# Radical Institute

AWS and Linux

Q. Create the Load balancer in AWS ?

1. First create the two instance which have different zone also save both the different name.

2. Second the create the target group where you need below details

Choose a target type : Instances

Target group name : XYZ

Protocol : Port

IP address type

Protocol version

3. Select the both the Ec2 instance and save it.

4. Create the load balancer, and check Load balancer types

* Application Load Balancer :Layer 7
* Network Load Balancer : layer 4

5. Fill basic details after that select the Availability Zones and subnets (minimum 2)

6. After that select the security group which are require and the last need to select the target group.

7. Then Create load balance after that wait until there activation.

### Q. **Difference Between Vertical Scalability and Horizontal Scalability in AWS**

|  |  |  |
| --- | --- | --- |
| Feature | Vertical Scalability | Horizontal Scalability |
| Definition | Increasing the capacity of a single instance (CPU, RAM, storage) | Adding more instances to distribute the load |
| How it Works | Upgrading an existing EC2 instance type (e.g., t2.medium → t2.large) | Adding multiple EC2 instances behind a load balancer |
| AWS Services Used | Auto Scaling (single instance), EC2 instance resizing | AWS Auto Scaling Groups, ELB (Elastic Load Balancer) |
| Cost Efficiency | Can be costly at a certain limit | More cost-efficient as instances can be scaled dynamically |
| Performance Impact | Can lead to downtime while upgrading | No downtime, as load is distributed |
| Scalability Limit | Limited by the max capacity of a single machine | Virtually unlimited by adding more machines |
| Best For | Applications with high processing needs but limited parallelism | Applications requiring high availability and redundancy |

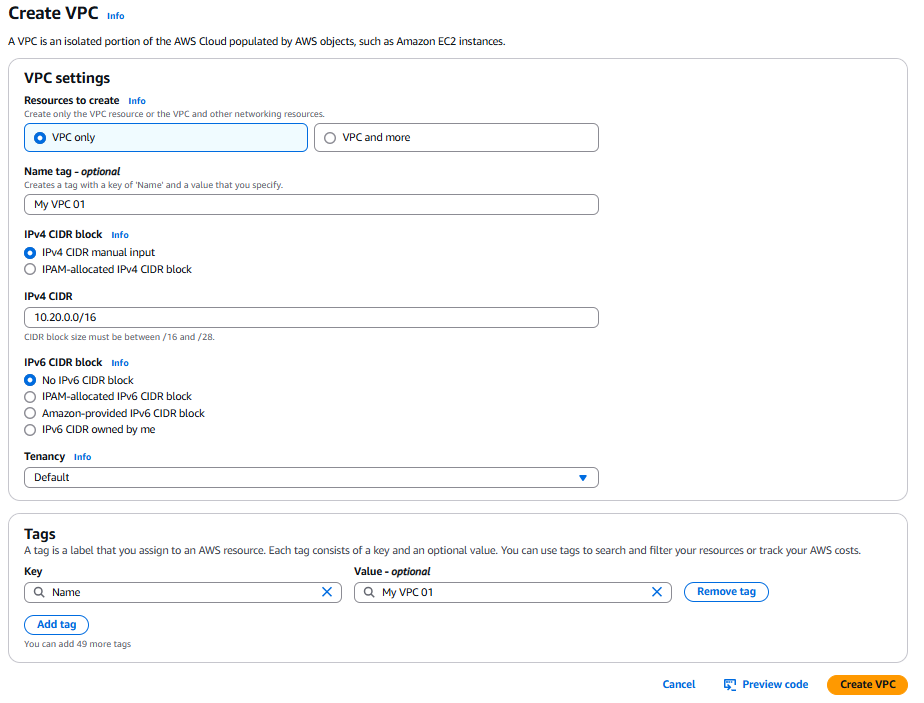
### **Q Difference Between Public and Private IP (Definition-wise)**

* **Public IP:** A globally unique IP address that allows devices to communicate over the internet. It is assigned by an ISP (Internet Service Provider) and can be accessed from anywhere in the world.
* **Private IP:** An IP address used within a private network (such as a home or office) that cannot be accessed directly from the internet. It is assigned by a local network router for internal communication

### **Q Create the virtual Private Cloud (VPC) also attached Internet gateway and route with the subnet?**

1. We need to first create VPC.

* First we need to define the name as well as IP



1. Second we need to create the subnet and add with the VPC which are created new make sure that the IP which are given to subnet is more than the VPC ip

Example:

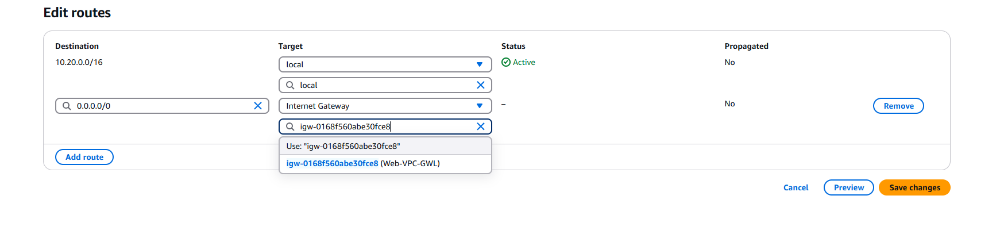
**IPv4 VPC CIDR block**

10.20.0.0/16

**IPv4 subnet CIDR block**

10.20.1.0/24

1. For connectivity of the internet we have create the Internet gateway after that we need to attached the respective VPC.
2. Than create one EC2 instance and add different VPC instead of the default and add security group after that the Launch the instance.
3. Once launch the instance we need to create the route against the VPC, which provide the actual connection between outside the local network



# Configure the Maven in Jenkin Server:

Note: Java and maven should be install in the Jenkin EC2 instance:

Maven:

[ec2-user@ip-172-31-14-252 ~]$ mvn --version

**Apache Maven 3.9.8 (36645f6c9b5079805ea5009217e36f2cffd34256)**

Maven home: /opt/apache-maven-3.9.8

Java version: 17.0.14, vendor: Amazon.com Inc., runtime: /usr/lib/jvm/java-17-amazon-corretto.x86\_64

Default locale: en\_US, platform encoding: UTF-8

OS name: "linux", version: "5.10.235-227.919.amzn2.x86\_64", arch: "amd64", family: "unix"

JaVa:

[ec2-user@ip-172-31-14-252 ~]$ java --version

openjdk 17.0.14 2025-01-21 LTS

OpenJDK Runtime Environment Corretto-17.0.14.7.1 (build 17.0.14+7-LTS)

OpenJDK 64-Bit Server VM Corretto-17.0.14.7.1 (build 17.0.14+7-LTS, mixed mode, sharing)

[ec2-user@ip-172-31-14-252 ~]$

1: We need to add tools in the Jenkin UI:

Path : Mange Jenkin -> Tools -> JDK installations -> add Jdk :Java-17->path: /usr/lib/jvm/java-17-amazon-corretto.x86\_64

Same add for the Maven as well as

Add Maven: Maven-3.9.8 -> path -> /opt/apache-maven-3.9.8

Then Save

2: We need to install the two pluggin:

* Maven Integration
* Maven invoker

3:Now we need to run the Job

* First we need go to configure
* Select **Invoke top-level Maven targets** t in the Build Step
* Select your Maven version as well as select goal like , clean package
* Then run the job

4: Check the workspace the package you have run are generated or not.

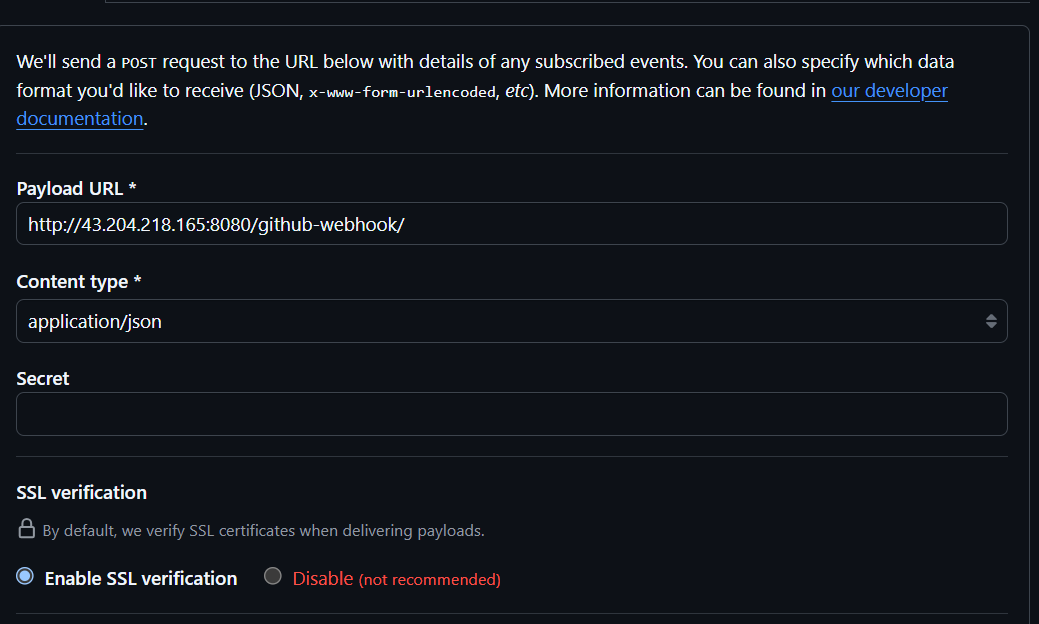
**Triggering Job**

**Q What is git Webhook?**

A **Git webhook** is a way for **Git repositories (like GitHub, GitLab, Bitbucket, etc.)** to **automatically notify external systems** (like Jenkins, CI/CD tools, servers, or apps) when certain events happen in the repository — such as a **push**, **pull request**, **merge**, etc.

For this we need to configure the webhook in github account;

Login git-> setting->webhook->fill details->**Which events would you like to trigger this webhook?**



Once done login you jenkin account and triggering your job as:

“GitHub hook trigger for GITScm polling”

Now, you can clone git hub resporitry and add the another file push the file in that resposirty from you local system.

You can see that job automatically run without manual interruption.

jbasone@jbasone-lap MINGW64 ~/Documents/DEVOPS/Radical/Maven/maven (master)

$ git push -u origin master

Enumerating objects: 4, done.

Counting objects: 100% (4/4), done.

Delta compression using up to 4 threads

Compressing objects: 100% (2/2), done.

Writing objects: 100% (3/3), 363 bytes | 363.00 KiB/s, done.

Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)

To https://github.com/Jayant-git-debug/Maven.git

d0e310d..ebff29c master -> master

branch 'master' set up to track 'origin/master'.

[ec2-user@ip-172-31-14-252 Maven-Project\_01]$ ls -ltr

total 8

drwxr-xr-x 4 jenkins jenkins 30 Apr 24 14:44 **src**

-rw-r--r-- 1 jenkins jenkins 2642 Apr 24 14:44 pom.xml

-rw-r--r-- 1 jenkins jenkins 17 Apr 24 14:44 jayant.md

drwxr-xr-x 9 jenkins jenkins 191 Apr 24 14:44 **target**

[ec2-user@ip-172-31-14-252 Maven-Project\_01]$

# Docker

We need to first create the ec2 instance and install the Docker on that EC2 machine once install we need to provide the user in the user group were we have freely use the docker without using of the docker command, below are the command:

sudo usermod -a -G docker ec2-user

Check user was added or not:

sudo cat /etc/group

docker:x:992:ec2-user

Some are commands important for the Docker:

docker images ls –a >> Docker images View

docker container >>View all container

docker system –help >>to check the Systeam information

docker image pull hello-world >> pulling image

docker container run hello-world >> create the container and run that

-a, --attach list Attach to STDIN, STDOUT or STDERR

-d, --detach Run container in background and print container ID

-e, --env list Set environment variables

-i, --interactive Keep STDIN open even if not attached

-m, --name string Assign a name to the container

-p, --publish list Publish a container's port(s) to the host

-t, --tty Allocate a pseudo-TTY

Command for generating the container:

docker image pull httpd

docker container run --name websrever -d -i -t -p 7000:80 httpd

docker container stop webserver >> Stop the container

docker container start webserver >> Start the container

docker container inspect webserver >> for detail for the image and container

docker container exec -it websrever /bin/bash >> execute the container

Path for changes >>> cd htdocs/

How to create the images and run as containerization?

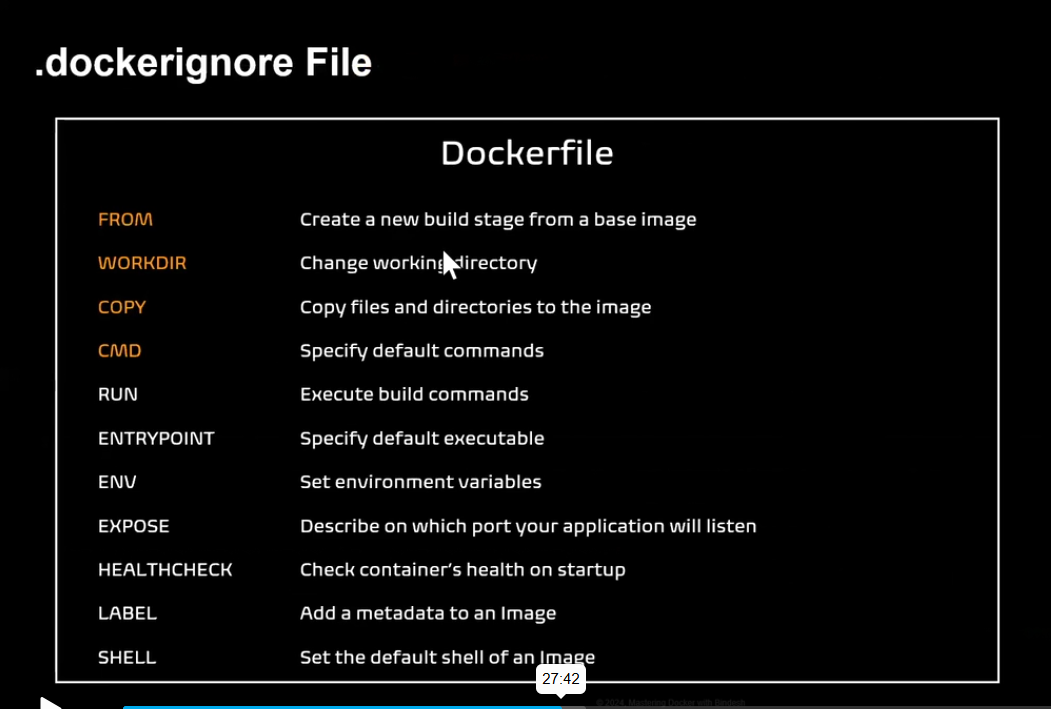
-First you need to login the EC2 instance where you have install Docker:

-Now Creating the directory in the Docker server

-Under the docker we can create the file name image, where we clone the git resposrity of the python pogram

-While creating the image we require the one text file which is help us to execute the command

-so creating the text file inside same path “Dockerfile”



-inside file we have pasting this command:

[ec2-user@ip-172-31-11-22 python-flask-app]$ cat Dockerfile

FROM python:3.11-alpine

LABEL maintainer="Jayant"

WORKDIR /app

COPY . .

RUN pip install -r requirements.txt

EXPOSE 8080

CMD ["python","src/app.py"]

-after saving the file we need to run the pogram:

docker image build -t pyflaskapp:v1 .

docker conatiner run --name "Mypython" -dit -p 5005:8080 pyflaskapp:v1

-you we will get the page automatically in the web page

Docker is a containerization tool.

Virtualization -- Fixed hardware

allocation.

Containerization - No Fixed HardwareProcess isolation ( Dependency in osis removed )

+++++++++++++++++++++++++

In comparison to the traditional virtualization functionalities of hypervisors,Docker containers eliminate the need for a separate guest operating system for every new virtual machine.Docker implements a high-level API to provide lightweight containers that run processes in isolation.

A Docker container enables rapid deployment with minimum run-time requirements. It also ensures better management and simplified portability.

This helps developers and operations team in rapid deployment of an application.

++++++++++++++++++++++++

Create Ubuntu Machine on AWS

All Traffic - anywhere Connect using git bash https://get.docker.com/

Go to Root Account

$ sudo su -

# curl -fsSL https://get.docker.com -o get-docker.sh ( this will download shell script in the machine)

# sh get-docker.sh ( This will execute the shell script, which will install docker )

How to check the docker is installed or not

# docker --version

We should be comformatable with four terms

1) Docker Images

Combinations of binaries / libraries which are necessary for one software application.

2) Docker Containers

When we run the Image, We get container.

3) Docker Host Machine on which docker is installed, is called as Docker host.

4) Docker Client Terminal used to run docker run commands ( Git bash )

On linux machine, git bash will work like docker client.

+++++++++++

Docker Commands

--------------------

Working on Images

-------------------------

1 To download a docker image docker pull image\_name

2 To see the list of docker images docker image ls

(or)

docker images

3 To delete a docker image from docker host docker rmi image\_name/image\_id

4) To upload a docker image into docker hub

docker push image\_name

5) To tag an image docker tag image\_name

ipaddress\_of\_local\_registry:5000/image\_name

6) To build an image from a customised container

docker commit container\_name/container\_id new\_image\_name

7) To create an image from docker file

docker build -t new\_image\_name

8) To search for a docker image docker search image\_name

9) To delete all images that are not attached to containers

docker system prune -a

++++++++++++++++++++++++++++++++++++++++++++++

Working on containers

-----------------------------

10) To see the list of all running continers

docker container ls

11) To see the list of running and stopped containers

docker ps -a

12) To start a container

docker start container\_name/container\_id

13) To stop a running container

docker stop container\_name/container\_id

14) To restart a running container

docker restart container\_name/container\_id

To restart after 10 seconds

docker restart -t 10 container\_name/container\_id

15) To delete a stopped container

docker rm container\_name/container\_id

16) To delete a running container

docker rm -f container\_name/container id

17) To stop all running containers

docker stop $(docker ps -aq)

18) To restart all containers

docker restart $(docker ps -aq)

19) To remove all stopped containers

docker rm $(docker ps -aq)

20) To remove all contianers(running and stopped)

docker rm -f $(docker ps -aq)

21) To see the logs generated by a container

docker logs container\_name/container\_id

22) To see the ports used by a container

docker port container\_name/container\_id

23) To get detailed info about a container

docker inspect container\_name/container\_id

24) To go into the shell of a running contianer which is moved into background

docker attach container\_name/container id

25) To execute anycommand in a

container docker exec -it container\_name/container\_id command

Eg: To launch the bash shell in a contianer

docker exec -it container\_name/container\_id bash

26) To create a container from a

docker image ( imp )

docker run image\_name

++++++++++++++++++++++++++++++++++++

++++++++++++

Run command options

-it for opening an interactive terminal in a container

--name Used for giving a name to a container

-d Used for running the container in detached mode as a background process

-e Used for passing environment varaibles to the container

-p Used for port mapping between port of container with the dockerhost port.

-P Used for automatic port mapping ie, it will map the internal port of the container with some port on host machine. This host port will be some number greater than 30000

-v Used for attaching a volume to the container

--volume-from Used for sharing volume between containers

--network Used to run the contianer on a specific network

--link Used for linking the container for creating a multi container architecture

--memory Used to specify the maximum amount of ram that the container can use

++++++++++++++++++++++++++++++++++++

++++++++++++++=

# docker images ( There are no images )

To download tomcat image

# docker pull tomee

# docker images

# docker pull ubuntu

If you do not specify the version, by default, we get latest version

I want to download jenkins

# docker pull jenkin

## Q. How to create the image of the container?

First the change the tag:

docker commit 2cf1b8f19566 mypython:v1

docker tag mypython:v1 jayant700/mypython:v1

Make sure that tag name is the same which are use for login the docker hub account

## Q. How to push the image in the docker hub?

docker push jayant700/mypython:v1

## Q What is Docker Volume?

- Docker containers are ephemeral ( temporary ) Where as the data processed by the container should be permanent. Generally, when a container is deleted all its data will be lost.

To preserve the data, even after deleting the container, we use volumes.

Volumes are of two types

**1) Simple docker volumes :** These volumes are used only when we want to access the data, even after the container is deleted. **But this data cannot be shared with other containers.**

**2) Docker volume containers (Sharable volume ):** These are also known as reusable volume. The volume used by one container can be shared with other containers. **Even if all the containers are deleted, data will still be available on the docker host.**

## Q What is Driver in Docker Volume?

* + In **Docker**, a **driver** is a software component or plugin that defines **how Docker interacts with various systems** — such as storage, networking, or logging.

QWhat is the process to create the volume, mount the container, delete the container and restore with new container ?

-First we need to create the volume:

[ec2-user@ip-172-31-11-22 ~]$ docker volume ls

DRIVER VOLUME NAME

[ec2-user@ip-172-31-11-22 ~]$ docker volume create datavolume

datavolume

[ec2-user@ip-172-31-11-22 ~]$ docker volume ls

DRIVER VOLUME NAME

local datavolume

-Then mount the location with the container

[ec2-user@ip-172-31-11-22 ~]$ docker container run --name linux01 -dit -v datavolume:/data alpine

[ec2-user@ip-172-31-11-22 ~]$ docker container ls -a

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

ae62c02583b4 alpine "/bin/sh" 10 seconds ago Up 9 seconds linux01

-Then check the container was mounted or not by using of inspect

docker container inspect linux01

"HostConfig": {

"Binds": [

"datavolume:/data"

],

-Once check create the two new file in the mount location

docker container exec -it linux01 /bin/sh

du –f

0 ./data

/ # cd /data/

/data # echo "this is second file" > Test02

/data # echo "this is the first file" > Test01

-Once file created then delete the specific container

docker container stop linux01

docker container rm linux01

-Then create the new conatiner and restore the previous mount location data

docker container run --name restored -dit -v datavolume:/app alpine

-Once mounted the data you will see that the data which was present in the preivous conatiner are present on the new container.

[ec2-user@ip-172-31-11-22 ~]$ docker container exec -it restored ls -ltr /app

total 8

-rw-r--r-- 1 root root 20 May 2 13:53 Test02

-rw-r--r-- 1 root root 23 May 2 13:53 Test01

[ec2-user@ip-172-31-11-22 ~]$

Q. What is step to processing the one package from the git to docker by using of the Jenkins?

- First we need to create the “Dockerfile” in the main place were we have other file place:



-Once created now install the Docker on the jenkin serever were you have request to process this docker file.

Inside the docker file content is :

# Build stage (optional if already using Maven separately)

FROM openjdk:17-alpine

WORKDIR /app

COPY target/\*.jar app.jar

CMD ["java", "-jar", "app.jar"]

-Once install the given the super user to docker as well by using of the below command :

sudo usermod -a -G docker ec2-user

-After that create the free style project by using of the jenkin

-Source Code Management >> provide git credanital

-Build Steps>>

Invoke top-level Maven targets >> Goals>> clean package

-Execute shell>>echo "Building Docker image..."

docker build -t jayant700/maven-docker-app .

echo "Running Docker container..."

docker container run --name project01 -dit jayant700/maven-docker-app

Login to the docker account ....

+ docker login -u jayant700 -p Jayant@123

Pushing image in Docker hub....

+ docker push jayant700/maven-docker-app:latest

Q. What is Docker networking?

--Docker networking refers to the way Docker containers communicate with each other and the outside world. By default, Docker provides several networking modes to help containers interact with each other and external systems in different ways. Each network type serves different use cases, and understanding these options is essential for managing how containers can connect, exchange data, and isolate themselves.

### Types of Docker Networks:

1. **Bridge Network (default network type)**:
   * **Use Case**: This is the default network type when you don't specify a network for your containers.
   * **How it works**: The containers on a bridge network can communicate with each other, but they cannot directly communicate with containers outside the host machine unless explicitly set up (e.g., using port forwarding).
   * **Container-to-Container Communication**: Containers in the same bridge network can communicate with each other using their container names or IP addresses.

Example command to run a container on the default bridge network:

docker run -d --name container1 my\_image

### Types of Docker Networks:

1. **Bridge Network (default network type)**:
   * **Use Case**: This is the default network type when you don't specify a network for your containers.
   * **How it works**: The containers on a bridge network can communicate with each other, but they cannot directly communicate with containers outside the host machine unless explicitly set up (e.g., using port forwarding).
   * **Container-to-Container Communication**: Containers in the same bridge network can communicate with each other using their container names or IP addresses.

Example command to run a container on the default bridge network:

docker network create --driver bridge web-network

docker container run -dit --name projectnetwork --network web-network alpine

1. **Host Network**:
   * **Use Case**: The host network is useful when you want a container to use the host machine’s network stack.
   * **How it works**: When a container is attached to the host network, it shares the same IP address as the Docker host. This means it can communicate with the outside world using the host's IP address, and it doesn’t need port mapping.
   * **Container-to-Container Communication**: Containers on the host network cannot communicate with other containers on different networks because they share the host’s network interface.
   * **Best for**: High-performance network communication, as containers can directly access the host's networking without any isolation.

Example command to run a container on the host network:

docker container run -dit --name hostnetwork --network host httpd

1. **None Network**:
   * **Use Case**: This network mode disables networking for a container entirely.
   * **How it works**: The container will have no network interface. This mode is rarely used, but it can be useful for running containers that don’t need networking (e.g., for testing purposes).

Example command to run a container with no networking:

docker container run -it --name nullnetwork02 --network none alpine:latest

1. **Overlay Network**:
   * **Use Case**: This is used in multi-host Docker environments (like Docker Swarm or Docker Compose).
   * **How it works**: Overlay networks allow containers running on different Docker hosts (machines) to communicate securely as if they were on the same host. Docker automatically sets up routing between the hosts.
   * **Best for**: Scalable applications where services need to be distributed across multiple machines.
   * **Security**: The overlay network encrypts traffic between containers in different hosts, providing secure communication.

Example command to create an overlay network:

docker network create --driver overlay my\_overlay\_network

1. **Macvlan Network**:
   * **Use Case**: The Macvlan network driver allows you to assign a MAC address to a container, making it appear as a physical device on the network.
   * **How it works**: This allows containers to have their own IP addresses on the local network, and they can communicate with other devices on the network as if they were physical machines.
   * **Best for**: Situations where you need containers to be directly accessible on the network and behave like separate physical devices.
   * **Note**: Macvlan networks are often used in scenarios where containers need to be on a separate physical network or when bridging physical hardware with virtual containers.

Example command to create a Macvlan network:

docker network create -d macvlan --subnet=192.168.1.0/24 --gateway=192.168.1.1 -o parent=eth0 my\_macvlan\_network

1. **Host-Only Network (for Docker Desktop on Windows/Mac)**:
   * **Use Case**: This network mode creates a bridge between the host and the container, allowing containers to communicate with the host while isolating them from other containers.
   * **Best for**: Local development environments where you want to allow the container to interact with the host but not the external network.

### Key Networking Commands in Docker:

1. **Create a network**:

docker network create my\_network

1. **List networks**:

docker network ls

1. **Inspect a network** (to view network details):

docker network inspect my\_network

1. **Connect a container to a network**:

docker network connect my\_network container\_name

1. **Disconnect a container from a network**:

docker network disconnect my\_network container\_name

### Docker Container Communication

* **Same Network**: Containers within the same network can communicate with each other using container names or IP addresses.
* **Different Networks**: Containers in different networks can't communicate unless explicitly linked or using Docker's network connection commands.

# Kubernates;

Q.1 What is kubernates?

-Kubernetes is a platform to orchestrate the deployment, scaling, and management of container-based applications.The correct pronunciation of Kubernetes is Koo-ber-netties or Koo-ber-nay-tace.

Kubernetes was a Google project, but joined the Cloud Native Computing Foundation (CNCF) and is now the de facto standard in the space of container-based applications.The core functionality is scheduling workloads in containers across your infrastructure.

Some of the other kubernetes capabilites are as follows:

* Providing authentication and authorization
* Debugging applications
* Accessing and ingesting logs
* Rolling updates
* Using Cluster Autoscaling
* Using the Horizontal Pod Autoscaler
* Replicating application instances
* Checking application health and readiness
* Monitoring resources
* Balancing loads
* Naming and service discovery
* Distributing secrets
* Mounting storage systems

Q. Kubernetes Distributions

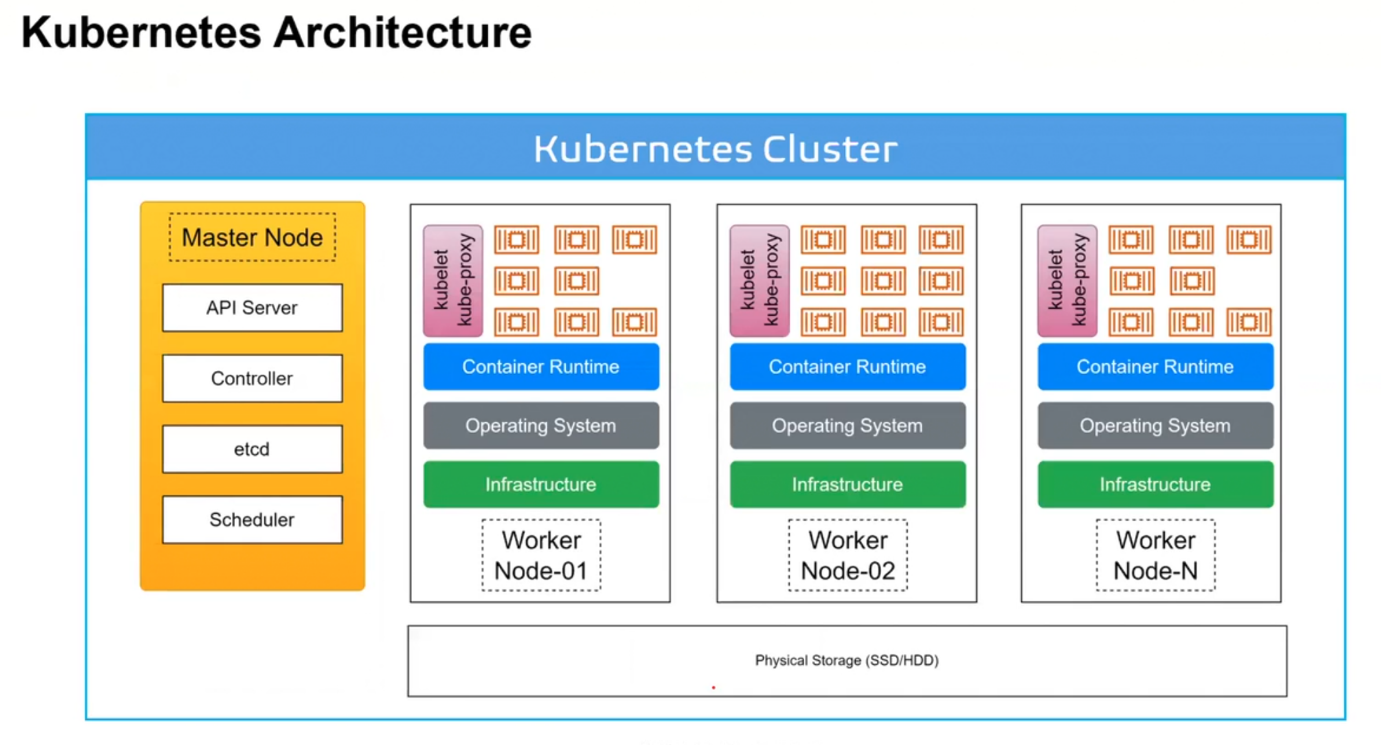
Some of the most popular Kubernetes distributions are as follows:

* minikube (for learning)**Note: Not Mention in interview**
* kubeadm
* k3s (IoT & Edge computing)
* k3d (Rancher Lab's minimal distro)
* Rancher Kubernetes (Open source)
* Docker Kubernetes Service (Open source)
* Amazon Elastic Kubernetes Service (Cloud managed)
* Azure Kubernetes Service (Cloud managed)
* Google Kubernetes Engine (Cloud managed)
* RedHat Openshift (Enterprise)
* Tanzu Kubernetes Grid (Enterprise)
* Mirantis (Enterprise)
* Elastisys Compliant Kubernetes (Enterprise)

Components of Kubernetes distribution

* Container Runtime
* Storage
* Networking

Q. Kubernates Architecture:

- 

**-Master Node :** Play important role in Cluster, its manages all the nodes in the cluster.

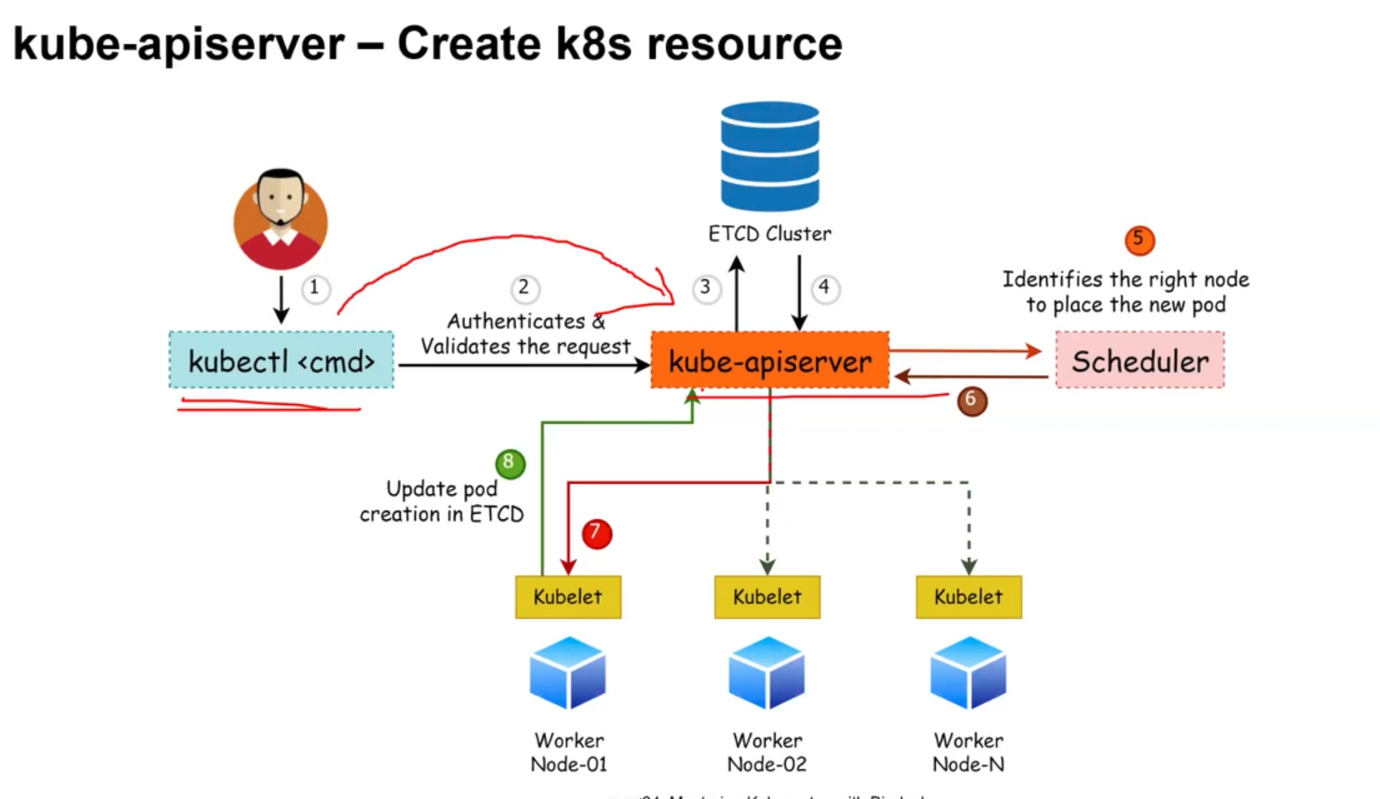
**-API Server:** To process any things in inside the nodes we are using API, like create , delete, passing arguments. Its take input from the users.

**-etcd**: It’s the data storage of the kubernates, if you use of the API the details will be fetched from the etcd.

**-Scheduler:** Its decide were the pods/container was deployed, in the worker node, its check were is space is available and which location cpablity to process pods.

**-kubelet** : **Kubelet** is a **core component** of the Kubernetes architecture. It is an **agent** that runs on **each node** in a Kubernetes cluster.

**-Controller:** Controller make the desired state of the kubernates cluster, which means its create the in any pods was deleted or failed.

****

Q. What is Pod?

* The smallest deployable unit in Kubernetes is a pod.
* The pod contains the actual application workload.
* It can be one or multiple containers.
* A pod in Kubernetes has a defined lifecycle.

Q. How to install the Kubernates?

-link : <https://github.com/kbindesh/kubernetes-masterclass/tree/main/Module-03_Setting_up_K8s_Cluster/01-minikube>

Q.What is object of the Kubernates?

-The fundamental building blocks of any containerized workload that is up and running in the Kubernetes cluster are called Kubernetes API primitives or Kubernetes objects.They are the API resource types defined in Kubernetes, including:

* pods
* ReplicaSets
* DaemonSets
* StatefulSets
* Job
* CronJob objects
* Deployments
* Namespaces
* Service
* ConfigMaps
* Volumes

Imperative vs Declarative approaches for K8s object management

There are a few ways to communicate with API servers in Kubernetes.

* Imperative management
  + Imperative commands
  + Imperative object configuration
* Declarative management

01. Imperative Commands

* When using imperative commands, a user operates directly on live objects in a cluster.
* The user provides operations to the kubectl command as arguments or flags.
* This is the recommended way to get started or to run a one-off task in a cluster.
* Because this technique operates directly on live objects, it provides no history of previous configurations.
* # Run an instance of nginx container by creating a Deployment object
* kubectl create deployment nginx --image nginx

or kubectl run tomcat-web --image httpd:latest

02. Imperative object configuration

In imperative object configuration, the kubectl command specifies the operation (create, replace, etc.), optional flags and at least one file name.

The file specified must contain a full definition of the object in YAML or JSON format.

# Create the objects defined in a .yaml configuration file

kubectl create -f nginx.yaml

# Delete the objects defined in two configuration files

kubectl delete -f nginx.yaml -f redis.yaml

# Update the objects defined in a configuration file

kubectl replace -f nginx.yaml

03. Declarative object configuration

* In declarative configuration, you define the object definition in YAML config files called a K8s manifest.
* When K8s manifests are applied to a kubernetes cluster, kubernetes creates an object based on the configuration.
* When using declarative object configuration, a user operates on object configuration files stored locally, however the user does not define the operations to be taken on the files.
* Create, update, and delete operations are automatically detected per-object by kubectl.
* This enables working on directories, where different operations might be needed for different objects.
* Example: Process all object configuration files in the configs directory

# diff to see what changes are going to be made, and then apply

kubectl diff -f configs/

kubectl apply -f configs/

* Recursively process directories

kubectl diff -R -f configs/

kubectl apply -R -f configs/

Q. What are Kubernetes Manifest Files?

* Kubernetes manifest files are YAML or JSON files that describe objects in your cluster.
* They're the primary way to manage your objects as they let you version configurations alongside your code, then declaratively apply them to your cluster.
* For example, the following basic manifest (pod.yml) describes a Pod called nginx that runs a container using the nginx:latest image:

apiVersion: V1 {What is version of the pod}

kind: Pod {What type obeject is it, is that volume or pod or network}

metadata: {What is the name of pod, or other specification given to pod}

name: webapp-pod { wirte now given the name }

spec: {what is the specfication given the Pod]

conatiner: {Now is the specficify the container may be container one ortwo}

- name: tomcat {name of the container}

image: httpd:latest { image of the container}

ports: {assign the port}

- conatinerPort: 80 {Given Port}

You can use kubectl to conviniently create the Pod from its manifest in any Kubernetes cluster, without having to remember the syntax of imperative commands like kubectl run.

# Using an imperative command

$ kubectl run nginx --image nginx:latest

# Declaratively applying a manifest file

$ kubectl apply -f pod.yaml

Manifests are the files that describe what k8s object will be deployed in a Kubernetes cluster.

Your actual deployments are created by applying your manifest files to your cluster, typically using kubectl or a package manager like Helm.

Why use kubernetes manifests?

* Declarative configuration
* Predictable outcomes
* Versioning k8s objects
* Collaborating on K8s objects
* Facilitate automations (CI/CD Pipelines)

Common Kubernetes manifest fields

All the kubernetes manifests have a few required fields and some object specific fields.

* apiVersion
* kind
* metadata
* spec

Q. What is tool use to check the pods?

-To get the pod details you can use:

kubectl get pod

NAME READY STATUS RESTARTS AGE

tomcat-web 1/1 Running 0 10s

- to get the simple view with all details:

kubectl get pod -o yaml

or

kubectl get pod -o Jason

or

kubectl get pod -o wide

NAME READY STATUS RESTARTS AGE IP NODE NOMINATED NODE READINESS GATES

tomcat-web 1/1 Running 0 50s 10.244.0.4 minikube <none> <none>

Q. How to define the pod ?

-To define the Pod we are using the describe command :

kubectl describe pod tomcat-web

Q. Write the manifest file creates the tomcat page?

* First write code outside the linux machine
* Then create the folder indide the instance
* Third make yml page by using of Vim or vi
* And save it afte that run the below command “kubectl apply -f pod.yaml”.

Q .What is replica and what is process of creating the replicaset?

* ReplicaSets help pods achieve higher availability since users can define a certain number of replicas using a ReplicaSet.
* The main capability of a ReplicaSet is to make sure the cluster keeps the exact number of replicas running in the Kubernetes cluster.
* If any of them were to fail, new ones would be deployed.

apiVersion: apps/v1

kind: ReplicaSet

metadata:

name: frontend

labels: //Label its help us to unique things in the all label

app: melonapp-rs

spec:

replicas: 3 //its define how many replica you required

selector: //Its bind with other bonds in replica

matchLabels:

app: melonapp-rs

template:

metadata:

labels:

app: melonapp-rs

spec:

containers:

- name: nginx

image: nginx

**Practical:**

apiVersion: apps/v1

kind: ReplicaSet

metadata:

name: tomcat-web

labels:

apptier: tomcat-frontend

spec:

replicas: 3

selector:

matchLabels:

apptier: tomcat-frontend

template:

metadata:

name: tomcat-web

labels:

apptier: tomcat-frontend

spec:

containers:

- name: tomcat-web

image: httpd:latest

ports:

- containerPort: 80

Q. How to get replicaset list?

- kubectl get replicaset

OR

kubectl get rs

# You will see the output indicating the number of DESIRED replica counts and the ones are in a READY state.

Q. How to Scale the application by revising the replica count?

-

- # Syntax

kubectl scale replicaset <replicaset\_name> --replicas=<revised\_count>

kubectl scale replicaset frontend --replicas=5

# You will get the response as "replicaset.apps/frontend scaled"

Q. How to Delete a ReplicaSet?

- # Syntax

kubectl delete replicaset <replicaset\_name>

# Example

kubectl delete replicaset frontend

Q. What is Deploying and managing applications?

- A Deployment is a kubernetes object which provides us a convenient way to define the desired state application deployment.

It provides us with a better way of upgrading the underlying instances seamlessly using:

* rolling updates
* undoing changes, and
* pausing and resuming changes as required

Deployments provide a way to define a desired state for the replica pod.

apiVersion: apps/v1

kind: Deployment //If you are change the version use depolyment

metadata:

name: nginx-deployment

labels:

app: nginx

spec:

replicas: 3

selector:

matchLabels:

app: nginx

template:

metadata:

labels:

app: nginx

spec:

containers:

- name: nginx

image: nginx:latest

ports:

- containerPort: 80

The following attributes are important to help you understand the preceding YAML definition:

spec.replicas gives us the number of replica pods

spec.template is the template pod descriptor that defines the pods that will be created

spec.selector is the deployment that will manage all pods whose labels match this selector

Q. What is Rollback deployment?

-Rollback allows us to revert our application to a previous state (deployment) seamlessly.

# To quickly recover if you need to perform a rollback

kubectl rollout undo deployments <deployment\_name>

kubectl rollout undo deployments app-deployment

# To verify the rolled back image (prev)

kubectl describe deployment <deployment\_name>

Q. How to check the history of deployment?

- ow History was used to provide the details how many time you have undo or roll back the deployment.

Kubectl rollout history deployment nigix-frontend

Kubernetes Networking Model: Kubernetes networking resolves the challenge of how to allow different Kubernetes components to communicate with each other. It also allows applications on Kubernetes to communicate with other applications, as well as the services outside of the Kubernetes cluster.

* **Container-to-Container communication**
* **Pod-to-Pod communication**
* **Pod-to-Service communication**
* **External-to-Service communication**
* **Node-to-Node communication**

Pod-to-Service communication

The service accepts traffic from both inside and outside of the cluster.

The effective communication between Pods and Services involves letting the service expose an application running on a set of pods.

**Kubernetes Service Types**

**ClusterIP**

A default service type for Kubernetes.

For internal communications, exposing the service makes it reachable within the cluster.

apiVersion: v1

kind: Service

metadata:

name: bin-cip-service //

spec:

type: ClusterIP //

selector:

app: nginx //Make sure provide the label

ports:

- protocol: TCP

port: 8080

targetPort: 80

[ec2-user@ip-172-31-6-71 K8S-Manifest]$ kubectl apply -f tomcat-culsterip-svc.yml

service/tomcat-clusterip-svc created

[ec2-user@ip-172-31-6-71 K8S-Manifest]$ kubectl get service

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE

kubernetes ClusterIP 10.96.0.1 <none> 443/TCP 2d10h

tomcat-clusterip-svc ClusterIP 10.104.33.108 <none> 8080/TCP 14s

[ec2-user@ip-172-31-6-71 K8S-Manifest]$ kubectl describe service tomcat-clusterip-svc

Name: tomcat-clusterip-svc

Namespace: default

Labels: <none>

Annotations: <none>

Selector: apptier=tomcat-frontend

Type: ClusterIP

IP Family Policy: SingleStack

IP Families: IPv4

IP: 10.104.33.108

IPs: 10.104.33.108

Port: <unset> 8080/TCP

TargetPort: 80/TCP

Endpoints: 10.244.0.27:80,10.244.0.29:80,10.244.0.28:80

Session Affinity: None

Internal Traffic Policy: Cluster

Events: <none>

[ec2-user@ip-172-31-6-71 K8S-Manifest]$

[ec2-user@ip-172-31-6-71 K8S-Manifest]$ kubectl apply -f tomcat-nodeport-svc.yml

**NodePort**

For both internal and external communication.

NodePort exposes the service on a static port on each worker node.

Example -

# For external communication

<node\_ip\_address>:<node\_port>

apiVersion: v1

kind: Service

metadata:

  name: tomcat-nodeport-svc

spec:

  type: NodePort

  selector:

    apptier: tomcat-frontend

  ports:

    - port: 8080

      nodePort: 30052

      targetPort: 80

[ec2-user@ip-172-31-6-71 K8S-Manifest]$ kubectl apply -f tomcat-nodeport-svc.yml

service/tomcat-nodeport-svc created

[ec2-user@ip-172-31-6-71 K8S-Manifest]$ kubectl get service

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE

kubernetes ClusterIP 10.96.0.1 <none> 443/TCP 2d10h

tomcat-clusterip-svc ClusterIP 10.104.33.108 <none> 8080/TCP 6m22s

tomcat-nodeport-svc NodePort 10.102.5.17 <none> 8080:30052/TCP 32s

[ec2-user@ip-172-31-6-71 K8S-Manifest]$ kubectl describe service tomcat-nodeport-svc

Name: tomcat-nodeport-svc

Namespace: default

Labels: <none>

Annotations: <none>

Selector: apptier=tomcat-frontend

Type: NodePort

IP Family Policy: SingleStack

IP Families: IPv4

IP: 10.102.5.17

IPs: 10.102.5.17

Port: <unset> 8080/TCP

TargetPort: 80/TCP

NodePort: <unset> 30052/TCP

Endpoints: 10.244.0.29:80,10.244.0.27:80,10.244.0.28:80

Session Affinity: None

External Traffic Policy: Cluster

Internal Traffic Policy: Cluster

Events: <none>

[ec2-user@ip-172-31-6-71 K8S-Manifest]$

[ec2-user@ip-172-31-6-71 K8S-Manifest]$ kubectl get node

NAME STATUS ROLES AGE VERSION

minikube Ready control-plane 2d10h v1.32.0

[ec2-user@ip-172-31-6-71 K8S-Manifest]$ minikube ip

192.168.49.2

[ec2-user@ip-172-31-6-71 K8S-Manifest]$ curl 192.168.49.2:30052

<html><body><h1>It works!</h1></body></html>

[ec2-user@ip-172-31-6-71 K8S-Manifest]$

**LoadBalancer**

Works with cloud providers, as it is backed by thier respective load balancer service offerings. Ex. AWS ALB

Underneath LoadBalancer, ClusterIP and NodePort are created, which are used for internal and external communication.

**ExternalName**

Maps the service to the contents with a CNAME record with its value.

It allows external traffic access through it.

[ec2-user@ip-172-31-6-71 K8S-Manifest]$ kubectl apply -f tomcat-culsterip-svc.yml

service/tomcat-clusterip-svc created

[ec2-user@ip-172-31-6-71 K8S-Manifest]$ kubectl get service

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE

kubernetes ClusterIP 10.96.0.1 <none> 443/TCP 2d10h

tomcat-clusterip-svc ClusterIP 10.104.33.108 <none> 8080/TCP 14s

[ec2-user@ip-172-31-6-71 K8S-Manifest]$ kubectl describe service tomcat-clusterip-svc

Name: tomcat-clusterip-svc

Namespace: default

Labels: <none>

Annotations: <none>

Selector: apptier=tomcat-frontend

Type: ClusterIP

IP Family Policy: SingleStack

IP Families: IPv4

IP: 10.104.33.108

IPs: 10.104.33.108

Port: <unset> 8080/TCP

TargetPort: 80/TCP

Endpoints: 10.244.0.27:80,10.244.0.29:80,10.244.0.28:80

Session Affinity: None

Internal Traffic Policy: Cluster

Events: <none>

[ec2-user@ip-172-31-6-71 K8S-Manifest]$

### 📌 **1. In Kubernetes (K8s):**

**What is NameSpace?**

-A **namespace** is used to **divide cluster resources** between multiple users or applications.

* It allows you to have **multiple environments** (like dev, test, prod) within the same cluster.
* Each namespace has its **own set of resources** like pods, services, and deployments.
* Default namespaces:
  + default
  + kube-system
  + kube-public

**Example:**

bash

kubectl get pods --namespace=dev

**What is HPA?**

-To update a workload resource such as a Deployment, you can use HPA.

HPA is a Kubernetes API primitive that scales the workloads automatically based on enduser's demand.

HPA is configured to fetch metrics provided by a metrics server based on the CPU and memory usage.

HPA scales the Deployment by increasing or decreasing the count of replicas, which is managed by an underlying ReplicaSet.

## imperative

kubectl autoscale deployment webapp-deployment --cpu-percent=50 --min=3 --max=10

## Declarative

apiVersion: autoscaling/v2

kind: HorizontalPodAutoscaler

metadata:

name: bin-hpa

spec:

scaleTargetRef:

apiVersion: apps/v1

kind: Deployment

name: bin-hpa

minReplicas: 3

maxReplicas: 10

metrics:

- type: Resource

resource:

name: cpu

target:

type: Utilization

averageUtilization: 50

Q What is DaemonSets?

-DaemonSets create a multiple copies of a pod, making sure that at least one copy of the pod is present on each node of your Kubernetes cluster.

If a new node is added to the cluster, a replica of that pod is automatically assigned to that node.

Likewise, when a node is removed, the pod is automatically removed.

Alias: ds

kubectl get daemonsets //For checking the nodes

apiVersion: apps/v1

kind: DaemonSet

metadata:

name: fluentd

namespace: kube-system

labels:

k8s-app: fluentd

spec:

selector:

matchLabels:

name: fluentd

template:

metadata:

labels:

name: fluentd

spec:

containers:

- name: fluentd

image: fluentd:latest

kubectl get daemonsets -n <namespace>

kubectl describe daemonsets <daemonset\_name> -n <namespace>

### **✅ What is Jenkins Pipeline?**

**Jenkins Pipeline** is a suite of plugins that supports **automation of continuous integration and continuous delivery (CI/CD)**. It allows you to define the entire build process (like compiling code, running tests, deploying) as **code**, usually in a file named Jenkinsfile.

Pipelines provide better visibility, control, and repeatability compared to freestyle jobs.

### 🔄 Types of Jenkins Pipelines

There are **two main types**:

#### 1. **Declarative Pipeline**

* Introduced to simplify pipeline creation.
* Uses a structured and predefined syntax.
* Easier to read and maintain.
* Defined using the pipeline {} block.

pipeline{

agent any

Jenkin tools which are added in the mange jenkin that’s call though the tools agent.

tools{

maven "Maven-3.9.8"

jdk 'Java-17'

}

environment{

Its help us to define that one enviromet where we have call though the name which is define.

BUILD\_FILE = 'Appserverfile'

ROLE = "Developer"

NAME = "Jayant"

}

stages{

stage('Invoke the files'){

In the stage we have define all the action which need to perform or execute as well we are called the env var as well though the stage.

steps{

sh 'echo $NAME have $ROLE for $BUILD\_FILE'

}

}

stage('Fetching the Details'){

steps{

echo 'fetching'

}

}

stage("Compilein gthe jobe"){

steps{

echo "Compling"

}

}

stage("Testing the Job"){

steps{

echo "Testing"

}

}

}

}

**Terraform**

**Q. What is Terraform?**

-Terraform is an infrastructure as code tool that lets you build, change, and version cloud and on-prem resources safely and efficiently.

**Q. How does Terraform work?**

-Terraform creates and manages resources on cloud platforms and other services through their application programming interfaces (APIs). Providers enable Terraform to work with virtually any platform or service with an accessible API.

**Ansible**

**Q. What is Ansible?**

* -**Ansible** is an **open-source automation tool** used for:
* **Configuration Management**
* **Application Deployment**
* **Task Automation**
* **IT Orchestration**

🔧 How it Works:

Ansible uses **YAML-based playbooks** to define automation tasks and connects to remote systems **over SSH** (no agent needed). It pushes configurations and executes commands in a simple, human-readable way.

Link: https://docs.ansible.com/

* **Agentless:** No software needs to be installed on target machines.
* **Simple Syntax:** Uses YAML for playbooks (easy to write and read).
* **Idempotent:** Ensures the same result every time it's run.
* **Scalable:** Can manage a few or thousands of servers.
* **Modular:** Uses modules to handle tasks like installing packages, copying files, restarting services, etc.