KUBERNETES

To check if kubernetes is installed :

kubeadm version

sudo service kubelet status

sudo kubeadm init --pod-network-cidr=192.168.0.0/16 (initiates the kubeadm manager in the master node with specific network details)

Add sudo to the join command and copy the join command and run them in the worker nodes

kubectl (shows all the modules which can be used using kubectl)

kubectl get nodes (lists the nodes)

kubectl get nodes -o wide (listes detailed information about the nodes)

kubectl describe node <hostname-of-node> (shows all the information about the node)

kubectl run <podname> --image <imagename> --port <portnumber>

ex: kubectl run nginxpod --image nginx --port 80

kubectl get pods (lists all the pods running)

kubectl get pods -o wide (lists detailed information of the pod)

kubectl describe pod <podname> (shows all the information of the pod with the containers)

kubectl delete pod <podname> (deletes the pod)

How to create multiple pods distributed across multiple worker nodes :

kubectl create deployment --help (shows all the options with the deployment)

Single pod :

kubectl create deployment <name> --image <imagename> (to deploy one container and one pod)

ex: kubectl create deployment mydep --image nginx

kubectl get deploy (shows all the deploys)

kubectl get deployments (shows all the deployments)

(both are same)

add -o wide (to see all the details of the nodes on which the containers are created )

Multipods :

kubectl create deployment multi-pod --image <imagename> --replicas <no.of-replicas>

kubectl get pods-o wide

kubectl exec -it <podcontainername> /bin/sh (enters into the pod container)

kubectl scale deployment multi-pod --replicas <number> (scales the number of pod containers)

kubectl get pods-o wide | grep -v mydep (-v shows the containers other than the word given)

kubectl delete deploy (to delete the deployment)

ACCESS TO THE PODS :

kubectl expose deployment multi-pod --port 80

kubectl get service

NETWORK EXPOSE:

kubectl expose deployment <servicename> --type NodePort --port 80

Note: host ports are pre allocated in kubernetes between 30000 - 32000... we do not have to specify

the host ports

kubectl get service (shows the new service formed with the port number)

curl http://localhost:<portnumber>

kubectl expose --help

NOTES:

kubernetes is another container runtime engine as docker

kubernetes architecture -

--> the master node is only used to manage the worker nodes... containers are not created on the

master node

--> we will install the kubelet on the worker nodes... master will send the commands to kubelet and

the kubelet will create the containers

CNI - Container Networking Interface

MOST OF THE KUBERNETES COMMANDS :

kubectl <action> <module>

--> inside the pod, we have an app container and for the functionality of the app we have three different

containers - one for login, another for services, another for monitoring... all of the cotainers

should be running on the same node which is not possible in docker...

--> since all the containers are on the same node, the storage and network can be shared

in kubernetes we can only manage the pods and not containers

DIFFERENCE BETWEEN DOCKER AND KUBERNETES :

Docker swarm

containers

service - many containers

publish service - access app

Kubernetes

pods - containers

deployments - many pods

service - access app | expose pods/containers

Exposing the deployments is called as service

LoadBalancer - Internal(ClusterIP) and external(Nodeport) - for others to access the port

Org./internet -> WEB (external) -> APP -> DB (app and db are internal accessed by developer)

WE CREATE PODS THROUGH INFRASTRUCTURE AS CODE AND NOT THROUGH COMMANDS IN REALTIME

TASKS :

Google kubernetes engine - GCP

Elastic kubernetes engine - AWS

Azure Kubernetes service - Azure Cloud

helm

kubelet - agent which is running on all the worker nodes

kubeadm - is a command -- admin command

kubectl - through this command we can control the cluster (ctl is control)

k8s - \* character in between the K and S

EPHEMERAL

Do not enter into the container -- to change the config of the container - write a dockerfile with

the changes to the image and then remove all the previous containers and then make the containers

using the new image

TRY kubectl get pods >> file1.txt (to move the output of the command into a file)

LANGUAGES TO LEARN - bash scripting, shell scripting, then python (system level python or OS level)

CI - continuous integration (Jenkinsfile)

Checkout code

Build -Jar file is formed

Dockerfile

Container image

tag image

Push the image

mvn clean install will build the jar file

tools used -- git maven Docker