



Red Hat build of MicroShift 4.12

CLI tools

Learning how to use the command-line tools for MicroShift

Red Hat build of MicroShift 4.12 CLI tools

Learning how to use the command-line tools for MicroShift

Legal Notice

Copyright © 2023 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux[®] is the registered trademark of Linus Torvalds in the United States and other countries.

Java[®] is a registered trademark of Oracle and/or its affiliates.

XFS[®] is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL[®] is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js[®] is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack[®] Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

Abstract

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

Table of Contents

CHAPTER 1. RED HAT BUILD OF MICROSHIFT CLI TOOLS	3
CHAPTER 2. GETTING STARTED WITH THE OPENSIFT CLI	4
2.1. INSTALLING THE OPENSIFT CLI	4
2.1.1. Installing the OpenShift CLI by downloading the binary	4
Installing the OpenShift CLI on Linux	4
Installing the OpenShift CLI on Windows	4
Installing the OpenShift CLI on macOS	5
2.1.2. Installing the OpenShift CLI by using Homebrew	5
2.1.3. Installing the OpenShift CLI by using an RPM	6
CHAPTER 3. CONFIGURING THE OPENSIFT CLI	7
3.1. ENABLING TAB COMPLETION	7
3.1.1. Enabling tab completion for Bash	7
3.1.2. Enabling tab completion for Zsh	7
CHAPTER 4. USING THE OC TOOL	9
4.1. ABOUT THE OPENSIFT CLI	9
4.2. USING THE OPENSIFT CLI IN RED HAT BUILD OF MICROSHIFT	9
4.2.1. Viewing pods	9
4.2.2. Viewing pod logs	9
4.2.3. Listing supported API resources	10
4.3. GETTING HELP	10
4.4. OC COMMAND ERRORS IN RED HAT BUILD OF MICROSHIFT	11
CHAPTER 5. USING OC AND KUBECTL COMMANDS	12
5.1. THE OC BINARY	12
5.2. THE KUBECTL BINARY	13
CHAPTER 6. OPENSIFT CLI COMMAND REFERENCE	14
6.1. OC COMMANDS LIST FOR RED HAT BUILD OF MICROSHIFT	14
6.1.1. oc apply	14
6.1.2. oc delete	14
6.1.3. oc get	15

CHAPTER 1. RED HAT BUILD OF MICROSHIFT CLI TOOLS

A user builds, deploys, and manages both applications and clusters while working with Red Hat build of MicroShift.

Red Hat build of MicroShift can use different command-line interface (CLI) tools that simplify these tasks by enabling users to perform various administration and development operations from the terminal. These tools expose simple commands to manage the deployments, as well as interact with each component of the system.

In addition to built-in **microshift** command types and Linux CLI tools, the optional OpenShift CLI (**oc**) tool with an enabled subset of commands is available for you to use if you are already familiar with OpenShift Container Platform and Kubernetes.

Additional resources

- [Installing the **oc** tool for MicroShift.](#)
- [OpenShift CLI \(oc\)](#): A full description of **oc** as provided by the OpenShift Container Platform documentation. Commands focused on multi-node deployments, projects, and developer tooling are not supported by Red Hat build of MicroShift.
- [Red Hat Enterprise Linux \(RHEL\)](#): The RHEL documentation for your specific use case.

CHAPTER 2. GETTING STARTED WITH THE OPENSIFT CLI

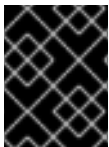
To use the OpenShift CLI (**oc**) tool, you must download and install it separately from your Red Hat build of MicroShift installation.

2.1. INSTALLING THE OPENSIFT CLI

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

2.1.1. Installing the OpenShift CLI by downloading the binary

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



IMPORTANT

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in Red Hat build of MicroShift 4.12. Download and install the new version of **oc**.

Installing the OpenShift CLI on Linux

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

Procedure

1. Navigate to the [OpenShift Container Platform downloads page](#) on the Red Hat Customer Portal.
2. Select the architecture from the **Product Variant** drop-down list.
3. Select the appropriate version from the **Version** drop-down list.
4. Click **Download Now** next to the **OpenShift v4.12 Linux Client** entry and save the file.
5. Unpack the archive:

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

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

```
$ echo $PATH
```

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

```
$ oc <command>
```

Installing the OpenShift CLI on Windows

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

Procedure

1. Navigate to the [OpenShift Container Platform downloads page](#) on the Red Hat Customer Portal.
2. Select the appropriate version from the **Version** drop-down list.
3. Click **Download Now** next to the **OpenShift v4.12 Windows Client** entry and save the file.
4. Unzip the archive with a ZIP program.
5. Move the **oc** binary to a directory that is on your **PATH**.
To check your **PATH**, open the command prompt and execute the following command:

```
C:\> path
```

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

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

Installing the OpenShift CLI on macOS

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

Procedure

1. Navigate to the [OpenShift Container Platform downloads page](#) on the Red Hat Customer Portal.
2. Select the appropriate version from the **Version** drop-down list.
3. Click **Download Now** next to the **OpenShift v4.12 macOS Client** entry and save the file.
4. Unpack and unzip the archive.
5. Move the **oc** binary to a directory on your PATH.
To check your **PATH**, open a terminal and execute the following command:

```
$ echo $PATH
```

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

```
$ oc <command>
```

2.1.2. Installing the OpenShift CLI by using Homebrew

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

Prerequisites

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

Procedure

- Run the following command to install the [openshift-cli](#) package:

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

2.1.3. Installing the OpenShift CLI by using an RPM

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

Prerequisites

- Must have root or sudo privileges.

Procedure

1. Register with Red Hat Subscription Manager:

```
# subscription-manager register
```

2. Pull the latest subscription data:

```
# subscription-manager refresh
```

3. List the available subscriptions:

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

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

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

5. Enable the repositories required by Red Hat build of MicroShift 4.12.

```
# subscription-manager repos --enable="rhocp-4.12-for-rhel-8-x86_64-rpms"
```



NOTE

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

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

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

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

```
$ oc <command>
```

CHAPTER 3. CONFIGURING THE OPENSIFT CLI

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

3.1. ENABLING TAB COMPLETION

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

3.1.1. Enabling tab completion for Bash

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

Prerequisites

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

Procedure

1. Save the Bash completion code to a file:

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

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

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

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

Tab completion is enabled when you open a new terminal.

3.1.2. Enabling tab completion for Zsh

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

Prerequisites

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

Procedure

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

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

```
| compdef _oc oc  
fi  
EOF
```

Tab completion is enabled when you open a new terminal.

CHAPTER 4. USING THE oc TOOL

The optional OpenShift CLI (**oc**) tool is available for you to use if you are already familiar with OpenShift Container Platform and Kubernetes.

4.1. ABOUT THE OPENSIFT CLI

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

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

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

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

4.2.1. Viewing pods

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



NOTE

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

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

Example output

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

4.2.2. Viewing pod logs

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

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

Example output

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

4.2.3. Listing supported API resources

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

```
$ oc api-resources
```

Example output

NAME	SHORTNAMES	APIGROUP	NAMESPACED	KIND
bindings		binding.k8s.io	true	Binding
componentstatuses	cs		false	ComponentStatus
configmaps	cm		true	ConfigMap
...				

4.3. GETTING HELP

You can get help with CLI commands and Red Hat build of MicroShift resources in the following ways.

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

Example: Get help for the **oc create** command

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

Example output

```
Create a resource by filename or stdin

JSON and YAML formats are accepted.

Usage:
  oc create -f FILENAME [flags]

...
```

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

Example: View documentation for the **Pod** resource

```
$ oc explain pods
```

Example output

```
KIND:   Pod
VERSION: v1

DESCRIPTION:
  Pod is a collection of containers that can run on a host. This resource is
```

created by clients and scheduled onto hosts.

FIELDS:

apiVersion <string>

APIVersion defines the versioned schema of this representation of an object. Servers should convert recognized schemas to the latest internal value, and may reject unrecognized values. More info:

<https://git.k8s.io/community/contributors/devel/api-conventions.md#resources>

...

4.4. OC COMMAND ERRORS IN RED HAT BUILD OF MICROSHIFT

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

Example output

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

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

The error following error message can be generated:

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

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

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

CHAPTER 5. USING OC AND KUBECTL COMMANDS

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

5.1. THE OC BINARY

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

- **Route resource**
The **Route** resource object is specific to Red Hat build of MicroShift distributions, and builds upon standard Kubernetes primitives.
- **Additional commands**
The additional command **oc new-app**, for example, makes it easier to get new applications started using existing source code or pre-built images.



IMPORTANT

If you installed an earlier version of the **oc** binary, you cannot use it to complete all of the commands in Red Hat build of MicroShift 4.12. If you want the latest features, you must download and install the latest version of the **oc** binary corresponding to your Red Hat build of MicroShift server version.

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

Table 5.1. Compatibility Matrix

	X.Y (oc Client)	X.Y+N footnote:versionpolicyn[Where N is a number greater than or equal to 1.] (oc Client)
X.Y (Server)	1	3
X.Y+N footnote:versionpolicyn[] (Server)	2	1

1

Fully compatible.

2

oc client might not be able to access server features.

3

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

5.2. THE KUBECTL BINARY

The **kubectl** binary is provided as a means to support existing workflows and scripts for new Red Hat build of MicroShift 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 Red Hat build of MicroShift cluster.

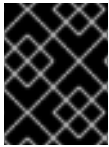
The **kubectl** binary is included in the archive if you download the **oc** binary.

For more information, see the [kubectl documentation](#).

CHAPTER 6. OPENSIFT CLI COMMAND REFERENCE

This reference provides descriptions and example commands for OpenShift CLI (**oc**) commands. You must have **cluster-admin** or equivalent permissions to use these commands.

Run **oc adm -h** to list all administrator commands or run **oc <command> --help** to get additional details for a specific command.



IMPORTANT

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

6.1. OC COMMANDS LIST FOR RED HAT BUILD OF MICROSHIFT

The following lists a few examples of **oc** commands you can use to administer, deploy, and observe a Red Hat build of MicroShift node.

6.1.1. oc apply

Apply a configuration to a resource by file name or stdin

Example usage

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

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

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

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

```
# Note: --prune is still in Alpha
# Apply the configuration in manifest.yaml that matches label app=nginx and delete all other resources that are not in the file and match label app=nginx
oc apply --prune -f manifest.yaml -l app=nginx
```

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

6.1.2. oc delete

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

Example usage

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

```

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

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

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

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

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

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

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

# Delete all pods
oc delete pods --all

```

6.1.3. oc get

Display one or many resources

Example usage

```

# List all pods in ps output format
oc get pods

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

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

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

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

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

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

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

```

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

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

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

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