xbmn svfr kfyy upcq

Ask question to anis:

1- Which configuration management tool will used.

2-Kubectl rollback vs helm version

What study tommorow :
Pipeline writing
Yaml file practice
Dockerfile practice

Quations: Claer scenn go up : ctr l
how statefulset (will see file)
vi editor : /enter and n

 # SAMPLES :

1-Deployment Sample yaml :

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: sample-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: sample-app
  template:
    metadata:
      labels:
        app: sample-app
    spec:
      containers:
      - name: sample-app
        image: your-registry/your-image:latest
        ports:
        - containerPort: 80
```

2-Deployment Sample yaml :

```
pipeline {
    agent any

    stages {
        stage('Checkout') {
            steps {
                checkout scm
```

```
            }
        }

        stage('Build and Test') {
            steps {
                // Your build and test commands here
            }
        }

        stage('Build Docker Image') {
            steps {
                script {
                    // Build and push Docker image
                    docker.build("your-image-name")
                    docker.withRegistry('https://registry.example.com',
'docker-credentials-id') {

docker.image("your-image-name").push("${env.BUILD_NUMBER}")
                    }
                }
            }
        }

        stage('Deploy to Kubernetes') {
            steps {
                script {
                    // Deploy to Kubernetes
                    sh 'kubectl apply -f deployment.yaml'
                }
            }
        }
    }
}
```

3-Dockerfile :

```
# Use an official Node.js runtime as a base image
FROM node:14

# Set the working directory in the container
WORKDIR /usr/src/app

# Copy package.json and package-lock.json to the working directory
COPY package*.json ./

# Install app dependencies
RUN npm install

# Copy the application code to the container
COPY . .
```

Expose the port the app runs on
EXPOSE 8080

Define the command to run the application
CMD ["npm", "start"]

#IMP Chanel: (M Prashant)
1- Linux Commands: https://www.youtube.com/watch?v=Byx4sgLR88E&t=5688s
2- Shell Scripting:
https://www.youtube.com/watch?v=TtGM9GfBuok&list=PL0tP8lerTbX3MeIyMxGW2sLhWnPdn_xhd
3- Jenkins: Guarav Sharma
https://www.youtube.com/watch?v=XCpIQlqIyd0&list=PL6XT0grm_Tfi21F8O0TvHmb78P2uEmhDq&index=6

Versions:
1-Docker : 24.0.5 (suo apt update and sudo apt install docker.io)
2-kubernetes : 1.24.17
3-Helm : 3.13.1
4-Ubuntu : 22.04.1
5-Eks Cluster : 1.24.
6-Terraform: 1.6.4
7-Aws: 5.26.0

#Path Details:
1- Authorised key path: /.ssh/authorized_keys
2- Nginx default html.index path : /usr/share/nginx/html (in case of nginx)
3- Httpd default htdocs path : /usr/local/apache2/htdocs (in case of httpd)
4- Log file path : /var/logs
5- Docker istallation path : /var/lib/docker/
6- Kube Confog Path : /.kube/config (: ls -al command)
7- Static Pod Path : /etc/kubernetes/manifests
8- git config --list --show-origin : to show configuration in git
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= security token for github
9- Webserver always hit on port no : 80
10-Helm Install path : /user/local/bin/helm
11- Jenkins always hit on port no : 8080
12- Promethious port : 9091
13- Sonarqube port : 9000
14- Jenkins wokspace path : /var/lib/jenkins/workspace
15- Grafana port : 3000

IMP:

1-UpperDir: local docker directory
2-Docker Inspect image or container name : all info abt img or contr
3-Kubectl describe podname : all info about pod with logs.

4-Join kubadm command with worker : kubeadm token create --print-join-command

5-Docker Networking Type:

- **a)Bridge: used within a single host**
- **b)Overlay: for multi-host communication**
- **c)MacVlan: used to connect Docker containers directly to host N/W interfaces.**

6-Ubuntu os:

normal to root : sudo -i

root to normal : sudo su ubuntu

7-kubeadm : the command to bootstrap the cluster

-kubelet: make sure component that runs on all of the machines in your cluster

-kubectl: the command line until to talk to your cluster.

8-EC2 Status :

-0/2 : hardware issue, memory issue

-1/2 : OS issue, Kernel issue, N/W,antivirus,EBS issue

9-Automatic Component VPC : NACL,SG, RT

10-DockerHub : It is registory for application images

11-K8s : Kubernetes is an open-source system that automates the management, scaling, and deployment of containerized applications, **We manage control plane.**

we can see static pods like ETCD,Kube-Controller,apiserver,kube-scheduler

12-Volumes : We create a volume for Pods. In the Kubernetes world, the PODs created in Kubernetes are transient in nature. When a POD is created to process data and then deleted, the data processed by it gets deleted as well. So for prevent from them we used persistent volume and claim them.

-Persistent Volume :

A Persistent Volume is a cluster-wide pool of storage volumes configured by an administrator to be used by users deploying application on the cluster. The users can now select storage from this pool using Persistent Volume Claims.

-Persistent Volume Claim: We Claims some volume out of persistence volume.

-**Volume Types:**

-a)Persistent Volume : Remains for long period, no data loss happened if restarts pods and pods delete situations.

-b)Non Persistent Volume : Remains short period, Data loss happened if restarts pods and pods delete situations.

Storage Class : With storage class you do not to create persistent volume claim seperately before you claiming it. Its k8s object who stores info about creating persistend volume for your pods.

13-K8s Restart policy :

-Always: The pod needs to be always running so, every time stops, a new one will be spawned

-OnFailure: Only if the container terminates with a non zero return code, it will be restarted. A

container that returns a 0 (success) doesn't need to be restarted
-Never: Do not attempt to restart the container.

14- HPA : (Horizontal Pod Autoscaling).

- On the basis of matrix scaling load and its increase pod as per target load cross.

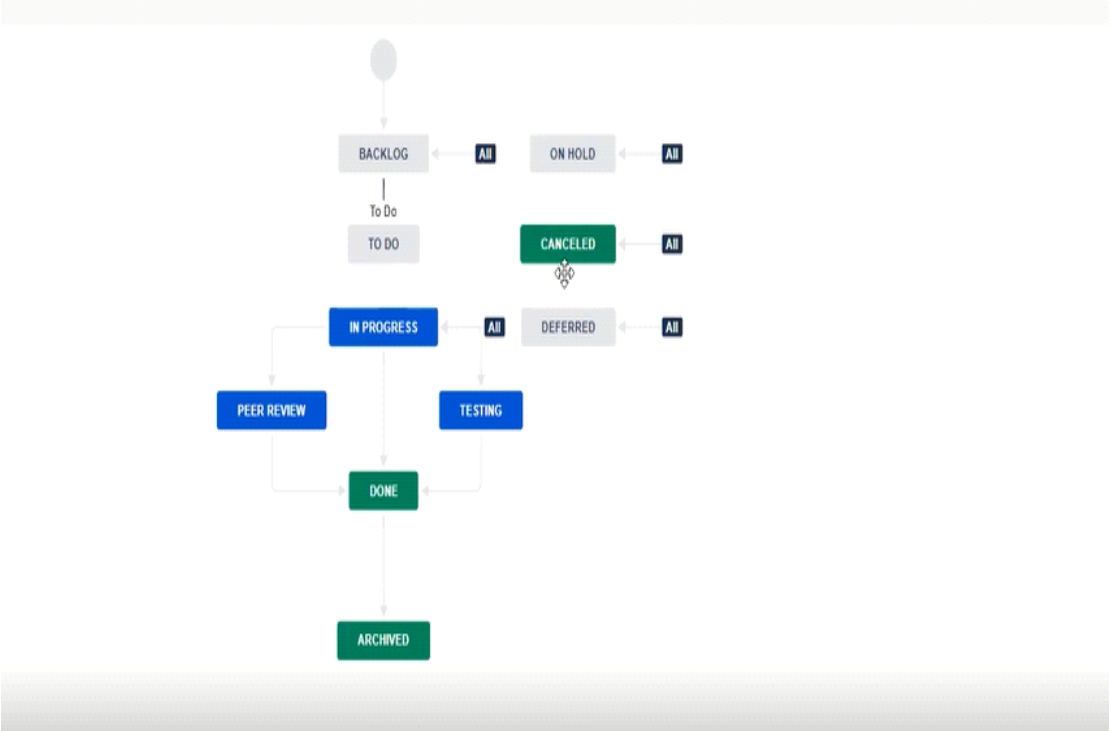
15- JIRA : **Ticketing tool**

- we work on jira as per priority :

PRIORITY : P1 (Critical)---->P2 (High)----->P3 (Normal)----->

p1- website not work, p2- website work but not some features, p3- install something, something want to change.

- JIRA WORKFLOW :



16- StatefulSet : Its kubernetes object which use pvc and store data.
Never happened data loss. used for statefull applications. Pod IDs are never change after restarting of pod. Pod will come with same IDs.

17- DEPLOYMENT : Deployment used for stateless appication. Pod IDs are always change after restarting of pod.

-Statefull Application : *data stored after pod deletion eg. Mysql, Jenkins*

-Stateless Application : *data loss after pod deletion. e.g Web servers.*

18- Kaniko : Tool which create image from dockerfile but without docker daemon. No need to install docker**.** **(Docker** use docker deamon to create image, Need to install docker**)**

19- Namespace : When we create namespace one default Service Account created.
20- In EKS if accedently loss kube config then :
aws eks update-kubeconfig --region region-code --name my-cluster

==========
==========
RIZWAN LECTURE NOTES :

Sat - 12th Aug

Trad.system vs cloud approach

1.Infrastructure ownership
2.Capital exp
3.Scalability
4.maintaince
5.Location dependancies

Traditional system vs cloud computing

Why linux and not windows (Windows : Enterprise) ?

Ans :

1. Secure os
2. Open source : (Free) ,
3. Perfect for programming
4. Easy to update
5. Better community support
6. Privacy

Types of linux :

Redhat
Debian
Ubuntu : ——> will use
Suse
Amazon linux

VPC and its components :

EC2 : ***An EC2 instance is a virtual server in the aws cloud*** it enables users to launch and run a wide range of applications and services in a scalable and flexible manner without the need to invest in expensive hardware upfront.
EC2 instances are charged on a pay-as-you-go basis, meaning users only pay for the compute resources they use. This makes EC2 instances a cost-effective and scalable solution for a wide range of use cases, from small-scale applications to large-scale enterprise workloads.

SG: security groups provide a simple and effective ***way to control network traffic to and from your EC2 instances*** in AWS.

SUBNETS : subnets in AWS provide a way to ***partition your resources within your VPC*** and help you to deploy your resources in a highly available and secure manner.

NACL : In AWS, Network Access Control Lists ***(NACLs) are an optional layer of security that act as a firewall for your VPC subnets***

Route table : route tables provide a powerful way to control ***how network traffic is directed within a VPC***. By configuring routes, you can ensure that traffic flows ***only to authorized destinations***

NAT GW : NAT gateway is a powerful service that enables resources in private subnets to access the internet securely and reliably. It's an important component of many AWS architectures, particularly those that require resources to be deployed in private subnets for security reasons.In short ***nat gateway is a managed service that allows resources in a private subnet to access the internet***

IGW : An internet gateway is a critical component of a VPC that ***provides a secure and reliable way to connect resources in your VPC to the internet***

NAT GW : will attach in subnet level
IGW : will attaching in VPC level

SUN - 13th Aug

Ipv4 = 32 bit number and divided into 4 groups

10.0.0.0/26

10 = 8 bits
0 = 8 bits
0 = 8 bits

0 = 8 bits

24 + 2 = 26

32-26=6

2 ^ x = no .of IP address

2 ^ 6 = 64

10.0.0.0/32

10.0.0 : network
0 : host

64 ip

64/4 = 16 ip for each subnet

Public subnet 1 = 16 ip
Public subnet 2 = 16 ip

Private subnet 1 = 16 ip
Private subnet 2 = 16 ip

2 ^ x = 16
2 ^ 4 = 16

Total bits - x = subnet mask
32 - x = subnet mask

32-4= 28

10.0.0.0/28 = 16 ip

10.0.0.0/28 —— 10.0.0.15/28 = 16 —> public subnet 1

10.0.0.16/28 —— 10.0.0.31/28 = 16 —> public subnet 2. 10.0.0.22

10.0.0.32/28 ---- 10.0.0.47/28 = 16 —> private subnet 1. 10.0.0.46

10.0.0.48/28 ---- 10.0.0.63/28 = 16 —> private subnet 2

2 ^ x = no .of IP address

Total bits - x = subnet mask

Clean up :

1. Terminate all ec2 instance which are running in custom VPC
2. Delete NAT GW
3. Release ELASTIC IP
4 . Delete VPC

PSCP CMD : pscp -P 22 -i <key name> <keyname> ubuntu@<ip add>:/home/ubuntu <enter>

File Permission

LINUX / MAC_Os = .pem
Windows : .ppk

r = read = 4
w = write = 2
x = execute 1

- rw- r-- r-- 1 rizwan.shaikh staff 1674 Aug 19 12:06 demo-key.pem

644

- === file(-) or directory(d)

rw- = owner
r-- = Group
r-- = other/everyone

chmod 400 demo-key.pem

Chmod <permission value> <file name>

- r— — —

-r--------@ 1 rizwan.shaikh staff 1674 Aug 19 12:06 demo-key.pem

Saqlain : ssh -i "demo-key.pem" ubuntu@ec2-3-86-15-53.compute-1.amazonaws.com

Rizwan : ssh ubuntu@ec2-3-86-15-53.compute-1.amazonaws.com

ssh-keygen -t rsa -C rizwanshaikh2352@gmail.com

==========

DOCKER

Trad approach problems :

Critical configuration
Issues with binaries and dependencies
Time consuming
Comparability issue
Operational cost

Hypervisor : a software that creates and runs virtual machines on a server

Cloud approach problems :

1 webserver : 1 ec2/VM
Issues with binaries and dependencies

Advantages of Docker (containerised approach)

Flexible : we can runt most complex application on docker in the form container
Its provide lightweight image
We can easily deploy docker on cloud
Secure and scalable
Container will update individually without disturbing other container

Docker is containerisation tool = Docker is container

Docker container using docker images to build the container

Docker container : it provides a package which includes everything needs to be run an

Application which called an image

Docker image includes everything : configuration , libraries , dependencies, application code

Docker Architecture :

Docker Client: docker client accepts commands from users and it will communicate to docker daemon for further steps
Docker Daemon / Docker server : Docker Daemon receives this commands from docker client and it will create container
Docker Host : provide an environment so that application can run (docker daemon , docker container , docker image)
Docker registry: manage and stores docker images

Ubuntu : apt : NGINX
CentOS : yum : HTTPD

docker run --name nginx -p 8080:80 -d nginx

docker run --name httpd -p 8082:80 -d httpd

8080 = host port
80 = container port

Users ——— > VM/host ——— > container
8080 ——> 80

27751fde47cf1622e39b1a602e48369cbb9bf74677ed9dfba213784c09fe783e. Container ID

ebe174bed9d7668be987f99e05d20731481870385bed8a802858ddecc6476395 Container ID

8082 ——> 80

Welcome to this docker session , I am your instructor - Rizwan

Html path = /usr/share/nginx/html (in case of nginx)

Two types of docker registry :
1. Public
2. Private

Create your own custom image
Requirement :
1 code (index.html) —— Developer
2. Create Dockerfile —— DevOps
3. Which web server want to use — nginx

Default code (html file) path - /usr/share/nginx/html

Dockerfile

==========
WHY k8s ?

challenges in docker

1. communication between 2 containers if those containers are running on different host

2. HA (Replicas)

3. self healing (if containers are in unhealthy state)…. etc.

Architecture:

cargobox --> actual goods

K8s Architecture (Master and Worker node)

Master :

Kube-apiserver : 1. it acts a a frontend component , 2. Expose k8s api , 3. All other master & worker node components will communicate to Kube-apiserver

Etcd : it stores all master and worker node information (cluster level information)

Kube-scheduler : - 1. Responsible for distributing pods across the worker nodes

Controllers/kube-controller manager : controllers are responsible for managing nodes and pods when those are go down

Worker :

Kubelet : is an agent running on every worker node , it will make sure that containers are running in a pod on node

Kube-proxy : maintains network rules , and this network rules allows communication to your pods

---

k8s resources :

1. Pod :

- pod represents a set of running containers
- its a smallest and simplest object on k8s
- one to one relationship: one pod = one container
- one to many relationships = one pod = many conatiners (default = 2 containers) ----> DB

POD lifecycle policy :

PENDING : kube-schedular is going to look appropriate resources on worker nodes

CREATING : node has assigned, now pod is creating state (inside this pod container is creating state) --- docker will pull the image and it will try to create container inside the pod.

RUNNING : during the running state -- pod and container both up & running

FAILED : when is pod is failed (inside the pod container is failed)

Succeeded : main purpose of pod is achieved.

2. ReplicaSet :

Nginx --- 5 pods (5 replicas) = 2 unhrealthy

- a replicaset maintain a set of replicas of pods running at any given time.

use : run a application with H.A

3. Deployment :

- its a k8s object that manges replicated applications/pods. (Workload Objects)

applications = Pods
replicated application : ReplicaSet

Deployment Strategies:

1. Rolling Update-------> its a default strategy in k8s. (Downtime = Zero)

2. Recreate - non- recommended (Downtime = Yes)

3. Blue-Green - create another deployment (Downtime = Zero)

Old workload = Blue === V1
New workload = green == V2

eg : Rolling Update :

1 application having 3 replicas in 3 different nodes

current application version : v1

client request : upgrade application from v1 to v2

Recommend version : v2

4. Service :

- its a k8s object and way to expose an application within k8s cluster

types of svc :

a. ClusterIP : its a default svc in k8s

when we use this type of svc ?

Ans : use of internal communication within the k8s cluster [we can use this svc for backend pods (i.e. DB)]

d. NodePort :

- expose pods/applications to the external network

ec2 instance

public ip : 10.0.0.2

docker run --name nginx -p 8080:80 -d nginx

access url : 10.0.0.2:8080 ----> 80

host ip : host port

8080 ----- host port

80 ---- container port

ec2 ---- Docker ---- host
ec2 ----- k8s ----- Node

NodePort :

host ip : host port (in case of docker)

k8s = nodeip: nodeport

nodeport range = 30000-32767

nodeport = ?

nodeip:nodeport

eg : 10.0.0.1:30000 --- app 1

eg : 10.0.2.0:31000 --- app2

eg : 10.0.2.0:32000 -- app3

eg : 10.1.2.0:30001 -- app4

container port = target port

how many ports are involved in nodeport svc ?

nodeport : (30000 to 32767)

service port : random (you can mention)
target port : random (you can mention)

eg : 10.0.0.1:30000 --- app 1

eg : 10.0.2.0:31000 --- app2

eg : 10.0.2.0:32000 -- app3

eg : 10.1.2.0:30001 -- app4

c. LoadBalancer.

url : xxxxxxxxxxxxxxxxxxxxxx

app 1 - xxxxxxxxxxxxxxxxxxxxxx/app1

app2 - xxxxxxxxxxxxxxxxxxxxxx/app2

app3 - xxxxxxxxxxxxxxxxxxxxxx/app3

app4 - xxxxxxxxxxxxxxxxxxxxxx/app4

ingress -- (will look this at the time of ingress)
https://www.jiosaavn.com/

app 1 = https://www.jiosaavn.com/new-releases

app2 = https://www.jiosaavn.com/charts

==========

YAML : yet another markup language

- its a human readable language
- its commonly used to configure application files.
- yaml is designed to be clean and easy to read
- similar to json
- to define yaml file you have use extension (.yml or .yaml)
- it stores information in key value pair

demo.txt , .pdf, .docx

Rules :

- Information will store in key=value pair
- space after colon (:) is mandatory to differentiate key and value

---

Namespace: (logically isolated environment)

- namespaces are a way to organise clusters into sub-clusters
- each namespaces logically separated from each others but with the ability to communicate with each other .
- k8s resources will deploy witin namespace only

Type of NS

kube-system : all master components
kube-node-lease = info about nodes if nodes are unhealthy state (heartbeats of nodes , availability of nodes)
default : by default every custom resources (deployment, pod , replicasets , service etc…..) will deploy in default namespace
kube-public = this ns is mostly reserved for cluster usage

Note : we can create n number of ns (as per qoata / resource availability)

Pod
Replicaset
Deployment (Pod+Replicaset)
Services (ClusterIP , NodePort (30000 to 32767) , LoadBalancer)
Namespace

There are 2 ways to create/update/modify resources in k8s :

Imperative way : using cmd line tool (using kubectl) — (single line CMD)

Declarative way : create resources using yaml file (manifest)
- commands :

kubectl create -f <file name>

k delete -f pod.yaml

To see any resources :

kubectl get <resources name>

kubectl apply -f <file name> = to change yaml and used again apply command.(manualy)

Base configuration : will provide initially

Actual configuration : -o yaml

kubectl edit : to change in yaml file= automatic change

In case pod use run cmd to create it

Create deployment and then try to edit pods whose managed by deployment (imperative)

Create pod or deployment and try to access that pod via NodePort service (via declarative approach)

--dry-run=client : hold resources to create

Kubeconfiguration file :

clusters: kubernetes
contexts: user@clustername
users: kubernetes-admin

kubernetes-admin@kubernetes

Context is a combination of cluster and user

There are 2 types of pods in k8s

Normal pods

Static pods — managed by kubelet

- we cant delete static pod using kubectl delete command
- static pods are created through manifest files (we cant create static pods using imperative approach)
- path is already defined (**/etc/kubernetes/manifests**)
- static pods follow particular naming convention (podname-ip address of CP node)
- will not use create, delete or apply command for static pods
- all master components are static pods

kube-apiserver-ip-172-31-23-112
etcd-ip-172-31-23-112
kube-controller-manager-ip-172-31-23-112
kube-scheduler-ip-172-31-23-112

10 nodes of cluster

Replicas : 10

Wn 1 : 2 pods
Wn 2 : 3 pods
Wn 3 : 1 pod
Wn 4 : 1 pod
Wn 5 : 3 pod
Wn 6 : other application pod
Wn x : other application pod

Logs-agent (har ek nodes ke logs ko collect karna)

Deployment: 10

Daemonset:

10 nodes : 10

20 : 20

nginx-server	2/2	Running	0	6s
nginx-server	1/2	Running	0	6s

there are 2 types of account in k8s

User account - humans — CSR
Service account - applications (pods)

username: lucky : declarative
Password: 1234

Username=lucky : imperative

Configmap (non sensitive) — pod status (completed), exec (no), logs (yes)
Secret (sensitive) - pod (running) , exec (yes) , logs (yes) . logs (yes)

DevOps Tools :

1-GIT : Git is an open-source distributed version control system.
-GitHub : GitHub is one place where project managers and developers coordinate, track, and update their work.(repository for app codes)

The Git workflow is divided into three states:

a-Working directory - Modify files in your working directory
b-Staging area (Index) - Stage the files and add snapshots of them to your staging area (ready to push for public/pub repository)
c-Git directory (Repository) - Perform a commit that stores the snapshots permanently to your Git directory

COMMANDS: git clean -f filename = to delete from wdir and all.

2-HELM : The package manager for Kubernetes

-The original goal of Helm was to provide users with a better way to manage all the Kubernetes YAML files we create on Kubernetes projects.

Below are the important component of helm tool:
-chart.yaml: This is where you'll put the information related to your chart

-values.yaml: this is the main file that contains defaults for variables. (it contains configurable parameters and their default values for a Helm chart. It allows users to customize the deployment of the chart by providing specific values for the defined parameters.)

-templates (dir): This is the place where you'll put all your manifest files. Everything in here will be passed on and created in Kubernetes.

-Some Commands :

-a-`helm show values stable/tomcat(chartname) :` **to check current details(like -o yaml in k8s)**

-b-`helm get values testchart2(releasname):` **to check what exact changes done.**

-c-`helm get manifest testchart2releasname):` **after change check current details**

3-EKS :

Amazon Elastic Kubernetes Service (Amazon EKS) is a managed service that eliminates the need to install, operate, and maintain your own Kubernetes **control plane** on Amazon Web Services (AWS). **Aws manage control plane. we can't see static pods like ETCD,Kube-Controller,apiserver,kube-scheduler.**

-Cluster : It is collection of virtual machines (VMs) to work together to perform a task. Clusters typically have a single head node (sometimes called a controller node), a number of compute nodes, and possibly some specialty nodes.

-Eksctl : ksctl is a simple command line tool for creating and managing Kubernetes clusters on Amazon EKS.

-Jumpbox/Baston host : A system on a network used to access and manage devices in a separate security zone. Its neither master node nor control node. Its simple EC2 m/c from where we will access cluster because we dont have master node like k8s to access cluster.

4-Ingress : (Single url multiple apps can access)

In Kubernetes, an Ingress is an object that allows access to Kubernetes services from outside the Kubernetes cluster.

{ **Genarally following ways we can acess or expose applicaion :**

-ClusterIP : exposes the Service on a cluster-internal IP. Access internally.

-NodePort: exposes the Service on each Node's IP at a static port. Access externally.

-LoadBalancer: exposes the Service externally using a cloud plateform. }

Following are the different component of Ingress :

-Ingress controller : An Ingress controller is responsible for fulfilling the Ingress, usually with a load balancer. Ingress controllers will usually track and communicate with endpoints behind services instead of using services directly.Ingress controller is the entity which grants (or remove) access, based on the changes in the services, pods and Ingress resources.

There are many different Ingress controllers, e.g. Nginx, Ambassador, EnRoute, HAProxy, AWS ALB, AKS Application Gateway (from GCP, AWS, and Azure).

-Ingress resource : The Ingress resource is a set of rules that map to Kubernetes services. Ingress resources are defined purely within Kubernetes as an object that other entities can watch and respond to

5-Jenkins : (automate task related to build,test and deploy).

Jenkins is a self-contained, open-source automation server that can be used to

automate all sorts of tasks related to building, testing, and delivering or deploying software. Jenkins can be installed through native system packages, Docker, or even run standalone by any machine with a Java Runtime Environment (JRE) installed.
we can use jenkins as a pod and also as ec2 server

-Continuous Integration : A software development process where the changes made to software are integrated into the main code as and when a patch is ready so that the software will be always ready to be - built, tested, deployed, monitored - continuously. (e.g : code change,bug fixes,new impro,new features).

-Continuous Delivery : This is a Software Development Process where the continuously integrated (CI) changes will be tested & deployed continuously into a specific environment, generally through a manual release process, after all the quality checks are successful. (e.g : sanity performance testing, Quality testing,load testing, latency testing by jmeter app).

-Continuous Deployment : A Software Development practice where the continuously integrated (CI) changes are deployed automatically into the target environment after all the quality checks are successful

-Jenkins Ways : Theire 2 ways we can use jenkins :
1- As a server (EC2) : We need to add credential in manage jenkins. cant see agent.
2- As a pod : We dont need to add credential in manage jenkins,its automaticaly detects kube config file.
JNLP agent : Will run pipeline. Once run pipeline agent pod will terminate.

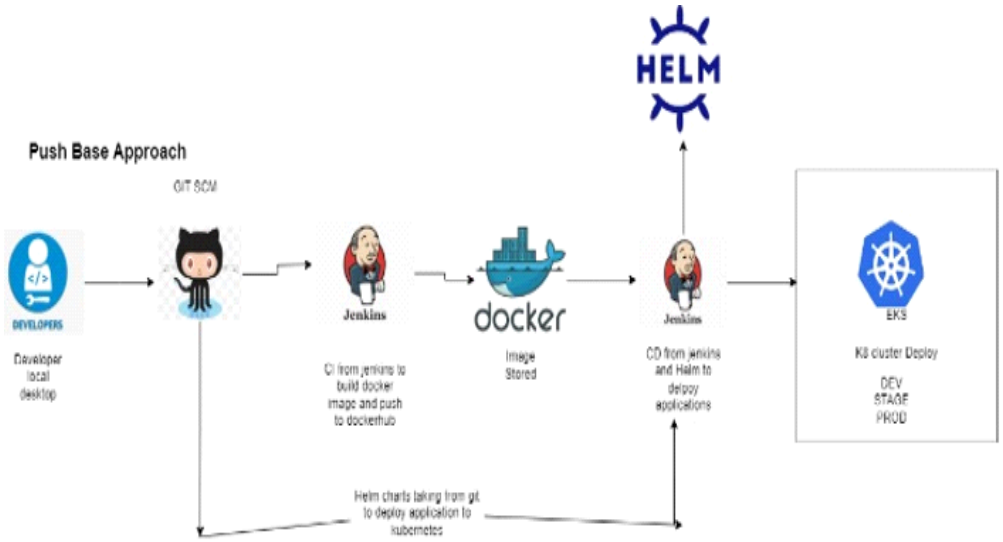
Based on the level of automation, the above three paradigms can be better represented as below -

Code -----> Build -----> Integrate -----> Release -----> Deploy

|----Continous Integration----------|
|----------Continous Delivery----------|
|----------Continous Deployment----------|

Jenkins Approch types : There are 2 types of approaches for Jenkins **:**

a : Push based approach :



b : Pull based approach

-Maven : Maven is written in Java and is used to build projects and generate artifact (dependencies) written in C#, Scala, Ruby, etc
-maven use pom.xml file to generate artifact.
-In poc environment we are storing artifacts in jenkins workspace.

-Revers Proxy : It acts like internal load balancer and route traffic.

IMP-Scanerio Based Questions :
Let's dive into some advanced scenario-based questions and power-packed answers to set you up for success:

Stateful Application Deployment:

Question: "How would you deploy a stateful application in Kubernetes, considering data persistence and scaling challenges?"

Answer: "I'd opt for a StatefulSet, ensuring ordered pod creation and stable network

identities. Leveraging Persistent Volumes (PVs) and Persistent Volume Claims (PVCs), I'd secure data persistence. Horizontal Pod Autoscaling (HPA) can dynamically adjust replicas, while considering the stateful nature of the app."

Pod Security Policies (PSP):
Question: "In a security-focused Kubernetes environment, how do you implement Pod Security Policies? Can you share an example scenario where PSPs enhance cluster security?"

Answer: "Pod Security Policies act as a crucial defense layer. I'd define policies to control pod capabilities, limit privilege escalation, and enforce container runtime constraints. For instance, I might restrict pod execution to specific namespaces, enhancing overall cluster security."

Multi-cluster Communication:
Question: "Imagine a scenario where you have multiple Kubernetes clusters. How would you design communication between services across these clusters?"

Answer: "For seamless cross-cluster communication, I'd explore solutions like service meshes or federation. Utilizing technologies such as Istio or Linkerd, I'd establish secure and efficient communication channels, ensuring services can interact across cluster boundaries."

Custom Resource Definitions (CRDs):
Question: "Share an advanced use case where Custom Resource Definitions (CRDs) play a pivotal role in extending Kubernetes functionality."

Answer: "Consider a scenario where I need to manage a specialized workload, like machine learning models. I'd create a CRD defining the custom resource for models, allowing seamless integration into the Kubernetes ecosystem. This way, I can leverage Kubernetes controllers to automate model deployment and scaling."

Kubernetes Cluster Upgrades:

Question: "How do you approach Kubernetes cluster upgrades in a production environment? Outline your strategy, emphasizing minimal downtime and rollback procedures."

Answer: "I'd adopt a phased approach, starting with a smaller subset of nodes to validate compatibility. Using tools like kubeadm, I'd perform the upgrade with careful

monitoring. Rolling updates ensure minimal downtime. Having a solid rollback plan, including snapshotting etcd and validating backups, provides a safety net."

Q: What is the role of a DevOps engineer in the software development lifecycle?

A: The role of a DevOps engineer is to bridge the gap between development and operations, focusing on automation, collaboration, and continuous improvement. We streamline the software delivery process, ensuring faster and more reliable releases.

Q: Can you explain the concept of continuous integration (CI) and continuous deployment (CD)?

A: Continuous Integration involves regularly merging code changes into a shared repository, automatically triggering builds and tests. Continuous Deployment extends this by automatically deploying the code to production if it passes all tests, ensuring a rapid and automated release process.

Q: How do you handle version control, and which tools are you familiar with?

A: I have experience with Git, which I use for version control. Git allows for efficient collaboration, branching, and merging. I also understand the importance of commit messages and maintaining a clean commit history for better traceability.

Q: Describe a situation where you optimized a CI/CD pipeline for better efficiency.

A: In my previous role, I optimized the CI/CD pipeline by parallelizing test suites and optimizing build scripts. This significantly reduced build times, providing faster feedback to developers and enabling quicker releases.

Q: How do you ensure the security of the CI/CD pipeline and the deployed applications?

A: I prioritize security by implementing code scanning tools in the pipeline, regularly updating dependencies, and following security best practices. Access controls and encryption are enforced, and I actively monitor for vulnerabilities in both the pipeline and deployed environments.

Q: Have you worked with any configuration management tools? If so, which ones?

A: Yes, I've worked with Ansible for configuration management. It allows me to automate the provisioning and configuration of infrastructure, ensuring consistency across different environments.

Q: What is Infrastructure as Code (IaC), and how have you implemented it in your work?

A: Infrastructure as Code involves managing and provisioning infrastructure through code. I've used Terraform to define and deploy infrastructure in a repeatable and version-controlled manner. This approach improves reproducibility and reduces manual errors.

Q: Can you explain the benefits of containerization and how you've utilized it?

A: Containerization, using tools like Docker, provides a consistent and portable environment. I've containerized applications to simplify deployment, enhance scalability, and ensure that they run consistently across various environments.

Q: How do you approach monitoring in a DevOps environment, and what tools have you used?

A: Monitoring is crucial for identifying and addressing issues promptly. I've used tools like Prometheus to monitor system metrics and set up alerts for proactive issue resolution. Monitoring helps ensure the health and performance of applications in production.

Q: Describe a scenario where you had to troubleshoot and resolve a production incident.

A: During a production incident, I used logging and monitoring tools to identify the root cause. I collaborated with the development team to implement a temporary fix for immediate resolution and later worked on a permanent solution to prevent similar incidents in the future.

Q: Explain the difference between continuous integration and continuous deployment.

A: Continuous Integration (CI) involves frequently integrating code changes into a shared repository with automated builds and tests. Continuous Deployment (CD) takes

CI a step further, automatically deploying code changes to production after passing tests.

Q: How do you handle secrets and sensitive information in your infrastructure and deployment scripts?

A: I utilize secure vaults and secret management tools like HashiCorp Vault or AWS Secrets Manager. Credentials are stored securely, and I ensure they are never hard-coded in scripts or configuration files.

Q: Describe a situation where you implemented or improved monitoring for a system.

A: I enhanced monitoring by implementing Prometheus for metrics collection and Grafana for visualization. This allowed us to proactively identify performance bottlenecks, predict potential issues, and maintain a reliable system.

Q: Have you worked with container orchestration tools? If so, which ones, and how did you use them?

A: Yes, I've worked with Kubernetes for container orchestration. I've deployed applications, managed clusters, and utilized features like auto-scaling to ensure optimal resource utilization.

Q: How do you approach infrastructure scaling, and what factors influence your decision to scale horizontally or vertically?

A: I assess the application's architecture and resource utilization. Horizontal scaling is preferred for distributed systems to handle increased load, while vertical scaling may be suitable for improving the performance of individual components.

Q: Can you explain blue-green deployments and how you've implemented them?

A: Blue-green deployments involve switching between two identical environments—one running the current production version (blue) and the other with the new version (green). I've implemented this by gradually routing traffic from blue to green, ensuring minimal downtime during releases.

Q: How do you ensure high availability in a cloud environment, and which cloud platforms are you familiar with?

A: I ensure high availability by distributing applications across multiple availability zones and regions. I've worked with AWS, Azure, and GCP, leveraging their services like auto-scaling, load balancing, and redundant storage to enhance system resilience.

Q: Describe your experience with scripting and automation tools.

A: I'm proficient in scripting languages like Bash and Python. I've used automation tools such as Ansible to streamline configuration management and deployment processes, reducing manual intervention and ensuring consistency.

Q: What strategies do you use for disaster recovery and backup planning?

A: I implement regular backups of critical data and configurations, ensuring quick recovery in case of data loss or system failures. I also devise and test disaster recovery plans to minimize downtime during unforeseen events.

Q: How do you stay updated with the latest trends and technologies in the DevOps domain?

A: I regularly attend webinars, conferences, and participate in online forums. Additionally, I follow industry blogs, subscribe to newsletters, and engage in continuous learning through online courses to stay informed about emerging technologies and best practices.

Q: Let's say a Kubernetes job should finish in 40 seconds; however, on a rare occasion, it takes 5 minutes. How can I make sure to stop the application if it exceeds more than 40 seconds?

A: You can set a timeoutSeconds field in the Kubernetes Job manifest. This field specifies the maximum allowed duration for the Job. If the Job exceeds this duration, it will be automatically terminated.

Q: How do you test a manifest without actually executing it?

A: Use the --dry-run=client option with kubectl apply or kubectl create to validate and generate the configuration without applying it to the cluster.

Q: How do you initiate a rollback for an application?

A: Use the kubectl rollout undo command, specifying the deployment or resource name. This will roll back to the previous version of the application.

Q: How do you package Kubernetes applications?

A: Kubernetes applications are packaged using YAML manifests. These manifests describe the desired state of the application and its components. Tools like Helm provide a higher-level abstraction for packaging and managing Kubernetes applications.

Q: What are init containers?

A: Init containers are containers that run and complete before the main application containers start. They are often used for setup or initialization tasks such as preparing data or waiting for a specific condition.

Q: What is node affinity and pod affinity?

A: Node affinity is a feature that allows you to constrain which nodes your pod is eligible to be scheduled based on node labels. Pod affinity, on the other hand, allows you to constrain which nodes a pod can be scheduled based on the presence of other pods.

Q: How do you drain the traffic from a Pod during maintenance?

A: Use the kubectl drain command to gracefully evict pods from a node before maintenance. This ensures that the workload is rescheduled to other nodes, minimizing downtime.

Q: I have one POD, and inside, 2 containers are running (Nginx and Wordpress). How can I access these 2 containers from the browser with an IP address?

A: You cannot access individual containers directly. Instead, expose the necessary ports through a Kubernetes Service, and access the service using the assigned ClusterIP.

Q: If I have multiple containers running inside a pod and I want to wait for a specific container to start before starting another one.

A: Containers within a pod start concurrently. If one container depends on the other, consider using Init Containers or implementing a mechanism within the application logic to wait for dependencies.

Q: What is the impact of upgrading kubelet if we leave the pods on the worker node - will it break running pods? Why?

A: Upgrading kubelet while leaving pods running may result in compatibility issues. It's recommended to drain the node, evicting pods to other nodes, then upgrade kubelet, and finally, allow pods to be rescheduled.

Q: How is a service that selects apps based on the label and has an externalIP?

A: Define a Service with the appropriate label selector. Set the externalIPs field in the Service manifest to assign an external IP address to the Service.

Q: Does the container restart when applying/updating the secret object (kubectl apply -f mysecret.yml)? If not, how is the new password applied to the database?

A: No, updating a secret does not trigger a container restart. The updated secret is automatically mounted into the pod, and the application can access the new credentials without a restart.

Q: How to take backup of nodes in K8S?

A: By using Valero tool we can take backup of nodes, PVCs. And store in s3 buckets.

Q: How should you connect an app pod with a database pod?

A: Use Kubernetes Services. Deploy the database as a service and configure the application pod to connect to the database service using the service's DNS name.

Q: How to configure a default ImagePullSecret for any deployment?

A: Create a Kubernetes secret containing Docker registry credentials and reference it in the imagePullSecrets field of the pod's service account or directly in the deployment.

Q: If you have a pod that is using a ConfigMap which you updated, and you want the container to be updated with those changes, what should you do?

A: Roll out an update to the pod using kubectl apply or kubectl rollout restart deployment <deployment-name>. This triggers a restart of the pod with the updated ConfigMap.

DevSecops : We used some tools for security, like OWASP, SONARQUBE, TRIVY.

OWASP : The Open Web Application Security Project (OWASP) is a non-profit organization founded in 2001, with the goal of helping website owners and security experts protect web applications from cyber attacks.

SONARQUBE : Used for code quality analysis. Check vulnerability.

TRIVY : Trivy is a simple and comprehensive vulnerability scanner for containers and other artifacts. A software vulnerability is a glitch, flaw, or weakness present in the software or in an Operating System. Also scan docker images.

Monitoring Tools : Promethious and Graphna

Promethious : Collecting matrices from your servers and clusters for e.g. EC2 memory use, CPU used, Latency matrics. Application used for event monitoring and setup alerts.

```
- job_name: 'node_exporter'
    static_configs:
      - targets: ['localhost:9100']
  - job_name: 'jenkins'
    metrics_path: '/prometheus'
    static_configs:
      - targets: ['18.215.171.31:8080'
```

1:11

Below are some components of promethious :

-Node exporter(node matrics) : Node exporter will works as a daemonsets and fetch data from nodes and send to the promethious. It will send data related to nodes like node cpu, node memory, nod has not ready etc.

-Kube state matrix (kubernetes matrix) : K8s matrics handle by kube_state_matrics.

-Alert Manager : Send notification alerts on emails e.g. Cpu usage, memory usage

Graphna :

Its create dashboards from promethious metrices. It use promethious as a datasource.

Terraform : IAC (Infrastructure as a code)(File extension is .tf)

- Terraform is an IAC tool, used primarily by DevOps teams to automate various infrastructure tasks.

Before Terraform issues like : provision manually, human write sysntax error, slow working, does not scaling, if any resource change in code than cannot trackable, Disaster recovery not possible, Inefficient.

- All above issues solve by terraform.

Disaster Recovery :

It means we have exact infrastructure in 2 regions. So if any region goes down we can acess servers from other regions.

Terraform Workflow :

1-Terraform init (create .terraform folder): (only once do this command)

It will download provider and initialise provider(download all respective configuration of cloud plateform like aws/azure/google cloud.) and also create .terraform folder.

- Providers : Its heart of terraform, cant work terraform without providers. Its in the form of .terraform folder.

2-Terraform validate : ***It will show if theire is any syntax error happens. It will show like syantax is invalid. And if syntax is ok then it will validate code.***

3-Terraform fmt : ***It will format your code as per terraform standard syntax like check spaces.***

4-Terraform plan : ***It will show execution plan like sequence wise. It will indicate like some resources need to added. It will show dry run output.***

5-Terraform apply : ***It will apply code to create resources. That means actual execution will happened.***

6-Terraform destroy : ***It will delete resources but one backup file will provide. When we run this command it delets data inside of .tfstate file which are create by terraform but it stores all that data into .tfstate.backup file for future uses.***

Terraform Blocks :

-In Terraform, a block is a fundamental unit used to define and configure different aspects of your infrastructure. Blocks are written in HashiCorp Configuration Language (HCL) and allow you to declare resources, providers, variables, outputs, and other configuration elements within your Terraform code.

Terraform Blocks Types :

1-Fundamental Blocks :

a-Terraform Block : Defines the overall configuration and settings for Terraform.

b-Provider Block : Specifies the cloud or infrastructure provider for the resources.

c-Resource Block : Declares and configures a specific infrastructure resource offered by the provider.

2-Variables Blocks :

a-Input Variable Block : Defines variables that users can input values for.

b-Output Values Block : Declares values that Terraform will display after applying confign.

c-Local Values Block : Specifies local computations or values within the Terraform configuration.

3-Calling And reference Blocks :

a-Data Source Block : Retrieves data from an external source or provider.
b-Modules Block : Organizes and reuses Terraform configurations as modular units.

Terraform.tfstate file : This file creates automatically, all the important configuration are store in this file. We always take backup of this file. If this gets deleted accidentally all resources will gets deleted.

Variables.tf file : When you do terraform apply variables are fetch from this files.

Terraform.tfvars file : When you do terraform apply the values take from this vars files instead of variable files. Its means it has high priority.

Remote State File :
A file in Terraform that stores the state of the infrastructure remotely, typically in a shared storage backend. It allows collaboration among team members and enables Terraform to manage infrastructure as code across multiple environments.

Terraform Backend File : Terraform Backend File: A file that configures the backend settings for Terraform, specifying where and how the Terraform state is stored, such as in a remote storage service like AWS S3 or HashiCorp Consul.

State Locking Feature :
A mechanism in Terraform that prevents multiple users or processes from concurrently modifying the same infrastructure state, ensuring consistency and avoiding conflicts.

Terraform Modules : Modules are container of multiple resources that are use together. It consist of collection of .tf files.
In short:

Reusable Components: Modules in Terraform are a set of Terraform configuration files grouped together to form a reusable component.

Encapsulation: Modules encapsulate infrastructure components, abstracting away complexity and providing a clean interface.

Parameterization: Modules can be parameterized, allowing you to customize their behavior by passing input variables.

Abstraction: Modules allow you to abstract infrastructure details, promoting code reusability and maintainability.

Terraform Modules Types:

a-Public : Publicaly available modules which anyone can use.

b-Private : Publicaly not available..

Terraform Import : (track manually created resourses into state files)

- Purpose**:** Used to import existing infrastructure into Terraform state.
- Scenario**:** When infrastructure is created outside Terraform, and you want to manage it using Terraform.
- Usage**:** `terraform import` command followed by the resource type and resource address.

IMP: We need to write resource block of same config running. And then use import command.

E.g. terraform import aws_instance.my_instanceID

Importing does not create Terraform code; it associates existing resources with Terraform state. Manual adjustments to the configuration may be necessary after import.

Terraform Provisioners:

Terraform, provisioners are a feature that allows you to execute scripts or tasks on a remote resource after it's been created. Provisioners can be useful for tasks such as bootstrapping, configuration management, or other actions that need to be performed on a resource after it has been provisioned.

1-File Provisioners:

It will act like scp command., Copy local to remote.

2-Local-Exec Provisioners:

- `local-exec` provisioners allow you to run commands locally on the machine where Terraform is executed. These commands can be used for tasks such as running scripts or configuring local settings.

3-Remote-Exec Provisioners:

- `remote-exec` provisioners allow you to run scripts or commands on the remote machine after it has been provisioned. This is often used for tasks like software installation or configuration management

Ansible : Configurations Management Tool

- E.g: Suppose you want to change 100s of EC2 servers time on one zone to other zone this has been example of configuration management tool. It will use playbook.yaml and inventory.yamls. If you perform any task on 100s of machines then ansible is the solution.

ARGOCD :

ArgoCD is an open-source declarative continuous delivery tool for Kubernetes. It allows you to maintain, visualize, and automate the deployment of applications and configurations in Kubernetes clusters. ArgoCD follows a GitOps approach, which means that the desired state of your applications and infrastructure is specified in Git repositories, and ArgoCD continuously monitors and syncs the actual state with the desired state.

Link: https://www.preethi-devops.com/syllabus

Kubeernetes: DevOps Pro

Terraform With EKS:

Q- What happened when we upgrade cluster though terraform?

A- First check official docs for new release docs. Upgrade control plane first. After that manually update following component: CoreDNS, Kube-proxy, VPC CNI