Kubernetes :-

Architecture :-

\* Master

\* Worker Node1 ( Kubelets )

\* Worker Node2

…. NodeN

\* API sever - Takes the commands from master

\* ETCD :- Stores the information about nodes

\* Scheduler :- Used for distributing work

\* Controller :- Controller manager

\* Container Runtime:- Docker

Kubectl - Kube control

\*\*\*\*\*\*\*\*

\* kubectl run hello-minikube - install something

\* kubecrl cluster-info - Gets the cluster info

\* kubectl get nodes - Get the details of installed nodes

Kubernetes installation :-

\* Minikube

\* Microk8s

\* Kubeadmin

Minikube :-

\* kubectl must be installed

\* Minikube.exe must be installed

\* Must have a hypervisor installed

\* Permanent Kubernetes can be installed on - Google Container Engine ( GKE )

\* Kubeadm - Used for multimode Kubernetes cluster in a local env

\* kubectl commands :-

\* kubectl create deployment hello-minikube --image=k8s.gcr.io/echoserver:1.10

\* kubectl expose deployment hello-minikube --type=NodePort --port=8080

\* kubectl get pods

\* kubectl get nodes

\* kubectl get nodes -o wide - To show the OS versions

\* kubectl delete services hello-minikube - Delete the service

\* kubectl delete deployment hello-minikube - Delete the deployments

Pods :-

\* Pods resides inside node

\* kubectl run nginx image=nginx

\* kubectl describe pod nginx

\* kubectl get pods -o wide

YAML:-

\* YAML in kubernetes

\* Create PODS using YAML

\* apiVersion: v1

\* kind: Pod

\* metadata:

name: samar-test-pod

labels:

app: myapp

type: front-end

\* spec:

containers:

- name: nginx-container

image: nginx

\* kubectl create -f pod-definitions.yml

\* kubectl edit pods <pod name >

\* kubectl get pods -o wide

Replication Controllers/

\* apiVersion: v1

\* kind: ReplicationController

\* metadata:

name: my app-rc

labels:

name: myapp

type: front-end

\* spec:

template:

metadata:

name: pod1

labels:

app: app1

type: front-end

spec:

containers:

- name: nginx

image: nginx

replicas: 3 ————> Number of pods

\* kubectl get replicationcontroller

\* kubectl delete replicationcontroller <>

\* kubectl get pods

Replica Set:- ( Need an extra selector )

\* apiVersion: apps/v1

\* kind: ReplicaSet

\* metadata:

name: my app-rc

labels:

name: myapp

type: front-end

\* spec:

template:

metadata:

name: pod1

labels:

app: app1

type: front-end

spec:

containers:

- name: nginx

image: nginx

replicas: 3

selector:

matchLabels:

type: front-end

\* kubectl get replicaset

\* kubectl delete replicaset <set name >

\* Replica Set is a process to minitor the pods

\* kubectl replace -f <file name > - To increase the replicas

\* kubectl scale replicaset <name> - -replicas=6 - - To scale the replicas

\* kubectl scale replicaset <replicaname > - -replicas=8

\* kubectl edit ReplicSet <ReplicaSet name>

Deployments

\* All remain same as ReplicaSet

\* kind : Deployment - This is the only change required

\* kubectl create -f <File name >

\* kubectl get all - Lists everything

\* Shortcut commands :-

\* kubectl create deployments http-frontend - -image=httpd:2.4-alpine

\* kubectl create deployments test - -image=httpd:2.4-alpine - -replicas=3

\* kubectl scale deployment test - -replicas=4 — > This will scale the pods

\* kubectl get deployment <name>

kubectl get deployment <name> -o wide

Deployments - Update/Rollback/Versioning

\* kubectl rollout status deployment/<deployment name >

\* kubectl rollout history deployment/<deployment name>

\* Deployment strategies :-

\* Destroy the 5 and deploy a new one

\* Rolling deploy/update - 1 down and 1 up

\* kubectl rollout undo deployment/<app name> - — > Will undo a rollout

\* kubectl set image deployment <app name> nginx=nginx:1.19 —record

Networking in Kubernetes :

\* Routings

\* Services

\* NodePort range :- 30000 - 32767

\* services.yaml template

aptVersion: v1

kind: Service

metadata:

spec:

type: NodePort

ports:

- targetPort: 80

port: 80

nodePort: 30008

selector:

app: test

type: front-end

\* Commands

\* kubectl get services

\* kubectl get svc

\* If you are running in Minikube type bellow commands to get the URL

minikube service <svc name> —url

\* ClusterIP

\* yaml File

apiVersion: v1

kind: Service

metadata:

spec:

type: ClusterIp

ports:

- targetPort: 80

port: 80

nodePort: 30008

\* Service Load Balancer

\* yaml file

apiVersion: v1

kind: Service

metadata:

spec:

type: LoadBalancer

ports:

- targetPort: 80

port: 80

nodePort: 30008

\* Shortest way to create a service :-

kubectl expose deployment <Deployment name> - -name=<> - -target-port= 8080 - -port=8080 - -NodePort=30080 - -type=Nodeport - - dry-run=client -o yaml > svc.yaml

Microservices Architecture :-

\* Designing a Voting App

\* Using docker

\* docker run -d --name=redis redis

\* docker run -d --name=db postgres

\* docker run -d --name=vote -p 5000:80 --link=redis:redis voting-app < - Deploy a app

\* docker run -d --name=result -p 5001:80 --link db:db result-app

\* docker run -d --name=worker --link db:db --link redis:redis worker

\* Using Kubernetes

\* Deploy PODs

\* Create Services ( ClusterIP)

\* redis

\* db

\* Create Services ( NodePort)

\* voting-app

\* result-app

\* Steps to run the yaml files:-

\* First create pod

\* Create the services

\* Create redis pod

\* Create redis service

\* Create Postgres pod

\* Create Postgres services

\* Create worker pod

\* Create woker service

\* Create the result pod

\* Create the result service

Note:-

\* minikube service voting-service - It will open up the services in web-browser

\* Kubernetes on Cloud

\* GKE ( Google Kubernetes Engine )

\* Deploy Clusters in GCP

\* AKS ( Azure Kubernetes Service )

\*

\* EKS ( Amazon Elastic Kubernets Services )

\* Create a Cluster

\* Create IAM roles for cluster

\* Open the IAM console at https://console.aws.amazon.com/iam/.

\* In the navigation panel, choose **Roles**.

\* Search the list of roles for eksClusterRole. If a role that includes eksClusterRole does not exist, then see Creating the Amazon EKS cluster role to create the role. If a role that includes eksClusterRole does exist, then select the role to view the attached policies.

\* Choose **Permissions**.

\* Ensure that the **AmazonEKSClusterPolicy** managed policy is attached to the role. If the policy is attached, your Amazon EKS cluster role is properly configured.

\* Choose **Trust Relationships**, **Edit Trust Relationship**.

\* Verify that the trust relationship contains the following policy. If the trust relationship matches the policy below, choose **Cancel**. If the trust relationship does not match, copy the policy into the **Policy Document** window and choose **Update Trust Policy**.

\* Make **sure to add this line in the trusted Policy** - "AWS": "arn:aws:iam::<Account ID >:user/<User Name>”

\* Create node groups

\* Create IAM roles for node cluster

\* **To create your Amazon EKS node role in the IAM console**

\* Open the IAM console at https://console.aws.amazon.com/iam/.

\* Choose **Roles**, then **Create role**.

\* Choose **EC2** from the list of **Common use cases** under **Choose a use case,** then choose **Next: Permissions**.

\* In the **Filter policies** box, enter AmazonEKSWorkerNodePolicy. Check the box to the left of **AmazonEKSWorkerNodePolicy.**

\* In the **Filter policies** box, enter AmazonEKS\_CNI\_Policy. Check the box to the left of **AmazonEKS\_CNI\_Policy.**

\* In the **Filter policies** box, enter AmazonEC2ContainerRegistryReadOnly. Check the box to the left of **AmazonEC2ContainerRegistryReadOnly.**

\* Choose **Next: Tags**.

\* (Optional) Add metadata to the role by attaching tags as key–value pairs. For more information about using tags in IAM, see Tagging IAM Entities in the *IAM User Guide*.

\* Choose **Next: Review**.

\* For **Role name**, enter a unique name for your role, such as NodeInstanceRole. For **Role description**, replace the current text with descriptive text such as Amazon EKS - Node Group Role, then choose **Create role**.

\* Create node groups

\* Create a key-pair to access the cluster

\* Install AWS cli

\* Install kubectl if you already haven’t installed

\* If you have aws account please set the access-ID, secret key and region

\* Run this command to add your cluster :-

\* aws eks --region <your region> update-kubeconfig --name <Cluster Name >

\* [ testing ] Update iam for your cluster :- aws --region <region-code> eks update-kubeconfig --name <cluster\_name> --role-arn arn:aws:iam::<aws\_account\_id>:role/<role\_name>

\* [ Testing ] Download the key file - curl -o aws-auth-cm.yaml https://amazon-eks.s3.us-west-2.amazonaws.com/cloudformation/2020-08-12/aws-auth-cm.yaml

\* [ Testing ] - Open the file with your favorite text editor. Replace <ARN of instance role (not instance profile)> with the Amazon Resource Name (ARN) of the IAM role associated with your nodes, and save the file. Do not modify any other lines in this file.

\* kubectl version --short --client —> To check there client version

\*\*\*\*\*\*\*\*\*

Note :- If you are getting bellow error :-

\*\*\*\*\*\*\*\*

An error occurred (AccessDenied) when calling the AssumeRole operation: User: arn:aws:iam::102868756097:user/dscdevopstest is not authorized to perform: sts:AssumeRole on resource: arn:aws:iam::dscdevopstest:role/eksClusterRole\_sam

Orig:-

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Principal": {

"Service": "eks.amazonaws.com"

},

"Action": "sts:AssumeRole"

}

]

}

New:-

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Principal": {

"Service": "eks.amazonaws.com”,

"AWS": "arn:aws:iam::<Account ID >:user/<User Name>”

},

"Action": "sts:AssumeRole"

}

]

}

\* https://docs.aws.amazon.com/eks/latest/userguide/getting-started-eksctl.html

\* https://docs.aws.amazon.com/eks/latest/userguide/troubleshooting.html#unauthorized

\* https://docs.aws.amazon.com/eks/latest/userguide/add-user-role.html

\* Try this if you are getting server unauthorized error ( So far it didn’t work for me ) :- https://aws.amazon.com/premiumsupport/knowledge-center/eks-api-server-unauthorized-error/

\* Once the above things resolved, try to make your deployments

\* kubectl create -f voting-app-deployment.yaml

\* kubectl create -f voting-app-service.yaml

\* kubectl create -f redis-app-deployment.yaml

\* kubectl create -f redis-app-service.yaml

\* kubectl create -f postgress-app-deployment.yaml

\* kubectl create -f postgress-app-service.yaml

\* kubectl create -f worker-app-deployment.yaml

\* kubectl create -f result-app-deployment.yaml

\* kubectl create -f result-app-service.yaml

\* verify the deployment - kubectl get deployment,svc

AKS ( Azure Kubernetes Service )

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Create an Azure Account

\* Select the Kubernetes service ( AKS)

\* Create the cluster

\* Select the proper subscription

\* Name

\* Version - default

\* Node size - 1

\* Auth ;- Service principle - Create a new one

\* Use cloud shell

\* Select the storage

kubeadm

\*\*\*\*\*\*\*\*\*\*

https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/install-kubeadm/

\* Clone this - https://github.com/kodekloudhub/certified-kubernetes-administrator-course.git

\* Once clone done - go to the folder

\* Open the Vagrantfile and make sure it has 1 master and 2 worker nodes set

\* run - vagrant up

\* SSH to nodes and master using - vagrant ssh kubemaster or vagrant ssh kubenode01

\* Login to master and run bellow commands

\* lsmod | grep br\_netfilter - if its not loaded run -

\* sudo mod probe br\_netfilter

\* Run the above commands in all nodes

\* Create bridges

\* Copy paste bellow configs in all nodes

cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf

net.bridge.bridge-nf-call-ip6tables = 1

net.bridge.bridge-nf-call-iptables = 1

EOF

sudo sysctl --system

\* Install Docker - https://kubernetes.io/docs/setup/production-environment/container-runtimes/

\* Make sure docker is running by doing “ systemctl status docker”

\* Install bellow contents

sudo apt-get update && sudo apt-get install -y apt-transport-https curl

curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -

cat <<EOF | sudo tee /etc/apt/sources.list.d/kubernetes.list

deb https://apt.kubernetes.io/ kubernetes-xenial main

EOF

sudo apt-get update

sudo apt-get install -y kubelet kubeadm kubectl

sudo apt-mark hold kubelet kubeadm kubectl

\* Create a cluster - https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/create-cluster-kubeadm/

\* Configure the pods by running this command :-

kubeadmin init will initiate the

kubeadm init --pod-network-cidr 10.244.0.0/16 --apiserver-advertise-address=<Master node ip>

\* Configure the config file under .kube/home folder

mkdir -p $HOME/.kube

sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

sudo chown $(id -u):$(id -g) $HOME/.kube/config

\* Copy paster the command to join the cluster as shown in the above upgrade

kubeadm join 192.168.56.2:6443 --token vd20cl.k0pqprwp9kuq4g7c \

--discovery-token-ca-cert-hash sha256:dcfe11f2b42b2469c487d9eacf2772763abe6d5b5639ac517f6862c7eabf3c99