OpenShift Enterprise 3.0 CLI Reference

OpenShift Enterprise 3.0 CLI Reference

Red Hat OpenShift Documentation Team

OpenShift Enterprise 3.0 CLI Reference

Legal Notice

Abstract

With the OpenShift Enterprise command line interface (CLI), you can create applications and manage OpenShift projects from a terminal. These topics show you how to use CLI.

Table of Contents

CHAPTER 1. OVERVIEW

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

- Working directly with project source code.
- Scripting OpenShift operations.
- Restricted by bandwidth resources and cannot use the web console.

The OpenShift CLI is available using the **oc** command:

```
$ oc <command>
```

You can download and unpack the CLI with an active OpenShift Enterprise subscription from the Red Hat Customer Portal.

Note

The CLI command examples presented through OpenShift documentation use **oc** command syntax. If the **oc** binary is not available on your workstation, you can alternatively substitute **`openshift cli`** in the examples if you have the **`openshift`** binary.

CHAPTER 2. GET STARTED WITH THE CLI

2.1. OVERVIEW

The OpenShift CLI exposes commands for managing your applications, as well as lower level tools to interact with each component of your system. This topic guides you through getting started with the CLI, including installation and logging in to create your first project.

2.2. PREREQUISITES

Certain operations require Git to be locally installed on a client. For example, the command to create an application using a remote Git repository:

```
$ oc new-app https://github.com/<your_user>/<your_git_repo>
```

Before proceeding, install Git on your workstation. See the official Git documentation for instructions per your workstation's operating system.

2.3. INSTALLING THE CLI

You can download and unpack the CLI from the Red Hat Customer Portal for use on Linux, MacOSX, and Windows clients. After logging in with your Red Hat account, you must have an active OpenShift Enterprise subscription to access the downloads page.

Download the CLI

2.4. BASIC SETUP AND LOGIN

The **`oc login`** command is the best way to initially set up the OpenShift CLI, and it serves as the entry point for most users. The interactive flow helps you establish a session to an OpenShift server with the provided credentials. The information is automatically saved in a CLI configuration file that is then used for subsequent commands.

The following example shows the interactive setup and login using the **`oc login`** command:

Example 2.1. Initial CLI Setup

```
$ oc login
OpenShift server [https://localhost:8443]:
https://openshift.example.com 1

Username: alice 2
Authentication required for https://openshift.example.com (openshift)
Password: ******
Login successful. 3

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

```
    $ oc new-project <projectname> 4

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

1

The command prompts for the OpenShift server URL.

2

The command prompts for login credentials: a user name and password.

3

A session is established with the server, and a session token is received.

4

If you do not have a project, information is given on how to create one.

When you have completed the CLI configuration, subsequent commands use the configuration file for the server, session token, and project information.

You can log out of CLI using the **`oc logout`** command:

```
$ oc logout
User, alice, logged out of https://openshift.example.com
```

If you log in after creating or being granted access to a project, a project you have access to is automatically set as the current default, until switching to another one:

```
$ oc login
Username: alice
Authentication required for https://openshift.example.com (openshift)
Password:
Login successful.

Using project "aliceproject".
```

Additional options are also available for the **`oc login`** command.

Note

If you have access to administrator credentials but are no longer logged in as the default system user **system:admin**, you can log back in as this user at any time as long as the credentials are still present in your CLI configuration file. The following command logs in and switches to the **default** project:

```
$ oc login -u system:admin -n default
```

2.5. CLI CONFIGURATION FILES

A CLI configuration file permanently stores **oc** options and contains a series of authentication mechanisms and OpenShift server connection information associated with nicknames.

As described in the previous section, the **`oc login`** command automatically creates and manages CLI configuration files. All information gathered by the command is stored in a configuration file located in ***~/.kube/config***. The current CLI configuration can be viewed using the following command:

Example 2.2. Viewing the CLI Configuration

```
$ oc config view
apiVersion: v1
clusters:
- cluster:
    server: https://openshift.example.com
  name: openshift
contexts:
- context:
    cluster: openshift
    namespace: aliceproject
    user: alice
  name: alice
current-context: alice
kind: Config
preferences: {}
users:
- name: alice
  user:
    token: NDM2N2MwODgtNjI1Yy10N3VhLTg1YmItYzI4NDEzZDUyYzVi
```

CLI configuration files can be used to setup multiple CLI profiles using various OpenShift servers, namespaces, and users so that you can switch easily between them. The CLI can support multiple configuration files; they are loaded at runtime and merged together along with any override options specified from the command line.

2.6. PROJECTS

A project in OpenShift contains multiple objects to make up a logical "application".

Most **oc** commands run in the context of a project. The **oc login** selects a default project during initial setup to be used with subsequent commands. Use the following command to display the project currently in use:

```
$ oc project
```

If you have access to multiple projects, use the following syntax to switch to a particular project by specifying the project name:

```
$ oc project <project_name>
```

For example:

```
$ oc project project02
Now using project 'project02'.

$ oc project project03
Now using project 'project03'.

$ oc project
Using project 'project03'.
```

The **oc status** command shows a high level overview of the project currently in use, with its components and their relationships, as shown in the following example:

```
$ oc status
In project OpenShift 3 Sample (test)

service database-test (172.30.17.113:6434 -> 3306)
  database-test deploys docker.io/library/mysql:latest
    #1 deployed 47 hours ago

service frontend-test (172.30.17.236:5432 -> 8080)
  frontend-test deploys origin-ruby-sample:test <-
    builds https://github.com/openshift/ruby-hello-world with
docker.io/openshift/ruby-20-centos7:latest
    not built yet
    #1 deployment waiting on image

To see more information about a service or deployment config, use 'oc
describe service <name>' or 'oc describe dc <name>'.
You can use 'oc get pods,svc,dc,bc,builds' to see lists of each of the
types described above.
```

2.7. WHAT'S NEXT?

After you have logged in, you can create a new application and explore some common CLI operations.

CHAPTER 3. MANAGING CLI PROFILES

3.1. OVERVIEW

A CLI configuration file allows you to configure different profiles, or *contexts*, for use with the OpenShift CLI. A context consists of user authentication and OpenShift server information associated with a *nickname*.

3.2. SWITCHING BETWEEN CLI PROFILES

Contexts allow you to easily switch between multiple users across multiple OpenShift servers, or *clusters*, when using issuing CLI operations. Nicknames make managing CLI configuration easier by providing short-hand references to contexts, user credentials, and cluster details.

After logging in with the CLI for the first time, OpenShift creates a ***~/.kube/config*** file if one does not already exist. As more authentication and connection details are provided to the CLI, either automatically during an **`oc login`** operation or by setting them explicitly, the updated information is stored in the configuration file:

Example 3.1. CLI Configuration File

```
apiVersion: v1
clusters: 1
- cluster:
    insecure-skip-tls-verify: true
    server: https://openshift1.example.com:8443
  name: openshift1.example.com:8443
- cluster:
    insecure-skip-tls-verify: true
    server: https://openshift2.example.com:8443
  name: openshift2.example.com:8443
contexts: 2
- context:
    cluster: openshift1.example.com:8443
    namespace: alice-project
    user: alice/openshift1.example.com:8443
  name: alice-project/openshift1.example.com:8443/alice
- context:
    cluster: openshift1.example.com:8443
    namespace: joe-project
    user: alice/openshift1.example.com:8443
  name: joe-project/openshift1/alice
current-context: joe-project/openshift1.example.com:8443/alice 3
kind: Config
preferences: {}
users: 4
- name: alice/openshift1.example.com:8443
  user:
    token: xZHd2piv5_9vQrg-SKXRJ2Dsl9SceNJdhNTljEKTb8k
```

1

The **clusters** section defines connection details for OpenShift clusters, including the address for their master server. In this example, one cluster is nicknamed **openshift1.example.com:8443** and another is nicknamed **openshift2.example.com:8443**.

2

This **contexts** section defines two contexts: one nicknamed **alice-project/openshift1.example.com:8443/alice**, using the **alice-project** project, **openshift1.example.com:8443** cluster, and **alice** user, and another nicknamed **joe-project/openshift1.example.com:8443/alice**, using the **joe-project** project, **openshift1.example.com:8443** cluster and **alice** user.

3

The **current-context** parameter shows that the **joe-project/openshift1.example.com:8443/alice** context is currently in use, allowing the **alice** user to work in the **joe-project** project on the **openshift1.example.com:8443** cluster.

4

The **users** section defines user credentials. In this example, the user nickname **alice/openshift1.example.com:8443** uses an access token.

The CLI can support multiple configuration files; they are loaded at runtime and merged together along with any override options specified from the command line.

After you are logged in, you can use the **oc status** command or the **oc project** command to verify your current working environment:

Example 3.2. Verifying the Current Working Environment

```
$ oc status
oc status
In project Joe's Project (joe-project)

service database (172.30.43.12:5434 -> 3306)
  database deploys docker.io/openshift/mysql-55-centos7:latest
    #1 deployed 25 minutes ago - 1 pod

service frontend (172.30.159.137:5432 -> 8080)
  frontend deploys origin-ruby-sample:latest <-
    builds https://github.com/openshift/ruby-hello-world with joe-
project/ruby-20-centos7:latest
    #1 deployed 22 minutes ago - 2 pods
```

```
To see more information about a service or deployment, use 'oc
describe service <name>' or 'oc describe dc <name>'.
You can use 'oc get all' to see lists of each of the types described
above.
```

```
$ oc project
Using project "joe-project" from context named "joe-
project/openshift1.example.com:8443/alice" on server
"https://openshift1.example.com:8443".
```

To log in using any other combination of user credentials and cluster details, run the **oc login** command again and supply the relevant information during the interactive process. A context is constructed based on the supplied information if one does not already exist.

If you are already logged in and want to switch to another project the current user already has access to, use the **oc project** command and supply the name of the project:

```
$ oc project alice-project
Now using project "alice-project" on server
"https://openshift1.example.com:8443".
```

At any time, you can use the **oc config view** command to view your current, full CLI configuration, as seen in the above output.

Additional CLI configuration commands are also available for more advanced usage.

Note

If you have access to administrator credentials but are no longer logged in as the default system user **system:admin**, you can log back in as this user at any time as long as the credentials are still present in your CLI configuration file. The following command logs in and switches to the **default** project:

```
$ oc login -u system:admin -n default
```

3.3. MANUALLY CONFIGURING CLI PROFILES

Note

This section covers more advanced usage of CLI configurations. In most situations, you can simply use the **oc login** and **oc project** commands to log in and switch between contexts and projects.

If you want to manually configure your CLI configuration files, you can use the **oc config** command instead of modifying the files themselves. The **oc config** command includes a number of helpful subcommands for this purpose:

Table 3.1. CLI Configuration Subcommands

Subcommand	Usage
set-credentials	Sets a user entry in the CLI configuration file. If the referenced user nickname already exists, the specified information is merged in. `$ oc config set-credentials <user_nickname> [--client-certificate=<path/to/certfile>] [--client-key=<path/to/keyfile>] [--token=<bearer_token>] [--username=<basic_user>] [--password=<basic_password>]`
set-cluster	Sets a cluster entry in the CLI configuration file. If the referenced cluster nickname already exists, the specified information is merged in. `$ oc config set-cluster <cluster_nickname> [--server=<master_ip_or_fqdn>] [--certificate-authority=<path/to/certificate/authority>] [--api-version=<apiversion>] [--insecure-skip-tls-verify=true]`
set-context	Sets a context entry in the CLI configuration file. If the referenced context nickname already exists, the specified information is merged in. `$ oc config set-context <context_nickname> [--cluster=<cluster_nickname>] [--user=<user_nickname>] [--namespace=<namespace>]`
use-context	Sets the current context using the specified context nickname. `$ oc config use-context <context_nickname>`
set	Sets an individual value in the the CLI configuration file. `$ oc config set <property_name> <property_value>` The **<property_name>** is a dot-delimited name where each token represents either an attribute name or a map key. The **<property_value>** is the new value being set.

Subcommand	Usage
unset	Unsets individual values in the CLI configuration file. `$ oc config unset <property_name>` The **<property_name>** is a dot-delimited name where each token represents either an attribute name or a map key.

view	Displays the merged CLI configuration currently in use.
	`$ oc config view`
	Displays the result of the specified CLI configuration file.
	`$ oc config view --config=<specific_filename>`

Example Usage

Consider the following configuration workflow. First, set credentials for a user nickname **alice** that uses an access token:

```
$ oc config set-credentials alice --
token=NDM2N2MwODgtNjI1Yy10N3VhLTg1YmItYzI4NDEzZDUyYzVi
```

Set a cluster entry named **openshift1**:

```
$ oc config set-cluster openshift1 --
server=https://openshift1.example.com
```

Set a context named **alice** that uses the **alice** user and the **openshift1** cluster:

```
$ oc config set-context alice --cluster=openshift1 --user=alice
```

Now that the **alice** context has been created, switch to that context:

```
$ oc config use-context alice
```

Set the **aliceproject** namespace for the **alice** context:

```
$ oc config set contexts.alice.namespace aliceproject
```

You can now view the configuration that has been created:

```
$ oc config view
apiVersion: v1
```

```
clusters:
- cluster:
    server: https://openshift1.example.com
  name: openshift1
contexts:
- context:
    cluster: openshift1
    namespace: aliceproject
    user: alice
  name: alice
current-context: alice (1)
kind: Config
preferences: {}
users:
- name: alice
  user:
    token: NDM2N2MwODgtNjI1Yy10N3VhLTg1YmItYzI4NDEzZDUyYzVi
```

1

The current context is set to **alice**.

All subsequent CLI operations will use the **alice** context, unless otherwise specified by overriding CLI options or until the context is switched.

3.4. LOADING AND MERGING RULES

When issuing CLI operations, the loading and merging order for the CLI configuration follows these rules:

1. CLI configuration files are retrieved from your workstation, using the following hierarchy and merge rules:
   - If the **`--config`** option is set, then only that file is loaded. The flag may only be set once and no merging takes place.
   - If **`$KUBECONFIG`** environment variable is set, then it is used. The variable can be a list of paths, and if so the paths are merged together. When a value is modified, it is modified in the file that defines the stanza. When a value is created, it is created in the first file that exists. If no files in the chain exist, then it creates the last file in the list.
   - Otherwise, the ***~/.kube/config*** file is used and no merging takes place.
2. The context to use is determined based on the first hit in the following chain:
   - The value of the **`--context`** option.
   - The **`current-context`** value from the CLI configuration file.
   - An empty value is allowed at this stage.

3. The user and cluster to use is determined. At this point, you may or may not have a context; they are built based on the first hit in the following chain, which is run once for the user and once for the cluster:
   - The value of the **`--user`** option for user name and the **`--cluster`** option for cluster name.
   - If the **`--context`** option is present, then use the context's value.
   - An empty value is allowed at this stage.
4. The actual cluster information to use is determined. At this point, you may or may not have cluster information. Each piece of the cluster information is built based on the first hit in the following chain:
   - The values of any of the following command line options:
     - **`--server`**,
     - **`--api-version`**
     - **`--certificate-authority`**
     - **`--insecure-skip-tls-verify`**
   - If cluster information and a value for the attribute is present, then use it.
   - If you do not have a server location, then there is an error.
5. The actual user information to use is determined. Users are built using the same rules as clusters, except that you can only have one authentication technique per user; conflicting techniques cause the operation to fail. Command line options take precedence over configuration file values. Valid command line options are:
   - **`--auth-path`**
   - **`--client-certificate`**
   - **`--client-key`**
   - **`--token`**
6. For any information that is still missing, default values are used and prompts are given for additional information.

CHAPTER 4. CLI OPERATIONS

4.1. OVERVIEW

This topic provides information on the CLI operations and their syntax. You must setup and login with the CLI before you can perform these operations.

4.2. COMMON OPERATIONS

The CLI allows interaction with the various objects that are managed by OpenShift. Many common **oc** operations are invoked using the following syntax:

```
$ oc <action> <object_type> <object_name_or_id>
```

This specifies:

- An ***<action>*** to perform, such as **get** or **describe**.
- The ***<object_type>*** to perform the action on, such as **service** or the abbreviated **svc**.
- The ***<object_name_or_id>*** of the specified ***<object_type>***.

For example, the **oc get** operation returns a complete list of services that are currently defined:

```
$ oc get svc
NAME              LABELS                                    SELECTOR
IP              PORT(S)
docker-registry   docker-registry=default                   docker-
registry=default   172.30.78.158   5000/TCP
kubernetes        component=apiserver,provider=kubernetes   <none>
172.30.0.2      443/TCP
kubernetes-ro     component=apiserver,provider=kubernetes   <none>
172.30.0.1      80/TCP
```

The **oc describe** operation can then be used to return detailed information about a specific object:

```
$ oc describe svc docker-registry
Name:   docker-registry
Labels:   docker-registry=default
Selector:  docker-registry=default
IP:   172.30.78.158
Port:   <unnamed> 5000/TCP
Endpoints:  10.1.0.2:5000
Session Affinity: None
No events.
```

Warning

Versions of **oc** prior to 3.0.2.0 did not have the ability to negotiate API versions against a server. So if you are using **oc** up to 3.0.1.0 with a server that only supports v1 or higher versions of the API, make sure to pass **`--api-version`** in order to point the **oc** client to the correct API endpoint. For example: **`oc get svc --api-version=v1`**.

4.3. BASIC CLI OPERATIONS

The following table describes basic **oc** operations and their general syntax:

Operation	Syntax	Description
`types`	**`oc types`**	Display an introduction to some core OpenShift concepts.
`login`	**`oc login`**	Log in to the OpenShift server.
`logout`	**`oc logout`**	End the current session.
`new-project`	**`oc new-project <project_name>`**	Create a new project.
`new-app`	**`oc new-app .`**	Creates a new application based on the source code in the current directory.
`status`	**`oc status`**	Show an overview of the current project.
`project`	**`oc project <project_name>`**	Switch to another project. Run without options to display the current project. To view all projects you have access to run **`oc projects`**.

4.4. APPLICATION MODIFICATION CLI OPERATIONS

Operation	Syntax	Description
`get`	`oc get <object_type> [<object_name_or_id>]`	Return a list of objects for the specified object type. If the optional **`<object_name_or_id>`** is included in the request, then the list of results is filtered by that value.
`describe`	`oc describe <object_type> <object_id>`	Returns information about the specific object returned by the query. A specific **`<object_name_or_id>`** must be provided. The actual information that is available varies as described in object type.
`edit`	`oc edit <object_type>/<object_type_name>`	Edit the desired object type.
	`OC_EDITOR="<text_editor>" oc edit \<object_type>/<object_type_name>`	Edit the desired object type with a specified text editor.
	`oc edit <object_type>/<object_type_name> \--output-version=<object_type_version> \-o <object_type_format>`	Edit the desired object in a specified format (eg: JSON).
`env`	`oc env <object_type>/<object_type_name> \<EN_VAR>=/<VALUE>`	Update the desired object type with a new environment variable
`volume`	`oc volume <object_type>/<object_type_name> \[--option]`	Modify a volume.
`label`	`oc label <object_type> <object_name_or_id> \<label>`	Update the labels on a object.

Operation	Syntax	Description
`expose`	`oc expose <object_type> <object_name_or_id>`	Look up a service and expose it as a route. There is also the ability to expose a deployment configuration, replication controller, service, or pod as a new service on a specified port. If no labels are specified, the new object will re-use the labels from the object it exposes.
`stop`	`oc stop -f <file_path>`	Gracefully shut down an object by ID or file name. Attempt to shut down and delete an object that supports graceful termination.
	`oc stop <object_type> <object_name_or_id>`	Gracefully shut down an object with the specified ID.
	`oc stop <object_type> -l <label>`	Gracefully shut down an object with the specified label.
	`oc stop all -l <label>`	Gracefully shut down all objects with the specified label.

`delete`	`oc delete -f <file_path>`
	`oc delete <object_type> <object_name_or_id>`
	`oc delete <object_type> -l <label>`
	`oc delete all -l <label>`
		Delete the specified object. An object configuration can also be passed in through STDIN. The **`oc delete all -l <label>`** operation deletes all objects matching the specified **`<label>`**, including the replication controller so that pods are not re-created.

4.5. BUILD AND DEPLOYMENT CLI OPERATIONS

One of the fundamental capabilities of OpenShift is the ability to build applications into a container from source. The following table describes the CLI operations for working with application builds:

OpenShift provides CLI access to inspect and manipulate deployment configurations using standard **`oc`** resource operations, such as **`get`**, **`create`**, and **`describe`**.

Operation	Syntax	Description
start-build	`oc start-build <buildConfig_name>`	Manually start the build process with the specified build configuration file.
	`oc start-build --from-build=<build_name>`	Manually start the build process by specifying the name of a previous build as a starting point.
	`oc start-build \<buildConfig_name> --follow` `oc start-build \--from-build=<build_name> --follow`	Manually start the build process by specifying either a configuration file or the name of a previous build and retrieves its build logs.
build-logs	`oc build-logs <build_name>`	Retrieve the build logs for the specified build.
deploy	`oc deploy <deploymentconfig>`	View a deployment, or manually start, cancel, or retry a deployment.
rollback	`oc rollback <deployment_name>`	Perform a rollback.
new-build	`oc new-build .`	Create a build config based on the source code in the current git repository (with a public remote) and a Docker image
cancel-build	`oc cancel-build <build_name>`	Stop a build that is in progress.
import-image	`oc import-image <imagestream>`	Import tag and image information from an external Docker image repository.

Operation	Syntax	Description
`scale`	`oc scale <object_type> <object_id> \--replicas=<#_of_replicas>`	Set the number of desired replicas for a replication controller or a deployment configuration to the number of specified replicas.
`tag`	`oc tag <current_image> <image_stream>`	Take an existing tag or image from an image stream, or a Docker image pull spec, and set it as the most recent image for a tag in one or more other image streams.

4.6. ADVANCED COMMANDS

Operation	Syntax	Description
`create`	`oc create -f <file_or_dir_path>`	Parse a configuration file and create one or more OpenShift objects based on the file contents. The `-f` flag can be passed multiple times with different file or directory paths. When the flag is passed multiple times, `oc create` iterates through each one, creating the objects described in all of the indicated files. Any existing resources are ignored.
`update`	`oc update -f <file_or_dir_path>`	Attempt to modify an existing object based on the contents of the specified configuration file. The `-f` flag can be passed multiple times with different file or directory paths. When the flag is passed multiple times, `oc update` iterates through each one, updating the objects described in all of the indicated files.
`process`	`oc process -f <template_file_path>`	Transform a project template into a project configuration file.
`export`	`oc export <object_type> [--options]`	Export resources to be used elsewhere
`policy`	`oc policy [--options]`	Manage authorization policies

Operation	Syntax	Description
`secrets`	`oc secrets [--options] path/to/ssh_key`	Configure secrets.

4.7. TROUBLESHOOTING AND DEBUGGING CLI OPERATIONS

Operation	Syntax	Description
`logs`	`oc logs -f <pod_name> <container_name>`	Retrieve the log output for a specific pod or container. This command does not work for other object types.
`exec`	`oc exec <pod_ID> \[-c <container_ID>] <command>`	Execute a command in an already-running container. You can optionally specify a container ID, otherwise it defaults to the first container.
`rsh`	`oc rsh <pod_ID>`	Open a remote shell session to a container.
`port-forward`	`oc port-forward <pod_ID> \<first_port_ID> <second_port_ID>`	Forward one or more local ports to a pod.
`proxy`	`oc proxy --port=<port_ID> \--www=<static_directory>`	Run a proxy to the Kubernetes API server

Important

For security purposes, the **`oc exec`** command does not work when accessing privileged containers. Instead, administrators can SSH into a node host, then use the **`docker exec`** command on the desired container.

4.8. OBJECT TYPES

The CLI supports the following object types, some of which have abbreviated syntax:

Object Type	Abbreviated Version
build	
buildConfig	bc
deploymentConfig	dc
imageStream	is
imageStreamTag	istag
imageStreamImage	isimage
event	ev
node	
pod	po
replicationController	rc
service	svc
persistentVolume	pv
persistentVolumeClaim	pvc

CHAPTER 5. REVISION HISTORY: CLI REFERENCE

5.1. THU MAY 19 2016

Affected Topic	Description of Change
Get Started with the CLI	Updated the example in the Projects section to use `https` for GitHub access.
Managing CLI Profiles	Updated the example in the Switching Between CLI Profiles section to use `https` for GitHub access.

5.2. TUE APR 19 2016

Affected Topic	Description of Change
Get Started with the CLI	Added a Prerequisites section.

5.3. TUE JUN 23 2015

OpenShift Enterprise 3.0 release.