OpenShift Container Platform 4.1

CLI reference

Learning how to use the OpenShift CLI

Last Updated: 2020-02-28

OpenShift Container Platform 4.1 CLI reference

Learning how to use the OpenShift CLI

Legal Notice

Abstract

This document provides information about installing, configuring, and using the OpenShift CLI (`oc`). It also contains a reference of CLI commands and examples of how to use them.

Table of Contents

CHAPTER 1. GETTING STARTED WITH THE CLI

1.1. ABOUT THE CLI

With the OpenShift Container Platform command-line interface (CLI), you can create applications and manage OpenShift Container Platform projects from a terminal. The CLI is ideal in situations where you:

- Work directly with project source code.
- Script OpenShift Container Platform operations.
- Are restricted by bandwidth resources and can not use the web console.

1.2. INSTALLING THE CLI

You can install the CLI in order to interact with OpenShift Container Platform using a command-line interface.

Procedure

1. From the Infrastructure Provider page on the Red Hat OpenShift Cluster Manager site, navigate to the page for your installation type and click **Download Command-line Tools**.
2. Click the folder for your operating system and architecture and click the compressed file.
3. Save the file to your file system.
4. Extract the compressed file.
5. Place it in a directory that is on your **PATH**.

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

```
$ oc <command>
```

1.3. LOGGING IN TO THE CLI

You can log in to the **oc** CLI to access and manage your cluster.

Prerequisites

- You must have access to an OpenShift Container Platform cluster.
- You must have installed the CLI.

Procedure

- Log in to the CLI using the **oc login** command and enter the required information when prompted.

```
$ oc login
Server [https://localhost:8443]: https://openshift.example.com:6443 1
The server uses a certificate signed by an unknown authority.
```

```
You can bypass the certificate check, but any data you send to the server could be
intercepted by others.
Use insecure connections? (y/n): y (2)

Authentication required for https://openshift.example.com:6443 (openshift)
Username: user1 (3)
Password: (4)
Login successful.

You don't have any projects. You can try to create a new project, by running

    oc new-project <projectname>

Welcome! See 'oc help' to get started.
```

1. Enter the OpenShift Container Platform server URL.
2. Enter whether to use insecure connections.
3. Enter the user name to log in as.
4. Enter the user's password.

You can now create a project or issue other commands for managing your cluster.

1.4. USING THE CLI

Review the following sections to learn how to complete common tasks using the CLI.

1.4.1. Creating a project

Use the **oc new-project** command to create a new project.

```
$ oc new-project my-project
Now using project "my-project" on server "https://openshift.example.com:6443".
```

1.4.2. Creating a new app

Use the **oc new-app** command to create a new application.

```
$ oc new-app https://github.com/sclorg/cakephp-ex
--> Found image 40de956 (9 days old) in imagestream "openshift/php" under tag "7.2" for "php"

...

    Run 'oc status' to view your app.
```

1.4.3. Viewing pods

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

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

```
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>
```

1.4.4. Viewing pod logs

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

```
$ oc logs cakephp-ex-1-deploy
--> Scaling cakephp-ex-1 to 1
--> Success
```

1.4.5. Viewing the current project

Use the **oc project** command to view the current project.

```
$ oc project
Using project "my-project" on server "https://openshift.example.com:6443".
```

1.4.6. Viewing the status for the current project

Use the **oc status** command to view information about the current project, such as Services, DeploymentConfigs, and BuildConfigs.

```
$ oc status
In project my-project on server https://openshift.example.com:6443

svc/cakephp-ex - 172.30.236.80 ports 8080, 8443
  dc/cakephp-ex deploys istag/cakephp-ex:latest <-
    bc/cakephp-ex source builds https://github.com/sclorg/cakephp-ex on openshift/php:7.2
    deployment #1 deployed 2 minutes ago - 1 pod

3 infos identified, use 'oc status --suggest' to see details.
```

1.4.7. Listing supported API resources

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

```
$ oc api-resources
NAME                              SHORTNAMES   APIGROUP                       NAMESPACED   KIND
bindings                                                                      true         Binding
componentstatuses                 cs                                          false        ComponentStatus
configmaps                        cm                                          true         ConfigMap
...
```

1.5. GETTING HELP

You can get help with CLI commands and OpenShift Container Platform resources in the following ways.

- Use **oc help** to get a list and description of all available CLI commands:

  Example: Get general help for the CLI

  ```
  $ oc help
  OpenShift Client

  This client helps you develop, build, deploy, and run your applications on any OpenShift or
  Kubernetes compatible
  platform. It also includes the administrative commands for managing a cluster under the 'adm'
  subcommand.

  Usage:
    oc [flags]

  Basic Commands:
    login           Log in to a server
    new-project     Request a new project
    new-app         Create a new application

  ...
  ```

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

  Example: Get help for the oc create command

  ```
  $ oc create --help
  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
  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

...
```

1.6. LOGGING OUT OF THE CLI

You can log out the CLI to end your current session.

- Use the **oc logout** command.

```
$ oc logout
Logged "user1" out on "https://openshift.example.com"
```

This deletes the saved authentication token from the server and removes it from your configuration file.

CHAPTER 2. CONFIGURING THE CLI

2.1. ENABLING TAB COMPLETION

After you install the **oc** CLI tool, you can enable tab completion to automatically complete **oc** commands or suggest options when you press Tab.

Prerequisites

- You must have the **oc** CLI tool installed.

Procedure

The following procedure enables tab completion for Bash.

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.

CHAPTER 3. EXTENDING THE CLI WITH PLUG-INS

You can write and install plug-ins to build on the default **oc** commands, allowing you to perform new and more complex tasks with the OpenShift Container Platform CLI.

3.1. WRITING CLI PLUG-INS

You can write a plug-in for the OpenShift Container Platform CLI in any programming language or script that allows you to write command-line commands. Note that you can not use a plug-in to overwrite an existing **oc** command.

IMPORTANT

OpenShift CLI plug-ins are currently a Technology Preview feature. Technology Preview features are not supported with Red Hat production service level agreements (SLAs), might not be functionally complete, and Red Hat does not recommend to use them for production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

See the Red Hat Technology Preview features support scope for more information.

Procedure

This procedure creates a simple Bash plug-in that prints a message to the terminal when the **oc foo** command is issued.

1. Create a file called **oc-foo**.
   When naming your plug-in file, keep the following in mind:

   - The file must begin with **oc-** or **kubectl-** in order to be recognized as a plug-in.
   - The file name determines the command that invokes the plug-in. For example, a plug-in with the file name **oc-foo-bar** can be invoked by a command of **oc foo bar**. You can also use underscores if you want the command to contain dashes. For example, a plug-in with the file name **oc-foo_bar** can be invoked by a command of **oc foo-bar**.

2. Add the following contents to the file.

```
#!/bin/bash

# optional argument handling
if [[ "$1" == "version" ]]
then
    echo "1.0.0"
    exit 0
fi

# optional argument handling
if [[ "$1" == "config" ]]
then
    echo $KUBECONFIG
    exit 0
fi

echo "I am a plugin named kubectl-foo"
```

After you install this plug-in for the OpenShift Container Platform CLI, it can be invoked using the **oc foo** command.

Additional resources

- Review the Sample plug-in repository for an example of a plug-in written in Go.
- Review the CLI runtime repository for a set of utilities to assist in writing plug-ins in Go.

3.2. INSTALLING AND USING CLI PLUG-INS

After you write a custom plug-in for the OpenShift Container Platform CLI, you must install it to use the functionality that it provides.

IMPORTANT

OpenShift CLI plug-ins are currently a Technology Preview feature. Technology Preview features are not supported with Red Hat production service level agreements (SLAs), might not be functionally complete, and Red Hat does not recommend to use them for production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

See the Red Hat Technology Preview features support scope for more information.

Prerequisites

- You must have the **oc** CLI tool installed.
- You must have a CLI plug-in file that begins with **oc-** or **kubectl-**.

Procedure

1. If necessary, update the plug-in file to be executable.

```
$ chmod +x <plugin_file>
```

2. Place the file anywhere in your **PATH**, such as **/usr/local/bin/**.

```
$ sudo mv <plugin_file> /usr/local/bin/.
```

3. Run **oc plugin list** to make sure that the plug-in is listed.

```
$ oc plugin list
The following compatible plugins are available:

/usr/local/bin/<plugin_file>
```

If your plug-in is not listed here, verify that the file begins with **oc-** or **kubectl-**, is executable, and is on your **PATH**.

4. Invoke the new command or option introduced by the plug-in.
For example, if you built and installed the **kubectl-ns** plug-in from the Sample plug-in repository, you can use the following command to view the current namespace.

```
$ oc ns
```

Note that the command to invoke the plug-in depends on the plug-in file name. For example, a plug-in with the file name of **oc-foo-bar** is invoked by the **oc foo bar** command.

CHAPTER 4. DEVELOPER CLI COMMANDS

4.1. BASIC CLI COMMANDS

4.1.1. explain

Display documentation for a certain resource.

Example: Display documentation for Pods

```
$ oc explain pods
```

4.1.2. login

Log in to the OpenShift Container Platform server and save login information for subsequent use.

Example: Interactive login

```
$ oc login
```

Example: Log in specifying a user name

```
$ oc login -u user1
```

4.1.3. new-app

Create a new application by specifying source code, a template, or an image.

Example: Create a new application from a local Git repository

```
$ oc new-app .
```

Example: Create a new application from a remote Git repository

```
$ oc new-app https://github.com/sclorg/cakephp-ex
```

Example: Create a new application from a private remote repository

```
$ oc new-app https://github.com/youruser/yourprivaterepo --source-secret=yoursecret
```

4.1.4. new-project

Create a new project and switch to it as the default project in your configuration.

Example: Create a new project

```
$ oc new-project myproject
```

4.1.5. project

Switch to another project and make it the default in your configuration.

Example: Switch to a different project

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

4.1.6. projects

Display information about the current active project and existing projects on the server.

Example: List all projects

```
$ oc projects
```

4.1.7. status

Show a high-level overview of the current project.

Example: Show the status of the current project

```
$ oc status
```

4.2. BUILD AND DEPLOY CLI COMMANDS

4.2.1. cancel-build

Cancel a running, pending, or new build.

Example: Cancel a build

```
$ oc cancel-build python-1
```

Example: Cancel all pending builds from the python BuildConfig

```
$ oc cancel-build buildconfig/python --state=pending
```

4.2.2. import-image

Import the latest tag and image information from an image repository.

Example: Import the latest image information

```
$ oc import-image my-ruby
```

4.2.3. new-build

Create a new **BuildConfig** from source code.

Example: Create a BuildConfig from a local Git repository

```
$ oc new-build .
```

Example: Create a BuildConfig from a remote Git repository

```
$ oc new-build https://github.com/sclorg/cakephp-ex
```

4.2.4. rollback

Revert an application back to a previous Deployment.

Example: Roll back to the last successful Deployment

```
$ oc rollback php
```

Example: Roll back to a specific version

```
$ oc rollback php --to-version=3
```

4.2.5. rollout

Start a new rollout, view its status or history, or roll back to a previous revision of your application.

Example: Roll back to the last successful Deployment

```
$ oc rollout undo deploymentconfig/php
```

Example: Start a new rollout for a DeploymentConfig with its latest state

```
$ oc rollout latest deploymentconfig/php
```

4.2.6. start-build

Start a build from a **BuildConfig** or copy an existing build.

Example: Start a build from the specified BuildConfig

```
$ oc start-build python
```

Example: Start a build from a previous build

```
$ oc start-build --from-build=python-1
```

Example: Set an environment variable to use for the current build

```
$ oc start-build python --env=mykey=myvalue
```

4.2.7. tag

Tag existing images into imagestreams.

Example: Configure the ruby image's latest tag to refer to the image for the 2.0 tag

```
$ oc tag ruby:latest ruby:2.0
```

4.3. APPLICATION MANAGEMENT CLI COMMANDS

4.3.1. annotate

Update the annotations on one or more resources.

Example: Add an annotation to a Route

```
$ oc annotate route/test-route haproxy.router.openshift.io/ip_whitelist="192.168.1.10"
```

Example: Remove the annotation from the Route

```
$ oc annotate route/test-route haproxy.router.openshift.io/ip_whitelist-
```

4.3.2. apply

Apply a configuration to a resource by file name or standard in (stdin) in JSON or YAML format.

Example: Apply the configuration in pod.json to a Pod

```
$ oc apply -f pod.json
```

4.3.3. autoscale

Autoscale a DeploymentConfig or ReplicationController.

Example: Autoscale to a minimum of two and maximum of five Pods

```
$ oc autoscale deploymentconfig/parksmap-katacoda --min=2 --max=5
```

4.3.4. create

Create a resource by file name or standard in (stdin) in JSON or YAML format.

Example: Create a Pod using the content in pod.json

```
$ oc create -f pod.json
```

4.3.5. delete

Delete a resource.

Example: Delete a Pod named parksmap-katacoda-1-qfqz4

```
$ oc delete pod/parksmap-katacoda-1-qfqz4
```

Example: Delete all Pods with the app=parksmap-katacoda label

```
$ oc delete pods -l app=parksmap-katacoda
```

4.3.6. describe

Return detailed information about a specific object.

Example: Describe a Deployment named example

```
$ oc describe deployment/example
```

Example: Describe all Pods

```
$ oc describe pods
```

4.3.7. edit

Edit a resource.

Example: Edit a DeploymentConfig using the default editor

```
$ oc edit deploymentconfig/parksmap-katacoda
```

Example: Edit a DeploymentConfig using a different editor

```
$ OC_EDITOR="nano" oc edit deploymentconfig/parksmap-katacoda
```

Example: Edit a DeploymentConfig in JSON format

```
$ oc edit deploymentconfig/parksmap-katacoda -o json
```

4.3.8. expose

Expose a Service externally as a Route.

Example: Expose a Service

```
$ oc expose service/parksmap-katacoda
```

Example: Expose a Service and specify the host name

```
$ oc expose service/parksmap-katacoda --hostname=www.my-host.com
```

4.3.9. get

Display one or more resources.

Example: List Pods in the `default` namespace

```
$ oc get pods -n default
```

Example: Get details about the `python` DeploymentConfig in JSON format

```
$ oc get deploymentconfig/python -o json
```

4.3.10. label

Update the labels on one or more resources.

Example: Update the `python-1-mz2rf` Pod with the label `status` set to `unhealthy`

```
$ oc label pod/python-1-mz2rf status=unhealthy
```

4.3.11. scale

Set the desired number of replicas for a ReplicationController or a DeploymentConfig.

Example: Scale the `ruby-app` DeploymentConfig to three Pods

```
$ oc scale deploymentconfig/ruby-app --replicas=3
```

4.3.12. secrets

Manage secrets in your project.

Example: Allow `my-pull-secret` to be used as an image pull secret by the `default` service account

```
$ oc secrets link default my-pull-secret --for=pull
```

4.3.13. serviceaccounts

Get a token assigned to a service account or create a new token or **`kubeconfig`** file for a service account.

Example: Get the token assigned to the `default` service account

```
$ oc serviceaccounts get-token default
```

4.3.14. set

Configure existing application resources.

Example: Sets the name of a secret on a BuildConfig

```
$ oc set build-secret --source buildconfig/mybc mysecret
```

4.4. TROUBLESHOOTING AND DEBUGGING CLI COMMANDS

4.4.1. attach

Attach the shell to a running container.

Example: Get output from the python container from Pod python-1-mz2rf

```
$ oc attach python-1-mz2rf -c python
```

4.4.2. cp

Copy files and directories to and from containers.

Example: Copy a file from the python-1-mz2rf Pod to the local file system

```
$ oc cp default/python-1-mz2rf:/opt/app-root/src/README.md ~/mydirectory/.
```

4.4.3. debug

Launch a command shell to debug a running application.

Example: Debug the python Deployment

```
$ oc debug deploymentconfig/python
```

4.4.4. exec

Execute a command in a container.

Example: Execute the ls command in the python container from Pod python-1-mz2rf

```
$ oc exec python-1-mz2rf -c python ls
```

4.4.5. logs

Retrieve the log output for a specific build, BuildConfig, DeploymentConfig, or Pod.

Example: Stream the latest logs from the python DeploymentConfig

```
$ oc logs -f deploymentconfig/python
```

4.4.6. port-forward

Forward one or more local ports to a Pod.

Example: Listen on port 8888 locally and forward to port 5000 in the Pod

```
$ oc port-forward python-1-mz2rf 8888:5000
```

4.4.7. proxy

Run a proxy to the Kubernetes API server.

Example: Run a proxy to the API server on port 8011 serving static content from ./local/www/

```
$ oc proxy --port=8011 --www=./local/www/
```

4.4.8. rsh

Open a remote shell session to a container.

Example: Open a shell session on the first container in the python-1-mz2rf Pod

```
$ oc rsh python-1-mz2rf
```

4.4.9. rsync

Copy contents of a directory to or from a running Pod container. Only changed files are copied using the **rsync** command from your operating system.

Example: Synchronize files from a local directory with a Pod directory

```
$ oc rsync ~/mydirectory/ python-1-mz2rf:/opt/app-root/src/
```

4.4.10. run

Create and run a particular image. By default, this creates a DeploymentConfig to manage the created containers.

Example: Start an instance of the perl image with three replicas

```
$ oc run my-test --image=perl --replicas=3
```

4.4.11. wait

Wait for a specific condition on one or more resources.

Example: Wait for the python-1-mz2rf Pod to be deleted

```
$ oc wait --for=delete pod/python-1-mz2rf
```

4.5. ADVANCED DEVELOPER CLI COMMANDS

4.5.1. api-resources

Display the full list of API resources that the server supports.

Example: List the supported API resources

```
$ oc api-resources
```

4.5.2. api-versions

Display the full list of API versions that the server supports.

Example: List the supported API versions

```
$ oc api-versions
```

4.5.3. auth

Inspect permissions and reconcile RBAC roles.

Example: Check whether the current user can read Pod logs

```
$ oc auth can-i get pods --subresource=log
```

Example: Reconcile RBAC roles and permissions from a file

```
$ oc auth reconcile -f policy.json
```

4.5.4. cluster-info

Display the address of the master and cluster services.

Example: Display cluster information

```
$ oc cluster-info
```

4.5.5. convert

Convert a YAML or JSON configuration file to a different API version and print to standard output (stdout).

Example: Convert `pod.yaml` to the latest version

```
$ oc convert -f pod.yaml
```

4.5.6. extract

Extract the contents of a ConfigMap or secret. Each key in the ConfigMap or secret is created as a separate file with the name of the key.

Example: Download the contents of the `ruby-1-ca` ConfigMap to the current directory

```
$ oc extract configmap/ruby-1-ca
```

Example: Print the contents of the `ruby-1-ca` ConfigMap to stdout

```
$ oc extract configmap/ruby-1-ca --to=-
```

4.5.7. idle

Idle scalable resources. An idled Service will automatically become unidled when it receives traffic or it can be manually unidled using the **oc scale** command.

Example: Idle the ruby-app Service

```
$ oc idle ruby-app
```

4.5.8. image

Manage images in your OpenShift Container Platform cluster.

Example: Copy an image to another tag

```
$ oc image mirror myregistry.com/myimage:latest myregistry.com/myimage:stable
```

4.5.9. observe

Observe changes to resources and take action on them.

Example: Observe changes to Services

```
$ oc observe services
```

4.5.10. patch

Updates one or more fields of an object using strategic merge patch in JSON or YAML format.

Example: Update the spec.unschedulable field for node node1 to true

```
$ oc patch node/node1 -p '{"spec":{"unschedulable":true}}'
```

NOTE

If you must patch a Custom Resource Definition, you must include the **--type merge** option in the command.

4.5.11. policy

Manage authorization policies.

Example: Add the edit role to user1 for the current project

```
$ oc policy add-role-to-user edit user1
```

4.5.12. process

Process a template into a list of resources.

Example: Convert template.json to a resource list and pass to oc create

```
$ oc process -f template.json | oc create -f -
```

4.5.13. registry

Manage the integrated registry on OpenShift Container Platform.

Example: Display information about the integrated registry

```
$ oc registry info
```

4.5.14. replace

Modify an existing object based on the contents of the specified configuration file.

Example: Update a Pod using the content in pod.json

```
$ oc replace -f pod.json
```

4.6. SETTINGS CLI COMMANDS

4.6.1. completion

Output shell completion code for the specified shell.

Example: Display completion code for Bash

```
$ oc completion bash
```

4.6.2. config

Manage the client configuration files.

Example: Display the current configuration

```
$ oc config view
```

Example: Switch to a different context

```
$ oc config use-context test-context
```

4.6.3. logout

Log out of the current session.

Example: End the current session

-

```
$ oc logout
```

4.6.4. whoami

Display information about the current session.

Example: Display the currently authenticated user

```
$ oc whoami
```

4.7. OTHER DEVELOPER CLI COMMANDS

4.7.1. help

Display general help information for the CLI and a list of available commands.

Example: Display available commands

```
$ oc help
```

Example: Display the help for the new-project command

```
$ oc help new-project
```

4.7.2. plugin

List the available plug-ins on the user's **PATH**.

Example: List available plug-ins

```
$ oc plugin list
```

4.7.3. version

Display the **oc** client and server versions.

Example: Display version information

```
$ oc version
```

CHAPTER 5. ADMINISTRATOR CLI COMMANDS

5.1. CLUSTER MANAGEMENT CLI COMMANDS

5.1.1. must-gather

Bulk collect data about the current state of your cluster to debug issues.

Example: Gather debugging information

```
$ oc adm must-gather
```

5.1.2. top

Show usage statistics of resources on the server.

Example: Show CPU and memory usage for Pods

```
$ oc adm top pods
```

Example: Show usage statistics for images

```
$ oc adm top images
```

5.2. NODE MANAGEMENT CLI COMMANDS

5.2.1. cordon

Mark a node as unschedulable. Manually marking a node as unschedulable blocks any new pods from being scheduled on the node, but does not affect existing pods on the node.

Example: Mark `node1` as unschedulable

```
$ oc adm cordon node1
```

5.2.2. drain

Drain a node in preparation for maintenance.

Example: Drain `node1`

```
$ oc adm drain node1
```

5.2.3. node-logs

Display and filter node logs.

Example: Get logs for NetworkManager

```
$ oc adm node-logs --role master -u NetworkManager.service
```

5.2.4. taint

Update the taints on one or more nodes.

Example: Add a taint to dedicate a node for a set of users

```
$ oc adm taint nodes node1 dedicated=groupName:NoSchedule
```

Example: Remove the taints with key `dedicated` from node `node1`

```
$ oc adm taint nodes node1 dedicated-
```

5.2.5. uncordon

Mark a node as schedulable.

Example: Mark `node1` as schedulable

```
$ oc adm uncordon node1
```

5.3. SECURITY AND POLICY CLI COMMANDS

5.3.1. certificate

Approve or reject certificate signing requests (CSRs).

Example: Approve a CSR

```
$ oc adm certificate approve csr-sqgzp
```

5.3.2. groups

Manage groups in your cluster.

Example: Create a new group

```
$ oc adm groups new my-group
```

5.3.3. new-project

Create a new project and specify administrative options.

Example: Create a new project using a node selector

```
$ oc adm new-project myproject --node-selector='type=user-node,region=east'
```

5.3.4. pod-network

Manage Pod networks in the cluster.

Example: Isolate project1 and project2 from other non-global projects

```
$ oc adm pod-network isolate-projects project1 project2
```

5.3.5. policy

Manage roles and policies on the cluster.

Example: Add the edit role to user1 for all projects

```
$ oc adm policy add-cluster-role-to-user edit user1
```

Example: Add the privileged security context constraint to a service account

```
$ oc adm policy add-scc-to-user privileged -z myserviceaccount
```

5.4. MAINTENANCE CLI COMMANDS

5.4.1. migrate

Migrate resources on the cluster to a new version or format depending on the subcommand used.

Example: Perform an update of all stored objects

```
$ oc adm migrate storage
```

Example: Perform an update of only Pods

```
$ oc adm migrate storage --include=pods
```

5.4.2. prune

Remove older versions of resources from the server.

Example: Prune older builds including those whose BuildConfigs no longer exist

```
$ oc adm prune builds --orphans
```

5.5. CONFIGURATION CLI COMMANDS

5.5.1. create-api-client-config

Create a client configuration for connecting to the server. This creates a folder containing a client certificate, a client key, a server certificate authority, and a **kubeconfig** file for connecting to the master as the provided user.

Example: Generate a client certificate for a proxy

```
$ oc adm create-api-client-config \
  --certificate-authority='/etc/origin/master/proxyca.crt' \
  --client-dir='/etc/origin/master/proxy' \
  --signer-cert='/etc/origin/master/proxyca.crt' \
  --signer-key='/etc/origin/master/proxyca.key' \
  --signer-serial='/etc/origin/master/proxyca.serial.txt' \
  --user='system:proxy'
```

5.5.2. create-bootstrap-policy-file

Create the default bootstrap policy.

Example: Create a file called policy.json with the default bootstrap policy

```
$ oc adm create-bootstrap-policy-file --filename=policy.json
```

5.5.3. create-bootstrap-project-template

Create a bootstrap project template.

Example: Output a bootstrap project template in YAML format to stdout

```
$ oc adm create-bootstrap-project-template -o yaml
```

5.5.4. create-error-template

Create a template for customizing the error page.

Example: Output a template for the error page to stdout

```
$ oc adm create-error-template
```

5.5.5. create-kubeconfig

Creates a basic **.kubeconfig** file from client certificates.

Example: Create a .kubeconfig file with the provided client certificates

```
$ oc adm create-kubeconfig \
  --client-certificate=/path/to/client.crt \
  --client-key=/path/to/client.key \
  --certificate-authority=/path/to/ca.crt
```

5.5.6. create-login-template

Create a template for customizing the login page.

Example: Output a template for the login page to stdout

```
$ oc adm create-login-template
```

5.5.7. create-provider-selection-template

Create a template for customizing the provider selection page.

Example: Output a template for the provider selection page to stdout

```
$ oc adm create-provider-selection-template
```

5.6. OTHER ADMINISTRATOR CLI COMMANDS

5.6.1. build-chain

Output the inputs and dependencies of any builds.

Example: Output dependencies for the perl imagestream

```
$ oc adm build-chain perl
```

5.6.2. completion

Output shell completion code for the **oc adm** commands for the specified shell.

Example: Display oc adm completion code for Bash

```
$ oc adm completion bash
```

5.6.3. config

Manage the client configuration files. This command has the same behavior as the **oc config** command.

Example: Display the current configuration

```
$ oc adm config view
```

Example: Switch to a different context

```
$ oc adm config use-context test-context
```

5.6.4. release

Manage various aspects of the OpenShift Container Platform release process, such as viewing information about a release or inspecting the contents of a release.

Example: Generate a changelog between two releases and save to changelog.md

```
$ oc adm release info --changelog=/tmp/git \
    quay.io/openshift-release-dev/ocp-release:4.1.0-rc.7 \
```

```
    quay.io/openshift-release-dev/ocp-release:4.1.0 \
    > changelog.md
```

5.6.5. verify-image-signature

Verify the image signature of an image imported to the internal registry using the local public GPG key.

Example: Verify the nodejs image signature

```
$ oc adm verify-image-signature \
    sha256:2bba968aedb7dd2aafe5fa8c7453f5ac36a0b9639f1bf5b03f95de325238b288 \
    --expected-identity 172.30.1.1:5000/openshift/nodejs:latest \
    --public-key /etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release \
    --save
```

CHAPTER 6. USAGE OF OC AND KUBECTL COMMANDS

Kubernetes' command line interface (CLI), **kubectl**, can be used to run commands against a Kubernetes cluster. Because OpenShift Container Platform is a certified Kubernetes distribution, you can use the supported **kubectl** binaries that ship with OpenShift Container Platform, or you can gain extended functionality by using the **oc** binary.

6.1. THE OC BINARY

The **oc** binary offers the same capabilities as the **kubectl** binary, but it extends to natively support additional OpenShift Container Platform features, including:

- **Full support for OpenShift Container Platform resources**
  Resources such as DeploymentConfigs, BuildConfigs, Routes, ImageStreams, and ImageStreamTags are specific to OpenShift Container Platform distributions, and build upon standard Kubernetes primitives.

- **Authentication**
  The **oc** binary offers a built-in **login** command that allows authentication and enables you to work with OpenShift Container Platform projects, which map Kubernetes namespaces to authenticated users. See Understanding authentication for more information.

- **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. Similarly, the additional command **oc new-project** makes it easier to start a project that you can switch to as your default.

6.2. THE KUBECTL BINARY

The **kubectl** binary is provided as a means to support existing workflows and scripts for new OpenShift Container Platform users coming from a standard Kubernetes environment, or for those who prefer to use the **kubectl** CLI. Existing users of **kubectl** can continue to use the binary to interact with Kubernetes primitives, with no changes required to the OpenShift Container Platform cluster.

For more information, see the kubectl docs.

---------- Last updated: 2020-02-28 ----------