docs.openshift.com

CLI Operations | CLI Reference

9-12 Minuten

You are viewing documentation for a release that is no longer supported. The latest supported version of version 3 is [3.11]. For the most recent version 4, see [4]

You are viewing documentation for a release that is no longer supported. The latest supported version of version 3 is [3.11]. For the most recent version 4, see [4]

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.

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.
```

> 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`.

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`.

Application Modification CLI Operations

Operation	Syntax	Description
`get`	`oc get <object_type> [<object_name_or_id>]`	Return a list of objec the specified object t

Operation	Syntax	Description
		If the optional *<object_name_or_* is included in the req then the list of results filtered by that value.
describe	`oc describe <object_type> <object_id>`	Returns information about the specific obj returned by the query specific *<object_name_or_* must be provided. Th actual information tha available varies as described in object ty
edit	`oc edit <object_type>/<object_type_name>`	Edit the desired obje type.
	`OC_EDITOR="<text_editor>" oc edit \ <object_type>/<object_type_name>`	Edit the desired obje type with a specified editor.
	`oc edit <object_type>/<object_type_name> \ --output-version= <object_type_version> \ -o <object_type_format>`	Edit the desired obje a specified format (eg JSON).
env	`oc env <object_type>/<object_type_name> \ <EN_VAR>=/<VALUE>`	Update the desired o type with a new environment variable

Operation	Syntax	Description
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 object.
expose	`oc expose <object_type> <object_name_or_id>`	Look up a service an expose it as a route. There is also the abil expose a deploymen configuration, replica controller, service, or as a new service on a specified port. If no la are specified, the nev object will re-use the labels from the objec exposes.
stop	`oc stop -f <file_path>`	Gracefully shut down object by ID or file na Attempt to shut down delete an object that supports graceful termination.
	`oc stop <object_type> <object_name_or_id>`	Gracefully shut down object with the specif ID.
	`oc stop <object_type> -l <label>`	Gracefully shut down object with the specif

Operation	Syntax	Description
		label.
	`oc stop all -l <label>`	Gracefully shut down objects with the spec 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 als passed in through STDIN. The `oc del all -l <label>` operation deletes all objects matching the specified `<label>`, including the replicati controller so that pod are not re-created.

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

Operation	Syntax	Description
		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

Operation	Syntax	Description
		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.
`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.

Advanced Commands

Operation	Syntax	Description
`create`	`oc create -f <file_or_dir_path>`	Parse a configuration file and create one or

Operation	Syntax	Description
		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.

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

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.

Operation	Syntax	Description
`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

> 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.

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`

Object Type	Abbreviated Version
imageStreamImage	isimage
event	ev
node	
pod	po
replicationController	rc
service	svc
persistentVolume	pv
persistentVolumeClaim	pvc