OpenShift Cheat Sheet

RED HAT® DEVELOPER PROGRAM

Table of Contents

What is Openshift?

From OpenShift.com:

> OpenShift is Red Hat's Platform-as-a-Service (PaaS) that allows developers to quickly develop, host, and scale applications in a cloud environment.

Openshift makes use of the Kubernetes upstream project to provide a secure, robust, and extendable manner for orchestrating applications. Openshift works to further the access management and build/deploy services that are provided in the upstream Kubernetes project. Development teams are empowered to own and maintain their applications through production environments, while operations teams can provide the guide rails for developers to have that application ownership in a multi-tenant environment.

Cheatsheet Guide

This guide is mostly focused on the developer experience, however several administrator tasks are detailed below. A high-level listing of operations from command line interface is provided, followed by a walkthrough of an example application build, deploy, and management. The command list is not exhaustive, but does cover the majority of operations a developer will need to understand to manage an application's lifecycle.

Command Overview

Login/User management

`oc login`	authenticate to an openshift cluster
`oc logout`	end the current session
`oc whoami`	show the current user context

Project management

`oc project`	show the current project context
`oc get projects`	show all project current login has access to
`oc status`	show overview of current project resources
`oc new-project`	create a new project in Openshift and change to that context

Resource management

`oc new-app`	create a new application from from source code, container image, or OpenShift template
`oc new-build`	create a new build configuration from source code
`oc label`	add/update/remove labels from an Openshift resource
`oc annotate`	add/update/remove annotations from an Openshift resource
`oc create`	create a new resource from filename or stdin
`oc get`	retrieve a resource (use -o for additional output options)
`oc replace`	replace an existing resource from filename or stdin
`oc delete`	delete a resource
`oc edit`	modify a resource from text editor
`oc describe`	retrieve a resource with details

Cluster management

`oc adm`	administrative functions for an openshift cluster
`oc adm router|registry`	install a router or registry
`oc adm policy`	manage role/scc to user/group bindings, as well as additional policy administration
`oc adm diagnostics`	run tests/validation against a cluster
`oc adm cordon/uncordon/drain`	unschedule/schedule/drain a node
`oc adm groups`	manage groups
`oc adm top`	show usage statistics of resources

Additional resource management

`oc patch`	Update fields for a resource with JSON or YAML segments
`oc extract`	get configmaps or secrets and save to disk
`oc set`	Modify miscellaneous application resources
`oc set probe`	Add a readiness/liveness probe on pod template/deployment configuration
`oc set volumes`	Manage volume types on a pod template/deployment configuration
`oc set build-hook`	Set a script/command to execute as part of the build process
`oc set build-secret`	set a secret to be included as part of the build process
`oc set env`	set environment variables on a pod template/deployment configuration/build configuration
`oc set image`	update the image for deployment configurations/ daemonsets
`oc set triggers`	set triggers for deployment configurations/build configurations

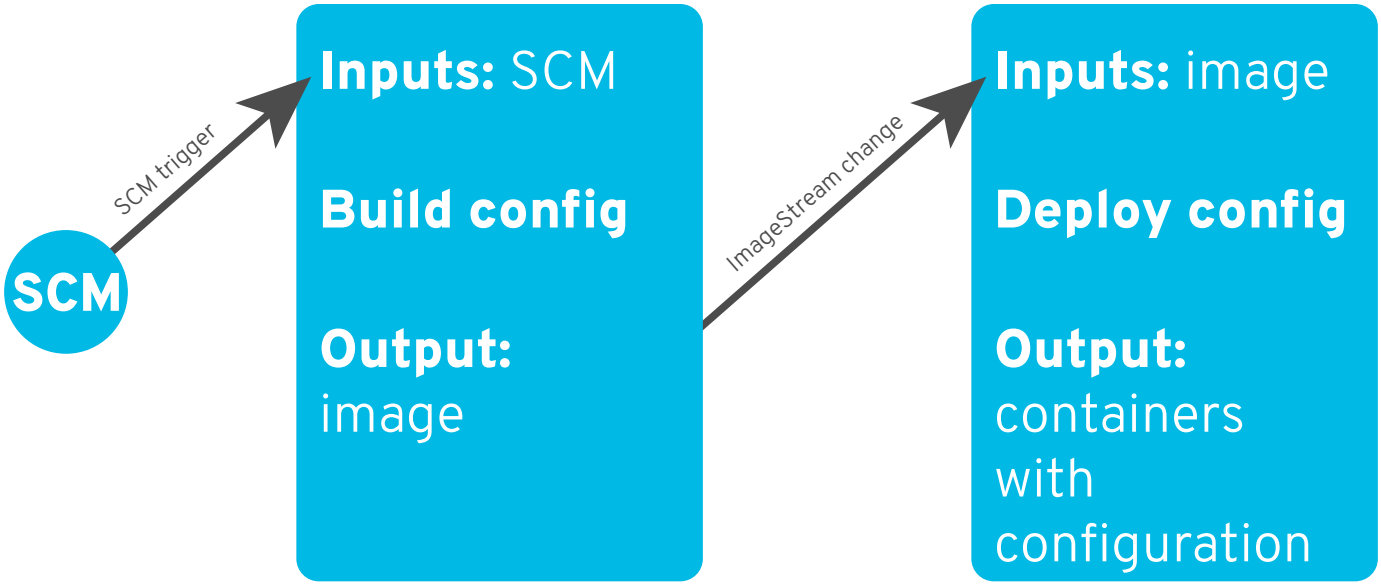
Operational commands

`oc logs`	retrieve the logs for a resource (build configurations, deployment configurations, and pods)
`oc rsh`	remote shell into a container
`oc rsync`	copy files to or from a container
`oc exec`	execute a command in a container
`oc run`	create a deployment configuration from image
`oc idle`	scale resources to zero replicas

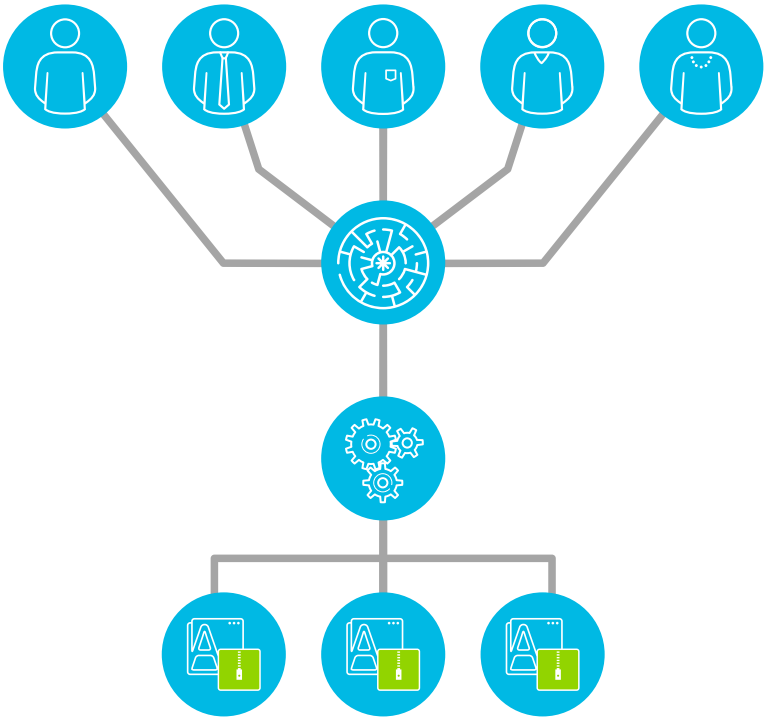
Build / Deploy

`oc rollout`	manage deployments from deployment configuration
`oc rollout latest`	start a new deployment with the latest state
`oc rollout undo`	perform a rollback operation
`oc rollout history`	oc rollout history - View historical information for a deployment configuration
`oc rollout status`	watch the status of a rollout until complete
`oc tag`	tag existing images into image streams
`oc start-build`	start a new build from a build configuration
`oc cancel-build`	cancel a build in progress
`oc import-image`	pull in images and tags from an external Docker registry
`oc scale`	change the number of pod replicas for a deployment

Simple Build/Deploy Overview



Simple Routing Overview



User traffic

Route

Service
Port mapping defined in Deploy Config

App Container
Image and number of replicas defined in Deploy Config

Volume
Volume configuration defined in Deploy Config

Examples

Login

First, we can login to the cluster to interact with Openshift via CLI

```
$ oc login -u myuser https://openshift.example.com
Authentication required for https://openshift.example.com
Username: myuser
Password:
```

Note that leaving the -p option off of login prompts for password. Additionally we can verify our user context:

```
$ oc whoami
myuser
```

Create Project

Let's list out our current available projects (those that we have at least view access for):

```
$ oc get projects
```

If this is our first login and no one has added us to any existing projects, there shouldn't be any projects listed. Let's create a project (allowed by self-provisioner role to all authenticated users, in the default Openshift policy installation).

```
$ oc new-project myproject --display-name='My Project' --description='cool project owned by myuser'
```

Now using project "myproject" on server "https://openshift.example.com:443".

To build a new example applicatin on Ruby you can add applications to this project with the 'new-app' command. For example, try:

```
oc new-app centos/ruby-22-centos7~https://github.com/openshift/ruby-ex.git
```

If you want to view the specifics of the project definition, output the full spec to YA

```
$ oc get project myproject1 -o yaml
apiVersion: v1
kind: Project
metadata:
 annotations:
  openshift.io/description: A really cool project owned by myuser
  openshift.io/display-name: My Project
  openshift.io/requester: myuser
  openshift.io/sa.scc.mcs: s0:c51,c20
  openshift.io/sa.scc.supplemental-groups: 1000000000/10000
  openshift.io/sa.scc.uid-range: 1000000000/10000
  creationTimestamp: 2017-02-10T15:36:18Z
 labels:
  name: myproject
  resourceVersion: "32381158"
  selfLink: /oapi/v1/projects/myproject
  uid: aa94c906-efa6-11e6-af71-02a55ffb157d
 spec:
  finalizers:
   - openshift.io/origin
   - kubernetes
 status:
  phase: Active
```

Add users to project

We can add additional users to our project by default, since self-provisioners get the "admin" role for any project they create:

```
$ oc adm policy add-role-to-user edit anotheruser
```

This allows anotheruser to edit resources within the project, but not manage policy

Create app from code and image

```
$ oc new-app centos/ruby-22-centos7~https://github.com/openshift/ruby-ex.git
--> Found Docker image 06f0cdc (2 days old) from Docker Hub for "centos/ruby-22-
centos7"
Ruby 2.2
--------
Platform for building and running Ruby 2.2 applicationsTags: builder, ruby, ruby22
* An image stream will be created as "ruby-22-centos7:latest" that will track the
source image
* A source build using source code from https://github.com/openshift/ruby-ex.git
will be created
* The resulting image will be pushed to image stream "ruby-ex:latest"
* Every time "ruby-22-centos7:latest" changes a new build will be triggered
* This image will be deployed in deployment config "ruby-ex"
* Port 8080/tcp will be load balanced by service "ruby-ex"
* Other containers can access this service through the hostname "ruby-ex"
--> Creating resources with label app=ruby-ex ...
imagestream "ruby-22-centos7" created
imagestream "ruby-ex" created
buildconfig "ruby-ex" created
deploymentconfig "ruby-ex" created
service "ruby-ex" created
--> Success
Build scheduled, use 'oc logs -f bc/ruby-ex' to track its progress.
Run 'oc status' to view your app.
```

The new-app command handles the majority of resource creation via template. Notice that deploymentconfig/buildconfig/service/imagestream were all set up.

Get resources

We can view the resources that were created as part of the new-app command, as well as the build/deploy resources that were created automatically. Notice that the new-app automatically started a new build of our code, and the deployment config watches successful builds to know when to next rollout/deploy. A good place to start with viewing application status is checking the pods in your project:

```
$ oc get pods
NAME READY STATUS RESTARTS AGE
ruby-ex-1-a7y56 1/1 Running 0 24m
ruby-ex-1-build 0/1 Completed 0 26m
```

This shows us the build pod completed successfully. Additionally we can see that there is one ready and running pod deployed with our application.

The status command shows us similar results:

```
$ oc status -v
In project My Project (myproject1) on server https://openshift.example.com:443
svc/ruby-ex - 172.30.36.21:8080
dc/ruby-ex deploys istag/ruby-ex:latest <-
bc/ruby-ex source builds https://github.com/openshift/ruby-ex.git on istag/ruby-
22-centos7:latest
deployment #1 deployed 26 minutes ago - 1 pod
Warnings:
* dc/ruby-ex has no readiness probe to verify pods are ready to accept traffic or
ensure deployment is successful.
try: oc set probe dc/ruby-ex --readiness ...
View details with 'oc describe <resource>/<name>' or list everything with 'oc get
all'.
```

Add a volume

If we want to attach a volume to our pods, the oc set volume command can be used:

```
$ oc set volume dc/ruby-ex --add --mount-path=/mnt/emptydir
info: Generated volume name: volume-7d1e8
deploymentconfigs/ruby-ex


$ oc get pods
NAME READY STATUS RESTARTS AGE
ruby-ex-1-a7y56 1/1 Running 0 2h
ruby-ex-1-build 0/1 Completed 0 2h
ruby-ex-2-deploy 0/1 ContainerCreating 0 5s
```

In this example, a simple emptyDir volume was attached, though the same command can be used for Persistent Volumes. Also notice that the deployment configuration has a ConfigChange trigger, so adding this volume automatically started a new deployment.

Edit resource

Making a change to any Openshift resource is simple. Let's change the /mnt/emptydir mountpath above to /mnt/appdata:

```
$ oc edit dc ruby-ex
# Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving this file
will be
# reopened with the relevant failures.
#
...
volumeMounts:
- mountPath: /mnt/emptydir /mnt/appdata
 name: volume-7d1e8
...
```

Saving the file in your text editor will update the resource, or report errors if validation did not succeed. Note that this change on the deployment config kicks off another deployment for our app.

Start build

If a new build from source is desired:

```
$ oc start-build ruby-ex
build "ruby-ex-2" started
```

Watch build

The build logs can be watched with the oc logs command (including -f option for follow):

```
$ oc logs -f bc/ruby-ex
Cloning "https://github.com/openshift/ruby-ex.git" ...
Commit: 855ab2de53ff897a19e1055f7554c64d19e02c50 (Merge pull request #6 from aj07/
typo)
Author: Ionut Palade <PI-Victor@users.noreply.github.com>
Date: Mon Dec 12 14:37:32 2016 +0100
---> Installing application source ...
---> Building your Ruby application from source ...
---> Running 'bundle install --deployment --without development:test' ...
Fetching gem metadata from https://rubygems.org/..............
Installing puma 3.4.0
Installing rack 1.6.4
Using bundler 1.7.8
Your bundle is complete!
Gems in the groups development and test were not installed.
It was installed into ./bundle
---> Cleaning up unused ruby gems ...


Pushing image 172.30.114.236:5000/myproject/ruby-ex:latest ...
Pushed 7/9 layers, 78% complete
Pushed 8/9 layers, 89% complete
Pushed 9/9 layers, 100% complete
Push successful
```

Start Deploy

Most configuration or image changes will automatically start a new deploy by default, but new deployments can be started manually as well:

```
$ oc rollout latest ruby-ex
deploymentconfig "ruby-ex" rolled out
```

Watch Deploy

The overall deployment status can be watched via oc logs command:

```
$ oc logs -f dc/ruby-ex
--> Scaling up ruby-ex-5 from 0 to 1, scaling down ruby-ex-4 from 1 to 0 (keep 1
pods available, don't exceed 2 pods)
Scaling ruby-ex-5 up to 1
Scaling ruby-ex-4 down to 0
--> Success
```

Additionally container logs can be viewed with oc logs:

```
$ oc logs ruby-ex-5-kgzvd
[1] Puma starting in cluster mode...
[1] * Version 3.4.0 (ruby 2.2.2-p95), codename: Owl Bowl Brawl
[1] * Min threads: 0, max threads: 16
[1] * Environment: production
[1] * Process workers: 8
[1] * Phased restart available
[1] * Listening on tcp://0.0.0.0:8080
[1] Use Ctrl-C to stop
[1] - Worker 2 (pid: 29) booted, phase: 0
[1] - Worker 1 (pid: 25) booted, phase: 0
[1] - Worker 5 (pid: 41) booted, phase: 0
[1] - Worker 3 (pid: 33) booted, phase: 0
[1] - Worker 0 (pid: 21) booted, phase: 0
[1] - Worker 4 (pid: 37) booted, phase: 0
[1] - Worker 6 (pid: 45) booted, phase: 0
[1] - Worker 7 (pid: 60) booted, phase: 0
```

Remote shell

Interacting directly with the container is simple with oc rsh:

```
$ oc rsh ruby-ex-5-kgzvd
sh-4.2$ ls
Gemfile Gemfile.lock README.md bundle config.ru
```

Create route

```
 $ oc expose service ruby-ex
route "ruby-ex" exposed
```

With no other options defined this will create a route for your application using the default route naming (ex: $appname-$projectname.openshift.example.com)

Idle app

We're done testing our application, so we can idle the service in order to save resources. This interacts with a Kubernetes service to set the pod replicas to 0, and when the service is next accessed will automatically boot up the pods again:

```
$ oc idle ruby-ex
Marked service myproject1/ruby-ex to unidle resource DeploymentConfig myproject1/
ruby-ex (unidle to 1 replicas)
Idled DeploymentConfig myproject1/ruby-ex
```

Delete app

If we're completely done with our application, we can delete resources within the project (or the project itself) to clean up:

```
$ oc delete services -l app=ruby-ex
service "ruby-ex" deleted


$ oc delete all -l app=ruby-ex
buildconfig "ruby-ex" deleted
imagestream "ruby-22-centos7" deleted
imagestream "ruby-ex" deleted
deploymentconfig "ruby-ex" deleted


$ oc delete project myproject
project "myproject" deleted
```

Additional Operations (not included in upstream guide)

Sometimes with automation there's a desire to control YAML resources in SCM, and ensure those resources match the current state as defined in code. However, we may not know if a resource requires an add or update operation via the Openshift CLI. The following can be used to idempotently add/update most resources regardless of their current state:

```
$ oc create -f myresource.yaml || oc replace -f myresource.yaml
```

Gather information on a project's pod deployment with node information

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

Note: Often can be used with 'watch' command to monitor in real-time the status of pods for a project.

See interesting actions within a project:

```
oc get events --sort-by='{.lastTimestamp}'
```

or in reverse order:

```
oc get events --sort-by='{.lastTimestamp}' | tac
```

About the author