Login and Configuration

#login with a user

oc login https://192.168.99.100:8443 -u developer -p developer

#login as system admin

oc login -u system:admin

#User Information

oc whoami

#View your configuration

oc config view

#Update the current context to have users login to the desired namespace:

oc config set-context `oc config current-context` --namespace=<project\_name>

## Basic Commands

#Create a new app from a GutHub Repository

oc new-app https://github.com/sclorg/cakephp-ex

#New app from a different branch

oc new-app --name=html-dev nginx:1.10~https://github.com/joe-speedboat/openshift.html.devops.git#mybranch

#Create objects from a file:

oc create -f myobject.yaml -n myproject

#Delete objects contained in a file:

oc delete -f myobject.yaml -n myproject

#Create or merge objects from file

oc apply -f myobject.yaml -n myproject

#Update existing object

oc patch svc mysvc --type merge --patch '{"spec":{"ports":[{"port": 8080, "targetPort": 5000 }]}}'

#Monitor Pod status

watch oc get pods

#Get a Specific Item (podIP) using a Go template

oc get pod example-pod-2 --template='{{.status.podIP}}'

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

oc get pods -o wide

#Hide inactive Pods

oc get pods --show-all=false

#Display all resources

oc get all,secret,configmap

#Get the Openshift Console Address

oc get -n openshift-console route console

#Get the Pod name from the Selector and rsh in it

POD=$(oc get pods -l app=myapp -o name) oc rsh -n $POD

#Exec single command in pod

oc exec $POD $COMMAND

#Copy from local folder byteman-4.0.12 in Pod wildfly-basic-1-mrlt5 under the folder /opt/wildfly

oc cp ./byteman-4.0.12 wildfly-basic-1-mrlt5:/opt/wildfly

## Image Streams

#List available IS for openshift project

oc get is -n openshift

#Import an image from an external registry

oc import-image --from=registry.access.redhat.com/jboss-amq-6/amq62-openshift -n openshift jboss-amq-62:1.3 --confirm

#List available IS and templates

oc new-app --list

## Templates Management

# Deploy resources contained in a template

oc process -f template.yaml | oc create -f -

#List parameters available in a template

oc process --parameters -f .template.yaml

## Setting environment variables

# Update deployment 'registry' with a new environment variable

oc set env dc/registry STORAGE\_DIR=/local

# List the environment variables defined on a build config 'sample-build'

oc set env bc/sample-build --list

# List the environment variables defined on all pods

oc set env pods --all --list

# Import environment from a secret

oc set env --from=secret/mysecret dc/myapp

## WildFly application example on OpeShift

# Create WildFli Image Stream

oc create -f https://raw.githubusercontent.com/wildfly/wildfly-s2i/wf-26.0/imagestreams/wildfly-centos7.json

# Create WildFly app from GitHub Repo

$ oc new-app wildfly:26.0~https://github.com/fmarchioni/ocpdemos --context-dir=wildfly-basic --name=wildfly-basic

#Expose Service with a Route

oc expose service wildfly-basic

#Add WildFly Helm Chart to the Repository

helm repo add wildfly https://docs.wildfly.org/wildfly-charts/

Then, here is how to use Helm to bootstrap an application from an Helm Chart:

# Image Streams and Configuration in the file sampleapp.yaml

helm install sample-app wildfly/wildfly -f sampleapp.yaml

## Create app from a Project with Dockerfile

Next, here is how to create an app from a Dockerfile using a Binary Build:

oc new-build --binary --name=mywildfly -l app=mywildfly

oc patch bc/mywildfly -p '{"spec":{"strategy":{"dockerStrategy":{"dockerfilePath":"Dockerfile"}}}}'

oc start-build mywildfly --from-dir=. --follow

oc new-app --image-stream=mywildfly

oc expose svc/mywildfly

## How to manage Nodes

## #Get Nodes list

oc get nodes

#Check on which Node your Pods are running

oc get pods -o wide

#Schedule an application to run on another Node

oc patch dc myapp -p '{"spec":{"template":{"spec":{"nodeSelector":{"kubernetes.io/hostname": "ip-10-0-0-74.acme.compute.internal"}}}}}'

#List all pods which are running on a Node

oc adm manage-node node1.local --list-pods

#Add a label to a Node

oc label node node1.local mylabel=myvalue

#Remove a label from a Node

oc label node node1.local mylabel-

#create a PersistentVolumeClaim (+update the DeploymentConfig to include a PV + update the DeploymentConfig to attach a volumemount into the specified mount-path)

## How to manage storage

oc set volume dc/file-uploader --add --name=my-shared-storage \

-t pvc --claim-mode=ReadWriteMany --claim-size=1Gi \

--claim-name=my-shared-storage --claim-class=ocs-storagecluster-cephfs \

--mount-path=/opt/app-root/src/uploaded \

-n my-shared-storage

#List storage classes

oc -n openshift-storage get sc

## Build Management

#Manual build from source

oc start-build ruby-ex

#Manual build from source and follow logs

oc start-build ruby-ex -F

#Stop a build that is in progress

oc cancel-build <build\_name>

#Changing the log level of a build:

oc set env bc/my-build-name BUILD\_LOGLEVEL=[1-5]

## How to manage Deployments

#Manual deployment

$ oc rollout latest ruby-ex

#Pause automatic deployment rollout

oc rollout pause dc $DEPLOYMENT

# Resume automatic deployment rollout

oc rollout resume dc $DEPLOYMENT

#Define resource requests and limits in DeploymentConfig

oc set resources deployment nginx --limits=cpu=200m,memory=512Mi --requests=cpu=100m,memory=256Mi

#Define livenessProve and readinessProve in DeploymentConfig

oc set probe dc/nginx --readiness --get-url=http://:8080/healthz --initial-delay-seconds=10

oc set probe dc/nginx --liveness --get-url=http://:8080/healthz --initial-delay-seconds=10

#Scale the number of Pods to 2

oc scale dc/nginx --replicas=2

#Define Horizontal Pod Autoscaler (hpa)

oc autoscale dc $DC\_NAME --max=4 --cpu-percent=10

## Managing Routes

#Create route

$ oc expose service ruby-ex

# Create Route and expose it through a custom Hostname

oc expose serviceruby-ex --hostname

#Read the Route Host attribute

oc get route my-route -o jsonpath --template="{.spec.host}"

## Managing Services

#Make a service idle. When the service is next accessed will automatically boot up the pods again:

$ oc idle ruby-ex

#Read a Service IP

oc get services rook-ceph-mon-a --template='{{.spec.clusterIP}}'

## Clean up resources

#Delete all resources

oc delete all --all

#Delete resources for one specific app

$ oc delete services -l app=ruby-ex

$ oc delete all -l app=ruby-ex

#CleanUp old docker images on nodes

#Keeping up to three tag revisions 1, and keeping resources (images, image streams and pods) younger than sixty minutes:

oc adm prune images --keep-tag-revisions=3 --keep-younger-than=60m

#Pruning every image that exceeds defined limits:

oc adm prune images --prune-over-size-limit

## Openshift Container Platform Troubleshooting

#How to inspect all resources in a namespace (produces resources tree in YAML files)

oc adm inspect ns/mynamespace

#run cluster diagnostics

oc adm diagnostics

#Collect must-gather

oc adm must-gather

#Check status of current project

oc status

#Get events for a project

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

# get the logs of the myrunning-pod-2-fdthn pod

oc logs myrunning-pod-2-fdthn<br />

# follow the logs of the myrunning-pod-2-fdthn pod

oc logs -f myrunning-pod-2-fdthn<br />

# tail the logs of the myrunning-pod-2-fdthn pod

oc logs myrunning-pod-2-fdthn --tail=50

#Check the integrated Docker registry logs:

oc logs docker-registry-n-{xxxxx} -n default | less

## Security

#Create a secret from the CLI

oc create secret generic oia-secret --from-literal=username=myuser

--from-literal=password=mypassword

# Use secret in deployment env

oc set env deployment/ --from secret/oia-secret

# You can also mount the Secret on a Volume

oc set volumes dc/myapp --add --name=secret-volume --mount-path=/opt/app-root/

--secret-name=oia-secret

## Managing user roles

oc adm policy add-role-to-user admin oia -n python

oc adm policy add-cluster-role-to-user cluster-reader system:serviceaccount:monitoring:default

oc adm policy add-scc-to-user anyuid -z default

## Misc commands

#Manage node state

oc adm manage node <node> --schedulable=false

#List installed operators

oc get csv

#Export in a template the IS, BC, DC and SVC

oc export is,bc,dc,svc --as-template=app.yaml

#Show user in prompt

function ps1(){

export PS1='[\u@\h($(oc whoami -c 2>/dev/null|cut -d/ -f3,1)) \W]\$ '

}

#backup openshift objects

oc get all --all-namespaces --no-headers=true | awk '{print $1","$2}' | while read obj

do

NS=$(echo $obj | cut -d, -f1)

OBJ=$(echo $obj | cut -d, -f2)

FILE=$(echo $obj | sed 's/\//-/g;s/,/-/g')

echo $NS $OBJ $FILE; oc export -n $NS $OBJ -o yaml > $FILE.yml

done