cluster information, to a local directory:

```
ocne cluster dump --output-directory $HOME/dump --skip-cluster --nodes
mynode1,mynode2
```

Example 4-45 Dump cluster information to an archive file

To dump cluster information, but not include node information, to an archive file:

```
ocne cluster dump --generate-archive $HOME/dump/cluster_dump.tgz --skip-nodes
```

ocne cluster info

Displays the overall cluster information, along with node level information.

```
ocne cluster info
[{-N|--nodes}] nodename, ...
[{-s|--skip-nodes }]
```

Where:

{-N|--nodes} ***nodename, ...***
A comma separated list of the nodes to include. The default is all nodes.

{-s|--skip-nodes}
Don't include data from the nodes.

Example 4-46 Get cluster info, including data from all nodes

To display the overall cluster information, along with node level information:

```
ocne cluster info
```

Example 4-47 Get cluster info, skipping the node data

```
ocne cluster info --skip-nodes
```

ocne cluster join

Joins a node to a cluster, or generates the Ignition files required to do so.

This command can be used for:

- Preprovisioned compute resources created with the `byo` provider. Nodes are migrated from one cluster to another cluster.
- Self provisioned compute created with the `byo` provider. Generates the materials needed to join a node to a cluster on first boot.

Nodes are migrated as worker nodes, unless you specify the `--role-control-plane` node option of the `ocne cluster join` command.

Set the location of the source cluster using the `--kubeconfig` command option. This option is required for this command.

```
ocne cluster join
[{-c|--config} path]
[{-d|--destination} path]
[{-N|--node} name]
[{-P|--provider} provider]
[{-r|--role-control-plane}]
```

Where:

{-c|--config} *path*
The path to a configuration file that contains the definition of the cluster. If this value isn't provided, a small (ephemeral) cluster might be created, using the default hypervisor, for the system where the command is run.

{-d|--destination} *path*
The path to the `kubeconfig` file for the destination cluster.

{-N|--node} *name*
The name of the node to move from the source cluster to the destination cluster, as seen from within Kubernetes. The name must be one of the nodes listed in the output of a `kubectl get nodes` command.

{-P|--provider} *provider*
The provider to use when interacting with the cluster. Options are:

- `libvirt` (the default)
- `oci`
- `byo`
- `none`

{-r|--role-control-plane}
Sets the role of the node to be a control plane node in the target cluster. If this option isn't used, the node is set as a worker node.

Example 4-48 Migrate a node to another BYO cluster

To migrate a node from one cluster to another BYO cluster:

```
ocne cluster join --kubeconfig $HOME/.kube/kubeconfig.mycluster --provider
byo --node source-worker-1  --destination $HOME/.kube/kubeconfig.targetcluster
```

Example 4-49 Migrate a node to another BYO cluster as a control plane node

To migrate a node from one cluster to another BYO cluster, and assign it as a control plane node:

```
ocne cluster join --kubeconfig $HOME/.kube/kubeconfig.mycluster --provider
byo --node source-worker-1 --destination $HOME/.kube/kubeconfig.targetcluster
--role-control-plane
```

Example 4-50 Migrate a node to another BYO cluster using a configuration file

To migrate a node from one cluster to another BYO cluster, using a configuration file:

```
ocne cluster join --kubeconfig $HOME/.kube/kubeconfig.mycluster --config
byo.yaml --node source-worker-1 --destination $HOME/.kube/
kubeconfig.targetcluster
```

Example 4-51 Generate Ignition file for a worker node in a BYO cluster

To generate the Ignition information required to add a worker node to a BYO cluster:

```
ocne cluster join --kubeconfig $HOME/.kube/kubeconfig.mycluster --config
byo.yaml > worker.ign
```

Example 4-52 Generate Ignition file for a control plane node in a BYO cluster

To generate the Ignition information required to add a control plane node to a BYO cluster:

```
ocne cluster join --kubeconfig $HOME/.kube/kubeconfig.mycluster --config
byo.yaml --role-control-plane  > control_plane.ign
```

ocne cluster {list|ls}

Lists all known Kubernetes clusters.

```
ocne cluster {list|ls}
```

Example 4-53 Lists all known clusters

To print a list of all known Kubernetes clusters:

```
ocne cluster list
```

Example 4-54 Lists all known clusters

To print a list of all known Kubernetes clusters:

```
ocne cluster ls
```

ocne cluster show

Lists the location of a `kubeconfig` file, the full cluster configuration information, or a specific cluster configuration field, of a Kubernetes cluster.

```
ocne cluster show
[{-a|--all}]
[{-C|--cluster-name} name]
[{-f|--field} path]
```

Where:

{-a|--all}
Shows the full cluster configuration.

{-C|--cluster-name} *name*
The name of the cluster. The default is `ocne`.

{-f|--field} *path*
The path to a configuration field in the cluster configuration. For example:

```
--field config.providerconfig.oci
```

Example 4-55 Show the `kubeconfig` file location for the default cluster

To show the location of the `kubeconfig` file for the default cluster:

```
ocne cluster show
```

Example 4-56 Show the `kubeconfig` file location for a cluster

To show the location of the `kubeconfig` file for a cluster:

```
ocne cluster show --cluster-name mycluster
```

Example 4-57 Show the full configuration for a cluster

To show the full configuration for a cluster:

```
ocne cluster show --cluster-name mycluster --all
```

Example 4-58 Show a configuration field for a cluster

To show a specific configuration field for a cluster:

```
ocne cluster show --cluster-name mycluster --field
config.providerconfig.libvirt
```

ocne cluster stage

Sets the target Kubernetes version for the cluster and all its nodes.

Nodes are configured to poll the container registry to pull and stage the Oracle Container Host for Kubernetes (OCK) image for the target Kubernetes version. After the image is staged, use the `ocne cluster update` command to install the image and reboot the nodes.

```
ocne cluster stage
[{-c|--config} path]
[{-r|--os-registry} registry]
[{-t|--transport} transport]
{-v|--version} version
```

Where:

{-c|--config} *path*
The path to a configuration file that contains the definition of the cluster. If this value isn't provided, a small (ephemeral) cluster might be created, using the default hypervisor, for the system where the command is run.

{-r|--os-registry} *registry*
The name of the container registry to use during the upgrade. The default is `container-registry.oracle.com/olcne`.

{-t|--transport} *transport*
The transport type to use during the upgrade. The default is `ostree-unverified-registry`. For a list of the available transports, see the upstream OSTree documentation.

{-v|--version} *version*
The target Kubernetes version. This must be the next available minor Kubernetes version.

Example 4-59 Set the target Kubernetes version in a cluster

To set the target Kubernetes version in a cluster, and stage the OCK image when it's available:

```
ocne cluster stage --version 1.31
```

ocne cluster start

Deploys a Kubernetes cluster using default or customized configuration.

Four primary types of deployments are available:

- Local virtualization
- Preprovisioned compute resources
- Self provisioned compute resources
- Cloud provider or other infrastructure automation

```
ocne cluster start
[{-u|--auto-start-ui} {true|false}]
[{-o|--boot-volume-container-image} URI]
[{-C|--cluster-name} name]
```

```
[{-c|--config} path]
[{-n|--control-plane-nodes} integer]
[{-i|--key} path]
[--load-balancer address]
[{-P|--provider} provider]
[{-s|--session} URI]
[{-v|--version} version]
[--virtual-ip IP]
[{-w|--worker-nodes} integer]
```

Where:

{-u|--auto-start-ui} {true|false}
Sets whether the UI is automatically loaded in a browser window after the cluster has started. The default is `true`.

{-o|--boot-volume-container-image} *URI*
The URI of a container image that contains the OS boot volume.

{-C|--cluster-name} *name*
The name of the cluster to start. The default is `ocne`.

{-c|--config} *path*
The path to a configuration file that contains the definition of the cluster. If this value isn't provided, a small (ephemeral) cluster might be created, using the default hypervisor, for the system where the command is run.

{-n|--control-plane-nodes} *integer*
The number of control plane nodes to provision for clusters deployed using local virtualization.

{-i|--key} *path*
The path to the SSH public key for the remote system. The default is `$HOME/.ssh/id_rsa.pub`.

--load-balancer *address*
The hostname or IP address of an external load balancer for the Kubernetes API Server.

{-P|--provider} *provider*
The provider to use when interacting with the cluster. Options are:

- `libvirt` (the default)
- `oci`
- `byo`
- `none`

{-s|--session} *URI*
The session URI for the `libvirt` provider.
For local sessions, use `qemu:///system`.
For remote sessions, use the format `qemu+ssh://`*`host`*`/system` where *host* is the name or IP address of the remote system.

{-v|--version} *version*
The Kubernetes version number to use.

`--virtual-ip` *`IP`*
The virtual IP address to use for the Kubernetes API Server when using the internal load balancer. If you don't set an IP address, one is set automatically using the subnet of the control plane nodes.

`{-w|--worker-nodes}` *`integer`*
The number of worker nodes to provision for clusters deployed using local virtualization.

Example 4-60 Create a default cluster using the `libvirt` provider

To create a libvirt cluster, using all default settings:

```
ocne cluster start
```

Example 4-61 Create a default cluster using the `oci` provider

To create a cluster on OCI, using the default settings and default configuration files on the system:

```
ocne cluster start --provider oci
```

Example 4-62 Create a `libvirt` cluster using a configuration file

To create a `libvirt` cluster using a configuration file:

```
ocne cluster start --config myconfig.yaml
```

Example 4-63 Create a cluster using the `byo` provider

To create a cluster using the `byo` provider:

```
ocne cluster start --provider byo
```

Example 4-64 Create a `libvirt` cluster with specified nodes and virtual IP

To create a cluster with a specified number of worker and control plane nodes, and a virtual IP address:

```
ocne cluster start --control-plane-nodes 3 --worker-nodes 5 --virtual-ip
192.168.0.100
```

Example 4-65 Create a remote `libvirt` cluster using a configuration file

To create a cluster on a remote host using a configuration file:

```
ocne cluster start --session qemu+ssh://myuser@myhost.example.com/system --
config myconfig.yaml
```

Example 4-66 Install the UI and application catalog into an existing cluster

To install the UI and application catalog into an existing Oracle CNE Release 1.*x* Kubernetes cluster:

```
ocne cluster start --provider none --kubeconfig $HOME/.kube/kubeconfig.ocne19
```

ocne cluster template

Creates a sample cluster template that can be customized as needed.

The output is displayed to the terminal. To save the output, pipe it to a file using `> filename.yaml`.

```
ocne cluster template
[{-c|--config} path]
[{-P|--provider} provider]
```

Where:

`{-c|--config}` *`path`*
The path to a configuration file that contains the definition of the cluster. If this value isn't provided, a small (ephemeral) cluster might be created, using the default hypervisor, for the system where the command is run.

`{-P|--provider}` *`provider`*
The provider to use when interacting with the cluster. Options are:

- `libvirt`
- `oci` (the default)
- `byo`
- `none`

Example 4-67 Generate a cluster template file

To generate and display a cluster template file for the `oci` provider:

```
ocne cluster template
```

Example 4-68 Generate and save a cluster template file

To generate and save a cluster template file:

```
ocne cluster template > mytemplate.yaml
```

Example 4-69 Generate and save a cluster template file using a configuration file

To generate and save a cluster template using a configuration file:

```
ocne cluster template --config myconfig.yaml > mytemplate.yaml
```

ocne completion

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

```
ocne completion command
[{-h|--help}]
[{-k|--kubeconfig} path]
[{-l|--log-level} {debug|error|info|trace}]
```

Where *command* is one of:

- bash
- fish
- powershell
- zsh

Where:

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

{-k|--kubeconfig} *path*
A Kubernetes client configuration file that describes the target cluster and how to access it. If this option is specified, all operations that work against an existing Kubernetes cluster use this cluster. This option takes precedence over the `KUBECONFIG` environment variable.

{-l|--log-level} {debug|error|info|trace}
A global flag that sets the level of logging to be displayed. The default is `info`.

ocne completion bash

Generates the command line completion script for the Bash shell.

Requires the `bash-completion` package.

```
ocne completion bash
[--no-descriptions]
```

Where:

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

Example 4-70 Generate syntax completion script for Bash shell

To generate a command line completion script for the Bash shell:

```
ocne completion bash
```

ocne completion zsh

Generates the command line completion script for the Zsh shell.

```
ocne completion zsh
[--no-descriptions]
```

Where:

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

Example 4-71 Generate syntax completion script for Zsh shell

To generate a command line completion script for the Zsh shell:

```
ocne completion zsh
```

ocne completion fish

Generates the command line completion script for the fish shell.

```
ocne completion fish
[--no-descriptions]
```

Where:

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

Example 4-72 Generate syntax completion script for fish shell

To generate a command line completion script for the fish shell:

```
ocne completion fish
```

ocne completion powershell

Generates the command line completion script for the PowerShell shell.

```
ocne completion powershell
[--no-descriptions]
```

Where:

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

Example 4-73 Generate syntax completion script for PowerShell shell

To generate a command line completion script for the PowerShell shell:

```
ocne completion powershell
```

ocne help

Displays help about any command.

```
ocne help command
[{-h|--help}]
[{-l|--log-level} {debug|error|info|trace}]
```

Where *command* is one of:

- application
- catalog
- cluster
- completion
- help
- image
- info
- node

Where:

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

{-l|--log-level} {debug|error|info|trace}
A global flag that sets the level of logging to be displayed. The default is `info`.

Example 4-74 Show help about a command

To display help about the `ocne cluster start` command:

```
ocne help cluster start
```

ocne image

Manages Oracle CNE images.

```
ocne image command
[{-h|--help}]
[{-k|--kubeconfig} path]
[{-l|--log-level} {debug|error|info|trace}]
```

Where *command* is one of:

- create
- upload

Where:

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

{-k|--kubeconfig} *path*
A Kubernetes client configuration file that describes the target cluster and how to access it. If this option is specified, all operations that work against an existing Kubernetes cluster use this cluster. This option takes precedence over the `KUBECONFIG` environment variable.

{-l|--log-level} {debug|error|info|trace}
A global flag that sets the level of logging to be displayed. The default is `info`.

ocne image create

Creates an Oracle Container Host for Kubernetes (OCK) image for a provider. The image is saved into the `$HOME/.ocne/images/` directory.

```
ocne image create
{-a|--arch} arch
[{-t|--type} provider]
[{-v|--version} version]
```

Where:

{-a|--arch} *arch*
The architecture of the image to create. The options are `amd64` or `arm64`.

{-t|--type} *provider*
The provider type. Options are:

- `oci`: Generates a bootable Qcow2 image that can be used for libvirt and OCI deployments.
- `olvm`: Generates a bootable Qcow2 image that can be used for Oracle Linux Virtualization Manager deployments.
- `ostree`: Creates an OSTree archive image that can be used for the Bring Your Own provider deployments.

The default is `oci`.

{-v|--version} *version*
The version of Kubernetes to embed in the image.

Example 4-75 Create an OCK image for OCI

To create an OCK image for the `amd64` architecture for the `oci` provider:

```
ocne image create --type oci --arch amd64
```

Example 4-76 Create an OCK image for the Bring Your Own provider

To create an OCK image for the `arm64` architecture for the `byo` provider:

```
ocne image create --type ostree --arch arm64
```

ocne image upload

Uploads an Oracle Container Host for Kubernetes (OCK) image to OCI object storage for a specific architecture.

```
ocne image upload
{-a|--arch} arch
[{-b|--bucket} name]
{-c|--compartment} name
[{-d|--destination} path]
{-f|--file} path
{-i|--image-name} name
{-t|--type} provider
{-v|--version} version
```

Where:

{-a|--arch} *arch*
The architecture of the image to upload. Options are `amd64` or `arm64`.

{-b|--bucket} *name*
The name or OCID of the object storage bucket. The default is `ocne-images`.

{-c|--compartment} *name*
The name or OCID of the compartment.

{-d|--destination} *path*
The path to the destination. This can be any target available with the Open Container Initiative transports and formats. See `containers-transports(5)` for available options. For example, the URI to a container registry:

```
--destination docker://myregistry.example.com/ock-ostree:latest
```

{-f|--file} *path*
The path to the image on the localhost.

{-i|--image-name} *name*
The name of the image to use when uploaded to the destination. The default is `ock`.

{-t|--type} *provider*
The provider type. Options are `oci` and `ostree`. The default is `oci`.

{-v|--version} *version*
The target Kubernetes version embedded in the image.

Example 4-77 Upload an OCK image to OCI using defaults

To upload an OCK image for the `oci` provider to OCI using the default options:

```
ocne image upload --compartment ocid1.compartment.oc1..UniqueID --
file $HOME/.ocne/images/boot.qcow2-1.31-amd64.oci --arch amd64
```

Example 4-78 Upload an OCK image to OCI

To upload an OCK image for the `oci` provider to OCI using more specific requirements:

```
ocne image upload --compartment ocid1.compartment.oc1..UniqueID --bucket my-
ocne-images --file $HOME/.ocne/images/boot.qcow2-1.31-arm64.oci --image-name
ock-arm64 --version 1.31 --arch arm64
```

Example 4-79 Upload an OSTree image to a container registry

To upload an OSTree archive image for the `byo` provider to a container registry:

```
ocne image upload --type ostree --file $HOME/.ocne/images/ock-1.31-amd64-
ostree.tar --destination docker://myregistry.example.com/ock-ostree:latest --
arch amd64
```

Example 4-80 Upload an OSTree image to a container registry

To upload an OSTree archive image for the `byo` provider to a local directory path:

```
ocne image upload --type ostree --file $HOME/.ocne/images/ock-1.31-amd64-
ostree.tar --destination dir:ock-ostree --arch amd64
```

ocne info

Displays CLI version information and the values of environment variables that can be set for the CLI.

```
ocne info
[{-h|--help}]
[{-l|--log-level} {debug|error|info|trace}]
```

Where:

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

`{-l|--log-level} {debug|error|info|trace}`
A global flag that sets the level of logging to be displayed. The default is `info`.

Example 4-81 Show information about the CLI

To show the CLI version and values of CLI environment variables:

```
ocne info
```

ocne node

Manages the life cycle of nodes in a Kubernetes cluster.

```
ocne node command
[{-h|--help}]
[{-k|--kubeconfig} path]
[{-l|--log-level} {debug|error|info|trace}]
```

Where *command* is one of:

- update

Where:

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

{-k|--kubeconfig} *path*
A Kubernetes client configuration file that describes the target cluster and how to access it. If this option is specified, all operations that work against an existing Kubernetes cluster use this cluster. This option takes precedence over the `KUBECONFIG` environment variable.

{-l|--log-level} {debug|error|info|trace}
A global flag that sets the level of logging to be displayed. The default is `info`.

ocne node update

Updates the staged Oracle Container Host for Kubernetes (OCK) image on a Kubernetes node.

Use the `ocne cluster info` command to ensure an updated Oracle Container Host for Kubernetes (OCK) image is staged and ready to install on a node.

```
ocne node update
[{-d|--delete-emptydir-data}]
[{-c|--disable-eviction}]
{-N|--node} name
[{-p|--pre-update-mode} mode]
[{-t|--timeout} minutes]
```

Where:

{-d|--delete-emptydir-data}
Deletes the pods that use `emptyDir` during node drain.

{-c|--disable-eviction}
Forces pods to be deleted during drain, bypassing `PodDisruptionBudget`.

{-N|--node} *name*
The name of the node to update, as seen from within Kubernetes. The name must be one of the nodes listed in the output of a `kubectl get nodes` command.

{-p|--pre-update-mode} ***mode***
Sets how to handle the preupdate steps. Options are:

- `default`: Runs the preupdate process and updates the node. This is the default.
- `only`: Run the preupdate process, but skip updating nodes. The `--node` option isn't required with this setting.
- `skip`: Prevents the preupdate process from being run.

{-t|--timeout} ***minutes***
Sets the timeout in minutes. The default is `30` minutes.

Example 4-82 Update a staged OCK image on a node

To update a staged OCK image on a node:

```
ocne node update --node mynode
```