**CLASS - 1**

**Testing Grade:**

Minikube

Kubeadm (kubernetes product)

**Production Grade:**

Kops -Kubernetes Operations (Kops) → **Will work on AWS and Google Cloud**

**Kops is not a Kubernetes product a Third Party generated,it will also not work on on-premises**

**Azure Cloud -> AKS**

**AWS -> EKS**

1. Domain Name - devopsk8s.xyz

2. AWS Account

3. S3 Bucket & Route53 Domain Integration

4. Deploy a Mgmt Server which holds all scripts.

5. KOPS Binary(K8S Cluster Mgmt) & KUBECTL Binary(K8S Cluster Ops)

6. SSH Public & Private Keys

7. AWS CLI and AWS Access/Secret Key

Route53:

Hosted zone -->Domain name ( devopsk8s.xyz) → Type (Public Hosted Zone)

Those nameservers place in godaddy → Manage DNS → Name servers → Change→ Enter name server as my own → name server→ select values for that hosted zone and place in nameservers in godaddy

**Step-1:**

Select ubuntu 18.04 Ubuntu

**Step-2:**

#!/bin/bash

curl https://get.docker.com/ | bash

**Step-3: (HOW TO INSTALL DOCKER)**

sudo -i

#To check Docker is installed or not Type below command

docker version

**Step-4: ( INSTALL CONTAINERS)**

#Installing Docker Image

docker pull nginx

#Check Image

docker images

#Using this image need to run container

docker run --rm -dit -p 8000:80 nginx:latest

#How to check Container running or not

docker ps

#Container is running and we can check in web page

<http://13.127.49.133:8000/>

**Step-5 (OPTIONAL) (INSTALLING SREE HARSHA DOCKER IMAGE)**

#Running Harsha’s Docker Image

docker pull sreeharshav/rollingupdate:v3

#Check Image

docker images

#Using this image need to run container (IF WE HAVE MULTIPLE CONTAINERS NEED TO USE DIFFERENT PORTS)

docker run --rm -dit --name sreecon1 -p 9000:80 sreeharshav/rollingupdate:v3

#How to check Container running or not

docker ps

#Container is running and we can check in web page

<http://13.127.49.133:9000/>

**Step-6 (HOW TO PREPARE DOCKER FILE)**

#create a file in ubuntu linux

nano Dockerfile

#Paste below commands

FROM ubuntu:18.04

RUN apt update && apt install nginx -y

RUN apt install -y wget curl net-tools nano

EXPOSE 80

CMD ["nginx", "-g", "daemon off;"]

#Run by executing this command to create repository (sreeharshav/newnginx:v1) → is a tag name (.) → is which all files inside the directory

docker build -t sreeharshav/newnginx:v1 .

#Check Docker images

docker images

#If required we can push this image to Docker Hub

#If required we can run container using the docker image

docker run --rm -dit --name newimage -p 8500:80 sreeharshav/nginx:v1

**CLASS - 2**

1. **YAML VS JSON Differences**
2. **VS Code advantages**
3. **Kubernetes Architecture**

**JSON:**

{

"public-clouds": [

{

"Companys name": "Amazon aws",

"Parent organization": "Amazon",

"Established-in": "14th-November",

"CEO": "rohith",

"Services": ["EC2","RDS","VPC"]

},

{

"Companys name": "Azure",

"Parent organization": "Microsoft",

"Established-in": "19th-Sep",

"CEO": "Rohith Achanta",

"Services": ["VNET","VSPHERE","MYSQL"]

},

{

"Companys name": "Google-Cloud",

"Parent organization": "Google",

"Established-in": "14th-November",

"CEO": "Sundhar pichai",

"Services": ["Gmail","Google-Drive","Google-Notes"]

},

{

"Companys name": "Alibaba-Cloud",

"Parent organization": "Alibaba",

"Established-in": "14th-November",

"CEO": "Akhil",

"Services": ["Alibaba-Clous Enterprises","Alibaba-Group"]

}

]

}

**PYTHON :**

---

public-cloud:

- Cloud: "AWS"

date of birth: "13th may"

ceo: "rohith"

services:

- vpc

ec2

s3

- Cloud: "Azure"

date of birth: "13th May"

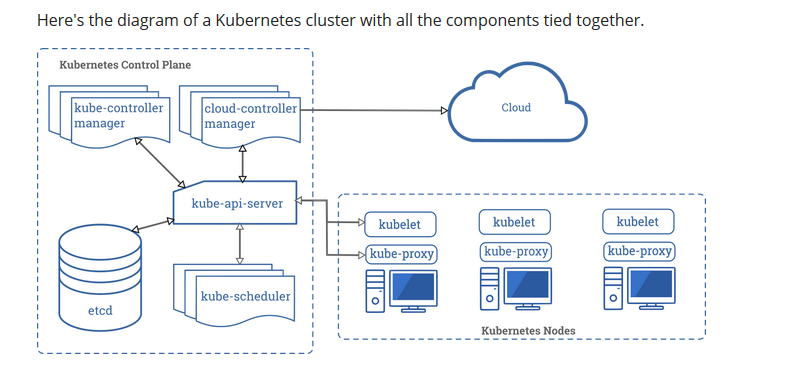
ceo: "andrew"

services:

- vnet

mysql

virtusl image

****

**CLASS - 3**

**Testing Grade:**

Minikube

Kubeadm

**Production Grade:**

Kops -Kubernetes Operations (Kops)

**Requirements or Pre-requisites :**

1. Domain Name - [cirrustribe.cloud](https://dcc.godaddy.com/control/cirrustribe.cloud/settings)

2. AWS Account

3. S3 Bucket & Route53 Domain Integration

4. Deploy a Mgmt Server which holds all scripts.

5. KOPS Binary(K8S Cluster Mgmt) & KUBECTL Binary(K8S Cluster Ops)

6. SSH Public & Private Keys

7. AWS CLI and AWS Access/Secret Key

**1. Domain Name -** [**cirrustribe.cloud**](https://dcc.godaddy.com/control/cirrustribe.cloud/settings)**:**

#Purchase a Domain

<https://dcc.godaddy.com/domains>

Domain Name : [**cirrustribe.cloud**](https://dcc.godaddy.com/control/cirrustribe.cloud/settings)

**2. AWS Account:**

**3. S3 Bucket & Route53 Domain Integration:**

#Create a Domain Hosted zone with domain name and with public access

#Edit DNS settings in GoDaddy

Domain in GoDaddy → manage DNS → add namesource values from route53 to dns

#Create S3 Bucket with same domain name

**4. Deploy a Mgmt Server which holds all scripts:**

**Create Management Server in VPC:**

**5. KOPS Binary(K8S Cluster Mgmt) & KUBECTL Binary(K8S Cluster Ops):**

**Kubernetes KOPS Github Link:**

<https://github.com/kubernetes/kops>

Installation:

sudo -i

#Go to <https://github.com/kubernetes/kops/releases/tag/v1.18.0-beta.1>

wget https://github.com/kubernetes/kops/releases/download/v1.18.0-beta.1/kops-linux-amd64

ll

chmod 700 kops-linux-amd64

mv kops-linux-amd64 kops

mv kops /usr/local/bin

#Go back to root folder

kops

kops version

#Install Kubectl

curl -LO https://storage.googleapis.com/kubernetes-release/release/`curl -s [https://storage.googleapis.com/kubernetes-release/release/stable.txt`/bin/linux/amd64/kubectl](https://storage.googleapis.com/kubernetes-release/release/stable.txt%60/bin/linux/amd64/kubectl)

#Permissions:

chmod 700 kubectl

mv kubectl /usr/local/bin/

kubectl

kubectl version

**6. SSH Public & Private Keys:**

(In root)

ls .ssh/

ssh-keygen

ll .ssh/

**7. AWS CLI and AWS Access/Secret Key:**

#To check version of the server

cat /etc/lsb-release

#Install unzip

apt update && apt install unzip -y

curl "https://awscli.amazonaws.com/awscli-exe-linux-x86\_64.zip" -o "awscliv2.zip"

unzip awscliv2.zip

sudo ./aws/install

aws --version

aws configure

Insert aws credentials

#If you have doubt on aws credentials,check

aws s3 ls

# JQ Install [Not Necessary]

apt install jq -y

aws ec2 describe-vpcs | jq

**KOPS DEPLOYMENT:**

kops create cluster --name=cirrustribe.cloud --state=s3://cirrustribe.cloud --zones=ap-south-1a --node-count=2 --node-size=t2.micro --master-size=t2.small --master-volume-size 20 --node-volume-size 10 --dns-zone=cirrustribe.cloud --yes

**#Review the Config**

kops get cluster --state s3://cirrustribe.cloud

kops get ig --name cirrustribe.cloud --state s3://cirrustribe.cloud

kops edit ig --name=cirrustribe.cloud master-ap-south-1a --state s3://cirrustribe.cloud

kops edit ig --name=cirrustribe.cloud nodes --state s3://cirrustribe.cloud

kops update cluster --name cirrustribe.cloud --yes --state s3://cirrustribe.cloud

kops rolling-update cluster --name cirrustribe.cloud --yes --state s3://cirrustribe.cloud

kops delete cluster --name=cirrustribe.cloud --state s3://cirrustribe.cloud --yes

**Config File:**

(Under root)

vim .kube/config

**CLASS - 4**

**K8S-B1-KOPS-ClusterMgmt-Exports\_KopsOptions**

Management Server

kops edit ig --**name=cirrustribe.cloud nodes** --**state s3://cirrustribe.cloud**

#Exporting name and kops\_state\_store to simplify command

nano .bashrc

export NAME=cirrustribe.cloud

export KOPS\_STATE\_STORE=s3://cirrustribe.cloud

#alias ku=’kubectl’

export EDITOR=nano

Logout and login again to refresh .bashrc file (or) source .bashrc

kops get ig

#Edit Master & Node configurations [OPTIONAL]

kops edit ig nodes

rootVolumesize: 10

#[OPTIONAL]

kops edit ig --name=cirrustribe.cloud nodes --state s3://cirrustribe.cloud

maxSize: 1

minSize: 1

#Deploy

kops update cluster --name cirrustribe.cloud --yes --state s3://cirrustribe.cloud

#Introducing tmux [OPTIONAL]

tmux

Press crtl+b release shift+%

#Config File

cat .kube/config

#To Login to Master & Nodes from Management server:

ssh -i .ssh/id\_rsa **admin**@ip

[This should be performed under .ssh Directory]

**Kops create secret sshpublickey admin -i /root/.ssh/id\_rsa.pub --name cirrustribe.cloud**

**kops update cluster --yes**

**kops rolling update cluster --yes**

**Current Server → Master [Under admin user]:**

kubectl get nodes

→ master

→ node

docker version

#Create extra node [OPTIONAL]

AWS → Autoscaling Groups → Check how many nodes

Management server:

kops edit ig nodes

max size: 2

minsize: 2

kops update cluster --name cirrustribe.cloud --yes --state s3://cirrustribe.cloud

#Cross-check in aws ui nodes will be deployed

AWS → Autoscaling Groups → Check how many nodes

#Check nodes from Management terminal

kubectl get nodes

#Script to edit master,node & update

nano k8s-script.sh

#!/bin/bash

kops edit ig --name=cirrustribe.cloud master-ap-south-1a --state s3://cirrustribe.cloud && kops edit ig --name=cirrustribe.cloud nodes --state s3://cirrustribe.cloud && kops update cluster --name cirrustribe.cloud --yes --state s3://cirrustribe.cloud

#Execute

bash k8s-script.sh

**#Whenever you need to Delete cluster,using this script make nodes and master 0**

bash k8s-script.sh

#Will show which all services used in kubernetes

kubectl get pods -A

#NAMESPACE - Using Namespace we can do logical separation of cluster,easy for mangmnt and resource allocation

kubectl get ns

#Example for creating Namespace [Generally NAMESPACE shouldnot be deleted]

kubectl create ns rohith

kubectl delete ns rohith

**PODS:**

Inside pod there might be number of containers and volumes

For Entire POD single ip will be assigned



**NODE:**

In a single node only single pod will exists



**CLASS - 5**

**KOPS-INSTALL-K8S-KUBEADM**

1. Ubuntu server
2. Install Required softwares and create an image
3. By using image create 3 machines and install cluster

**DOCKER INSTALLATION & kubelet kubeadm kubectl Installation:**

**FOR CONTROL PLANE:**

**==================**

**Make sure you create the master server with 2CPU(Use t2.medium)**

**Take Ubuntu 16.04 (Tag name - Kube Image)**

Userdata

#!bin/bash

curl https://get.docker.com/ | bash

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

Check with below commands:

docker version

kubectl version

kubeadm

**Create an Image:**

KubeImage Instance → Image → Create Image → KubeAdminImage (Image Name) → Create Image → Server Reboots

AMI → It will create Image

Once Image created **delete** kubeImage Server



**Master Creation:**

Once Image has been created,Go to AMI → Right click → Launch → **t2.medium** → Number of Instances → 1 [Master] → Choose any **Public**/Private subnets → K8s-Master [Tag-Name]

**Node Creation:**

My AMI → Launch → t2.micro → Number of Instances → 2 → Choose different Public subnet → Tag names → [Node-1,Node-2]

**Login to master via SSH:**

sudo -i

kubeadm init --pod-network-cidr 192.168.0.0/16

Run below commands from the user (**In my case - root**) you are working :

mkdir -p $HOME/.kube

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

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

kubectl get nodes

#kubectl will get the pods which are in kube-system Namespace → n denotes to name space

kubectl get pods -n kube-system

#To check where master was running

kubectl cluster-info

alias ku=’kubectl’

kubectl get pods -A

kubectl get nodes

root@ip-10-1-1-81:~# kubectl get nodes

NAME STATUS ROLES AGE VERSION

ip-10-1-1-81 NotReady master 2m16s v1.18.3

#To get network between pods,execute below command

kubectl apply -f <https://docs.projectcalico.org/v3.14/manifests/calico.yaml>

root@ip-10-1-1-81:~# kubectl get nodes

NAME STATUS ROLES AGE VERSION

ip-10-1-1-81 Ready master 5m47s v1.18.3

kubectl get pods -A

**NODE-1:**

BELOW COMMAND NEEDED TO BE EXECUTED IN THE WORKER NODE:

kubeadm join 10.1.1.200:6443 --token ynogss.v28um8uq3pbza9mo \

--discovery-token-ca-cert-hash sha256:e5c238a5f964a05cdf2e09adfe8d3458eaabb6925327fa76849ae8463d0ea84f

**NODE-2:**

BELOW COMMAND NEEDED TO BE EXECUTED IN THE WORKER NODE:

kubeadm join 10.1.1.200:6443 --token ynogss.v28um8uq3pbza9mo \

--discovery-token-ca-cert-hash sha256:e5c238a5f964a05cdf2e09adfe8d3458eaabb6925327fa76849ae8463d0ea84f

**Labels:[Login to Master Server]**

kubectl get nodes

kubectl label node ip-10-1-2-42 node-role.kubernetes.io/worker1=worker1

kubectl label node ip-10-1-2-45 node-role.kubernetes.io/worker2=worker1

**To Delete Label:**

kubectl label node ip-10-1-2-42 node-role.kubernetes.io/worker1-

kubectl label node ip-10-1-2-45 node-role.kubernetes.io/worker2-

**Config File in Local system:**

cat .kube/config

#Copy file [apiVersion to Client-key-data] and paste it in C drive → Folder → new Text doc → kubeadm → Edit with notepad++ → Paste

**Environmental Variables on local system:**

Local system → cmd prompt → sysdm.cpl → Advanced → Environmental Variables → Path → Edit → Add folder path here → close cmd prompt and open it again

**Cmd prompt :**

kubectl

cd C:\*path-to-folder*

*C:/Terraform>kubectl.exe --kubeconfig=kubeadm get nodes*

**Alias in Local System:**

Cmd → sysdm.cpl → Advanced → Environmental variables → Variables → New -->KUBECONFIG[Variable Name] → Variable Value [Folder path] → ok → restart cmdprmpt

**To check pods & nodes from command prompt:**

C:\Terraform>cd ..

C:\>kubectl get pods -A

C:\>kubectl get nodes

**Adding Additional server:**

Launch instance → AMI → VPC → Public Subnet 3 → Node 3 [Tag Name]

**Attaching Node to Cluster: [MASTER - NODE]:**

kubeadm token create --print-join-command

[Copy kubeadm join and paste in Node-3]

kubectl get nodes

kubectl run testpod --image=index.docker.io/sreeharshav/rollingupdate:v3

kubectl get pods

#This command let us now where pod was running [In which ip or node it is running]

kubectl get pods -o wide

#If you need to view it in yaml format then

ku get pods -o yaml

#If we need to execute any command in pod

ku exec -it testpod -- df -h

#By default pod will not expose to inside & outside.To expose to inside execute below command

kubectl expose pod testpod --port=8000 --target-port=80

kubectl get service

#To check whether it is exposed internally

curl http://[TESTPOD IP]:8000

#To expose to Outside execute below command

[Service short form svc]

kubectl get svc

kubectl delete svc testpod

kubectl expose pod testpod --port=8000 --target-port=80 --type=NodePort

kubectl get svc

#Check from Internet

publicip:32712

**\*\* IF YOU EXPOSE SERVICE ALL NODES WILL BE EXPOSED AND YOU CAN CONNECT WITH ANY NODE PUBLIC- IP**

#Draining Nodes

kubectl drain ip-10-1-2-5 --force --ignore-daemonsets

#After draining pod,need to delete node

kubectl delete node ip-10-1-2-5

#Same like above delete pod and node at a time

kubectl drain ip-10-1-2-42 --force --ignore-daemonsets && kubectl delete node ip-10-1-2-42

# **#Class - 6**

# **K8S-B1-KOPS-Install-K8S-PODs**

How to check version of Ubuntu installed :

cat /etc/lsb-release

**If you need to connect Master from your local computer:**

sudo -i

cat .ssh/id\_rsa

#Copy that private key paste it in notepad and save .pem in local folder

#Open Puttygen and load select that .pem and save **private key** as .ppk extension

#Connect as admin@ip & ssh as .ppk

admin@ip

kubectl get nodes



Extra Container will run as a Side car and it save logs or create proxy

#POD creation using Docker image

kubectl run testpod1 --image=index.docker.io/sreeharshav/rollingupdate:v1

ku get pods

ku describe pod testpod1

#Alias

alias dpod=’kubectl describe pod’

#To expose pod and connect from Outside → Expose is also called as service

ku expose pod testpod1 --port=8000 --target-port=80 --type=Nodeport

#Service

ku get svc

#Port Forwarding

ku port-forward pod/testpod1 8888:80

#If you need to test connection internally

curl [http://localhost:8888:80](about:blank)

#MANAGEMENT SERVER Config file:

cat .ssh/config

#Copy entire config and paste it in → Visual Studio code → File → Open Folder → place in any folder → save as config →

#Go to command prompt and check

cd C:\Terraform

kubectl.exe --kubeconfig=config get pods

#We can also do port forwarding from local command prompt

kubectl.exe --kubeconfig=config port-forward testpod1 8000:80

#If you need to check deployed pod in yaml or json

ku get pods -o yaml

#If you want to enter to testpod1 bash

kubectl exec -it testpod1 -- bash

apt update

apt install iputils-ping

ping [www.google.com](http://www.google.com)

#Check communication between pods

ping 100.96.4.6

ctrl+pq

Management server:

kops get clusters

kops get cluster devopsk8s.xyz

kops get cluster devopsk8s.xyz -o yml

nano pod.yaml

apiVersion: v1

kind: Pod

metadata:

name: memory-demo

namespace: mem-example

spec:

containers:

- name: container1

image: index.docker.io/sreeharshav/testcontainer:v1

- name: container2

image: index.docker.io/sreeharshav/testcontainer:v1

- name: container3

image: index.docker.io/sreeharshav/testcontainer:v1

kubectl create -f pod.yaml

ku describe pod memory-demo

#ku get pods

#ku edit pod memory-demo

ku delete pod memory-demo testpod1 testpod2

#Namespace creating

ku create ns ns1

ku create ns ns2

ku get ns

nano pod.yml

#Add namespace under name and delete last two containers

Example:

name : memory-demo

namespace: ns1

#Delete container 2 and container 3

ku create -f pod.yaml

nano pod.yaml

#change namespace as ns2

ku create -f pod.yaml

ku get pods -n ns1

ku describe pod -n ns2 memory-demo

bash b1-k8s-script.sh

**7.K8S-B1-KOPS-INSTALL-K8S-PODs-LABELS-SERVICE**

Management server:

ku get pods

kubectl run hello --image=httpd:latest

ku get pods

ku get pod hello -o yaml

nano pod.yaml

apiVersion: v1

kind: Pod

metadata:

name: mydemopod\

labels:

env: prod

owner: sree

spec:

containers:

- name: container1

image: index.docker.io/sreeharshav/rollingupdate:v3

kubectl create -f pod.yaml

ku describe pod mydemopod

ku get pods

Random vs Arbitary:

Random:

10,23,43,12 → Random

Arbitary:

When you call specific label and get the output

Example : ku get pods -l **env=dev**

ku describe pod mydemopod #-------> shows messages

ku describe call specific label kubetcl

**MINIKUBE:**

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

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

echo "deb https://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee -a /etc/apt/sources.list.d/kubernetes.list

sudo apt-get update

sudo apt-get install -y kubectl conntrack

curl https://get.docker.com | bash

curl -Lo minikube https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64 && chmod +x minikube

sudo mv minikube /usr/local/bin

set NO\_PROXY=localhost,127.0.0.1,10.96.0.0/12,192.168.99.1/24,192.168.39.0/24

minikube start --vm-driver=none