Openshift Container platform (OCP)by Redhat

Openshift container platform = Openshift

Reference links

|  |  |
| --- | --- |
| All object definitions are available here | <https://docs.openshift.com/container-platform/3.11/rest_api/apps_openshift_io/deploymentconfig-apps-openshift-io-v1.html> |
| <https://gitlab.com/practical-openshift/labs> | Clone the code of udemy trainer here |
| Quay.io  This is by redhat | Image repository (same like docker repository, here also we can store all the image ) |

Installation

Install docker for desktop software

Install WSL software (windows subsystem for linux)

Then to test run a command called “docker run hello-world” here hello-world is the image name in “windows power shell” and docker desktop should open correctly

Unlike minikube u cant install the Openshift in ur windows machine, bec it needs 9GB RAM, 30 GB harddisk, 4 virtual cpu, so that’s why better use the Redhat developer sand box version

|  |
| --- |
| 1. Create a dev free cluster in Redhat- then few namespaces also will be created in the cluster |
| 1. Download oc command line tool <https://console-openshift-console.apps.rm2.thpm.p1.openshiftapps.com/command-line-tools>   Or login to oc and right click on ? and select command line tools – download oc tool into desktop and add that to path environment variable |
| 1. To see all namespaces <https://console-openshift-console.apps.rm2.thpm.p1.openshiftapps.com/topology/all-namespaces?view=graph>   or go to topology u can see all the namespaces |

Similarities and differences

Same like kubernetes

* Openshift also have a cluster (bunch of server machines)
* Openshift also have objects/resources (among 15, 10 are Openshift exclusive, 5 are common in both)

Openshift architecture & Terminologies

Ocp primary job is to run containers

Here host is nothing but some worker node

|  |  |
| --- | --- |
|  |  |
|  |  |

Here plane means category, in ocp total servers are categorized into 2 –

Control plane- the servers under this category are responsible for running core Openshift processes - REST API, Data persistence, monitoring processes

Data plane – the servers in this category are responsible for running our custom apps like java programs

Developer will interact with REST api present in control plane

Containerization

Container is nothing but some isolated process space inside a hard disk where our app can run

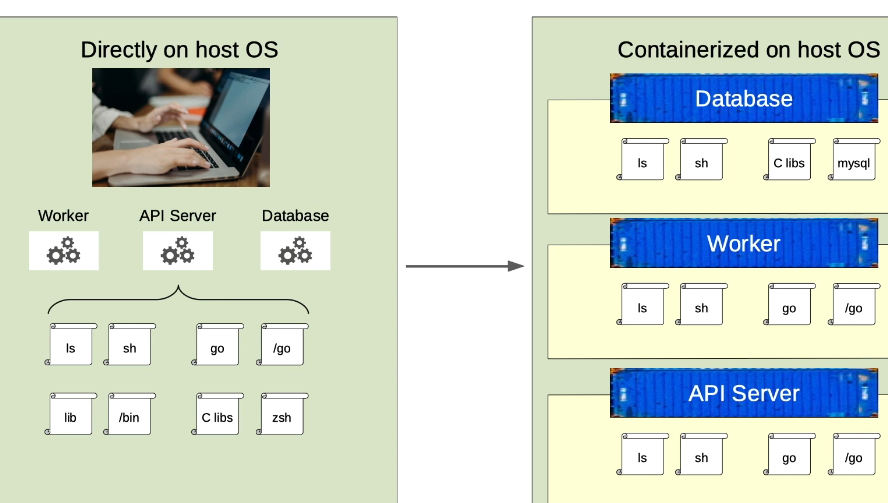
Why containers are needed?

Ex:- if spring needs java 11 , and python needs java 17 , in same machine or os u can’t have 2 java versions,

So if u use containerization, an isolated space will be created inside a hard disk

Container is bundled with all of its dependencies

Docker daemon is something that runs container



OS uses project object to divide applications/ segregate apps

User quota represents upper bound of RAM,cpu and all other resources , for that quota it will reserve the space

Commands

|  |  |
| --- | --- |
| Explain command vs  Describe commands | To see all commands just type “oc” it will only show those commands  There is some small difference between explain and describe  Explain – is to see the specification , describe- this is to see the properties of object after its creation, like “desc table tablename” in database |
| To see doc /explanation of any os/kubernetes resources | oc explain <any ocp / kub object /resource type>  Ex:- oc explain pod or oc explain imagestream   |  | | --- | | You can even see full specification  Every object have a field called spec, that’s what we are fetching here  If u see the object def by hitting “oc explain <any ocp object>” it will show all the fields infor | | E:\study related\pods>oc explain pod  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 ..  kind <string>  Kind is a string value ...  metadata <ObjectMeta>  Standard object's metadata. More info:  https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata  spec <PodSpec>  Specification of the desired behavior of the pod. More info:  https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#spec-and-status  status <PodStatus>  Most recently observed status of the pod..  oc explain pod.spec (note,as per above definition, this pod has a field name spec this spec field may not exist in few objects like configmap)  here inside spec we have another variable called “subdomain” to see that definition  “oc explain pod.spec.subdomain” | |  | |
| Login to oc server cluster | <https://console-openshift-console.apps.rm2.thpm.p1.openshiftapps.com/topology/all-namespaces?view=graph>  oc login –-token=<your token here > --server=cluster url  (you can get this command from oc browser login using above url and right top right corner – click on ur pic – select – copy login command)  oc login –u developer  # Uses the pre-configured OpenShift cluster  oc login  # Allows you to log in to any OpenShift cluster  oc login <cluster address>  # Log out  oc logout |
| To see who am I (to get user id) | Oc whoami |
| To see project  (Here project is nothing but namespace) | |  |  | | --- | --- | | To see all the available projects | oc projects | | To see on which project u are | oc project | | To switch to another project” (like switching to another cluster) | oc project <target project name here> | | To see status of all objects like pods, deploymentConfig, configmap,.. | oc status | | To create a new project | oc new-project <new proj name here > | |  | oc port-forward pod/lab-pod 8080 | | To see openshift cli /server version | oc version | |

OS objects

All object

|  |  |
| --- | --- |
|  | All object definitions are available here  <https://docs.openshift.com/container-platform/3.11/rest_api/apps_openshift_io/deploymentconfig-apps-openshift-io-v1.html> |

Projects

OS uses projects to group related resources

Here project is nothing but namespace, like how namespace contains all kub resources, here project contains all

|  |  |
| --- | --- |
|  | To see the projects and namespaces  oc get projects  here under project tab u can see all resources |

### Pods

Generally pod means group of whales, in ocp 1 pod is nothing but group of containers, 1 pod can have 1 or more containers

In real world assume container as a whale fish

In real world, too much tightly inter-dependent applications (one can’t live without other) stay in same pod, if pod goes down both will be down, else both will be up na

Additional containers are called side car containers

|  |  |
| --- | --- |
| To see full specification of this pod object like what fields to write in this definition | oc explain pod.spec  oc explain pod.spec.containers.env |
| Create sample pod object  apiVersion: v1  kind: Pod  metadata:  name: hello-world-pod (this will be final pod name u can see in oc get pods )  labels:  app: hello-world-pod  spec:  containers:  - env:  - name: MESSAGE  value: Hi! I'm an environment variable  image: quay.io/practicalopenshift/hello-world  imagePullPolicy: Always  name: hello-world-override  resources: {}  we can create this pod object with this command  “oc create –f pods/pod.yaml”  here –f means file | “oc create –f pods/pod.yaml”  Ex:- oc apply -f ./pod.yaml (here even apply keyword will work)  This command will internally hit the RESTful post api and it will create a pod object in os project  In kubernetes it is kubectl apply –f “path to yaml file” |
| To see available running /stopped pods | oc get pods  oc get po (here po is the shortcut for pods)  To see pod real time updates (we don’t need to refresh- but we can see changing pod statuses)  oc get pods --watch |
| Describing a pod | oc describe pod <pod name>  here you can see pod details such as 🡪 how many containers in that pod,  for each container u can see – image name, env variables….and that pod cpu details |
| To go into the pod and run a command  Here rsh – remote shell session  “oc rsh” command is to establish a remote shell session with running container  q- if 2 containers are there with which container it will open session | oc rsh hello-world-pod  after going inside pod to hit that container “wget localhost:8080” – this will generate a file called “index.html” inside pod to to open that “cat index.html”  if u want to come outside from that pod “exit” |
| To delete the pod | oc delete pod <pod-name>  in the command we should mention the type of resource- bec internally they might have enabled/disabled delete grant for certain object ,  so we must enter resource type |
|  |  |

### Configmap

|  |  |
| --- | --- |
| Oc e |  |

Deploymentconfig

Deployment config is nothing but collection of pods, this object brings extra configuration to your pods

In real time we will use this definition instead of pod definition, this is like parent object for pod object

Ex:- if u directly delete pod, sometime pod wont get deleted because in deploymentconfig we would have set the replicas as 2, (oc explain deploymentConfig.spec.replicas)

so in that case delete this deploymet object , so that pods will get deleted automatically

While running commands instead of deploymentconfig , u can always use dc

This DC object will create the pods

|  |  |
| --- | --- |
| To see full documentation | oc explain deploymentconfig  or  oc explain dc |
| Deploy application with single command with image name  If u just know image name, with this single command u can create pod, DC  ( without writing the pod definition) | oc new-app <image name with repository> --as-deployment-config  oc new-app quay.io/practicalopenshift/hello-world --as-deployment-config |

When u run this command below will be the log

--> Found container image 34b5ac8 (4 years old) from quay.io for "quay.io/practicalopenshift/hello-world"

\* An image stream tag will be created as "hello-world:latest" that will track this image

\* This image will be deployed in deployment config "hello-world"

\* Port 8080/tcp will be load balanced by service "hello-world"

\* Other containers can access this service through the hostname "hello-world"

--> Creating resources ...

imagestream.image.openshift.io "hello-world" created

Warning: apps.openshift.io/v1 DeploymentConfig is deprecated in v4.14+, unavailable in v4.10000+

deploymentconfig.apps.openshift.io "hello-world" created

service "hello-world" created

--> Success

Application is not exposed. You can expose services to the outside world by executing one or more of the commands below:

'oc expose service/hello-world'

Run 'oc status' to view your app.