Red Hat build of MicroShift 4.15

CLI tools

Learning how to use the command-line tools for MicroShift

Last Updated: 2024-05-08

Red Hat build of MicroShift 4.15 CLI tools

Learning how to use the command-line tools for MicroShift

Legal Notice

Abstract

This document provides information about using the command-line tools for MicroShift. Installing and configuring optional CLI tools such as `oc` and `kubectl` are detailed. A reference of CLI commands and examples of how to use them are also included.

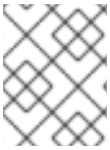
Table of Contents

CHAPTER 1. RED HAT BUILD OF MICROSHIFT CLI TOOLS INTRODUCTION

You can use different command-line interface (CLI) tools to build, deploy, and manage Red Hat build of MicroShift clusters and workloads. With CLI tools, you can complete various administration and development operations from the terminal to manage deployments and interact with each component of the system.

CLI tools available for use with Red Hat build of MicroShift are the following:

- Kubernetes CLI (**kubectl**)
- The OpenShift CLI (**oc**) tool with an enabled subset of commands
- Built-in **microshift** command types



NOTE

Commands for multi-node deployments, projects, and developer tooling are not supported by Red Hat build of MicroShift.

1.1. ADDITIONAL RESOURCES

- Installing the OpenShift CLI tool for MicroShift.
- Detailed description of the OpenShift CLI (oc).
- Red Hat Enterprise Linux (RHEL) documentation for specific use cases .
- Cluster access with kubeconfig

CHAPTER 2. GETTING STARTED WITH THE OPENSHIFT CLI

To use the OpenShift CLI (**oc**) tool, you must download and install it separately from your MicroShift installation.

2.1. INSTALLING THE OPENSHIFT CLI

You can install the OpenShift CLI (**oc**) either by downloading the binary or by using Homebrew.

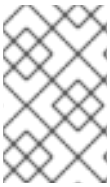
2.1.1. Installing the OpenShift CLI by downloading the binary

You can install the OpenShift CLI (**oc**) to interact with Red Hat build of MicroShift from a command-line interface. You can install **oc** on Linux, Windows, or macOS.

IMPORTANT

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in Red Hat build of MicroShift 4.15. Download and install the new version of **oc**.

Installing the OpenShift CLI on Linux
You can install the OpenShift CLI (**oc**) binary on Linux by using the following procedure.



NOTE

Red Hat build of MicroShift version numbering matches OpenShift Container Platform version numbering. Use the **oc** binary that matches your MicroShift version and has the appropriate RHEL compatibility.

Procedure

1. Navigate to the OpenShift Container Platform downloads page on the Red Hat Customer Portal.
2. Select the architecture from the **Product Variant** drop-down list.
3. Select the appropriate version from the **Version** drop-down list.
4. Click **Download Now** next to the **OpenShift v4.15 Linux Client** entry and save the file.
5. Unpack the archive:

```
$ tar xvf <file>
```

6. Place the **oc** binary in a directory that is on your **PATH**.
   To check your **PATH**, execute the following command:

```
$ echo $PATH
```

Verification

- After you install the OpenShift CLI, it is available using the **oc** command:

```
$ oc <command>
```

Installing the OpenShift CLI on Windows

You can install the OpenShift CLI (**oc**) binary on Windows by using the following procedure.

NOTE

Red Hat build of MicroShift version numbering matches OpenShift Container Platform version numbering. Use the **oc** binary that matches your MicroShift version and has the appropriate RHEL compatibility.

Procedure

1. Navigate to the OpenShift Container Platform downloads page on the Red Hat Customer Portal.
2. Select the appropriate version from the **Version** drop-down list.
3. Click **Download Now** next to the **OpenShift v4.15 Windows Client** entry and save the file.
4. Unzip the archive with a ZIP program.
5. Move the **oc** binary to a directory that is on your **PATH**.
   To check your **PATH**, open the command prompt and execute the following command:

```
C:\> path
```

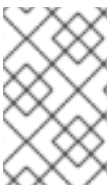
Verification

- After you install the OpenShift CLI, it is available using the **oc** command:

```
C:\> oc <command>
```

Installing the OpenShift CLI on macOS

You can install the OpenShift CLI (**oc**) binary on macOS by using the following procedure.



NOTE

Red Hat build of MicroShift version numbering matches OpenShift Container Platform version numbering. Use the **oc** binary that matches your MicroShift version and has the appropriate RHEL compatibility.

Procedure

1. Navigate to the OpenShift Container Platform downloads page on the Red Hat Customer Portal.
2. Select the appropriate version from the **Version** drop-down list.
3. Click **Download Now** next to the **OpenShift v4.15 macOS Client** entry and save the file.
4. Unpack and unzip the archive.
5. Move the **oc** binary to a directory on your PATH.

To check your **PATH**, open a terminal and execute the following command:

```
$ echo $PATH
```

Verification

- After you install the OpenShift CLI, it is available using the **oc** command:

```
$ oc <command>
```

2.1.2. Installing the OpenShift CLI by using Homebrew

For macOS, you can install the OpenShift CLI (**oc**) by using the Homebrew package manager.

Prerequisites

- You must have Homebrew (**brew**) installed.

Procedure

- Run the following command to install the openshift-cli package:

```
$ brew install openshift-cli
```

2.1.3. Installing the OpenShift CLI by using an RPM

For Red Hat Enterprise Linux (RHEL), you can install the OpenShift CLI (**oc**) as an RPM if you have an active Red Hat build of MicroShift subscription on your Red Hat account.

NOTE

It is not supported to install the OpenShift CLI (**oc**) as an RPM for Red Hat Enterprise Linux (RHEL) 9. You must install the OpenShift CLI for RHEL 9 by downloading the binary.

Prerequisites

- Must have root or sudo privileges.

Procedure

1. Register with Red Hat Subscription Manager:

```
# subscription-manager register
```

2. Pull the latest subscription data:

```
# subscription-manager refresh
```

3. List the available subscriptions:

```
# subscription-manager list --available --matches '*OpenShift*'
```

4. In the output for the previous command, find the pool ID for an Red Hat build of MicroShift subscription and attach the subscription to the registered system:

```
# subscription-manager attach --pool=<pool_id>
```

5. Enable the repositories required by Red Hat build of MicroShift 4.15.

```
# subscription-manager repos --enable="rhocp-4.15-for-rhel-8-x86_64-rpms"
```

6. Install the **openshift-clients** package:

```
# yum install openshift-clients
```

After you install the CLI, it is available using the **oc** command:

```
$ oc <command>
```

CHAPTER 3. CONFIGURING THE OPENSHIFT CLI

Configure **oc** based on your preferences for working with it.

3.1. ENABLING TAB COMPLETION

You can enable tab completion for the Bash or Zsh shells.

3.1.1. Enabling tab completion for Bash

After you install the OpenShift CLI (**oc**), you can enable tab completion to automatically complete **oc** commands or suggest options when you press Tab. The following procedure enables tab completion for the Bash shell.

Prerequisites

- You must have the OpenShift CLI (**oc**) installed.
- You must have the package **bash-completion** installed.

Procedure

1. Save the Bash completion code to a file:

```
$ oc completion bash > oc_bash_completion
```

2. Copy the file to **/etc/bash_completion.d/**:

```
$ sudo cp oc_bash_completion /etc/bash_completion.d/
```

   You can also save the file to a local directory and source it from your **.bashrc** file instead.

Tab completion is enabled when you open a new terminal.

3.1.2. Enabling tab completion for Zsh

After you install the OpenShift CLI (**oc**), you can enable tab completion to automatically complete **oc** commands or suggest options when you press Tab. The following procedure enables tab completion for the Zsh shell.

Prerequisites

- You must have the OpenShift CLI (**oc**) installed.

Procedure

- To add tab completion for **oc** to your **.zshrc** file, run the following command:

```
$ cat >>~/.zshrc<<EOF
if [ $commands[oc] ]; then
  source <(oc completion zsh)
```

```
  compdef _oc oc
fi
EOF
```

Tab completion is enabled when you open a new terminal.

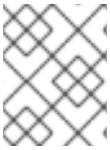
CHAPTER 4. USING THE OC TOOL

The optional OpenShift CLI (**oc**) tool provides a subset of **oc** commands for Red Hat build of MicroShift deployments. Using **oc** is convenient if you are familiar with OpenShift Container Platform and Kubernetes.

4.1. ABOUT THE OPENSHIFT CLI

With the OpenShift command-line interface (CLI), the **oc** command, you can deploy and manage MicroShift projects from a terminal. The CLI **oc** tool is ideal in the following situations:

- Working directly with project source code
- Scripting Red Hat build of MicroShift operations
- Managing projects while restricted by bandwidth resources



NOTE

A **kubeconfig** file must exist for the cluster to be accessible. The values are applied from built-in default values or a **config.yaml**, if one was created.

4.2. USING THE OPENSHIFT CLI IN RED HAT BUILD OF MICROSHIFT

Review the following sections to learn how to complete common tasks in Red Hat build of MicroShift using the **oc** CLI.

4.2.1. Viewing pods

Use the **oc get pods** command to view the pods for the current project.

NOTE

When you run **oc** inside a pod and do not specify a namespace, the namespace of the pod is used by default.

```
$ oc get pods -o wide
```

Example output

```
NAME                READY   STATUS      RESTARTS   AGE     IP          NODE
NOMINATED NODE
cakephp-ex-1-build  0/1     Completed   0          5m45s   10.131.0.10   ip-10-0-141-74.ec2.internal
<none>
cakephp-ex-1-deploy   0/1     Completed   0          3m44s   10.129.2.9    ip-10-0-147-65.ec2.internal
<none>
cakephp-ex-1-ktz97    1/1     Running     0          3m33s   10.128.2.11   ip-10-0-168-105.ec2.internal
<none>
```

4.2.2. Viewing pod logs

Use the **oc logs** command to view logs for a particular pod.

```
$ oc logs cakephp-ex-1-deploy
```

Example output

```
--> Scaling cakephp-ex-1 to 1
--> Success
```

4.2.3. Listing supported API resources

Use the **oc api-resources** command to view the list of supported API resources on the server.

```
$ oc api-resources
```

Example output

```
NAME                              SHORTNAMES   APIGROUP                       NAMESPACED   KIND
bindings                                                                      true         Binding
componentstatuses                 cs                                          false        ComponentStatus
configmaps                        cm                                          true         ConfigMap
...
```

4.3. GETTING HELP

You can get help with CLI commands and MicroShift resources in the following ways.

- Use **oc help --flag** to get information about a specific CLI command:

  Example: Get help for the oc create command

  ```
  $ oc create --help
  ```

  Example output

  ```
  Create a resource by filename or stdin

  JSON and YAML formats are accepted.

  Usage:
    oc create -f FILENAME [flags]

  ...
  ```

- Use the **oc explain** command to view the description and fields for a particular resource:

  Example: View documentation for the Pod resource

  ```
  $ oc explain pods
  ```

  Example output

```
KIND:     Pod
VERSION:  v1

DESCRIPTION:
     Pod is a collection of containers that can run on a host. This resource is
     created by clients and scheduled onto hosts.

FIELDS:
   apiVersion <string>
     APIVersion defines the versioned schema of this representation of an
     object. Servers should convert recognized schemas to the latest internal
     value, and may reject unrecognized values. More info:
     https://git.k8s.io/community/contributors/devel/api-conventions.md#resources

...
```

4.4. OC COMMAND ERRORS IN RED HAT BUILD OF MICROSHIFT

Not all OpenShift CLI (oc) tool commands are relevant for Red Hat build of MicroShift deployments. When you use **oc** to make a request call against an unsupported API, the **oc** binary usually generates an error message about a resource that cannot be found.

Example output

For example, when the following **new-project** command is run:

```
$ oc new-project test
```

The following error message can be generated:

```
Error from server (NotFound): the server could not find the requested resource (get
projectrequests.project.openshift.io)
```

And when the **get projects** command is run, another error can be generated as follows:

```
$ oc get projects
error: the server doesn't have a resource type "projects"
```

CHAPTER 5. USING OC AND KUBECTL COMMANDS

The Kubernetes command-line interface (CLI), **kubectl**, can be used to run commands against a Kubernetes cluster. Because Red Hat build of MicroShift is a certified Kubernetes distribution, you can use the supported **kubectl** CLI tool that ships with Red Hat build of MicroShift, or you can gain extended functionality by using the **oc** CLI tool.

5.1. THE KUBECTL CLI TOOL

You can use the **kubectl** CLI tool to interact with Kubernetes primitives on your Red Hat build of MicroShift cluster. You can also use existing **kubectl** workflows and scripts for new Red Hat build of MicroShift users coming from another Kubernetes environment, or for those who prefer to use the **kubectl** CLI.

The **kubectl** CLI tool is included in the archive if you download the **oc** CLI tool.

For more information, read the Kubernetes CLI tool documentation.

5.2. THE OC CLI TOOL

The **oc** CLI tool offers the same capabilities as the **kubectl** CLI tool, but it extends to natively support additional Red Hat build of MicroShift features, including:

- **Route resource**
  The **Route** resource object is specific to Red Hat build of MicroShift distributions, and builds upon standard Kubernetes primitives.

- **Additional commands**
  The additional command **oc new-app**, for example, makes it easier to get new applications started using existing source code or pre-built images.

IMPORTANT

If you installed an earlier version of the **oc** CLI tool, you cannot use it to complete all of the commands in Red Hat build of MicroShift 4.15. If you want the latest features, you must download and install the latest version of the **oc** CLI tool corresponding to your Red Hat build of MicroShift version.

Non-security API changes will involve, at minimum, two minor releases (4.1 to 4.2 to 4.3, for example) to allow older **oc** binaries to update. Using new capabilities might require newer **oc** binaries. A 4.3 server might have additional capabilities that a 4.2 **oc** binary cannot use and a 4.3 **oc** binary might have additional capabilities that are unsupported by a 4.2 server.

Table 5.1. Compatibility Matrix

	X.Y (**oc** Client)	**X.Y+N** footnote:versionpolicyn[Where **N** is a number greater than or equal to 1.] (**oc** Client)
X.Y (Server)	1	3

X.Y+N footnote:versionpolicyn[] (Server)	2	1

1 Fully compatible.

2 **oc** client might not be able to access server features.

3 **oc** client might provide options and features that might not be compatible with the accessed server.

CHAPTER 6. OPENSHIFT CLI COMMAND REFERENCE

Descriptions and example commands for OpenShift CLI (**oc**) commands are included in this reference document. You must have **cluster-admin** or equivalent permissions to use these commands. To list administrator commands and information about them, use the following commands:

- Enter the **oc adm -h** command to list all administrator commands:

  Command syntax

  ```
  $ oc adm -h
  ```

- Enter the **oc <command> --help** command to get additional details for a specific command:

  Command syntax

  ```
  $ oc <command> --help
  ```

IMPORTANT

Using **oc <command> --help** lists details for any **oc** command. Not all **oc** commands apply to using Red Hat build of MicroShift.

6.1. OPENSHIFT CLI (OC) DEVELOPER COMMANDS

6.1.1. oc annotate

Update the annotations on a resource

Example usage

```
# Update pod 'foo' with the annotation 'description' and the value 'my frontend'
# If the same annotation is set multiple times, only the last value will be applied
oc annotate pods foo description='my frontend'

# Update a pod identified by type and name in "pod.json"
oc annotate -f pod.json description='my frontend'

# Update pod 'foo' with the annotation 'description' and the value 'my frontend running nginx',
overwriting any existing value
oc annotate --overwrite pods foo description='my frontend running nginx'

# Update all pods in the namespace
oc annotate pods --all description='my frontend running nginx'

# Update pod 'foo' only if the resource is unchanged from version 1
oc annotate pods foo description='my frontend running nginx' --resource-version=1

# Update pod 'foo' by removing an annotation named 'description' if it exists
# Does not require the --overwrite flag
oc annotate pods foo description-
```

6.1.2. oc api-resources

Print the supported API resources on the server

Example usage

```
# Print the supported API resources
oc api-resources

# Print the supported API resources with more information
oc api-resources -o wide

# Print the supported API resources sorted by a column
oc api-resources --sort-by=name

# Print the supported namespaced resources
oc api-resources --namespaced=true

# Print the supported non-namespaced resources
oc api-resources --namespaced=false

# Print the supported API resources with a specific APIGroup
oc api-resources --api-group=rbac.authorization.k8s.io
```

6.1.3. oc api-versions

Print the supported API versions on the server, in the form of "group/version"

Example usage

```
# Print the supported API versions
oc api-versions
```

6.1.4. oc apply

Apply a configuration to a resource by file name or stdin

Example usage

```
# Apply the configuration in pod.json to a pod
oc apply -f ./pod.json

# Apply resources from a directory containing kustomization.yaml - e.g. dir/kustomization.yaml
oc apply -k dir/

# Apply the JSON passed into stdin to a pod
cat pod.json | oc apply -f -

# Apply the configuration from all files that end with '.json' - i.e. expand wildcard characters in file
names
oc apply -f '*.json'

# Note: --prune is still in Alpha
# Apply the configuration in manifest.yaml that matches label app=nginx and delete all other
```

```
# resources that are not in the file and match label app=nginx
oc apply --prune -f manifest.yaml -l app=nginx

# Apply the configuration in manifest.yaml and delete all the other config maps that are not in the file
oc apply --prune -f manifest.yaml --all --prune-allowlist=core/v1/ConfigMap
```

6.1.5. oc apply edit-last-applied

Edit latest last-applied-configuration annotations of a resource/object

Example usage

```
# Edit the last-applied-configuration annotations by type/name in YAML
oc apply edit-last-applied deployment/nginx

# Edit the last-applied-configuration annotations by file in JSON
oc apply edit-last-applied -f deploy.yaml -o json
```

6.1.6. oc apply set-last-applied

Set the last-applied-configuration annotation on a live object to match the contents of a file

Example usage

```
# Set the last-applied-configuration of a resource to match the contents of a file
oc apply set-last-applied -f deploy.yaml

# Execute set-last-applied against each configuration file in a directory
oc apply set-last-applied -f path/

# Set the last-applied-configuration of a resource to match the contents of a file; will create the annotation if it does not already exist
oc apply set-last-applied -f deploy.yaml --create-annotation=true
```

6.1.7. oc apply view-last-applied

View the latest last-applied-configuration annotations of a resource/object

Example usage

```
# View the last-applied-configuration annotations by type/name in YAML
oc apply view-last-applied deployment/nginx

# View the last-applied-configuration annotations by file in JSON
oc apply view-last-applied -f deploy.yaml -o json
```

6.1.8. oc attach

Attach to a running container

Example usage

```
# Get output from running pod mypod; use the 'oc.kubernetes.io/default-container' annotation
# for selecting the container to be attached or the first container in the pod will be chosen
oc attach mypod

# Get output from ruby-container from pod mypod
oc attach mypod -c ruby-container

# Switch to raw terminal mode; sends stdin to 'bash' in ruby-container from pod mypod
# and sends stdout/stderr from 'bash' back to the client
oc attach mypod -c ruby-container -i -t

# Get output from the first pod of a replica set named nginx
oc attach rs/nginx
```

6.1.9. oc auth can-i

Check whether an action is allowed

Example usage

```
# Check to see if I can create pods in any namespace
oc auth can-i create pods --all-namespaces

# Check to see if I can list deployments in my current namespace
oc auth can-i list deployments.apps

# Check to see if service account "foo" of namespace "dev" can list pods
# in the namespace "prod".
# You must be allowed to use impersonation for the global option "--as".
oc auth can-i list pods --as=system:serviceaccount:dev:foo -n prod

# Check to see if I can do everything in my current namespace ("*" means all)
oc auth can-i '*' '*'

# Check to see if I can get the job named "bar" in namespace "foo"
oc auth can-i list jobs.batch/bar -n foo

# Check to see if I can read pod logs
oc auth can-i get pods --subresource=log

# Check to see if I can access the URL /logs/
oc auth can-i get /logs/

# List all allowed actions in namespace "foo"
oc auth can-i --list --namespace=foo
```

6.1.10. oc auth reconcile

Reconciles rules for RBAC role, role binding, cluster role, and cluster role binding objects

Example usage

```
# Reconcile RBAC resources from a file
oc auth reconcile -f my-rbac-rules.yaml
```

6.1.11. oc auth whoami

Experimental: Check self subject attributes

Example usage

```
# Get your subject attributes.
oc auth whoami

# Get your subject attributes in JSON format.
oc auth whoami -o json
```

6.1.12. oc cluster-info

Display cluster information

Example usage

```
# Print the address of the control plane and cluster services
oc cluster-info
```

6.1.13. oc cluster-info dump

Dump relevant information for debugging and diagnosis

Example usage

```
# Dump current cluster state to stdout
oc cluster-info dump

# Dump current cluster state to /path/to/cluster-state
oc cluster-info dump --output-directory=/path/to/cluster-state

# Dump all namespaces to stdout
oc cluster-info dump --all-namespaces

# Dump a set of namespaces to /path/to/cluster-state
oc cluster-info dump --namespaces default,kube-system --output-directory=/path/to/cluster-state
```

6.1.14. oc completion

Output shell completion code for the specified shell (bash, zsh, fish, or powershell)

Example usage

```
# Installing bash completion on macOS using homebrew
## If running Bash 3.2 included with macOS
brew install bash-completion
## or, if running Bash 4.1+
brew install bash-completion@2
## If oc is installed via homebrew, this should start working immediately
## If you've installed via other means, you may need add the completion to your completion directory
oc completion bash > $(brew --prefix)/etc/bash_completion.d/oc
```

```
# Installing bash completion on Linux
## If bash-completion is not installed on Linux, install the 'bash-completion' package
## via your distribution's package manager.
## Load the oc completion code for bash into the current shell
source <(oc completion bash)
## Write bash completion code to a file and source it from .bash_profile
oc completion bash > ~/.kube/completion.bash.inc
printf "
# Kubectl shell completion
source '$HOME/.kube/completion.bash.inc'
" >> $HOME/.bash_profile
source $HOME/.bash_profile

# Load the oc completion code for zsh[1] into the current shell
source <(oc completion zsh)
# Set the oc completion code for zsh[1] to autoload on startup
oc completion zsh > "${fpath[1]}/_oc"


# Load the oc completion code for fish[2] into the current shell
oc completion fish | source
# To load completions for each session, execute once:
oc completion fish > ~/.config/fish/completions/oc.fish

# Load the oc completion code for powershell into the current shell
oc completion powershell | Out-String | Invoke-Expression
# Set oc completion code for powershell to run on startup
## Save completion code to a script and execute in the profile
oc completion powershell > $HOME\.kube\completion.ps1
Add-Content $PROFILE "$HOME\.kube\completion.ps1"
## Execute completion code in the profile
Add-Content $PROFILE "if (Get-Command oc -ErrorAction SilentlyContinue) {
oc completion powershell | Out-String | Invoke-Expression
}"
## Add completion code directly to the $PROFILE script
oc completion powershell >> $PROFILE
```

6.1.15. oc config current-context

Display the current-context

Example usage

```
# Display the current-context
oc config current-context
```

6.1.16. oc config delete-cluster

Delete the specified cluster from the kubeconfig

Example usage

```
# Delete the minikube cluster
oc config delete-cluster minikube
```

6.1.17. oc config delete-context

Delete the specified context from the kubeconfig

Example usage

```
# Delete the context for the minikube cluster
oc config delete-context minikube
```

6.1.18. oc config delete-user

Delete the specified user from the kubeconfig

Example usage

```
# Delete the minikube user
oc config delete-user minikube
```

6.1.19. oc config get-clusters

Display clusters defined in the kubeconfig

Example usage

```
# List the clusters that oc knows about
oc config get-clusters
```

6.1.20. oc config get-contexts

Describe one or many contexts

Example usage

```
# List all the contexts in your kubeconfig file
oc config get-contexts

# Describe one context in your kubeconfig file
oc config get-contexts my-context
```

6.1.21. oc config get-users

Display users defined in the kubeconfig

Example usage

```
# List the users that oc knows about
oc config get-users
```

6.1.22. oc config new-admin-kubeconfig

Generate, make the server trust, and display a new admin.kubeconfig.

Example usage

```
# Generate a new admin kubeconfig
oc config new-admin-kubeconfig
```

6.1.23. oc config new-kubelet-bootstrap-kubeconfig

Generate, make the server trust, and display a new kubelet /etc/kubernetes/kubeconfig.

Example usage

```
# Generate a new kubelet bootstrap kubeconfig
oc config new-kubelet-bootstrap-kubeconfig
```

6.1.24. oc config refresh-ca-bundle

Update the OpenShift CA bundle by contacting the apiserver.

Example usage

```
# Refresh the CA bundle for the current context's cluster
oc config refresh-ca-bundle

# Refresh the CA bundle for the cluster named e2e in your kubeconfig
oc config refresh-ca-bundle e2e

# Print the CA bundle from the current OpenShift cluster's apiserver.
oc config refresh-ca-bundle --dry-run
```

6.1.25. oc config rename-context

Rename a context from the kubeconfig file

Example usage

```
# Rename the context 'old-name' to 'new-name' in your kubeconfig file
oc config rename-context old-name new-name
```

6.1.26. oc config set

Set an individual value in a kubeconfig file

Example usage

```
# Set the server field on the my-cluster cluster to https://1.2.3.4
oc config set clusters.my-cluster.server https://1.2.3.4

# Set the certificate-authority-data field on the my-cluster cluster
```

```
oc config set clusters.my-cluster.certificate-authority-data $(echo "cert_data_here" | base64 -i -)

# Set the cluster field in the my-context context to my-cluster
oc config set contexts.my-context.cluster my-cluster

# Set the client-key-data field in the cluster-admin user using --set-raw-bytes option
oc config set users.cluster-admin.client-key-data cert_data_here --set-raw-bytes=true
```

6.1.27. oc config set-cluster

Set a cluster entry in kubeconfig

Example usage

```
# Set only the server field on the e2e cluster entry without touching other values
oc config set-cluster e2e --server=https://1.2.3.4

# Embed certificate authority data for the e2e cluster entry
oc config set-cluster e2e --embed-certs --certificate-authority=~/.kube/e2e/kubernetes.ca.crt

# Disable cert checking for the e2e cluster entry
oc config set-cluster e2e --insecure-skip-tls-verify=true

# Set custom TLS server name to use for validation for the e2e cluster entry
oc config set-cluster e2e --tls-server-name=my-cluster-name

# Set proxy url for the e2e cluster entry
oc config set-cluster e2e --proxy-url=https://1.2.3.4
```

6.1.28. oc config set-context

Set a context entry in kubeconfig

Example usage

```
# Set the user field on the gce context entry without touching other values
oc config set-context gce --user=cluster-admin
```

6.1.29. oc config set-credentials

Set a user entry in kubeconfig

Example usage

```
# Set only the "client-key" field on the "cluster-admin"
# entry, without touching other values
oc config set-credentials cluster-admin --client-key=~/.kube/admin.key

# Set basic auth for the "cluster-admin" entry
oc config set-credentials cluster-admin --username=admin --password=uXFGweU9l35qcif

# Embed client certificate data in the "cluster-admin" entry
oc config set-credentials cluster-admin --client-certificate=~/.kube/admin.crt --embed-certs=true
```

```
  # Enable the Google Compute Platform auth provider for the "cluster-admin" entry
  oc config set-credentials cluster-admin --auth-provider=gcp

  # Enable the OpenID Connect auth provider for the "cluster-admin" entry with additional args
  oc config set-credentials cluster-admin --auth-provider=oidc --auth-provider-arg=client-id=foo --auth-provider-arg=client-secret=bar

  # Remove the "client-secret" config value for the OpenID Connect auth provider for the "cluster-admin" entry
  oc config set-credentials cluster-admin --auth-provider=oidc --auth-provider-arg=client-secret-

  # Enable new exec auth plugin for the "cluster-admin" entry
  oc config set-credentials cluster-admin --exec-command=/path/to/the/executable --exec-api-version=client.authentication.k8s.io/v1beta1

  # Define new exec auth plugin args for the "cluster-admin" entry
  oc config set-credentials cluster-admin --exec-arg=arg1 --exec-arg=arg2

  # Create or update exec auth plugin environment variables for the "cluster-admin" entry
  oc config set-credentials cluster-admin --exec-env=key1=val1 --exec-env=key2=val2

  # Remove exec auth plugin environment variables for the "cluster-admin" entry
  oc config set-credentials cluster-admin --exec-env=var-to-remove-
```

6.1.30. oc config unset

Unset an individual value in a kubeconfig file

Example usage

```
  # Unset the current-context
  oc config unset current-context

  # Unset namespace in foo context
  oc config unset contexts.foo.namespace
```

6.1.31. oc config use-context

Set the current-context in a kubeconfig file

Example usage

```
  # Use the context for the minikube cluster
  oc config use-context minikube
```

6.1.32. oc config view

Display merged kubeconfig settings or a specified kubeconfig file

Example usage

```
  # Show merged kubeconfig settings
```

```
oc config view

# Show merged kubeconfig settings and raw certificate data and exposed secrets
oc config view --raw

# Get the password for the e2e user
oc config view -o jsonpath='{.users[?(@.name == "e2e")].user.password}'
```

6.1.33. oc cp

Copy files and directories to and from containers

Example usage

```
# !!!Important Note!!!
# Requires that the 'tar' binary is present in your container
# image.  If 'tar' is not present, 'oc cp' will fail.
#
# For advanced use cases, such as symlinks, wildcard expansion or
# file mode preservation, consider using 'oc exec'.

# Copy /tmp/foo local file to /tmp/bar in a remote pod in namespace <some-namespace>
tar cf - /tmp/foo | oc exec -i -n <some-namespace> <some-pod> -- tar xf - -C /tmp/bar

# Copy /tmp/foo from a remote pod to /tmp/bar locally
oc exec -n <some-namespace> <some-pod> -- tar cf - /tmp/foo | tar xf - -C /tmp/bar

# Copy /tmp/foo_dir local directory to /tmp/bar_dir in a remote pod in the default namespace
oc cp /tmp/foo_dir <some-pod>:/tmp/bar_dir

# Copy /tmp/foo local file to /tmp/bar in a remote pod in a specific container
oc cp /tmp/foo <some-pod>:/tmp/bar -c <specific-container>

# Copy /tmp/foo local file to /tmp/bar in a remote pod in namespace <some-namespace>
oc cp /tmp/foo <some-namespace>/<some-pod>:/tmp/bar

# Copy /tmp/foo from a remote pod to /tmp/bar locally
oc cp <some-namespace>/<some-pod>:/tmp/foo /tmp/bar
```

6.1.34. oc create

Create a resource from a file or from stdin

Example usage

```
# Create a pod using the data in pod.json
oc create -f ./pod.json

# Create a pod based on the JSON passed into stdin
cat pod.json | oc create -f -

# Edit the data in registry.yaml in JSON then create the resource using the edited data
oc create -f registry.yaml --edit -o json
```

6.1.35. oc create clusterrole

Create a cluster role

Example usage

```
  # Create a cluster role named "pod-reader" that allows user to perform "get", "watch" and "list" on pods
  oc create clusterrole pod-reader --verb=get,list,watch --resource=pods

  # Create a cluster role named "pod-reader" with ResourceName specified
  oc create clusterrole pod-reader --verb=get --resource=pods --resource-name=readablepod --resource-name=anotherpod

  # Create a cluster role named "foo" with API Group specified
  oc create clusterrole foo --verb=get,list,watch --resource=rs.apps

  # Create a cluster role named "foo" with SubResource specified
  oc create clusterrole foo --verb=get,list,watch --resource=pods,pods/status

  # Create a cluster role name "foo" with NonResourceURL specified
  oc create clusterrole "foo" --verb=get --non-resource-url=/logs/*

  # Create a cluster role name "monitoring" with AggregationRule specified
  oc create clusterrole monitoring --aggregation-rule="rbac.example.com/aggregate-to-monitoring=true"
```

6.1.36. oc create clusterrolebinding

Create a cluster role binding for a particular cluster role

Example usage

```
  # Create a cluster role binding for user1, user2, and group1 using the cluster-admin cluster role
  oc create clusterrolebinding cluster-admin --clusterrole=cluster-admin --user=user1 --user=user2 --group=group1
```

6.1.37. oc create configmap

Create a config map from a local file, directory or literal value

Example usage

```
  # Create a new config map named my-config based on folder bar
  oc create configmap my-config --from-file=path/to/bar

  # Create a new config map named my-config with specified keys instead of file basenames on disk
  oc create configmap my-config --from-file=key1=/path/to/bar/file1.txt --from-file=key2=/path/to/bar/file2.txt

  # Create a new config map named my-config with key1=config1 and key2=config2
  oc create configmap my-config --from-literal=key1=config1 --from-literal=key2=config2

  # Create a new config map named my-config from the key=value pairs in the file
```

```
oc create configmap my-config --from-file=path/to/bar

# Create a new config map named my-config from an env file
oc create configmap my-config --from-env-file=path/to/foo.env --from-env-file=path/to/bar.env
```

6.1.38. oc create cronjob

Create a cron job with the specified name

Example usage

```
# Create a cron job
oc create cronjob my-job --image=busybox --schedule="*/1 * * * *"

# Create a cron job with a command
oc create cronjob my-job --image=busybox --schedule="*/1 * * * *" -- date
```

6.1.39. oc create deployment

Create a deployment with the specified name

Example usage

```
# Create a deployment named my-dep that runs the busybox image
oc create deployment my-dep --image=busybox

# Create a deployment with a command
oc create deployment my-dep --image=busybox -- date

# Create a deployment named my-dep that runs the nginx image with 3 replicas
oc create deployment my-dep --image=nginx --replicas=3

# Create a deployment named my-dep that runs the busybox image and expose port 5701
oc create deployment my-dep --image=busybox --port=5701
```

6.1.40. oc create ingress

Create an ingress with the specified name

Example usage

```
# Create a single ingress called 'simple' that directs requests to foo.com/bar to svc
# svc1:8080 with a tls secret "my-cert"
oc create ingress simple --rule="foo.com/bar=svc1:8080,tls=my-cert"

# Create a catch all ingress of "/path" pointing to service svc:port and Ingress Class as
"otheringress"
oc create ingress catch-all --class=otheringress --rule="/path=svc:port"

# Create an ingress with two annotations: ingress.annotation1 and ingress.annotations2
oc create ingress annotated --class=default --rule="foo.com/bar=svc:port" \
--annotation ingress.annotation1=foo \
--annotation ingress.annotation2=bla
```

```
# Create an ingress with the same host and multiple paths
oc create ingress multipath --class=default \
--rule="foo.com/=svc:port" \
--rule="foo.com/admin/=svcadmin:portadmin"

# Create an ingress with multiple hosts and the pathType as Prefix
oc create ingress ingress1 --class=default \
--rule="foo.com/path*=svc:8080" \
--rule="bar.com/admin*=svc2:http"

# Create an ingress with TLS enabled using the default ingress certificate and different path types
oc create ingress ingtls --class=default \
--rule="foo.com/=svc:https,tls" \
--rule="foo.com/path/subpath*=othersvc:8080"

# Create an ingress with TLS enabled using a specific secret and pathType as Prefix
oc create ingress ingsecret --class=default \
--rule="foo.com/*=svc:8080,tls=secret1"

# Create an ingress with a default backend
oc create ingress ingdefault --class=default \
--default-backend=defaultsvc:http \
--rule="foo.com/*=svc:8080,tls=secret1"
```

6.1.41. oc create job

Create a job with the specified name

Example usage

```
# Create a job
oc create job my-job --image=busybox

# Create a job with a command
oc create job my-job --image=busybox -- date

# Create a job from a cron job named "a-cronjob"
oc create job test-job --from=cronjob/a-cronjob
```

6.1.42. oc create namespace

Create a namespace with the specified name

Example usage

```
# Create a new namespace named my-namespace
oc create namespace my-namespace
```

6.1.43. oc create poddisruptionbudget

Create a pod disruption budget with the specified name

Example usage

```
  # Create a pod disruption budget named my-pdb that will select all pods with the app=rails label
  # and require at least one of them being available at any point in time
  oc create poddisruptionbudget my-pdb --selector=app=rails --min-available=1

  # Create a pod disruption budget named my-pdb that will select all pods with the app=nginx label
  # and require at least half of the pods selected to be available at any point in time
  oc create pdb my-pdb --selector=app=nginx --min-available=50%
```

6.1.44. oc create priorityclass

Create a priority class with the specified name

Example usage

```
  # Create a priority class named high-priority
  oc create priorityclass high-priority --value=1000 --description="high priority"

  # Create a priority class named default-priority that is considered as the global default priority
  oc create priorityclass default-priority --value=1000 --global-default=true --description="default
priority"

  # Create a priority class named high-priority that cannot preempt pods with lower priority
  oc create priorityclass high-priority --value=1000 --description="high priority" --preemption-
policy="Never"
```

6.1.45. oc create quota

Create a quota with the specified name

Example usage

```
  # Create a new resource quota named my-quota
  oc create quota my-quota --
hard=cpu=1,memory=1G,pods=2,services=3,replicationcontrollers=2,resourcequotas=1,secrets=5,persis
tentvolumeclaims=10

  # Create a new resource quota named best-effort
  oc create quota best-effort --hard=pods=100 --scopes=BestEffort
```

6.1.46. oc create role

Create a role with single rule

Example usage

```
  # Create a role named "pod-reader" that allows user to perform "get", "watch" and "list" on pods
  oc create role pod-reader --verb=get --verb=list --verb=watch --resource=pods

  # Create a role named "pod-reader" with ResourceName specified
  oc create role pod-reader --verb=get --resource=pods --resource-name=readablepod --resource-
name=anotherpod
```

```
# Create a role named "foo" with API Group specified
oc create role foo --verb=get,list,watch --resource=rs.apps

# Create a role named "foo" with SubResource specified
oc create role foo --verb=get,list,watch --resource=pods,pods/status
```

6.1.47. oc create rolebinding

Create a role binding for a particular role or cluster role

Example usage

```
# Create a role binding for user1, user2, and group1 using the admin cluster role
oc create rolebinding admin --clusterrole=admin --user=user1 --user=user2 --group=group1

# Create a role binding for serviceaccount monitoring:sa-dev using the admin role
oc create rolebinding admin-binding --role=admin --serviceaccount=monitoring:sa-dev
```

6.1.48. oc create route edge

Create a route that uses edge TLS termination

Example usage

```
# Create an edge route named "my-route" that exposes the frontend service
oc create route edge my-route --service=frontend

# Create an edge route that exposes the frontend service and specify a path
# If the route name is omitted, the service name will be used
oc create route edge --service=frontend --path /assets
```

6.1.49. oc create route passthrough

Create a route that uses passthrough TLS termination

Example usage

```
# Create a passthrough route named "my-route" that exposes the frontend service
oc create route passthrough my-route --service=frontend

# Create a passthrough route that exposes the frontend service and specify
# a host name. If the route name is omitted, the service name will be used
oc create route passthrough --service=frontend --hostname=www.example.com
```

6.1.50. oc create route reencrypt

Create a route that uses reencrypt TLS termination

Example usage

```
# Create a route named "my-route" that exposes the frontend service
```

```
  oc create route reencrypt my-route --service=frontend --dest-ca-cert cert.cert

  # Create a reencrypt route that exposes the frontend service, letting the
  # route name default to the service name and the destination CA certificate
  # default to the service CA
  oc create route reencrypt --service=frontend
```

6.1.51. oc create secret docker-registry

Create a secret for use with a Docker registry

Example usage

```
  # If you don't already have a .dockercfg file, you can create a dockercfg secret directly by using:
  oc create secret docker-registry my-secret --docker-server=DOCKER_REGISTRY_SERVER --
docker-username=DOCKER_USER --docker-password=DOCKER_PASSWORD --docker-
email=DOCKER_EMAIL

  # Create a new secret named my-secret from ~/.docker/config.json
  oc create secret docker-registry my-secret --from-file=.dockerconfigjson=path/to/.docker/config.json
```

6.1.52. oc create secret generic

Create a secret from a local file, directory, or literal value

Example usage

```
  # Create a new secret named my-secret with keys for each file in folder bar
  oc create secret generic my-secret --from-file=path/to/bar

  # Create a new secret named my-secret with specified keys instead of names on disk
  oc create secret generic my-secret --from-file=ssh-privatekey=path/to/id_rsa --from-file=ssh-
publickey=path/to/id_rsa.pub

  # Create a new secret named my-secret with key1=supersecret and key2=topsecret
  oc create secret generic my-secret --from-literal=key1=supersecret --from-literal=key2=topsecret

  # Create a new secret named my-secret using a combination of a file and a literal
  oc create secret generic my-secret --from-file=ssh-privatekey=path/to/id_rsa --from-
literal=passphrase=topsecret

  # Create a new secret named my-secret from env files
  oc create secret generic my-secret --from-env-file=path/to/foo.env --from-env-file=path/to/bar.env
```

6.1.53. oc create secret tls

Create a TLS secret

Example usage

```
  # Create a new TLS secret named tls-secret with the given key pair
  oc create secret tls tls-secret --cert=path/to/tls.cert --key=path/to/tls.key
```

6.1.54. oc create service clusterip

Create a ClusterIP service

Example usage

```
# Create a new ClusterIP service named my-cs
oc create service clusterip my-cs --tcp=5678:8080

# Create a new ClusterIP service named my-cs (in headless mode)
oc create service clusterip my-cs --clusterip="None"
```

6.1.55. oc create service externalname

Create an ExternalName service

Example usage

```
# Create a new ExternalName service named my-ns
oc create service externalname my-ns --external-name bar.com
```

6.1.56. oc create service loadbalancer

Create a LoadBalancer service

Example usage

```
# Create a new LoadBalancer service named my-lbs
oc create service loadbalancer my-lbs --tcp=5678:8080
```

6.1.57. oc create service nodeport

Create a NodePort service

Example usage

```
# Create a new NodePort service named my-ns
oc create service nodeport my-ns --tcp=5678:8080
```

6.1.58. oc create serviceaccount

Create a service account with the specified name

Example usage

```
# Create a new service account named my-service-account
oc create serviceaccount my-service-account
```

6.1.59. oc create token

Request a service account token

Example usage

```
  # Request a token to authenticate to the kube-apiserver as the service account "myapp" in the current namespace
  oc create token myapp

  # Request a token for a service account in a custom namespace
  oc create token myapp --namespace myns

  # Request a token with a custom expiration
  oc create token myapp --duration 10m

  # Request a token with a custom audience
  oc create token myapp --audience https://example.com

  # Request a token bound to an instance of a Secret object
  oc create token myapp --bound-object-kind Secret --bound-object-name mysecret

  # Request a token bound to an instance of a Secret object with a specific uid
  oc create token myapp --bound-object-kind Secret --bound-object-name mysecret --bound-object-uid 0d4691ed-659b-4935-a832-355f77ee47cc
```

6.1.60. oc debug

Launch a new instance of a pod for debugging

Example usage

```
  # Start a shell session into a pod using the OpenShift tools image
  oc debug

  # Debug a currently running deployment by creating a new pod
  oc debug deploy/test

  # Debug a node as an administrator
  oc debug node/master-1

  # Launch a shell in a pod using the provided image stream tag
  oc debug istag/mysql:latest -n openshift

  # Test running a job as a non-root user
  oc debug job/test --as-user=1000000

  # Debug a specific failing container by running the env command in the 'second' container
  oc debug daemonset/test -c second -- /bin/env

  # See the pod that would be created to debug
  oc debug mypod-9xbc -o yaml

  # Debug a resource but launch the debug pod in another namespace
  # Note: Not all resources can be debugged using --to-namespace without modification. For example,
  # volumes and service accounts are namespace-dependent. Add '-o yaml' to output the debug pod
```

```
definition
  # to disk.  If necessary, edit the definition then run 'oc debug -f -' or run without --to-namespace
  oc debug mypod-9xbc --to-namespace testns
```

6.1.61. oc delete

Delete resources by file names, stdin, resources and names, or by resources and label selector

Example usage

```
  # Delete a pod using the type and name specified in pod.json
  oc delete -f ./pod.json

  # Delete resources from a directory containing kustomization.yaml - e.g. dir/kustomization.yaml
  oc delete -k dir

  # Delete resources from all files that end with '.json' - i.e. expand wildcard characters in file names
  oc delete -f '*.json'

  # Delete a pod based on the type and name in the JSON passed into stdin
  cat pod.json | oc delete -f -

  # Delete pods and services with same names "baz" and "foo"
  oc delete pod,service baz foo

  # Delete pods and services with label name=myLabel
  oc delete pods,services -l name=myLabel

  # Delete a pod with minimal delay
  oc delete pod foo --now

  # Force delete a pod on a dead node
  oc delete pod foo --force

  # Delete all pods
  oc delete pods --all
```

6.1.62. oc describe

Show details of a specific resource or group of resources

Example usage

```
  # Describe a node
  oc describe nodes kubernetes-node-emt8.c.myproject.internal

  # Describe a pod
  oc describe pods/nginx

  # Describe a pod identified by type and name in "pod.json"
  oc describe -f pod.json

  # Describe all pods
  oc describe pods
```

```
# Describe pods by label name=myLabel
oc describe po -l name=myLabel

# Describe all pods managed by the 'frontend' replication controller
# (rc-created pods get the name of the rc as a prefix in the pod name)
oc describe pods frontend
```

6.1.63. oc diff

Diff the live version against a would-be applied version

Example usage

```
# Diff resources included in pod.json
oc diff -f pod.json

# Diff file read from stdin
cat service.yaml | oc diff -f -
```

6.1.64. oc edit

Edit a resource on the server

Example usage

```
# Edit the service named 'registry'
oc edit svc/registry

# Use an alternative editor
KUBE_EDITOR="nano" oc edit svc/registry

# Edit the job 'myjob' in JSON using the v1 API format
oc edit job.v1.batch/myjob -o json

# Edit the deployment 'mydeployment' in YAML and save the modified config in its annotation
oc edit deployment/mydeployment -o yaml --save-config

# Edit the deployment/mydeployment's status subresource
oc edit deployment mydeployment --subresource='status'
```

6.1.65. oc events

List events

Example usage

```
# List recent events in the default namespace.
oc events

# List recent events in all namespaces.
oc events --all-namespaces
```

```
# List recent events for the specified pod, then wait for more events and list them as they arrive.
oc events --for pod/web-pod-13je7 --watch

# List recent events in given format. Supported ones, apart from default, are json and yaml.
oc events -oyaml

# List recent only events in given event types
oc events --types=Warning,Normal
```

6.1.66. oc exec

Execute a command in a container

Example usage

```
# Get output from running the 'date' command from pod mypod, using the first container by default
oc exec mypod -- date

# Get output from running the 'date' command in ruby-container from pod mypod
oc exec mypod -c ruby-container -- date

# Switch to raw terminal mode; sends stdin to 'bash' in ruby-container from pod mypod
# and sends stdout/stderr from 'bash' back to the client
oc exec mypod -c ruby-container -i -t -- bash -il

# List contents of /usr from the first container of pod mypod and sort by modification time
# If the command you want to execute in the pod has any flags in common (e.g. -i),
# you must use two dashes (--) to separate your command's flags/arguments
# Also note, do not surround your command and its flags/arguments with quotes
# unless that is how you would execute it normally (i.e., do ls -t /usr, not "ls -t /usr")
oc exec mypod -i -t -- ls -t /usr

# Get output from running 'date' command from the first pod of the deployment mydeployment,
using the first container by default
oc exec deploy/mydeployment -- date

# Get output from running 'date' command from the first pod of the service myservice, using the first
container by default
oc exec svc/myservice -- date
```

6.1.67. oc explain

Get documentation for a resource

Example usage

```
# Get the documentation of the resource and its fields
oc explain pods

# Get the documentation of a specific field of a resource
oc explain pods.spec.containers
```

6.1.68. oc expose

Expose a replicated application as a service or route

Example usage

```
# Create a route based on service nginx. The new route will reuse nginx's labels
oc expose service nginx

# Create a route and specify your own label and route name
oc expose service nginx -l name=myroute --name=fromdowntown

# Create a route and specify a host name
oc expose service nginx --hostname=www.example.com

# Create a route with a wildcard
oc expose service nginx --hostname=x.example.com --wildcard-policy=Subdomain
# This would be equivalent to *.example.com. NOTE: only hosts are matched by the wildcard;
subdomains would not be included

# Expose a deployment configuration as a service and use the specified port
oc expose dc ruby-hello-world --port=8080

# Expose a service as a route in the specified path
oc expose service nginx --path=/nginx
```

6.1.69. oc extract

Extract secrets or config maps to disk

Example usage

```
# Extract the secret "test" to the current directory
oc extract secret/test

# Extract the config map "nginx" to the /tmp directory
oc extract configmap/nginx --to=/tmp

# Extract the config map "nginx" to STDOUT
oc extract configmap/nginx --to=-

# Extract only the key "nginx.conf" from config map "nginx" to the /tmp directory
oc extract configmap/nginx --to=/tmp --keys=nginx.conf
```

6.1.70. oc get

Display one or many resources

Example usage

```
# List all pods in ps output format
oc get pods

# List all pods in ps output format with more information (such as node name)
oc get pods -o wide
```

```
# List a single replication controller with specified NAME in ps output format
oc get replicationcontroller web

# List deployments in JSON output format, in the "v1" version of the "apps" API group
oc get deployments.v1.apps -o json

# List a single pod in JSON output format
oc get -o json pod web-pod-13je7

# List a pod identified by type and name specified in "pod.yaml" in JSON output format
oc get -f pod.yaml -o json

# List resources from a directory with kustomization.yaml - e.g. dir/kustomization.yaml
oc get -k dir/

# Return only the phase value of the specified pod
oc get -o template pod/web-pod-13je7 --template={{.status.phase}}

# List resource information in custom columns
oc get pod test-pod -o custom-
columns=CONTAINER:.spec.containers[0].name,IMAGE:.spec.containers[0].image

# List all replication controllers and services together in ps output format
oc get rc,services

# List one or more resources by their type and names
oc get rc/web service/frontend pods/web-pod-13je7

# List status subresource for a single pod.
oc get pod web-pod-13je7 --subresource status
```

6.1.71. oc image append

Add layers to images and push them to a registry

Example usage

```
# Remove the entrypoint on the mysql:latest image
oc image append --from mysql:latest --to myregistry.com/myimage:latest --image '{"Entrypoint":null}'

# Add a new layer to the image
oc image append --from mysql:latest --to myregistry.com/myimage:latest layer.tar.gz

# Add a new layer to the image and store the result on disk
# This results in $(pwd)/v2/mysql/blobs,manifests
oc image append --from mysql:latest --to file://mysql:local layer.tar.gz

# Add a new layer to the image and store the result on disk in a designated directory
# This will result in $(pwd)/mysql-local/v2/mysql/blobs,manifests
oc image append --from mysql:latest --to file://mysql:local --dir mysql-local layer.tar.gz

# Add a new layer to an image that is stored on disk (~/mysql-local/v2/image exists)
oc image append --from-dir ~/mysql-local --to myregistry.com/myimage:latest layer.tar.gz

# Add a new layer to an image that was mirrored to the current directory on disk ($(pwd)/v2/image
```

```
exists)
  oc image append --from-dir v2 --to myregistry.com/myimage:latest layer.tar.gz

  # Add a new layer to a multi-architecture image for an os/arch that is different from the system's
os/arch
  # Note: The first image in the manifest list that matches the filter will be returned when --keep-
manifest-list is specified
  oc image append --from docker.io/library/busybox:latest --filter-by-os=linux/s390x --to
myregistry.com/myimage:latest layer.tar.gz

  # Add a new layer to a multi-architecture image for all the os/arch manifests when keep-manifest-list
is specified
  oc image append --from docker.io/library/busybox:latest --keep-manifest-list --to
myregistry.com/myimage:latest layer.tar.gz

  # Add a new layer to a multi-architecture image for all the os/arch manifests that is specified by the
filter, while preserving the manifestlist
  oc image append --from docker.io/library/busybox:latest --filter-by-os=linux/s390x --keep-manifest-
list --to myregistry.com/myimage:latest layer.tar.gz
```

6.1.72. oc image extract

Copy files from an image to the file system

Example usage

```
  # Extract the busybox image into the current directory
  oc image extract docker.io/library/busybox:latest

  # Extract the busybox image into a designated directory (must exist)
  oc image extract docker.io/library/busybox:latest --path /:/tmp/busybox

  # Extract the busybox image into the current directory for linux/s390x platform
  # Note: Wildcard filter is not supported with extract; pass a single os/arch to extract
  oc image extract docker.io/library/busybox:latest --filter-by-os=linux/s390x

  # Extract a single file from the image into the current directory
  oc image extract docker.io/library/centos:7 --path /bin/bash:.

  # Extract all .repo files from the image's /etc/yum.repos.d/ folder into the current directory
  oc image extract docker.io/library/centos:7 --path /etc/yum.repos.d/*.repo:.

  # Extract all .repo files from the image's /etc/yum.repos.d/ folder into a designated directory (must
exist)
  # This results in /tmp/yum.repos.d/*.repo on local system
  oc image extract docker.io/library/centos:7 --path /etc/yum.repos.d/*.repo:/tmp/yum.repos.d

  # Extract an image stored on disk into the current directory ($(pwd)/v2/busybox/blobs,manifests
exists)
  # --confirm is required because the current directory is not empty
  oc image extract file://busybox:local --confirm

  # Extract an image stored on disk in a directory other than $(pwd)/v2 into the current directory
  # --confirm is required because the current directory is not empty ($(pwd)/busybox-mirror-
dir/v2/busybox exists)
```

```
  oc image extract file://busybox:local --dir busybox-mirror-dir --confirm

  # Extract an image stored on disk in a directory other than $(pwd)/v2 into a designated directory
(must exist)
  oc image extract file://busybox:local --dir busybox-mirror-dir --path /:/tmp/busybox

  # Extract the last layer in the image
  oc image extract docker.io/library/centos:7[-1]

  # Extract the first three layers of the image
  oc image extract docker.io/library/centos:7[:3]

  # Extract the last three layers of the image
  oc image extract docker.io/library/centos:7[-3:]
```

6.1.73. oc image info

Display information about an image

Example usage

```
  # Show information about an image
  oc image info quay.io/openshift/cli:latest

  # Show information about images matching a wildcard
  oc image info quay.io/openshift/cli:4.*

  # Show information about a file mirrored to disk under DIR
  oc image info --dir=DIR file://library/busybox:latest

  # Select which image from a multi-OS image to show
  oc image info library/busybox:latest --filter-by-os=linux/arm64
```

6.1.74. oc image mirror

Mirror images from one repository to another

Example usage

```
  # Copy image to another tag
  oc image mirror myregistry.com/myimage:latest myregistry.com/myimage:stable

  # Copy image to another registry
  oc image mirror myregistry.com/myimage:latest docker.io/myrepository/myimage:stable

  # Copy all tags starting with mysql to the destination repository
  oc image mirror myregistry.com/myimage:mysql* docker.io/myrepository/myimage

  # Copy image to disk, creating a directory structure that can be served as a registry
  oc image mirror myregistry.com/myimage:latest file://myrepository/myimage:latest

  # Copy image to S3 (pull from <bucket>.s3.amazonaws.com/image:latest)
  oc image mirror myregistry.com/myimage:latest
s3://s3.amazonaws.com/<region>/<bucket>/image:latest
```

```
  # Copy image to S3 without setting a tag (pull via @<digest>)
  oc image mirror myregistry.com/myimage:latest s3://s3.amazonaws.com/<region>/<bucket>/image

  # Copy image to multiple locations
  oc image mirror myregistry.com/myimage:latest docker.io/myrepository/myimage:stable \
  docker.io/myrepository/myimage:dev

  # Copy multiple images
  oc image mirror myregistry.com/myimage:latest=myregistry.com/other:test \
  myregistry.com/myimage:new=myregistry.com/other:target

  # Copy manifest list of a multi-architecture image, even if only a single image is found
  oc image mirror myregistry.com/myimage:latest=myregistry.com/other:test \
  --keep-manifest-list=true

  # Copy specific os/arch manifest of a multi-architecture image
  # Run 'oc image info myregistry.com/myimage:latest' to see available os/arch for multi-arch images
  # Note that with multi-arch images, this results in a new manifest list digest that includes only
  # the filtered manifests
  oc image mirror myregistry.com/myimage:latest=myregistry.com/other:test \
  --filter-by-os=os/arch

  # Copy all os/arch manifests of a multi-architecture image
  # Run 'oc image info myregistry.com/myimage:latest' to see list of os/arch manifests that will be
mirrored
  oc image mirror myregistry.com/myimage:latest=myregistry.com/other:test \
  --keep-manifest-list=true

  # Note the above command is equivalent to
  oc image mirror myregistry.com/myimage:latest=myregistry.com/other:test \
  --filter-by-os=.*

  # Copy specific os/arch manifest of a multi-architecture image
  # Run 'oc image info myregistry.com/myimage:latest' to see available os/arch for multi-arch images
  # Note that the target registry may reject a manifest list if the platform specific images do not all
  # exist. You must use a registry with sparse registry support enabled.
  oc image mirror myregistry.com/myimage:latest=myregistry.com/other:test \
  --filter-by-os=os/arch \
  --keep-manifest-list=true
```

6.1.75. oc kustomize

Build a kustomization target from a directory or URL

Example usage

```
  # Build the current working directory
  oc kustomize

  # Build some shared configuration directory
  oc kustomize /home/config/production

  # Build from github
  oc kustomize https://github.com/kubernetes-sigs/kustomize.git/examples/helloWorld?ref=v1.0.6
```

6.1.76. oc label

Update the labels on a resource

Example usage

```
# Update pod 'foo' with the label 'unhealthy' and the value 'true'
oc label pods foo unhealthy=true

# Update pod 'foo' with the label 'status' and the value 'unhealthy', overwriting any existing value
oc label --overwrite pods foo status=unhealthy

# Update all pods in the namespace
oc label pods --all status=unhealthy

# Update a pod identified by the type and name in "pod.json"
oc label -f pod.json status=unhealthy

# Update pod 'foo' only if the resource is unchanged from version 1
oc label pods foo status=unhealthy --resource-version=1

# Update pod 'foo' by removing a label named 'bar' if it exists
# Does not require the --overwrite flag
oc label pods foo bar-
```

6.1.77. oc logs

Print the logs for a container in a pod

Example usage

```
# Start streaming the logs of the most recent build of the openldap build config
oc logs -f bc/openldap

# Start streaming the logs of the latest deployment of the mysql deployment config
oc logs -f dc/mysql

# Get the logs of the first deployment for the mysql deployment config. Note that logs
# from older deployments may not exist either because the deployment was successful
# or due to deployment pruning or manual deletion of the deployment
oc logs --version=1 dc/mysql

# Return a snapshot of ruby-container logs from pod backend
oc logs backend -c ruby-container

# Start streaming of ruby-container logs from pod backend
oc logs -f pod/backend -c ruby-container
```

6.1.78. oc observe

Observe changes to resources and react to them (experimental)

Example usage

```
# Observe changes to services
oc observe services

# Observe changes to services, including the clusterIP and invoke a script for each
oc observe services --template '{ .spec.clusterIP }' -- register_dns.sh

# Observe changes to services filtered by a label selector
oc observe services -l regist-dns=true --template '{ .spec.clusterIP }' -- register_dns.sh
```

6.1.79. oc patch

Update fields of a resource

Example usage

```
# Partially update a node using a strategic merge patch, specifying the patch as JSON
oc patch node k8s-node-1 -p '{"spec":{"unschedulable":true}}'

# Partially update a node using a strategic merge patch, specifying the patch as YAML
oc patch node k8s-node-1 -p $'spec:\n unschedulable: true'

# Partially update a node identified by the type and name specified in "node.json" using strategic merge patch
oc patch -f node.json -p '{"spec":{"unschedulable":true}}'

# Update a container's image; spec.containers[*].name is required because it's a merge key
oc patch pod valid-pod -p '{"spec":{"containers":[{"name":"kubernetes-serve-hostname","image":"new image"}]}}'

# Update a container's image using a JSON patch with positional arrays
oc patch pod valid-pod --type='json' -p='[{"op": "replace", "path": "/spec/containers/0/image", "value":"new image"}]'

# Update a deployment's replicas through the scale subresource using a merge patch.
oc patch deployment nginx-deployment --subresource='scale' --type='merge' -p '{"spec":{"replicas":2}}'
```

6.1.80. oc plugin list

List all visible plugin executables on a user's PATH

Example usage

```
# List all available plugins
oc plugin list
```

6.1.81. oc policy add-role-to-user

Add a role to users or service accounts for the current project

Example usage

```
# Add the 'view' role to user1 for the current project
```

```
oc policy add-role-to-user view user1

# Add the 'edit' role to serviceaccount1 for the current project
oc policy add-role-to-user edit -z serviceaccount1
```

6.1.82. oc policy scc-review

Check which service account can create a pod

Example usage

```
# Check whether service accounts sa1 and sa2 can admit a pod with a template pod spec specified in my_resource.yaml
# Service Account specified in myresource.yaml file is ignored
oc policy scc-review -z sa1,sa2 -f my_resource.yaml

# Check whether service accounts system:serviceaccount:bob:default can admit a pod with a template pod spec specified in my_resource.yaml
oc policy scc-review -z system:serviceaccount:bob:default -f my_resource.yaml

# Check whether the service account specified in my_resource_with_sa.yaml can admit the pod
oc policy scc-review -f my_resource_with_sa.yaml

# Check whether the default service account can admit the pod; default is taken since no service account is defined in myresource_with_no_sa.yaml
oc policy scc-review -f myresource_with_no_sa.yaml
```

6.1.83. oc policy scc-subject-review

Check whether a user or a service account can create a pod

Example usage

```
# Check whether user bob can create a pod specified in myresource.yaml
oc policy scc-subject-review -u bob -f myresource.yaml

# Check whether user bob who belongs to projectAdmin group can create a pod specified in myresource.yaml
oc policy scc-subject-review -u bob -g projectAdmin -f myresource.yaml

# Check whether a service account specified in the pod template spec in myresourcewithsa.yaml can create the pod
oc policy scc-subject-review -f myresourcewithsa.yaml
```

6.1.84. oc port-forward

Forward one or more local ports to a pod

Example usage

```
# Listen on ports 5000 and 6000 locally, forwarding data to/from ports 5000 and 6000 in the pod
oc port-forward pod/mypod 5000 6000
```

```
  # Listen on ports 5000 and 6000 locally, forwarding data to/from ports 5000 and 6000 in a pod
selected by the deployment
  oc port-forward deployment/mydeployment 5000 6000

  # Listen on port 8443 locally, forwarding to the targetPort of the service's port named "https" in a pod
selected by the service
  oc port-forward service/myservice 8443:https

  # Listen on port 8888 locally, forwarding to 5000 in the pod
  oc port-forward pod/mypod 8888:5000

  # Listen on port 8888 on all addresses, forwarding to 5000 in the pod
  oc port-forward --address 0.0.0.0 pod/mypod 8888:5000

  # Listen on port 8888 on localhost and selected IP, forwarding to 5000 in the pod
  oc port-forward --address localhost,10.19.21.23 pod/mypod 8888:5000

  # Listen on a random port locally, forwarding to 5000 in the pod
  oc port-forward pod/mypod :5000
```

6.1.85. oc proxy

Run a proxy to the Kubernetes API server

Example usage

```
  # To proxy all of the Kubernetes API and nothing else
  oc proxy --api-prefix=/

  # To proxy only part of the Kubernetes API and also some static files
  # You can get pods info with 'curl localhost:8001/api/v1/pods'
  oc proxy --www=/my/files --www-prefix=/static/ --api-prefix=/api/

  # To proxy the entire Kubernetes API at a different root
  # You can get pods info with 'curl localhost:8001/custom/api/v1/pods'
  oc proxy --api-prefix=/custom/

  # Run a proxy to the Kubernetes API server on port 8011, serving static content from ./local/www/
  oc proxy --port=8011 --www=./local/www/

  # Run a proxy to the Kubernetes API server on an arbitrary local port
  # The chosen port for the server will be output to stdout
  oc proxy --port=0

  # Run a proxy to the Kubernetes API server, changing the API prefix to k8s-api
  # This makes e.g. the pods API available at localhost:8001/k8s-api/v1/pods/
  oc proxy --api-prefix=/k8s-api
```

6.1.86. oc rollback

Revert part of an application back to a previous deployment

Example usage

```
# Perform a rollback to the last successfully completed deployment for a deployment config
oc rollback frontend

# See what a rollback to version 3 will look like, but do not perform the rollback
oc rollback frontend --to-version=3 --dry-run

# Perform a rollback to a specific deployment
oc rollback frontend-2

# Perform the rollback manually by piping the JSON of the new config back to oc
oc rollback frontend -o json | oc replace dc/frontend -f -

# Print the updated deployment configuration in JSON format instead of performing the rollback
oc rollback frontend -o json
```

6.1.87. oc rollout cancel

Cancel the in-progress deployment

Example usage

```
# Cancel the in-progress deployment based on 'nginx'
oc rollout cancel dc/nginx
```

6.1.88. oc rollout history

View rollout history

Example usage

```
# View the rollout history of a deployment
oc rollout history dc/nginx

# View the details of deployment revision 3
oc rollout history dc/nginx --revision=3
```

6.1.89. oc rollout latest

Start a new rollout for a deployment config with the latest state from its triggers

Example usage

```
# Start a new rollout based on the latest images defined in the image change triggers
oc rollout latest dc/nginx

# Print the rolled out deployment config
oc rollout latest dc/nginx -o json
```

6.1.90. oc rollout pause

Mark the provided resource as paused

Example usage

```
# Mark the nginx deployment as paused. Any current state of
# the deployment will continue its function, new updates to the deployment will not
# have an effect as long as the deployment is paused
oc rollout pause dc/nginx
```

6.1.91. oc rollout restart

Restart a resource

Example usage

```
# Restart a deployment
oc rollout restart deployment/nginx

# Restart a daemon set
oc rollout restart daemonset/abc

# Restart deployments with the app=nginx label
oc rollout restart deployment --selector=app=nginx
```

6.1.92. oc rollout resume

Resume a paused resource

Example usage

```
# Resume an already paused deployment
oc rollout resume dc/nginx
```

6.1.93. oc rollout retry

Retry the latest failed rollout

Example usage

```
# Retry the latest failed deployment based on 'frontend'
# The deployer pod and any hook pods are deleted for the latest failed deployment
oc rollout retry dc/frontend
```

6.1.94. oc rollout status

Show the status of the rollout

Example usage

```
# Watch the status of the latest rollout
oc rollout status dc/nginx
```

6.1.95. oc rollout undo

Undo a previous rollout

Example usage

```
# Roll back to the previous deployment
oc rollout undo dc/nginx

# Roll back to deployment revision 3. The replication controller for that version must exist
oc rollout undo dc/nginx --to-revision=3
```

6.1.96. oc rsh

Start a shell session in a container

Example usage

```
# Open a shell session on the first container in pod 'foo'
oc rsh foo

# Open a shell session on the first container in pod 'foo' and namespace 'bar'
# (Note that oc client specific arguments must come before the resource name and its arguments)
oc rsh -n bar foo

# Run the command 'cat /etc/resolv.conf' inside pod 'foo'
oc rsh foo cat /etc/resolv.conf

# See the configuration of your internal registry
oc rsh dc/docker-registry cat config.yml

# Open a shell session on the container named 'index' inside a pod of your job
oc rsh -c index job/scheduled
```

6.1.97. oc rsync

Copy files between a local file system and a pod

Example usage

```
# Synchronize a local directory with a pod directory
oc rsync ./local/dir/ POD:/remote/dir

# Synchronize a pod directory with a local directory
oc rsync POD:/remote/dir/ ./local/dir
```

6.1.98. oc run

Run a particular image on the cluster

Example usage

```
# Start a nginx pod
oc run nginx --image=nginx
```

```
  # Start a hazelcast pod and let the container expose port 5701
  oc run hazelcast --image=hazelcast/hazelcast --port=5701

  # Start a hazelcast pod and set environment variables "DNS_DOMAIN=cluster" and
"POD_NAMESPACE=default" in the container
  oc run hazelcast --image=hazelcast/hazelcast --env="DNS_DOMAIN=cluster" --
env="POD_NAMESPACE=default"

  # Start a hazelcast pod and set labels "app=hazelcast" and "env=prod" in the container
  oc run hazelcast --image=hazelcast/hazelcast --labels="app=hazelcast,env=prod"

  # Dry run; print the corresponding API objects without creating them
  oc run nginx --image=nginx --dry-run=client

  # Start a nginx pod, but overload the spec with a partial set of values parsed from JSON
  oc run nginx --image=nginx --overrides='{ "apiVersion": "v1", "spec": { ... } }'

  # Start a busybox pod and keep it in the foreground, don't restart it if it exits
  oc run -i -t busybox --image=busybox --restart=Never

  # Start the nginx pod using the default command, but use custom arguments (arg1 .. argN) for that
command
  oc run nginx --image=nginx -- <arg1> <arg2> ... <argN>

  # Start the nginx pod using a different command and custom arguments
  oc run nginx --image=nginx --command -- <cmd> <arg1> ... <argN>
```

6.1.99. oc scale

Set a new size for a deployment, replica set, or replication controller

Example usage

```
  # Scale a replica set named 'foo' to 3
  oc scale --replicas=3 rs/foo

  # Scale a resource identified by type and name specified in "foo.yaml" to 3
  oc scale --replicas=3 -f foo.yaml

  # If the deployment named mysql's current size is 2, scale mysql to 3
  oc scale --current-replicas=2 --replicas=3 deployment/mysql

  # Scale multiple replication controllers
  oc scale --replicas=5 rc/foo rc/bar rc/baz

  # Scale stateful set named 'web' to 3
  oc scale --replicas=3 statefulset/web
```

6.1.100. oc secrets link

Link secrets to a service account

Example usage

```
# Add an image pull secret to a service account to automatically use it for pulling pod images
oc secrets link serviceaccount-name pull-secret --for=pull

# Add an image pull secret to a service account to automatically use it for both pulling and pushing build images
oc secrets link builder builder-image-secret --for=pull,mount
```

6.1.101. oc secrets unlink

Detach secrets from a service account

Example usage

```
# Unlink a secret currently associated with a service account
oc secrets unlink serviceaccount-name secret-name another-secret-name ...
```

6.1.102. oc set data

Update the data within a config map or secret

Example usage

```
# Set the 'password' key of a secret
oc set data secret/foo password=this_is_secret

# Remove the 'password' key from a secret
oc set data secret/foo password-

# Update the 'haproxy.conf' key of a config map from a file on disk
oc set data configmap/bar --from-file=../haproxy.conf

# Update a secret with the contents of a directory, one key per file
oc set data secret/foo --from-file=secret-dir
```

6.1.103. oc set env

Update environment variables on a pod template

Example usage

```
# Update deployment config 'myapp' with a new environment variable
oc set env dc/myapp STORAGE_DIR=/local

# List the environment variables defined on a build config 'sample-build'
oc set env bc/sample-build --list

# List the environment variables defined on all pods
oc set env pods --all --list

# Output modified build config in YAML
oc set env bc/sample-build STORAGE_DIR=/data -o yaml

# Update all containers in all replication controllers in the project to have ENV=prod
```

```
oc set env rc --all ENV=prod

# Import environment from a secret
oc set env --from=secret/mysecret dc/myapp

# Import environment from a config map with a prefix
oc set env --from=configmap/myconfigmap --prefix=MYSQL_ dc/myapp

# Remove the environment variable ENV from container 'c1' in all deployment configs
oc set env dc --all --containers="c1" ENV-

# Remove the environment variable ENV from a deployment config definition on disk and
# update the deployment config on the server
oc set env -f dc.json ENV-

# Set some of the local shell environment into a deployment config on the server
oc set env | grep RAILS_ | oc env -e - dc/myapp
```

6.1.104. oc set image

Update the image of a pod template

Example usage

```
  # Set a deployment config's nginx container image to 'nginx:1.9.1', and its busybox container image
to 'busybox'.
  oc set image dc/nginx busybox=busybox nginx=nginx:1.9.1

  # Set a deployment config's app container image to the image referenced by the imagestream tag
'openshift/ruby:2.3'.
  oc set image dc/myapp app=openshift/ruby:2.3 --source=imagestreamtag

  # Update all deployments' and rc's nginx container's image to 'nginx:1.9.1'
  oc set image deployments,rc nginx=nginx:1.9.1 --all

  # Update image of all containers of daemonset abc to 'nginx:1.9.1'
  oc set image daemonset abc *=nginx:1.9.1

  # Print result (in YAML format) of updating nginx container image from local file, without hitting the
server
  oc set image -f path/to/file.yaml nginx=nginx:1.9.1 --local -o yaml
```

6.1.105. oc set image-lookup

Change how images are resolved when deploying applications

Example usage

```
  # Print all of the image streams and whether they resolve local names
  oc set image-lookup

  # Use local name lookup on image stream mysql
  oc set image-lookup mysql
```

```
# Force a deployment to use local name lookup
oc set image-lookup deploy/mysql

# Show the current status of the deployment lookup
oc set image-lookup deploy/mysql --list

# Disable local name lookup on image stream mysql
oc set image-lookup mysql --enabled=false

# Set local name lookup on all image streams
oc set image-lookup --all
```

6.1.106. oc set probe

Update a probe on a pod template

Example usage

```
# Clear both readiness and liveness probes off all containers
oc set probe dc/myapp --remove --readiness --liveness

# Set an exec action as a liveness probe to run 'echo ok'
oc set probe dc/myapp --liveness -- echo ok

# Set a readiness probe to try to open a TCP socket on 3306
oc set probe rc/mysql --readiness --open-tcp=3306

# Set an HTTP startup probe for port 8080 and path /healthz over HTTP on the pod IP
oc set probe dc/webapp --startup --get-url=http://:8080/healthz

# Set an HTTP readiness probe for port 8080 and path /healthz over HTTP on the pod IP
oc set probe dc/webapp --readiness --get-url=http://:8080/healthz

# Set an HTTP readiness probe over HTTPS on 127.0.0.1 for a hostNetwork pod
oc set probe dc/router --readiness --get-url=https://127.0.0.1:1936/stats

# Set only the initial-delay-seconds field on all deployments
oc set probe dc --all --readiness --initial-delay-seconds=30
```

6.1.107. oc set resources

Update resource requests/limits on objects with pod templates

Example usage

```
# Set a deployments nginx container CPU limits to "200m and memory to 512Mi"
oc set resources deployment nginx -c=nginx --limits=cpu=200m,memory=512Mi

# Set the resource request and limits for all containers in nginx
oc set resources deployment nginx --limits=cpu=200m,memory=512Mi --
requests=cpu=100m,memory=256Mi

# Remove the resource requests for resources on containers in nginx
oc set resources deployment nginx --limits=cpu=0,memory=0 --requests=cpu=0,memory=0
```

```
# Print the result (in YAML format) of updating nginx container limits locally, without hitting the server
oc set resources -f path/to/file.yaml --limits=cpu=200m,memory=512Mi --local -o yaml
```

6.1.108. oc set route-backends

Update the backends for a route

Example usage

```
# Print the backends on the route 'web'
oc set route-backends web

# Set two backend services on route 'web' with 2/3rds of traffic going to 'a'
oc set route-backends web a=2 b=1

# Increase the traffic percentage going to b by 10%% relative to a
oc set route-backends web --adjust b=+10%%

# Set traffic percentage going to b to 10%% of the traffic going to a
oc set route-backends web --adjust b=10%%

# Set weight of b to 10
oc set route-backends web --adjust b=10

# Set the weight to all backends to zero
oc set route-backends web --zero
```

6.1.109. oc set selector

Set the selector on a resource

Example usage

```
# Set the labels and selector before creating a deployment/service pair.
oc create service clusterip my-svc --clusterip="None" -o yaml --dry-run | oc set selector --local -f - 'environment=qa' -o yaml | oc create -f -
oc create deployment my-dep -o yaml --dry-run | oc label --local -f - environment=qa -o yaml | oc create -f -
```

6.1.110. oc set serviceaccount

Update the service account of a resource

Example usage

```
# Set deployment nginx-deployment's service account to serviceaccount1
oc set serviceaccount deployment nginx-deployment serviceaccount1

# Print the result (in YAML format) of updated nginx deployment with service account from a local file, without hitting the API server
oc set sa -f nginx-deployment.yaml serviceaccount1 --local --dry-run -o yaml
```

6.1.111. oc set subject

Update the user, group, or service account in a role binding or cluster role binding

Example usage

```
# Update a cluster role binding for serviceaccount1
oc set subject clusterrolebinding admin --serviceaccount=namespace:serviceaccount1

# Update a role binding for user1, user2, and group1
oc set subject rolebinding admin --user=user1 --user=user2 --group=group1

# Print the result (in YAML format) of updating role binding subjects locally, without hitting the server
oc create rolebinding admin --role=admin --user=admin -o yaml --dry-run | oc set subject --local -f -
--user=foo -o yaml
```

6.1.112. oc set volumes

Update volumes on a pod template

Example usage

```
# List volumes defined on all deployment configs in the current project
oc set volume dc --all

# Add a new empty dir volume to deployment config (dc) 'myapp' mounted under
# /var/lib/myapp
oc set volume dc/myapp --add --mount-path=/var/lib/myapp

# Use an existing persistent volume claim (PVC) to overwrite an existing volume 'v1'
oc set volume dc/myapp --add --name=v1 -t pvc --claim-name=pvc1 --overwrite

# Remove volume 'v1' from deployment config 'myapp'
oc set volume dc/myapp --remove --name=v1

# Create a new persistent volume claim that overwrites an existing volume 'v1'
oc set volume dc/myapp --add --name=v1 -t pvc --claim-size=1G --overwrite

# Change the mount point for volume 'v1' to /data
oc set volume dc/myapp --add --name=v1 -m /data --overwrite

# Modify the deployment config by removing volume mount "v1" from container "c1"
# (and by removing the volume "v1" if no other containers have volume mounts that reference it)
oc set volume dc/myapp --remove --name=v1 --containers=c1

# Add new volume based on a more complex volume source (AWS EBS, GCE PD,
# Ceph, Gluster, NFS, ISCSI, ...)
oc set volume dc/myapp --add -m /data --source=<json-string>
```

6.1.113. oc tag

Tag existing images into image streams

Example usage

```
  # Tag the current image for the image stream 'openshift/ruby' and tag '2.0' into the image stream
'yourproject/ruby with tag 'tip'
  oc tag openshift/ruby:2.0 yourproject/ruby:tip

  # Tag a specific image
  oc tag
openshift/ruby@sha256:6b646fa6bf5e5e4c7fa41056c27910e679c03ebe7f93e361e6515a9da7e258cc
yourproject/ruby:tip

  # Tag an external container image
  oc tag --source=docker openshift/origin-control-plane:latest yourproject/ruby:tip

  # Tag an external container image and request pullthrough for it
  oc tag --source=docker openshift/origin-control-plane:latest yourproject/ruby:tip --reference-
policy=local

  # Tag an external container image and include the full manifest list
  oc tag --source=docker openshift/origin-control-plane:latest yourproject/ruby:tip --import-
mode=PreserveOriginal

  # Remove the specified spec tag from an image stream
  oc tag openshift/origin-control-plane:latest -d
```

6.1.114. oc version

Print the client and server version information

Example usage

```
  # Print the OpenShift client, kube-apiserver, and openshift-apiserver version information for the
current context
  oc version

  # Print the OpenShift client, kube-apiserver, and openshift-apiserver version numbers for the current
context
  oc version --short

  # Print the OpenShift client version information for the current context
  oc version --client
```

6.1.115. oc wait

Experimental: Wait for a specific condition on one or many resources

Example usage

```
  # Wait for the pod "busybox1" to contain the status condition of type "Ready"
  oc wait --for=condition=Ready pod/busybox1

  # The default value of status condition is true; you can wait for other targets after an equal delimiter
(compared after Unicode simple case folding, which is a more general form of case-insensitivity):
  oc wait --for=condition=Ready=false pod/busybox1

  # Wait for the pod "busybox1" to contain the status phase to be "Running".
```

```
  oc wait --for=jsonpath='{.status.phase}'=Running pod/busybox1

  # Wait for the pod "busybox1" to be deleted, with a timeout of 60s, after having issued the "delete"
command
  oc delete pod/busybox1
  oc wait --for=delete pod/busybox1 --timeout=60s
```

6.2. OPENSHIFT CLI (OC) ADMINISTRATOR COMMANDS

6.2.1. oc adm inspect

Collect debugging data for a given resource

Example usage

```
  # Collect debugging data for a kubernetes service
  oc adm inspect service/kubernetes

  # Collect debugging data for a node
  oc adm inspect node/<node_name>

  # Collect debugging data for logicalvolumes in a CRD
  oc adm inspect crd/logicalvolumes.topolvm.io

  # Collect debugging data for routes.route.openshift.io in a CRD
  oc adm inspect crd/routes.route.openshift.io
```

6.2.2. oc adm release extract

Extract the contents of an update payload to disk

Example usage

```
  # Use git to check out the source code for the current cluster release to DIR
  oc adm release extract --git=DIR

  # Extract cloud credential requests for AWS
  oc adm release extract --credentials-requests --cloud=aws

  # Use git to check out the source code for the current cluster release to DIR from linux/s390x image
  # Note: Wildcard filter is not supported; pass a single os/arch to extract
  oc adm release extract --git=DIR quay.io/openshift-release-dev/ocp-release:4.11.2 --filter-by-
os=linux/s390x
```

6.2.3. oc adm release info

Display information about a release

Example usage

```
  # Show information about the cluster's current release
  oc adm release info
```

```
# Show the source code that comprises a release
oc adm release info 4.11.2 --commit-urls

# Show the source code difference between two releases
oc adm release info 4.11.0 4.11.2 --commits

# Show where the images referenced by the release are located
oc adm release info quay.io/openshift-release-dev/ocp-release:4.11.2 --pullspecs

# Show information about linux/s390x image
# Note: Wildcard filter is not supported; pass a single os/arch to extract
oc adm release info quay.io/openshift-release-dev/ocp-release:4.11.2 --filter-by-os=linux/s390x
```

6.2.4. oc adm taint

Update the taints on nodes

Example usage

```
# Update node 'foo' with a taint with key 'dedicated' and value 'special-user' and effect 'NoSchedule'
# If a taint with that key and effect already exists, its value is replaced as specified
oc adm taint nodes foo dedicated=special-user:NoSchedule

# Remove from node 'foo' the taint with key 'dedicated' and effect 'NoSchedule' if one exists
oc adm taint nodes foo dedicated:NoSchedule-

# Remove from node 'foo' all the taints with key 'dedicated'
oc adm taint nodes foo dedicated-

# Add a taint with key 'dedicated' on nodes having label mylabel=X
oc adm taint node -l myLabel=X  dedicated=foo:PreferNoSchedule

# Add to node 'foo' a taint with key 'bar' and no value
oc adm taint nodes foo bar:NoSchedule
```