Chapter 2. Kubernetes and OpenShift Command-line Interfaces and APIs

Abstract

Goal	Access an OpenShift cluster by using the command line and query its Kubernetes API resources to assess the health of a cluster.
Sections	• The Kubernetes and OpenShift Command-line Interfaces (and Guided Exercise) • Inspect Kubernetes Resources (and Guided Exercise) • Assess the Health of an OpenShift Cluster (and Guided Exercise)
Lab	• Kubernetes and OpenShift Command-line Interfaces and APIs

The Kubernetes and OpenShift Command-line Interfaces

Objectives

- Access an OpenShift cluster by using the Kubernetes and OpenShift command-line interfaces.

The Kubernetes Command-line Interface

You can manage an OpenShift cluster from the web console or by using the `kubectl` or `oc` command-line interfaces (CLI).
The `kubectl` commands are native to Kubernetes, and are a thin wrapper over the Kubernetes API. The OpenShift `oc` commands are a superset of the `kubectl` commands, and add commands for the OpenShift-specific features. In this course, examples of both the `kubectl` and the `oc` commands are shown, to highlight the differences between the commands.

With the `oc` command, you can create applications and manage Red Hat OpenShift Container Platform (RHOCP) projects from a terminal. The OpenShift CLI is ideal in the following situations:

- Working directly with project source code
- Scripting OpenShift Container Platform operations
- Managing projects that are restricted by bandwidth
- When the web console is unavailable
- Working with OpenShift resources, such as `routes` and `projects`

Kubernetes Command-line Tool

The `oc` CLI installation also includes an installation of the `kubectl` CLI, which is the recommended method for installing the `kubectl` CLI for OpenShift users.

You can also install the `kubectl` CLI independently of the `oc` CLI. You must use a `kubectl` CLI version that is within one minor version difference of your cluster. For example, a `v1.30` client can communicate with `v1.29`, `v1.30`, and `v1.31` control planes. Using the latest compatible version of the `kubectl` CLI helps to avoid unforeseen issues.

To perform a manual installation of the `kubectl` binary for a Linux installation, you must first download the latest release by using the `curl` command.

```
[user@host ~]$ curl -LO "https://dl.k8s.io/release/$(curl -L \
  -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
```

Then, you must download the `kubectl` checksum file and validate the `kubectl` binary against the checksum file.

```
[user@host ~]$ curl -LO "https://dl.k8s.io/$(curl -L \
-s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl.sha256"
```

```
[user@host ~]$ echo "$(cat kubectl.sha256)  kubectl" | sha256sum --check
kubectl: OK
```

If the check fails, then the `sha256sum` command exits with nonzero status, and prints a `kubectl: FAILED` message.

You can then install the `kubectl` CLI.

```
[user@host ~]$ sudo install -o root -g root -m 0755 kubectl \
  /usr/local/bin/kubectl
```

> **NOTE**
>
> If you do not have `root` access on the target system, then you can still install the `kubectl` CLI to the `~/.local/bin` directory. For more information, refer to https://kubernetes.io/docs/tasks/tools/install-kubectl-linux/.

Finally, use the `kubectl version` command to verify the installed version. This command prints the client and server versions. Use the `--client` option to view the client version only.

```
[user@host ~]$ kubectl version --client
Client Version: v1.31.1
Kustomize Version: v5.4.2
```

Alternatively, a distribution that is based on Red Hat Enterprise Linux (RHEL) can install the `kubectl` CLI with the following command:

```
[user@host ~]$ cat <<EOF | sudo tee /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=https://pkgs.k8s.io/core:/stable:/v1.31/rpm/
enabled=1
gpgcheck=1
gpgkey=https://pkgs.k8s.io/core:/stable:/v1.31/rpm/repodata/repomd.xml.key
EOF
[user@host ~]$ sudo yum install -y kubectl
```

To view a list of the available `kubectl` commands, use the `kubectl --help` command.

```
[user@host ~]$ kubectl --help
kubectl controls the Kubernetes cluster manager.

 Find more information at:
https://kubernetes.io/docs/reference/kubectl/

Basic Commands (Beginner):
  create        Create a resource from a file or from stdin
  expose        Take a replication controller, service, deployment or pod and
expose it as a new Kubernetes Service
  run           Run a particular image on the cluster
  set           Set specific features on objects

Basic Commands (Intermediate):
...output omitted...
```

You can also use the `--help` option with any command to view detailed information about the command, including its purpose, examples, available subcommands, and options.

For example, the following command provides information about the `kubectl create` command and its usage.

```
[user@host ~]$ kubectl create --help
Create a resource from a file or from stdin.

 JSON and YAML formats are accepted.

Examples:
  # Create a pod using the data in pod.json
  kubectl create -f ./pod.json

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

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

Available Commands:
  clusterrole             Create a cluster role
  clusterrolebinfing      Create a cluster role binding for a particular cluster role
...output omitted...
```

Kubernetes uses many resource components to support applications. The `kubectl explain` command provides detailed information about the attributes of a given resource. For example, use the following command to learn more about the attributes of a `pod` resource.

```
[user@host ~]$ kubectl explain pod
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.
...output omitted...
```

Refer to the Kubernetes documentation at https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands/ for further information about the `kubectl` commands.

The OpenShift Command-line Interface

The main method of interacting with an RHOCP cluster is by using the `oc` command.

You can download the `oc` CLI from the OpenShift web console to ensure that the CLI tools are compatible with the RHOCP cluster. From the OpenShift web console, go to **Help** → **Command Line tools**. The **Help** menu is represented by a `?` icon. The web console provides several installation options for the `oc` client, such as downloads for the following operating systems:

- x86_64 Windows, Mac, and Linux systems
- ARM 64 Linux and Mac systems
- Linux for IBM Z, IBM Power, and little endian

Command Line Tools

Copy login command

oc - OpenShift Command Line Interface (CLI)

With the OpenShift command line interface, you can create applications and manage OpenShift projects from a terminal.

The oc binary offers the same capabilities as the kubectl binary, but it is further extended to natively support OpenShift Container Platform features.

- Download oc for Linux for x86_64
- Download oc for Mac for x86_64
- Download oc for Windows for x86_64
- Download oc for Linux for ARM 64
- Download oc for Mac for ARM 64
- Download oc for Linux for IBM Power, little endian
- Download oc for Linux for IBM Z
- LICENSE

The basic usage of the `oc` command is through its subcommands in the following syntax:

```
[user@host ~]$ oc command
```

Because the `oc` CLI is a superset of the `kubectl` CLI, the `version`, `--help`, and `explain` commands work the same for both CLIs. However, the `oc` CLI includes additional commands that are not included in the `kubectl` CLI, such as the `oc login` and `oc new-project` commands.

Developers who are familiar with Kubernetes can use the `kubectl` utility to manage a RHOCP cluster. This course uses the `oc` command-line utility, to take advantage of additional RHOCP features. The `oc` commands manage resources that are exclusive to RHOCP, such as projects, routes, and image streams.

Authentication Methods

Before you can interact with your RHOCP cluster, you must authenticate your requests. The authentication layer identifies the user that is associated with requests to the RHOCP API. After authentication, the authorization layer uses information about the requesting user to determine whether the request is allowed.

A user in OpenShift is an entity that can send requests to the RHOCP API. An RHOCP `User` object represents an actor that can be granted permissions in the system by adding roles to the user or to the user's groups. Typically, this entity represents the account of a developer or an administrator.

Several types of users can exist.

Regular users
Most interactive RHOCP users are represented by this user type. An RHOCP `User` object represents a regular user.

System users
Infrastructure uses system users to interact with the API securely. Some system users are automatically created, including the cluster administrator, with access to everything. By default, unauthenticated requests use an anonymous system user.

Service accounts
`ServiceAccount` objects represent accounts that allow applications to access the API without sharing regular user credentials. RHOCP creates service accounts automatically when a project is created. Project administrators can create additional service accounts to define access to the contents of each project.

Each user must authenticate to access a cluster. After authentication, the policy determines what the user is authorized to do.

> **NOTE**
>
> Authentication and authorization are covered in greater detail in the *Red Hat OpenShift Administration II: Operating a Production Kubernetes Cluster* (DO280) course.

Use the `oc login` command to authenticate your requests. The `oc login` command provides role-based authentication and authorization that protects the RHOCP cluster from unauthorized access.

The `oc login` command supports several authentication methods.

Username and Password Authentication
The syntax for username and password authentication is as follows:

```
[user@host ~]$ oc login -u username api-url
```

Then, the `oc login` command prompts for the password.

For example, in this course, you can use the following command:

```
[user@host ~]$ oc login -u admin https://api.ocp4.example.com:6443
Console URL: https://api.ocp4.example.com:6443/console
Authentication required for https://api.ocp4.example.com:6443 (openshift)
Username: admin
Password: redhatocp
Login successful.
...output omitted...
```

Token-based Authentication
The RHOCP control plane includes a built-in OAuth server. To authenticate to the API, users obtain OAuth access tokens. Token authentication is the recommended method for production environments, automation, and CI/CD pipelines to improve security and to eliminate storing passwords in scripts. You can use token-based authentication to log in to the cluster, as follows:

```
[user@host ~]$ oc login --token=your-token --server=cluster-url
```

To retrieve a token, open a web browser and go to https://oauth-openshift.apps.*cluster-url*/oauth/token/request Then, click **Display token** to display the login command.

Web Authentication
You can also use the `--web` option to log in to the cluster by using a web browser, as follows:

```
[user@host ~]$ oc login --web api-url
```

This command opens a browser window to the OpenShift authentication page, to log in with your username and password, or other authentication method. Then, the command automatically returns the authentication token to the command line, to complete the login process.

> **IMPORTANT**
>
> For simplicity, this course uses the username and password authentication method by providing the password as plain text, as follows:
>
> ```
> [user@host ~]$ oc login -u admin -p redhatocp \
>   https://api.ocp4.example.com:6443
> ```
>
> However, this method is not secure, and Red Hat does not recommended it for production environments. Thus, in production environments use a more secure method, such as token-based authentication.

Managing Resources at the Command Line

After authenticating to the RHOCP cluster, you can create a project with the `oc new-project` command. Projects provide isolation between your application resources. Projects are Kubernetes namespaces with additional annotations that provide multitenancy scoping for applications.

```
[user@host ~]$ oc new-project myapp
```

Several essential commands can manage RHOCP and Kubernetes resources, as described here. Unless otherwise specified, the following commands are compatible with both the `oc` and `kubectl` CLIs.

Some commands require a user with cluster administrator access. The following list includes several useful `oc` commands for cluster administrators.

oc cluster-info
The `cluster-info` command prints the address of the control plane and other cluster services. The `oc cluster-info dump` command expands the output to include helpful details for debugging cluster problems.

```
[user@host ~]$ oc cluster-info
Kubernetes control plane is running at https://api.ocp4.example.com:6443
...output omitted...
```

oc api-versions
The structure of cluster resources has a corresponding API version, which the `oc api-versions` command displays. The command prints the supported API versions on the server, in the form of "group/version".

In the following example, the group is `admissionregistration.k8s.io` and the version is `v1`:

```
[user@host ~]$ oc api-versions
admissionregistration.k8s.io/v1
...output omitted...
```

oc get clusteroperator
The cluster operators that Red Hat ships serve as the architectural foundation for RHOCP. RHOCP installs cluster operators by default. Use the `oc get clusteroperator` command to see a list of the cluster operators:

```
[user@host ~]$ oc get clusteroperator
NAME                       VERSION    AVAILABLE PROGRESSING DEGRADED SINCE ...
authentication             4.18.6     True      False       False    18d
baremetal                  4.18.6     True      False       False    18d
...output omitted...
```

Other useful commands are available to both regular and administrator users:

oc get
Use the `get` command to retrieve information about resources in the selected project. Generally, this command shows only the most important characteristics of the resources, and omits detailed information.

The `oc get` *`RESOURCE_TYPE`* command displays a summary of all resources of the specified type.

For example, the following command returns the list of the `pod` resources in the current project:

```
[user@host ~]$ oc get pod
NAME                          READY   STATUS    RESTARTS   AGE
quotes-api-6c9f758574-nk8kd   1/1     Running   0          39m
quotes-ui-d7d457674-rbkl7     1/1     Running   0          67s
```

You can use the `oc get RESOURCE_TYPE RESOURCE_NAME` command to export a resource definition. Typical use cases include creating a backup or modifying a definition. The `-o yaml` option prints the object representation in YAML format. You can change to JSON format by providing a `-o json` option.

oc get all

Use the `oc get all` command to retrieve a summary of the most important resources in a project. This command iterates through the major resource types for the current project, and prints a summary of their information:

```
[user@host ~]$ oc get all
...output omitted...
NAME                          READY   STATUS    RESTARTS   AGE
pod/mysql-6bb757c595-qj5ng    1/1     Running   0          16s

NAME            TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)             AGE
service/mysql   ClusterIP   172.30.173.61   <none>        3306/TCP,33060/TCP   2m21s

NAME                    READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/mysql   1/1     1            1           2m21s

NAME                               DESIRED   CURRENT   READY   AGE
replicaset.apps/mysql-55dc44b4c9   0         0         0       2m21s
replicaset.apps/mysql-6bb757c595   1         1         1       16s
replicaset.apps/mysql-7d9b6b5996   0         0         0       2m21s

NAME                                  IMAGE REPOSITORY                           TAGS     UPDATED
imagestream.image.openshift.io/mysql   image-registry.../mysql-openshift/mysql   latest   2 minutes ago
```

oc describe

If the summaries from the `get` command are insufficient, then you can use the `oc describe RESOURCE_TYPE RESOURCE_NAME` command to retrieve additional information. Unlike the `get` command, you can use the `describe` command to iterate through all the different resources by type. Although most major resources can be described, this function is not available across all resources. The following example demonstrates describing a pod resource:

```
[user@host ~]$ oc describe pod mysql-6bb757c595-qj5ng
Name:             mysql-6bb757c595-qj5ng
Namespace:        mysql-openshift
Priority:         0
Service Account:  default
Node:             master01/192.168.50.10
Start Time:       Mon, 16 Jun 2025 16:07:00 -0400
Labels:           db=mysql
                  deployment=mysql
...output omitted...
Status:             Running
SeccompProfile:     RuntimeDefault
IP:                 10.129.0.85
...output omitted...
```

oc explain

To learn about the fields of an API resource object, use the `oc explain` command. This command describes the purpose and the fields that are associated with each supported API resource. You can also use this command to print the documentation of a specific field of a resource. Fields are identified via a JSONPath identifier. The following example prints the documentation for the `.spec.containers.resources` field of the `pod` resource type:

```
[user@host ~]$ oc explain pods.spec.containers.resources
KIND:     Pod
VERSION:  v1

FIELD: resources <ResourceRequiremnts>

DESCRIPTION:
     Compute Resources required by this container. Cannot be updated. More info:
     https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/

     ResourceRequirements describes the compute resource requirements.

FIELDS:
   claims       <[]ResourceClaim>
     Claims lists the names of resources, defined in spec.resourceClaims, that
     are used by this container.

     This is an alpha field nd requires enabling the DynamicResourceAllocation
     feature gate.

     This field is immutable. It can only be set for containers.

   limits       <map[string]Quantity>
     Limits describes the maximum amount of compute resources allowed. More
     info:
     https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/

   requests     <map[string]Quantity>
     Requests describes the minimum amount of compute resources required. If
     Requests is omitted for a container, it defaults to Limits if that is
     explicitly specified, otherwise to an implementation-defined value. Requests
    cannot exceed Limits. More info:
     https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/
```

Add the `--recursive` flag to display all fields of a resource without descriptions. Information about each field is retrieved from the server in OpenAPI format.

oc create

Use the `create` command to create a RHOCP resource in the current project. This command creates resources from a resource definition. Typically, this command is paired with the `oc get` *`RESOURCE_TYPE RESOURCE_NAME`* `-o yaml` command for editing definitions. Developers commonly use the `-f` flag to indicate the file that contains the JSON or YAML representation of an RHOCP resource.

For example, to create resources from the `pod.yaml` file, use the following command:

```
[user@host ~]$ oc create -f pod.yaml
pod/quotes-pod created
```

RHOCP resources in the YAML format are discussed later.

oc status

The `oc status` command provides a high-level overview of the current project. The command shows services, deployments, build configurations, and active deployments. Information about any misconfigured components is also shown. The `--suggest` option shows additional details for any identified issues.

oc delete

Use the `delete` command to delete an existing RHOCP resource from the current project. You must specify the resource type and the resource name.

For example, to delete the `quotes-ui` pod, use the following command:

```
[user@host ~]$ oc delete pod quotes-ui
pod/quotes-ui deleted
```

A fundamental understanding of the RHOCP architecture is needed here, because deleting managed resources, such as pods, results in the automatic creation of new instances of those resources. When a project is deleted, it deletes all the resources and applications within it.

Each of these commands is executed in the current selected project. To execute commands in a different project, you must include the `--namespace` or `-n` options.

```
[user@host ~]$ oc get pods -n openshift-apiserver
NAME                         READY   STATUS    RESTARTS   AGE
apiserver-68c9485699-ndqlc   2/2     Running   2          18d
```

Refer to the references for a complete list of `oc` commands.

REFERENCES

Kubernetes Documentation - Install and Set Up kubectl on Linux

For more information, refer to the *Getting Started with the OpenShift CLI* chapter in the Red Hat OpenShift Container Platform 4.18 *CLI Tools* documentation at https://docs.redhat.com/en/documentation/openshift_container_platform/4.18/html-single/cli_tools/index#cli-about-cli_cli-developer-commands

For more information, refer to the *CLI Developer Commands* chapter in the Red Hat OpenShift Container Platform 4.18 *CLI Tools* documentation at https://docs.redhat.com/en/documentation/openshift_container_platform/4.18/html-single/cli_tools/index#cli-developer-commands

For more information, refer to the *Understanding Authentication* chapter in the Red Hat OpenShift Container Platform 4.18 *Authentication and Authorization* documentation at https://docs.redhat.com/en/documentation/openshift_container_platform/4.18/html-single/authentication_and_authorization/index#understanding-authentication