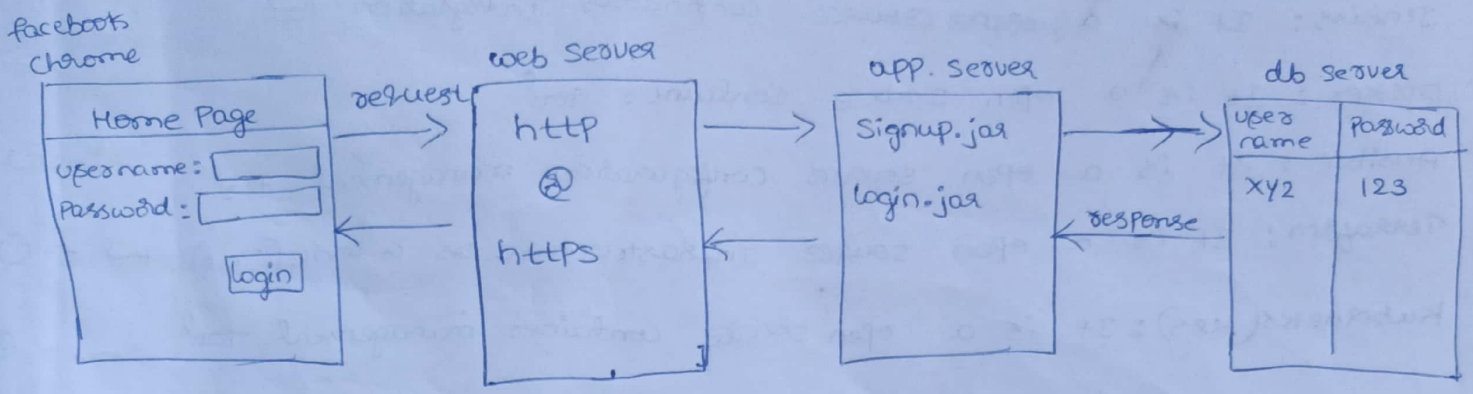
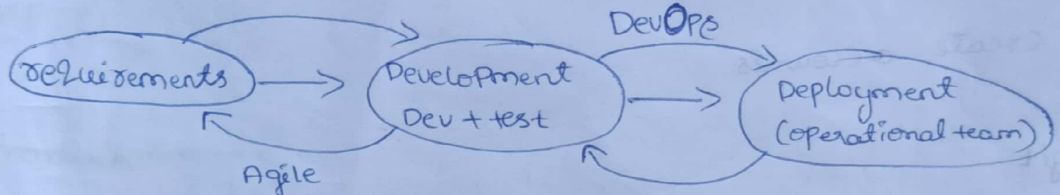
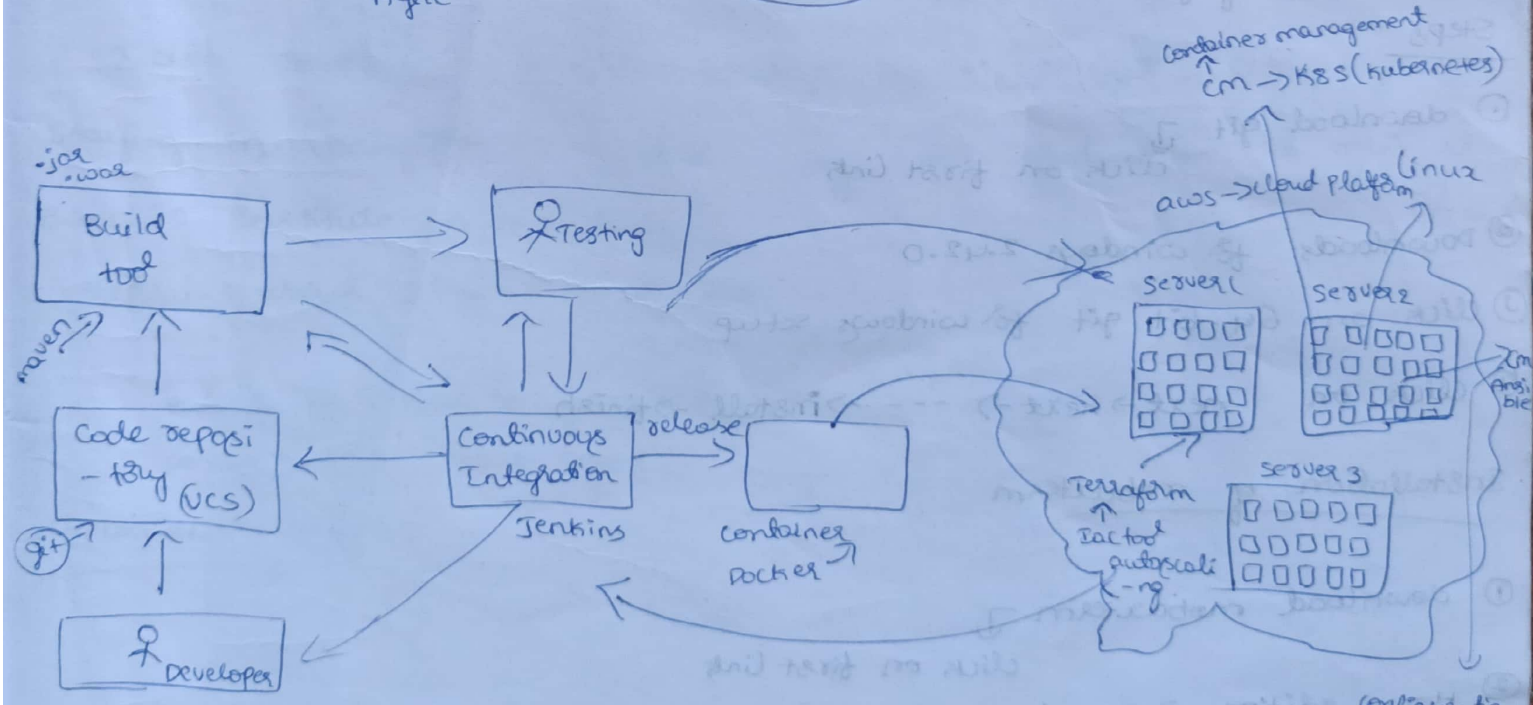
Three tier Architecture



Introduction





VCS → Version Control System

IaC → Infrastructure as a Code

git : It is a open source version control tool

maven: It is a open source Build management tool

Jenkins: It is a open source continuous Integration tool

Docker: It is a open source container tool

Ansible: It is a open source configuration management tool

Terraform: It is a open source Infrastructure as a code (IaC) tool

Kubernetes(K8S): It is a open source container management tool.

Software Requirements:

① git (global Information tracker) } download & install
② mobaxterm

③ dockerhub } Create accounts
④ aws

Installation of git

Steps

① download git ↓ click on first link
② Downloads for windows 2.42.0
③ click on 64-bit git for windows setup
④ click on next → next → --- → install → finish

Installation of mobaxterm

① download mobaxterm ↓ click on first link
② Home edition → Download now
③ click on Portable Edition

create account in github

① type github -> first link

② signup

create account in Docker

-> Docker hub

-> signup

create account on aws

① amazon management console ↓ first link

② create an account

③ mail id
user name
verify email address

Pan
select no

type -> individual

④ root password

⑤ step (1 of 5)

Introduction

I am Sowmya, I am belongs to Sira but currently

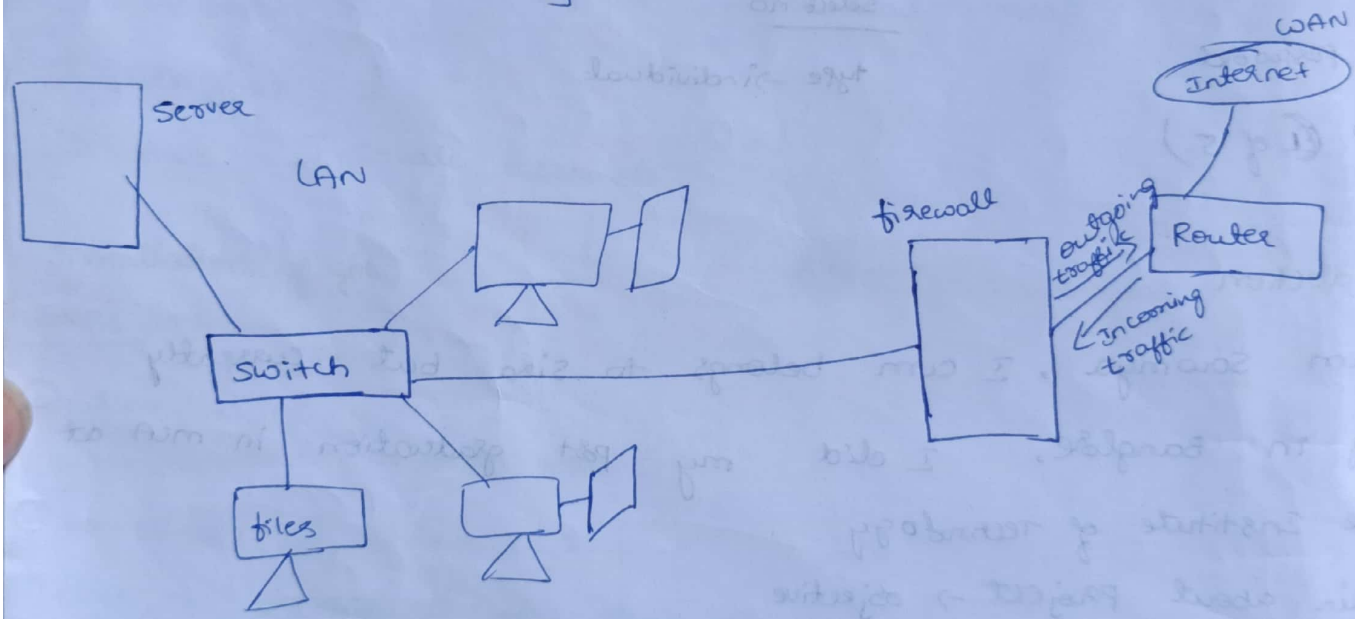
post graduation in MCA at

Computer Networking

Computer Networking is a process of connecting billions of computing devices with the network and it will allow to share resources and files to each other.

Types of network

① PAN [Personal Area Network]
② LAN [Local Area Network]
③ MAN [Metropolitan Area network]
④ WAN [Wide Area Network]



Switch :- Switch is a hardware device which is used to connect multiple computing devices within the network

Firewall :- Firewall is a network security device that monitors and filters incoming and outgoing traffic in the network

Router :- Router is a gateway that passes/process the data from one network to another network.

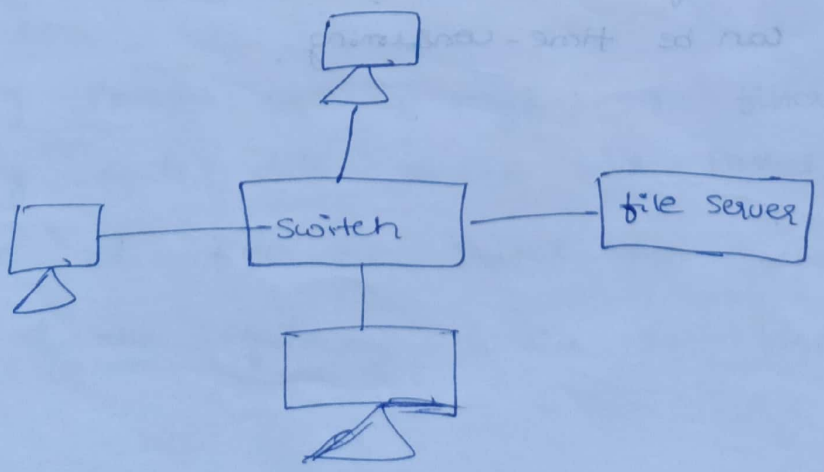
Gateway :- Gateway is a hardware device which is used to connect two similar (&) dissimilar networks.

Assignments

1. what is LAN? Explain breifly
2. what is WAN? explain breifly.
3. Advantages & disadvantages of gateway.

(1) LAN

* A local Area Network is a computer network that connects devices within a small geographic area, such as building, school, office
* It acts as a communication infrastructure, allowing devices to share resources and exchange information seamlessly.
* we can establish LAN n/w in 2 ways either wired & wireless, In wired LAN we use the ethernet cable to connect the devices, the wireless LAN is the wi fi
* The LAN network size is limited to small geographical area
* The Data transfer rate is high 100mbps to 1000 mbps
* LAN uses only one type of transmission medium i.e., coaxial cables.
* The number of computers connected to a LAN is usually restricted.
* IEEE 802.3 (a) Ethernet is the most common LAN, which uses twisted pair cable (a) fibre optic cable.



wireless LANS

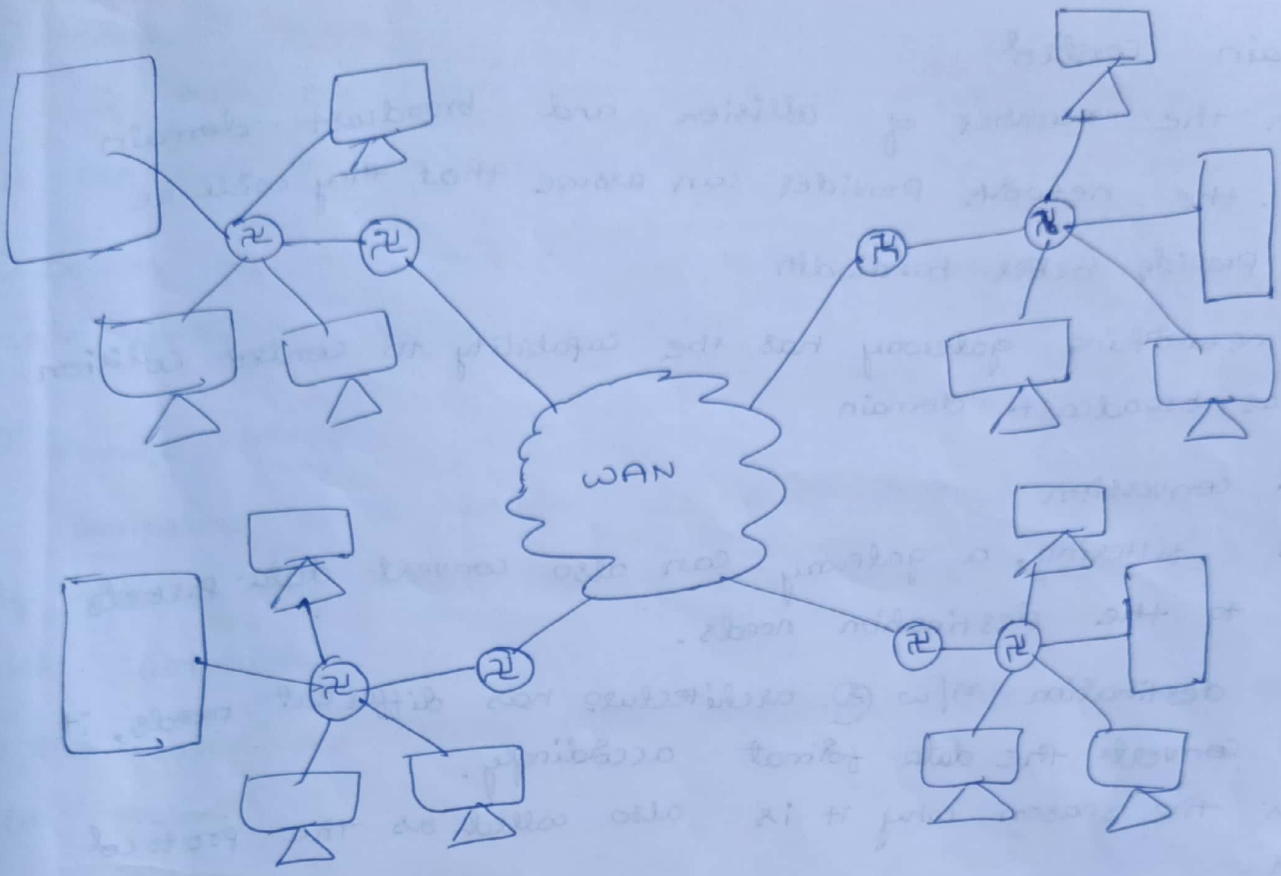
wireless LAN's use high-frequency radio waves instead of cables for communication

(2) what is WAN? explain breifly

* A wide Area Network is a computer network that is used to connect devices over a large geographic area comprising a region, a country a continent (or) even the whole world.
* WAN's uses various technologies & infrastructure they are fibre optics, satellites & the internet to establish the connection over long distances
* The main Purpose of WAN's are to connect the devices in a wide range
* TO Provide security WAN uses various technologies like Encryption, virtual private networks (VPNs) & other security measures
* Building and maintaining a WAN is expensive due to the infrastructure and services involved

Disadvantages

* The cost is high
* Difficult to maintain the networks
* It faces the more security issues than LAN
* Troubleshooting WAN can be time-consuming.



Advantages and Disadvantages of gateway.

Gateway is a device which is used to connect similar & dissimilar networks

Advantages

① Connectivity: gateways enable the connection of networks for exchanging | transferring the data.

② Security: gateway provide security since they allow user authentication

③ Filtering Process: Gateway peafms the filtering process by inspecting each data packet that passed through the packet. If we not filter the packet then whatever the services that arrives at the gateway carries the risk of theft

④ Domain control.

when the number of collision and broadcast domain increases. the network provider can assure that they will be able to provide better bandwidth.

⇒ A networking gateway has the capability to control collision as well as broadcast domain

⑤ Protocol conversion

Besides filtering, a gateway can also convert data packets according to the destination needs.

⇒ If the destination n/w ② architecture has different needs, it can also convert the data format accordingly.

That is the reason why it is also called as the protocol converter.

Disadvantages of Gateway

① Implementation

⇒ Generally gateways on default are installed on the routers itself. This makes it more difficult for the n/w administrators to install ② configure them.

⇒ The cost of implementation is also high.

② configuration

configuration of devices through a gateway is made even more difficult.

⇒ There must be special system administration for this purpose

③ Time delay.

Gateway networks always causes time delay since information must be translated.

⇒ There is no way can a instant transfer takeplace.

④ Connection Failure:

If there are possibilities of failure occuring at the gateway, it can lead to communication loss.

⇒ Devices on the oppopite side will no longer be able to communi-cate until the problem is resolved.

⑤ Troubleshooting.

computers on a networks are with different protocols Therefore, if there is any problem each of these computers needed to be troubleshoo-ted individually.

⇒ This makes the process more complicated since different tools must be present

Server

Server is a main computer which will be used to share the resources to each other devices within the network.

Types of server

majorly we have 3 types

① web server: It will process the request in the form of http & https [Apache tomcat, Nginx]

[Apach]

② Application server: Application Server is used to run the applicati in the server

③ Database server: Database Server is used to store the data

Server Scaling :- Server Scaling is a process of adjusting Power of computing devices like scale-up and scale-down

Types of server scaling

① horizontal scaling: Increasing the number of servers

② Vertical Scaling: Increasing the system configuration like RAM, Storage and CPU

Note: Vertical scaling is not yet Preferred for any type of operation because it will have Single Point of Contact.

VPN [Virtual Private Network]

Virtual Private Network is a private wide area n/w built on the Internet.

⇒ It allows the creation of secured tunnel b/w the different n/ws using in the Internet

⇒ By using VPN a client can connect to the Organization network remotely.

Advantages of VPN

* secure that network
* Hide Private Information
* Provide n/w Scalability
* cost effective
* Prevent data threatening

Advantages of servers

The advantages of servers are

① centralized data storage: servers provide a centralized location for data storage, making it easier to manage and access data for multiple users.

② High Performance: Server Hardware is designed for reliability and performance, allowing them to handle heavy workloads and run resource-intensive applications efficiently.

③ Data security: servers can be equipped with security features and protocols to protect data from unauthorized access, ensuring data integrity.

④ Remote access: servers can be accessed remotely, enabling users to work from different locations and access resources and data over the internet.

⑤ Scalability: servers can be easily scaled to accomodate growing storage and processing needs, making them suitable for business of all sizes.

⑥ Centralized Backup: Data on servers can be regularly backed up, reducing the risk of data loss due to hardware failures & other issues.

⑦ Resource sharing: servers allow multiple users to share resources such as printers, files & applications enhancing collaboration in a networked environment.

⑧ Redundancy and Fault tolerance.
Servers can be configured with redundancy and fault-tolerant features to minimize downtime in case of hardware failures

⑨ Centralized management: servers often come with management tools that simplify system administration updates and maintenance

⑩ customization: servers can be tailored to specific business needs allowing for the installation of specialized software and configurations.

⑪ virtualization: server virtualization technology enables the creation of multiple virtual servers on a single physical machine optimizing resource utilization.

Internet

Internet is a wide area n/w which is used to connect billions of computing devices.

Protocols

Protocols are the set of rules which is used to formatting and processing the data.

few Important Protocols

① TCP : Transmission control protocol
② IP : Internet protocol
③ http : hypertext transfer protocol
④ https : hypertext transfer protocol secured
⑤ POP : Post office protocol
⑥ SMTP : simple mail Transfer protocol
⑦ UDP : User Datagram Protocol
⑧ FTP : File transfer protocol

IP

IP stands for Internet protocol

⇒ An IP address is a unique number provided to each and every devices.

⇒ we have 2 types of IP address

(i) IPV4

(ii) IPV6

Difference b/w IPV4 and IPV6

Info	IPV4	IPV6
* octet	* 4 octets	* 8 octets
* bits	* each octet contain 8 bits	* Each octet contains 16 bits
* Total length	* 32 bits	* 128 bits
* range	* 0 - 255	* 0 - 65535
* devices	* 4 billion devices	* 340 trillion
* Example	* 192.168.1.20	* fe80:def:78d8:def:c3dd:9695 dfgd:dfst

URL [Uniform Resource Locator]

* https://www.facebook.com/login/?
  - https → protocol
  - www → subdomain
  - facebook → main domain
  - com → TOP level domain
  - login/ → parameter
  - ? → query

IPV4 classes

IPV4 class	Ip address	Usage
Class A	0.0.0.0 - 126.255.255.255	Usage for larger n/ws
Class B	128.0.0.0 - 191.255.255.255	medium range networks
Class C	192.0.0.0 - 223.255.255.255	local area networks
Class D	224.0.0.0 - 239.255.255.255	reserved for multicasting
Class E	240.0.0.0 - 255.255.255.254	research & development.

127 → localhost

IPV4 classification

1. Public Ip
2. Private Ip

Cloud computing.

Cloud computing is a delivery of on-demand IT services over a Internet which follows ~~Pays~~ Pay as you go basis

Key characteristics of Cloud computing.

1. on-demand self service
2. scalability
3. flexibility
4. automation
5. Larger-network access
6. availability.

assignment

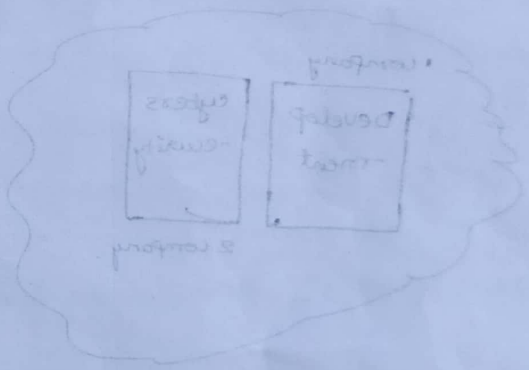
Difference b/w on-premises and on-demand services

Info	On-Premise Services	on-demand Services
location	on-premise services are installed & run on the Organizations own hardware and servers	on-demand services are hosted on remote servers & accessed via internet ⇒ users do not need to manage the physical infrastructure
scalability	Scaling on-premises solutions can be costly & time-consuming	on-demand services are typically more scalable, allowing Organizations to easily adjust resources up & down based on their needs without significant up-front costs.

Accessibility	on-premise solutions are limited to on-site access which can be a limitation for remote work & collabration	on-demand services can be accessed from anywhere with an internet connection making them more suitable for remote work & collabration
Initial costs	These solutions typically require a significant upfront investment in hardware, software & IT personnel	These services often have lower up-front costs as organizations pay for what they use on a subscription basis
Flexibility	These provide more control & customization options but can be less flexible	These services offer greater flexibility in terms of scalability & the ability to adopt new features & technologies quickly.

classification of cloud computing.

1. Based on deployment model
2. Based on service model.



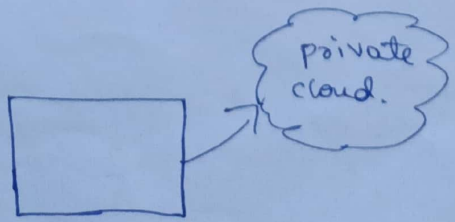
1. Deployment model

we have 4 types.

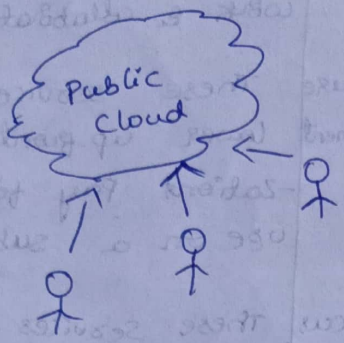
i. Private cloud
ii. Public cloud
iii. Community cloud
iv. hybrid cloud.

The cloud deployment model identifies the specific type of cloud environment based on ownership, scale, access and purpose.

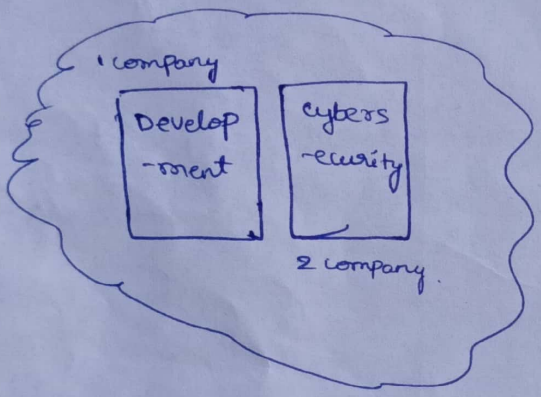
i. private cloud: Resources are managed and used by the organisation



(ii) Public cloud :- Resources available for the general public under Pay as you go basis

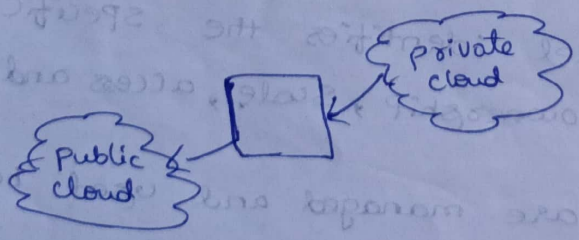


(iii) Community cloud :- Resources shared by the several organization working in the same industry.

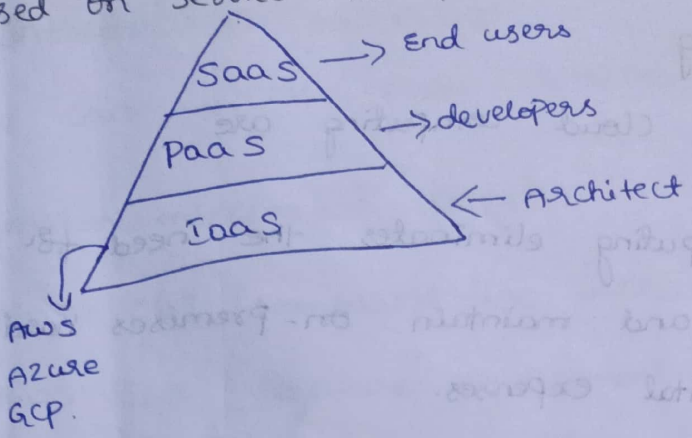


(iv) Hybrid cloud : It is a combination of public and private cloud.

* This cloud is partially managed by the service provider and partially managed by organization



Based on service model.



SaaS [Software as a service]

It replaces the traditional on-device softwares and applications can be accessed directly from the web browser without any downloads.

⇒ without downloading the any application we can access those application through web browser

Ex: whatsapp web.

Paas [Platform as a service]

It allows Agani̇cations to build, run and manage applications without any IT infrastructure.

* This makes it easier and faster development, testing and deployment.

Iaas [Infrastructure as a service]

It is a self service model for managing remote data centre, Iaas provides virtualized computing resources over the Internet hosted by a third party such as Amazon web service [AWS], microsoft Azure and Google cloud platform (GCP)

Assignments

Advantages of cloud computing (6 Points)

Advantages of cloud computing.

=) The several advantages of cloud computing are

* Cost-efficiency: Cloud computing eliminates the need for 3ganizations to invest in and maintain on-premises hardware and Software, reducing capital expenses.

* Scalability: Cloud services can easily scale up & down to meet changing business demands.

This flexibility allows 3ganizations to only pay for the resources they use.

* Accessibility: Cloud services can be accessed from anywhere with an internet connection, enabling remote work and collabration among teams in different location.

* Reliability:- leading cloud Providers offer high levels of service uptime and data redundancy, reducing the risk of downtime due to hardware failures & maintainance.

* Security:- Cloud Providers invest heavily in security measures including data encryption, access controls and compliance certifications.

* Collabration and integration: Cloud services often provide tools for collabration, data sharing and integration with other Software applications.

Introduction to AWS

We can access AWS services by using 3 ways.

1. Amazon management console
2. AWS CLI [Command line Interface]
3. AWS SDK [Software Development Kit]

AWS Stands for Amazon web services which is broadly adopted cloud Platform that provides all the IT services based on demand on Pay-as you go basis.

Benefits of AWS

1. Security
2. Availability
3. Performance
4. Scalability
5. flexibility.
6. Global Footprint

AWS Global Infrastructure.

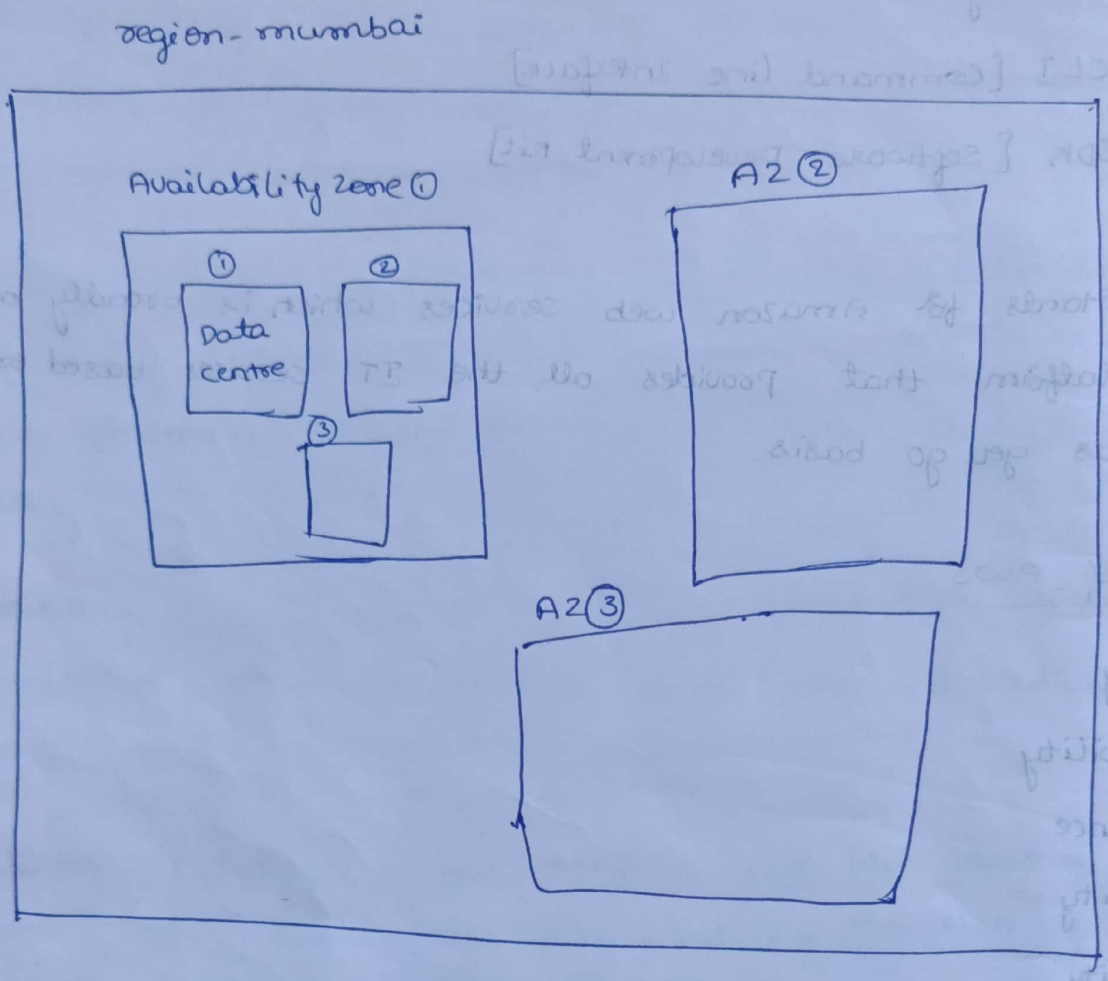
The most secure, extensive and reliable Global cloud Infrastructure for all your applications.

1. Region: Region is the geographical area which consists of 2 or more Availability Zones.

2. Availability Zone: Availability zone is a area location which is designed in a Isolated from each other.

⇒ Each availability zone is designed as an Independent failure zone

③ Edge locations : Edge locations are Aws data centres designed to deliver the services.



Aws Services.

Aws service provides total 212 services

Compute → Amazon EC2 → Elastic compute cloud

Database → Amazon RDS → Relational Database service

management and Governance → Amazon cloud watch

Networking & content Delivery → Amazon VPC
↳ Route 53
↳ Elastic load Balancing (ELB)

Storage → Amazon Simple Storage service (S3)
→ Amazon Elastic Block store (EBS)
→ Amazon Elastic File System (EFS)

Compute Service

EC2 [Elastic compute cloud]

* EC2 stands for Elastic compute cloud and it is one of our web service and it will provide scalable, resizable and High-Performance virtual servers.

Use of Amazon EC2 Service

Launch one windows Instance

① chrome
↳ Amazon management console

② sign in to the console
↳ select Root User

③ top Right → select Region → mumbai (AP-South-1)

④ In search bar → type EC2
↳ select first one EC2 → virtual Server

Zones

Zone-name

ap-south-1a
ap-South-1b
ap-South-1c

⑤ Instances → Instance is called virtual servers.

Launch one windows Instance and connect to any terminals

① click on Launch Instances [Amazon EC2 allows you to create virtual machines @ Instances that runs on Aws cloud]

② AmI : select windows

AmI [Amazon machine Image]

Image : Image is a template which contain software configuration such as operating system and applications.

AmI : An AmI is a template that contains software configu-ration such as operating system, application server and applications

③ Instance type :

↳ select t2.micro which is free tier eligible.

④ Key-Pair → used for provide security to our server.

⇒ It is a file which contains private keys which is used to connect instance securely.

⇒ Key Pair will have the extension .pem [privacy enhanced mail]

Click-on → create new key-pair

↳ Enter key-pair name

↳ key pair type

⊙RSA

↳ Extension

⊙ .pem

Click

create key-pair

⑤ Network settings

⇒ Security groups

↳ A security group is a set of firewall rules that control the traffic for your instance.

⇒ don't do anything in security group.

⑥ Storage

don't do anything.

⑦ Summary

finally click on launch instance.

States of server

① Stop Instance
② Start Instance
③ Reboot Instance
④ Terminate Instance

Connect Instance to Remote Desktop [Because AMI is a windows]

① In search bar → type Remote desktop connection.

② select public IPV4 DNS.

ec2-35-154-157-124.ap-south-1.compute-amazonaws → copy this and Paste it on computer : when you open remote desktop connection.

③ Create credentials of windows Instance

⇒ select Instance

⇒ click on Action

⇒ In that click on security.

⇒ In that click on get windows Password.

upload private key that you downloaded when you done key-pair

click on

Decrypt Password.

Note : In copy username & Password and Paste in notepad.

④ In Remote desktop connection

↳ click on → show option

↳ In that Paste → user name and password

click ok → click on yes

⑤ finally terminate the server

Launch Amazon linux Instance

System Requirements

① AMI → Amazon linux2

② Instance type → t2.micro.
↳ family of instance type [we have many family majorly we go for t2 & t3]

⇒ login to your AWS console
↳ select root user.

② type EC2 Instance

③ click on launch instance

③ ~~type~~ type server name → linux (name of the server)

④ select AMI as linux 2 → Amazon linux 2 AMI (HVM)

⑤ Instance type → t2.micro

⑥ select key-pair → Create new key Pair

⑦ click on launch instance.

View-all Instances

linux Instance we can connect in 4 ways.

① EC2 Instance Connect

⇒ Select the Instance → click on connect → EC2 instance connect

click connect

It will open one terminal

Sudo su - [to come from EC2 user to root user]

⑤ mobaxterm

In downloads → extract the mobaxterm file and go inside that folder double click on that icon automatically it will run

⇒ select private n/w access and allow access

TO connect to mobaxterm

click on session

↳ click on SSH → it will ask remote host : open instance dashboard & copy Public IPv4 address & Past it here

⇒ specify username : ec2-user

we need to go for

↳ advance SSH setting → in last we have to provide Path of private key. → for private key checks

Bookmark settings → session-name : type session-name

finally click on ACCEPT

③ git bash

⇒ open downloads → Right click → click on open gitbash here.

use command $ pwd [present working directory]

⇒ come to instance dashboard → select instance → click on connect → SSH client → copy the example Path

come to gitbash → right click & Paste it and click on enter

type yes

exit

exit

④ command Prompt.

open downloads → type cmd in search bar

C:\Users\downloads > paste the SSH client example path

it will directly connect to Linux ec2-user.

exit

exit

How to get TestPage

To open our server in any one browser we need to connect it to http @ else it will not be able to process any request using your instance IPV4 address.

To check whether your server is processing your request

=> Launch one linux instance

=> Copy the Public IPV4 address

=> Paste it in the browser

=> click on enter.

① Connect instance to any one of the terminal

② Connect -> EC2 Instance Connect -> Connect

③ Change to root user => `sudo su -`

④ next you need to update the System

`yum update -y`

⑤ next you need to install http to send request

`yum install httpd -y`

⑥ To check the status of http is active @ not use

=> `systemctl status httpd`

⑦ To start the http service

`systemctl start httpd`

⑧ To stop the http service

=> `systemctl stop httpd`

Upto this we have only outgoing traffic. to allow incoming traffic we need to set the firewall.

=> For this in instance dashboard.
=> go to security
=> click on security groups below link
=> edit inbound rules
=> add rule
=> type http
=> select anywhere IPV4
=> save rules
=> Then finally refresh the page that you entered the public IPV4 address to see whether the response is send from the server or not

TO create 5 instances at a time and getting the test page for every instance without executing those commands each time

follow these steps

① launch instance
② provide server name
③ select AMI as linux
④ select linux2 server
⑤ Instance type -> select t2.micro.
⑥ Provide key-pair
⑦ In network settings click on Edit

In that for first

① create security group

Security group-name-required.

Provide security name

for ex: sigma-all-traffic

~~Inbound security group rules~~

for next use the created security group.

Inbound security group rules.

type
↳ select → all-traffic

source-type

Anywhere.

(3) In advanced settings.

go to last → in user data

type → #!/bin/bash
yum update -y
yum install httpd -y
systemctl start httpd
systemctl enable httpd

(4) In summary provide no of instances you want and click on launch instance.

View all Instance

Select that Instance Public IPv4 address and Paste it in chrome browser you will get the Test Page.

IAM Service

IAM stands for Identity and access management and it is a one of our web service provided by aws.

⇒ IAm service providing majorly 4 components i.e.,

users, groups, Policies and Roles.

TO create IAM user

An IAM user is an identity with long-term credentials that is used to interact with aws in an account.

⇒ login to the Amazon management console as root user.
⇒ search IAM service
⇒ click create user
⇒ provide user name.

=> select provide user access to AWS.

=> select I want create an IAM user

=> select Auto-generated password

=> select user must create a new password at next sign-in

=> click on next.

=> don't do anything in set permission.

=> Review the summary.

create user

After creating the user then click on download .csv file then view all users.

Assign policy for the IAM user / to use EC2 service by the IAM user

=> you need to click on user.

=> policies we have 2 types.

(i) AWS managed

(ii) customer managed

=> In permission tab click on add permission

⇓

click on add permissions

=> select ~~click~~ on attach policies directly.

scroll down

select the policy => Amazon EC2 full access => click on next.

finally click add permission

create Policies

A policy is an object in AWS that defines permissions

=> go to policies

=> create policies

=> service → select EC2

=> manual actions

☑ select all EC2 actions

=> Resources

◉ All

next

=> Policy name

EC2fullaccess

=> description - optional

=> create Policy

create groups

=> A user group is a collection of IAM users.

=> use groups to specify permissions for a collection of users

=> go to users group

=> create group

=> provide the group name

=> select the users

=> select the permission

click on create group

create Roles

An IAM role is an identity you can create that has specific permissions that are valid for short duration

=) when we have 2 services that time we use the roles

1. create role
2. select the service → AWS
3. service → usecase
   - select EC2

   next

4. select the permission which you want to assign i.e, ec2fullaccess that you have created
5. provide role name → ec2access
6. create role

Cloud storage

we have 3 types

1. Block storage → EBS [Elastic Block Store]

The block storage is EBS we use in Aws and it is in volumes which is a block level storage.

2. object storage → S3 [Simple Storage service]
3. file storage → EFS [Elastic File System]

Volume

Volume is a block level storage

=> we have 2 types of volume

(i) Instance store volume

(ii) named volume

Instance store volume : Instance store volumes are the temporary block level storage which will be used by Instances

=> It will be created while launching the Instance and it will be deleted while terminating the Instance.

Note : (1) Volumes are availability zone specific

(2) we can have one instance with multiple volumes but we can't have one volume with multiple - instances in AWS

To create volume

=> Elastic block store

=> Volume -> create volume -> volume type (SSD gp3)

=> size (100 gb)

=> check availability zone

=> create volume

For attaching volume to any Instance

=> Select volume -> actions -> attach volume -> Instance name ->

=> attach volume

=> For removing volume first detatch the volume from the instance and then delete the volume

TO create customized AMI

=> you need to launch one instance

=> select AMI as amazon linux 2

=> Instance type t2.micro

=> security group allow-all-traffic

=> take a backup

~~how to take~~

How to take a backup [snapshot]

=> In Elastic block store -> go to snapshots -> create snapshots ->

⊙ Volume and select volume which is in running

create snapshot

TO create AMI

=> select the snapshot -> click on action -> click on create Image from snapshot

create Image

=> By using customized Image launch one Instance

① select AMI that is created by you

② launch Instance from AMI

when we need to go for customized AMI

=> when the client is given the requirement to create 50 @ more servers with the same configuration to all servers => It will be difficult to do manually, on that time you can create one Instance with the given configuration and take a snapshot of that and create your own AMI and launch 50 @ more Instances by using that AMI

Object Storage.

S3 stands for simple storage service and it is a one of our web service provided by AWS.

=> S3 is a scalable storage in the cloud.

=> An Amazon S3 is an object storage service that offers industry-leading scalability, data availability, security and performan

=> Every object in S3 is stored in a Bucket.

TO create S3 Bucket

Bucket: Buckets are containers for data stored in S3.

=> S3 Bucket is a Region-specific

=> S3 service is a global specific

① search S3

② Create Bucket

③ Provide name for bucket

④ select region-mumbai

⑤ object ownership
- ⊙ ACLs enabled
  - ↳ Access control list

⑥ object ownership
- ⊙ object writer.

⑦ Block all public access → uncheck the checkbox

⑧ Bucket versioning: versioning is a means of keeping multiple variants of an object in the same bucket.

⑨ ⊙ Enable

⑩ create bucket

To upload any files inside bucket

objects : objects are the fundamental entities stored in amazon S3

⇒ click on bucket.
⇒ click on upload
⇒ add file
⇒ select file
⇒ click on open
⇒ upload.

how to access objects in publically.

⇒ click on that object
⇒ Set the Permission for the object to access Publically.
⇒ click on permission
⇒ click on edit
⇒ check the checkbox i-e, Read & read option for Public access
⇒ ☑ I understood
⇒ click on save changes.

To check whether your uploaded file has Public access or not.

⇒ click the file → set permissions → copy the url and Paste it in another tab
⇒ If you get the file or image what you have uploaded then it has the public access

Static Deployment

uploading file to S3 bucket by creating Instance

Steps

① create one Linux Instance

② create one S3 bucket

③ Connect instance to terminal
↳ Connect → EC2 Instance Connect → Connect

④ type sudo su -

⑤ create one file either text @ html file
ex: sample.txt @ sample.html

⑥ TO create file use command
⇒ touch sample.txt

⑦ TO get inside the file use command
⇒ vi sample.txt

⑧ TO insert data → type i

⑨ Then type any data you want

⑩ TO come out from that file use
esc :wq

⑪ TO read the content use
⇒ cat sample.txt

⑫ TO ~~create~~ Connect EC2 & S3 service we need to create role

(13) Search for IAm service

(14) select ⊙ Aws Service

(15) usecase / select EC2

(16) click on next

(17) ~~select~~ Permission as → S3-full-access

(18) Provide role name

(19) create role

TO attach IAm role to EC2

(1) select Instance

(2) click on Actions

(3) click on Security

(4) ~~xx~~ Click on modify IAm role

(5) ~~click~~ select the IAm role → that you created

(6) update IAm role

After doing all these → to upload file

In terminal type → aws S3 cp sample.txt (file name) S3://sigmain/sample.txt (bucketname).

once execute this command ~~copy~~ open the uploaded file and give permission to access Publically → click on Permission → select read & read option → save

then copy the url and Paste it in new tab you will get the web page @ the text what you have in the created file

Autoscaling EC2.

If any one of the server is getting down then automatically other server will replace it.

① first we need to create template

② second Create Autoscaling group.

Steps

① select region - mumbai

TO Create Template

② click on create ↑ launch template

③ Provide template name

④ Version description → optional

⑤ go to quickstart → select AMI as Amazon linux.
select Linux 2 version

⑥ Instance type → select t2.micro

⑦ Key pair → select the key pair that you created

⑧ Select security group that you created → that will enable all-traff

⑨ In advanced details → go to last

inside user-data Paste

```
#!/bin/bash
yum update -y
yum install -y httpd
systemctl start httpd
systemctl enable httpd
EC2AZ=$(curl -s http://169.254.169.254/latest/meta-data/placement/availability-zone)
```

echo ' <h1> This Amazon EC2 instance is located in this Availability zone: AZID </h1> ' > /var/www/html/index.txt
sed "s/AZID/$EC2AZ/" /var/www/html/index.txt > /var/www/html/index.html

(10) create launch template

To create autoscaling group

=> click on autoscaling group :- Autoscaling groups are collections of Amazon EC2 instances that enables automatic scaling.

(1) create auto scaling group

(2) Provide autoscaling group name -> appserverhttpd.

(3) launch template
↳ select the template that you have created

(4) click on next

(5) Availability zones
select all zones -> ap-south-1a
ap-south-1b
ap-south-1c

(6) next
don't do anything in configuration

(7) next

(8) Desired capacity
2

(9) Scaling
minimum — 2 — maximum — 2

(10) next

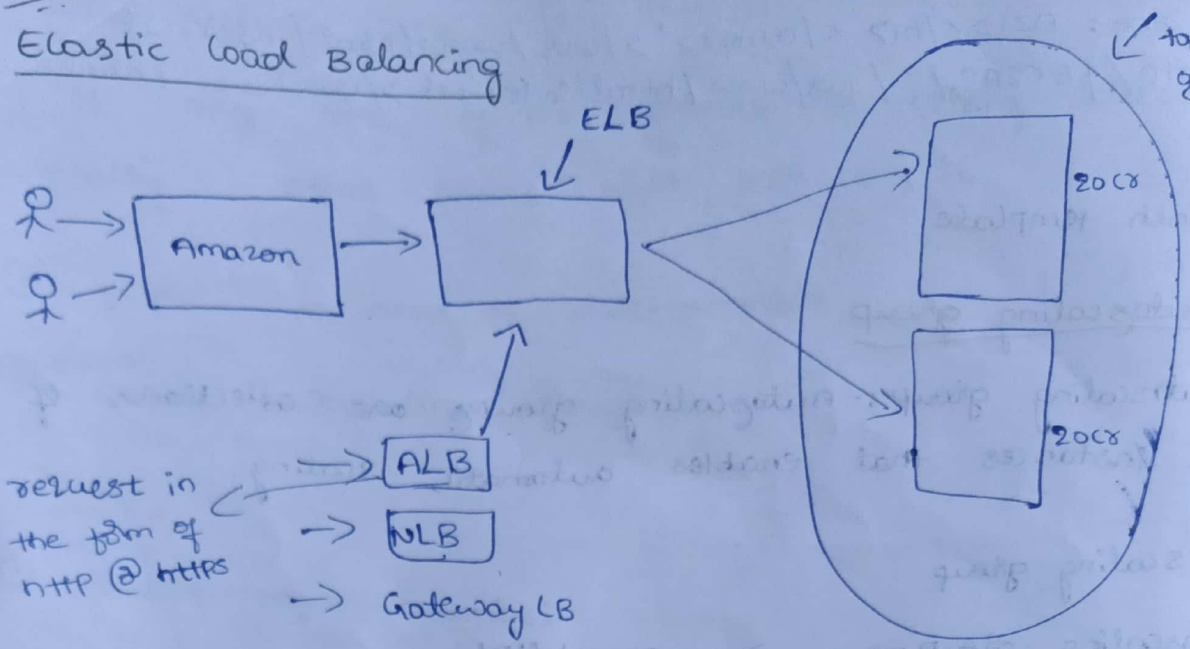
(11) next

(12) Create autoscaling group.

when any of the server is getting down then automatically another Instance will replace that server TO do this we use autoscaling

Imp

Elastic Load Balancing



=> First create template then create autoscaling group and then create the target group & ALB

TO create target groups

① Click on create target group
=> your load balancer routes requests to the targets in target group.

② target type
Instances

③ Provide target name -> webapprequests

④ Note :- For the http protocol 80 is the default port no
=> For https protocol 443 is the default port no

⑤ select IPV4

⑥ In last click on next

⑦ select those 2 servers which you created by autoscaling group.

⑧ click on include as pending below

⑨ click on create group.

3 types of load balancers we have

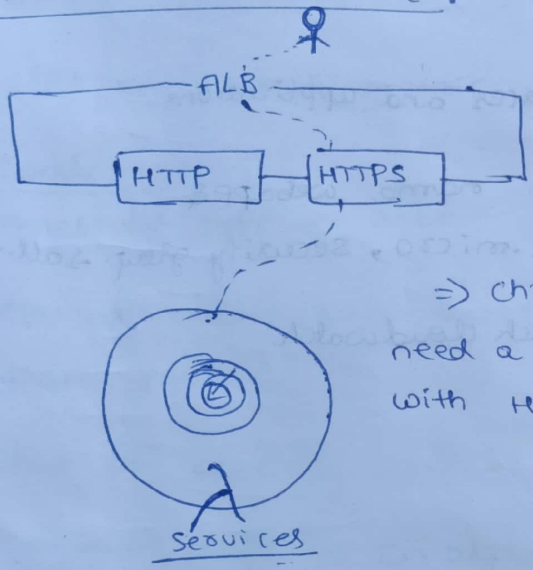
① Application load balancer

② Network load balancer

③ Gateway load balancer

Load balancers: Elastic load Balancing scales your load balancer capacity automatically in response to changes in incoming traffic the server

Application load Balancer:



⇒ choose an application load balancer when you need a flexible feature set for your applications with HTTP and HTTPS. traffic

In load balancers

① click on create load balancer → In ALB click on create ALB
ALB distributes incoming HTTP & HTTPS traffic across multiple targets. such as Amazon EC2 instances

② Provide ALB name → Amazon http

③ mappings
select all-3 availability zones.

④ security groups
select the security group that you created.

⑤ Default-action
select the target group that you created

⑥ click on create load balance

Note: DNS stands for Domain name System
⇒ Copy DNS url and Paste it in new tab

Delete

=> First delete load balancer -> select -> click on action -> delete load balancer

=> delete target group -> select target group => click on action -> delete target group.

=> delete autoscaling group

=> delete templates

Cloud watch

Cloud watch is used to monitor resources and applications

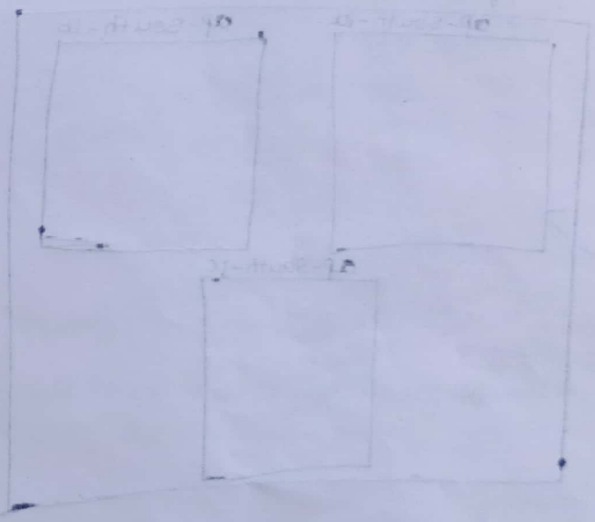
1. First launch one instance with the name webapp & AMI as linux 2, Instance type - t2.micro, security group -> all-traffic
2. create dashboard in cloudwatch -> search cloudwatch
3. click on dashboard
4. click on create dashboard
5. provide dashboard name
6. click on create-dashboard
7. select metrics
8. widget type

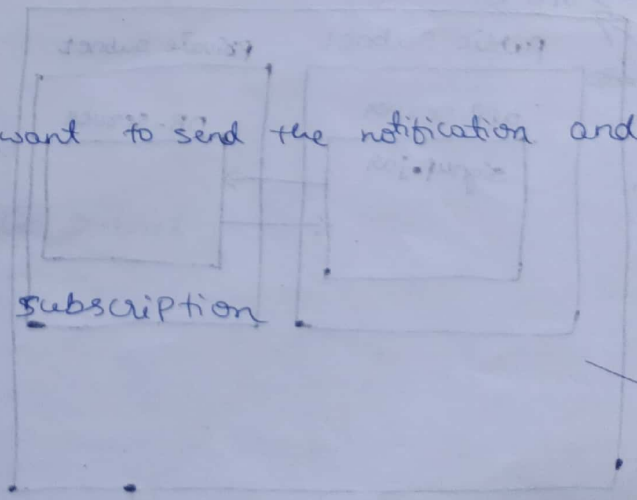
=> line

9. click on next
10. click on EC2
11. click per-Instance metrics
12. copy instance-id and paste it in searchbar and select the created instance based on cpu-utilization
13. click on Create widget
14. click on save

create Alarams to set the remainder

1. click on alarams
2. click on In alaram
3. click on create alaram
4. Select the metric.
5. Slick on EC2
6. clicks on per-instance metric
7. copy instance-id and paste it in search bar and selelt the created instance based on CPU-utilization
8. click on select metric.
9. Conditions
   Threshold type → static
   CPU utilization is → lower than 1
10. click on next
11. alaram state → In alaram
12. send a notification to the following SNS topic
    ⇒select create new topic
13. Provide new-topic name
14. provide email-id that you want to send the notification and click on create topic
15. go the mail and confirm subscription
16. click on add EC2 action
17. select Stop this Instance.
18. click on next
19. provide alaram name
20. click on next
21. click on create alaram





Virtual private cloud [VPC]

It is a one of the web service provided by AWS

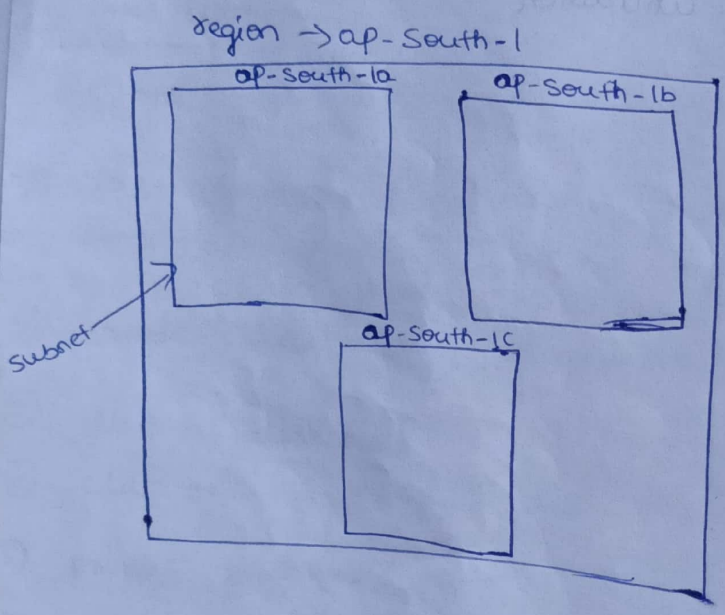
=> VPC is a isolated cloud resource

Types of VPC

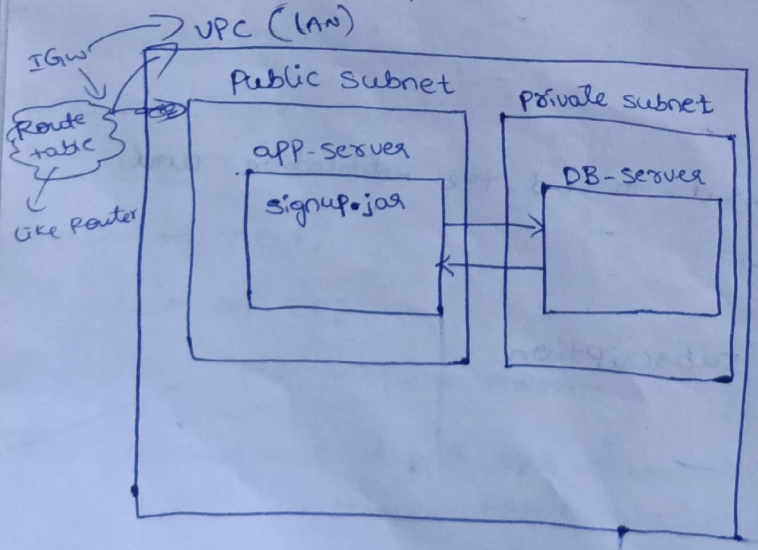
i) Default VPC

ii) Custom VPC

Default-VPC



Custom-VPC



Note :-

=> Default-VPC is called for region

=> Default-subnets is called for availability zones

Components of VPC Service

majorly we have 4 components

i) VPC
ii) Subnet
iii) Route-table
iv) Internet gateway.

VPC

An VPC is an isolated portion of an Aws cloud by Aws objects such as Amazon EC2 instances

TO create VPC

(i) Search VPC
(ii) click on create VPC
(iii) Provide VPC name
(iv) IPV4 CIDR [classless Inter Domain routes]
10.0.0.0/24 → range of IP address [will have the range from 16-32]
(v) click on [create VPC]

Create Subnets

Subnet will defines the range of Ip address

(i) click on create subnet
(ii) specify VPC which you created
(iii) Provide subnet-name → Public Subnet.
(iv) Availability zone
Select 1a
(v) IPV4 Subnet CIDR block
10.0.0.0/25
(vi) click on [create subnet]

Route-table

A Route-table specifies how packets are forwarded b/w the subnets within your VPC to the Internet.

(i) click on create route-table
(ii) Provide route-table name
(iii) select VPC that you created
(iv) click on route table

Internet Gateway

An Internet Gateway is a horizontally scaled and highly available VPC component that enables communication b/w your VPC and the internet.

click on
(i) ↑ create Internet-gateway.
(ii) Provide gateway name
(iii) click-on create Internet gateway.

Attach Internet gateway to the created VPC

=> select the Internet gateway
=> click on actions
=> click on attach-to VPC
=> select VPC which you created
=> click on attach-internet gateway.

Launch one instance with region-mumbai

AmI -> Linux 2

Instance type -> t2.micro

KeyPair -> select keypair which you created

Network settings -> click on edit -> select VPC you created -> select the subnet you created -> auto-assign-Public IP -> ~~Create~~ Enable

=> create security-group -> all-traffic -> anywhere

=> Launch Instance

DO - configuration in Route tables

① click on route - table

=> select route - table that you created

=> click on action

=> click on edit - subnet associations

=> select the Particular subnet

=> click on save associations

TO add Internet gateway to the Route table

=> select the route - table

=> click on routes

=> click on Edit routes

=> click on add routes

=> destination

0	sdect 0.0.0.0/0

=> Target -> select Internet gateway -> select your Internet gateway.

=> click on save changes

How to delete

=> First terminate the Instance

=> In route table -> edit subnet association -> uncheck -> save

=> select Internet - gateway -> detatch from VPC

=> delete Internet gateway

=> delete subnet

=> remove the added Internet gateway in route table -> routes -> Edit routes -> remove

=> delete route table

=> delete VPC

Private subnets

(i) click on create subnet

(ii) Select VPC you created

(iii) Provide private subnet name

(iv) select availability-zone → 1a

(v) Provide IPv4 CIDR → ~~10.0.0.0~~ 10.0.0.128/25

(vi) click on create subnet

we need another route table & not need the Internet gate
because it allow to connect b/w
Internet & VPC so as the datab
is secured we don't need that
-way instead we create NAT g

(i) create route-table

(ii) Provide name

(iii) select VPC

(iv) click on create route table

Create one Linux Instance

=> Provide name

=> select Amn Linux 2

=> select Instance type t2.micro

=> n/w settings → Edit

* select VPC you created
* select the private subnet
* disable Public access
* security-group → all-traffic

=> ~~Creo~~ Launch instance

you need to associate private subnet to the private route table
=> select route-table → action → Edit subnet association → select the private subnet → save changes

As the Public IP is disabled we cannot connect it to any terminal so we need one gateway i.e, NAT gateway

→ N/w Address Translation.

we need to create NAT gateway.

NAT gateway is used to enable instances in a private subnet to connect to services your VPC.

(i) click on create NAT gateway.

(ii) Provide NAT gateway name

(iii) select the Public subnet that you created.

(iv) Connectivity type

⇒ Public

(v) click on allocate Elastic IP

↳ It is a Static IP provided by AWS

(vi) create NAT gateway.

Add NAT gateway to the Route table.

⇒ select private-route table → click on routes → edit routes → add routes → select 0.0.0.0/0 → target - NAT gateway → select NAT gateway which you created → save

To connect to the terminal

⇒ select public-subnet instance → ~~to~~ Connect it to the terminal → sudo su -

⇒ select private-subnet instance → click on connect → SSH client → copy example Path link → come to terminal that you opened & Paste it → it will denied the access for that.

⇒ create one file → Vi Private-KeyPairname.Pem → copy the entire content ~~of~~ the file & Paste it in the file → esc → :wq

⇒ type → chmod 400 KeyPairname.Pem (filename)

⇒ Copy the (SSH client) example Path of private-subnet instance and Paste it in the terminal → now it will connect to the terminal

=> type ping google.com -> to show how much data is transmitted & lost

=> to stop the transmission press ctrl + C

=> If the data is not transmitting then go to the route table -> select the private-subnet route table -> routes -> edit routes -> add routes -> 0.0.0.0/0 -> target = NAT gateway -> save

How to delete

=> First terminate the Instance

-> delete Elastic IP -> actions -> release Elastic IP

-> delete NAT gateway

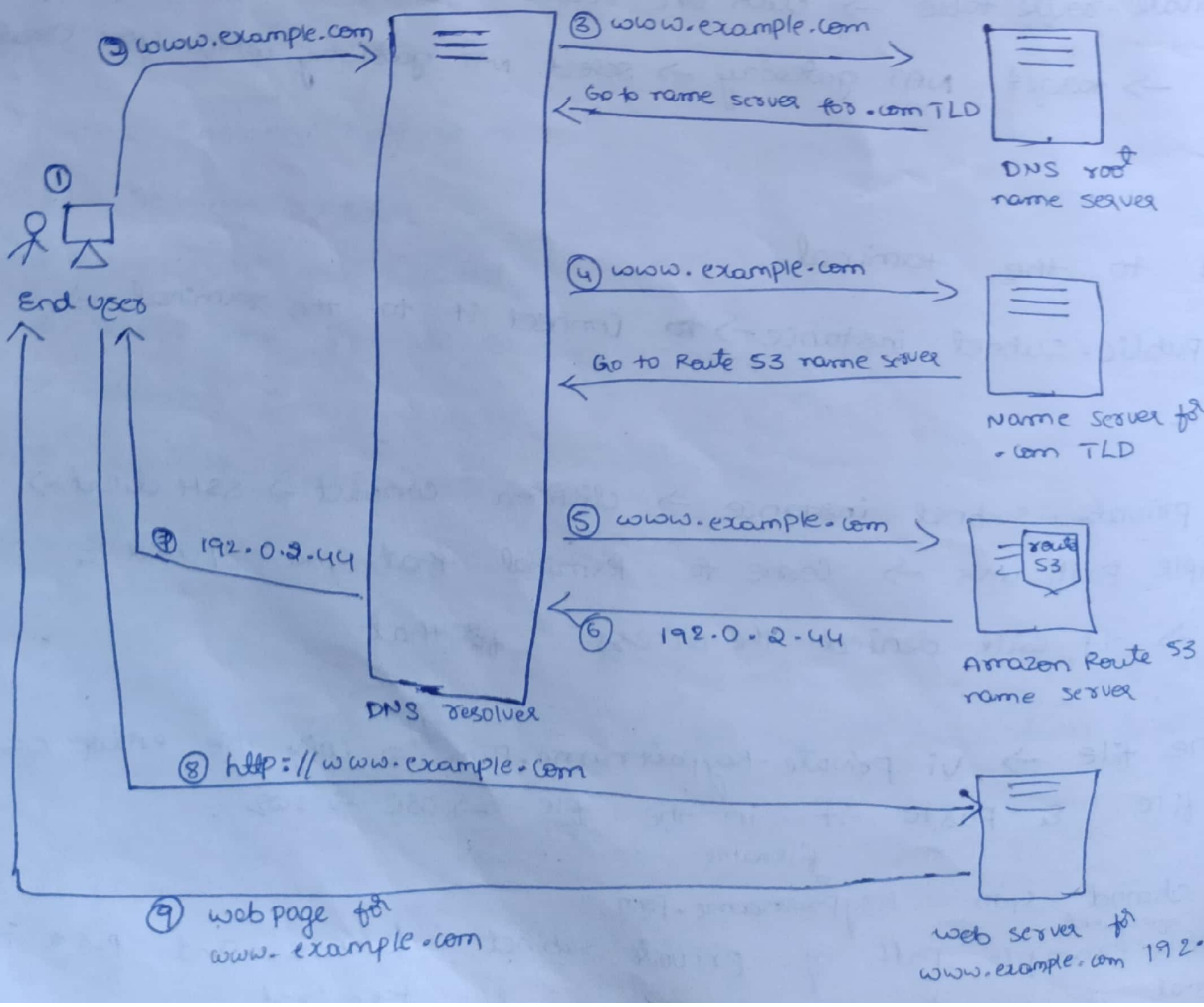
=> delete route table

=> delete subnet

=> delete VPC

Domain Name System

Domain Name system is a system of converting from domain name to IPV4 address.



Route 53

It is a scalable DNS and domain name registration

(i) Search Route 53

(ii) click on create hosted zone

↳ A hosted zone is a container that holds information about how you want to route traffic for a domain, such as example.com

(iii) Provide domain name

(iv) Description - optional

(v) type of hosted zone

Public

(vi) click on create hosted zone

TO specify server to the hosted zone you need to create record

=> click on created hosted zone

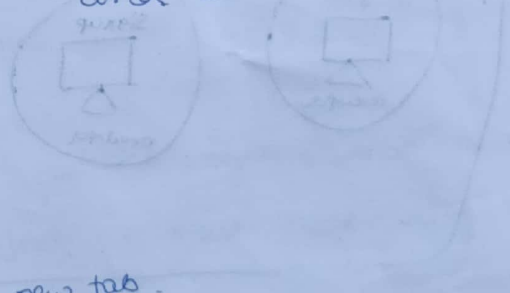
=> click on create record

=> provide subdomain name

=> value
192.120.10.18

=> click on create record.

Record type -> A
Route traffic to an IPV4 address and some AWS resources



=> copy the record url and paste it on new tab.

Version control system ?

Version control system is a tool that helps to manage changes to source code & any set of files

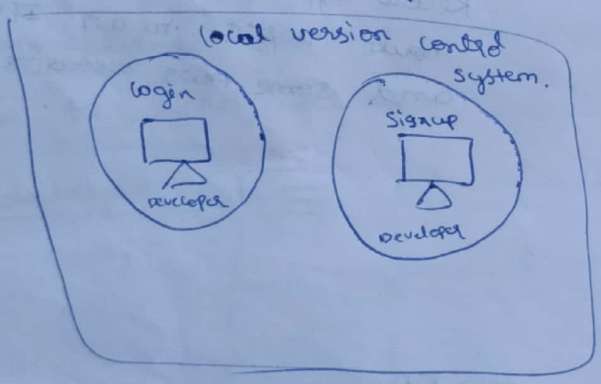
=> It maintains a historical record of modifications, enabling collaboration among multiple contributors

=> Developers use VCS to track changes, revert to previous states and work concurrently on different branches, promoting efficient and organized software development

There are 3 types of version control system, they are

(i) local version control system
(ii) centralized version control system
(iii) Distributed version control system

Local version control system:



=> In this type of version control system we have multiple source codes like signup & login, each code will be developed by different different developers in their local system.

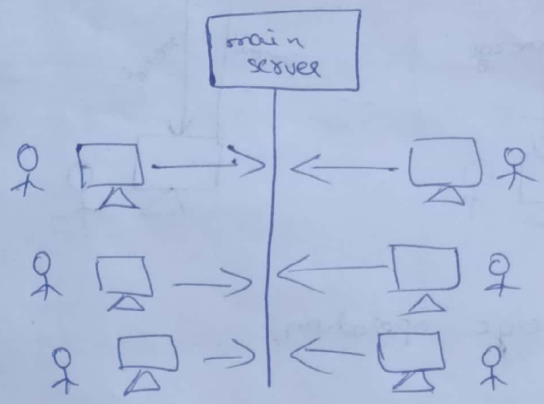
=> By using LVCS's [local version control system], we can track the changes to files and directories within a project. allowing the users to see the previous states & compare different versions

=> The drawbacks of LVCS are it lacks the feature of collaborative work.

centralized version control system

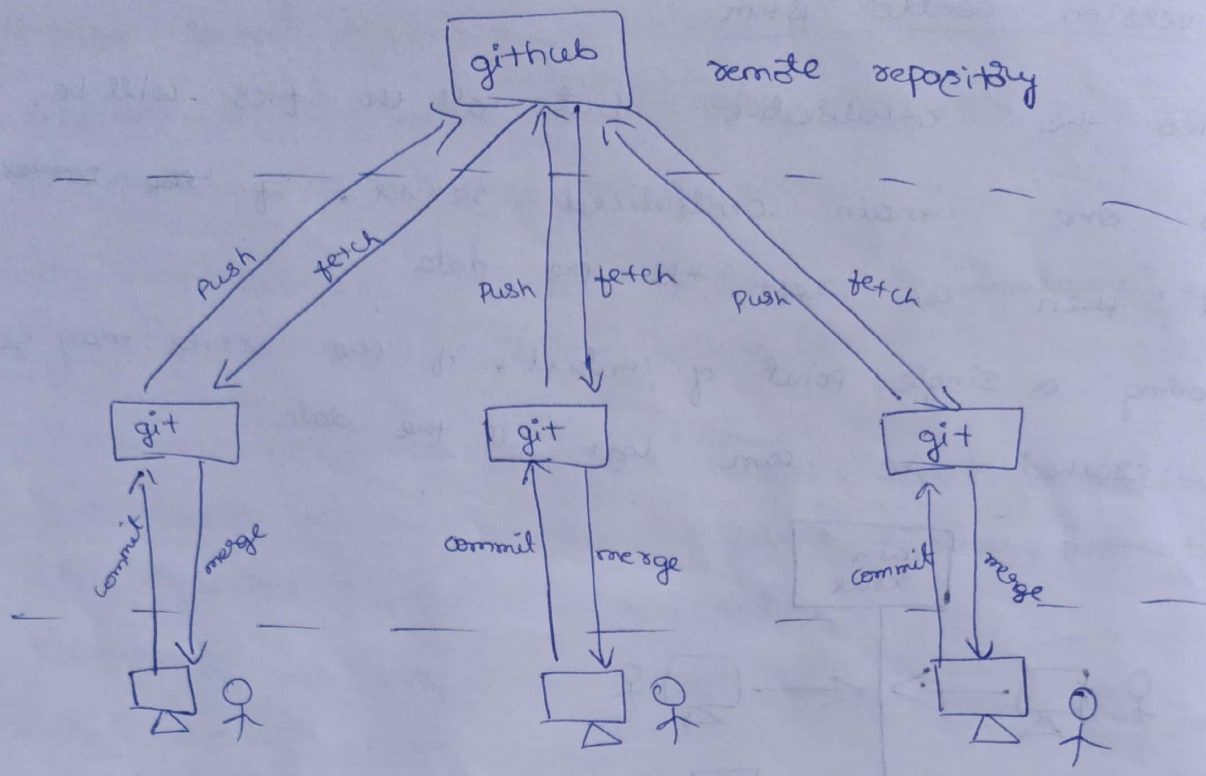
* It enables the collaborative but all the files will be shared to one main centralized server, ~~if the server get damaged then we lost all the data~~

=> It is having a single point of contact, if the server may get damaged (or) crashed we can lost all the data



Distributed version control system:

* git is a widely used distributed version control system.
* It also enables the collaborative work by storing all the files in global repository.
* It is a type of version control system that allows multiple developers to collaborate on a project without necessarily being connected to a central server.
* Each user has a complete copy of the projects repository, enabling them to work independently.
* popular DVCS are Git and mercurial



Pull → complete fetch & merge operation.

=) first the developers need to install the git into their local systems called as local repository

=) Then they need to create account in github [global repository]

=) TO Push the files from local device to local repository [git] they need to "commit" the files.

=) TO move the files from local repository to global repository they need to use "Push"

=) Agin ~~from~~ to retrieve the files the global repository to local repository they need to use "fetch" and again from local repository to local device they need to ~~use~~ perform "merge" operation complete this is known as pull operation

what is git and github.

=> git is a tool and it is a local repository., we need to install this

=> github is an global repository and we need to create account in this

Difference b/w git and github

git	github
* git is a tool	* github is a code hosting platform
* It is a local repository	* It is a global repository
* git is a command line Interface tool	* github is a graphical user Interface
* git works locally	* It requires internet connection
* git is a software	* github is a service

git operations

Configuration of git

=> create one folder in desktop / anywhere.

=> open that folder -> right click -> open gitbash here

=> to know the Path where you are present type -> Pwd
↳ Present working directory

=> git init -> It will create (initialize) the empty repository.

=> ls -> list of files

=> ~~create one file~~ vi Sample.txt

=> i -> type some data inside that file.

=> esc :wq
↳ write & quit

=> cat Sample.txt -> to view the data inside file

=> git status -> to know status of the file in where the file is present
=> If it is present in workspace -> the filename is shown in red color.
=> If it shown in green color then it is present in staging area

=> $git add sample.txt -> to add the file from workspace to staging

=> $git status

=> $git commit -m "first Project" -> to add the file to local repository

"first project" -> It is a message which shows the versions (modifications)

It will not work for first time, so we need to configure git

Configuration

git config --global user.email "githubmailid"

git config --global user.name "username"
↳ take from github repository

Provide id & name in same format

=> $git commit -m "first Project"

Create one remote repository in github.

=> Login to your github account

=> Click on new / create a new repository.

=> Provide repository name

=> Description -> optional

=> Select ● Public

=> create repository.

In terminal

=> $ git remote -v => to show whether the connection is established b/w local git & global remote repository

=> If it shows (fetch) & (Push) which means the connection has been established, else (it not shows any line directly it shows the $ symbol to type another command) then we need to use below command

=> $git remote add origin "url"

(needed only when we are establishing connection b/w local & global repository) -> origin name like variable name.

come to github and there you will have the url for created repository copy it and paste it.

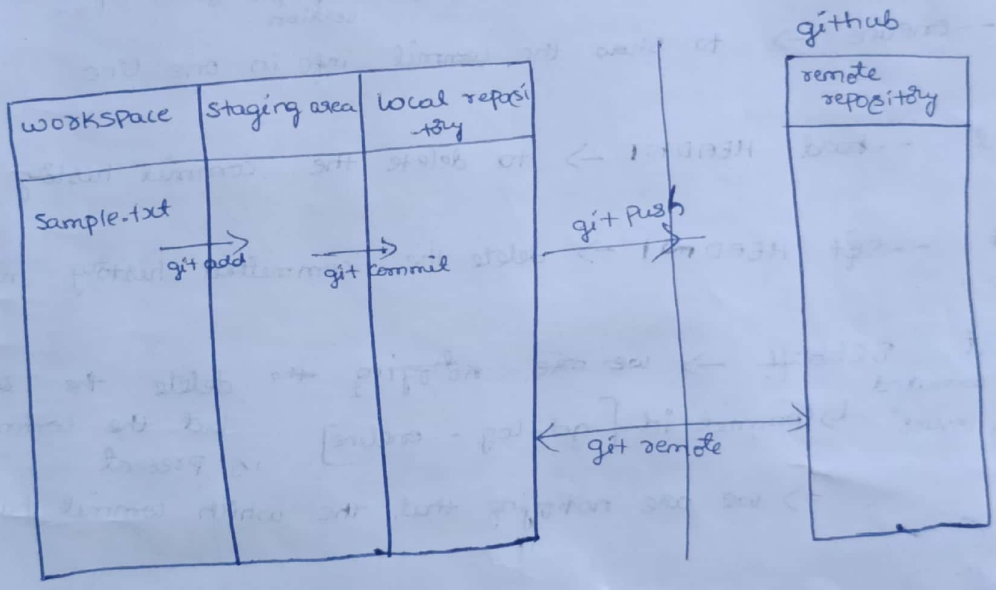
=> $git push origin master

It will not push the files to remote repository because still now you are not authorized so you need to authorize by providing username & password after that the files will be pushed.

=> come to terminal -> there you have the message that the files are pushed.

=> to know whether the file is pushed (or) not -> open the repository there you have the file.

=> $git log -> to know how many commits you have done.

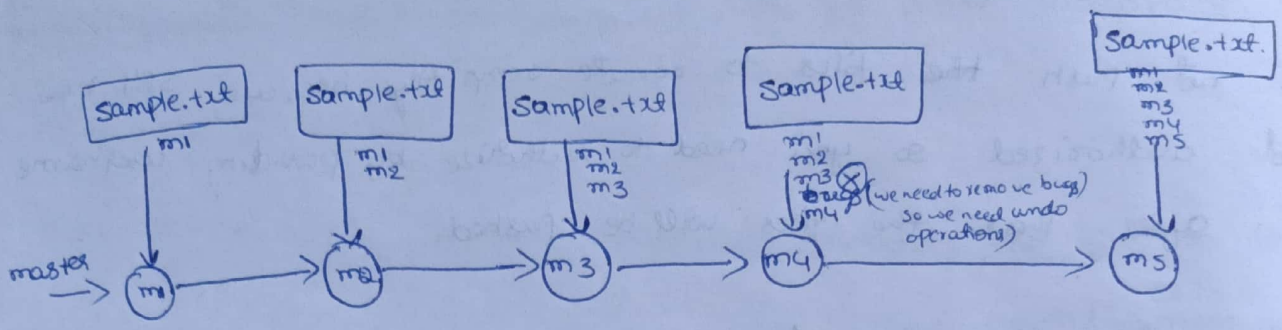


workspace -> untracked area

staging area -> tracked area

=> echo "m3" >> filename => shortcut to add data to the file

=>



Undo commands

1) $git reset Sample.txt → move file from Staging area to workspace
[before committing the file when we know if there are any bugs in the file then we move that file from staging to workspace]

2) $git checkout sample.txt → to remove the last added content in the file
[If we add the last contents as bugs then we can remove that line by using this command]
⇒ to remove the content of file from updated version to previous version

3) $git log --oneline → to show the commit info in one line

4) $git reset --hard HEAD m1 → to delete the commit history with data

5) $ git reset --soft HEAD m1 → delete the ~~committed~~ history not the data

6) $git revert 8Cbbaff → ~~we are notifying the~~ delete the ~~commit~~ data
after giving this command type "having issues" ↳ commit id [git log --oneline] but the commit history is present
⇒ we are notifying that the which commit has issues.

⇒ $git diff local repository master origin/master global
compare from this to this

⇒ $git diff origin/master master.

when we done upto here we the contents till now.

=> cat sample.txt

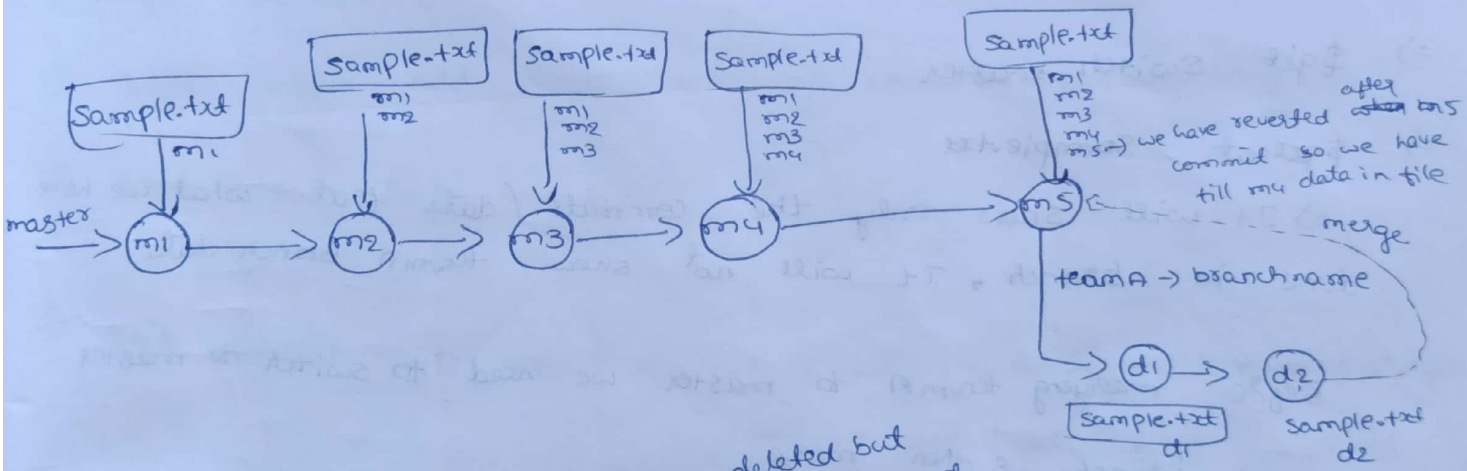
m1
m2
m3
m4

=> Still we are present in master branch

git branch

There are 2 default branches

i) master
ii) main



=> we have reverted after m5 commit So we have data till m4 in file. -> The data has been deleted but commit history is present

=> Push it to global repository

=> Now the scenario is we need to change the branch called teamA and then perform those operation [add & commit]

=> $git branch -> to know the branches of git

=> $git branch teamA -> branch name -> to create the branch in git

=> $git switch teamA -> to switch the branch from master to teamA

=> $cat sample.txt -> It will show the data present in master branch

=> $echo "d1" >> sample.txt

=> $git add.

=> $git commit -m "d1"

=> $echo "d2" >> sample.txt

=> $git add.

=> $git commit -m "d2"

=> $git switch master

=> $cat sample.txt

=> It will show only the commits / data that what we have done in branch, It will not show teamA branch data

=> Before merging teamA to master we need to switch to master branch & then merge

=> $git merge teamA

=> $git checkout -b dev -> to create new branch & switch the branch.

=> $git branch -d dev -> delete the branch which has merged

create new folder -> open gitbash here ->

=> $git clone url -> to take backup of file sample.txt

↳ come to github and you have code there in last you have copy option => copy it & paste it

to access the files change the directory

=> $cd flipkart.git

=> $cat sample.txt

we have deleted teamA branch and now again creating branch called teamA and adding the data d1 & d2 in two different commits

⇒ $git checkout -b teamA

⇒ $echo "d1" ⇒ Sample.txt

⇒ $git add .

⇒ $git commit -m "d1"

⇒ add d2 data also & perform commit

⇒ $git remote -v

⇒ $git remote add origin url [to establish the connection]

⇒ $git push origin teamA

⇒ $git branch -D teamA ⇒ -D is used to delete the branch that has not merged [delete only from local repository]

⇒ before deleting from global you need to delete in local repository

⇒ to delete from global repository

$git push origin --delete teamA → branch name

assignment

Assigning tags to each commit

⇒ $git tag V1 → tag name ⇒ It will add the tag to the current commit

⇒ $git tag V2 commit code → respective commit ⇒ This will add the tag to the specified commit

(committed tag name)

⇒ provide tag to all commits

⇒ $git push origin tag name ⇒ It will push the ~~respective~~ specified tag to global repository

⇒ $git push origin --tags → assignment. → to push all the tags at a time

=> To fetch data from global repository to local repository.

=> first delete all the data & commits

=> $git reset hard head~6

=> $git fetch origin

=> $git pull origin teamA to fetch the data [Pull operation]

Ignore the files that was not been added to commit:

=) we have multiple files with different different file extensions [.html, .js, .txt]

=> If we doesn't want to ~~ignore the file~~ commit the files that are having extension .js then we need to use .gitignore so that what are the files that are mentioned in this will not get committed remaining files will be committed.

~~=> we cannot remove the individual files that has not been~~

=> git init

=> touch a.html -> create empty file

=> touch b.html

=> touch c.html

=> touch demo.js

=> touch demo1.js

=> touch sample.txt

=> ~~$vi .gitignore~~ $vi .gitignore

(If will open one terminal

specify the files that you want to ignore from commit -> demo.js, demo2.js -> *.js

esc :wq

~~esc~~

=> ls -a -> to show the list of ignored files.

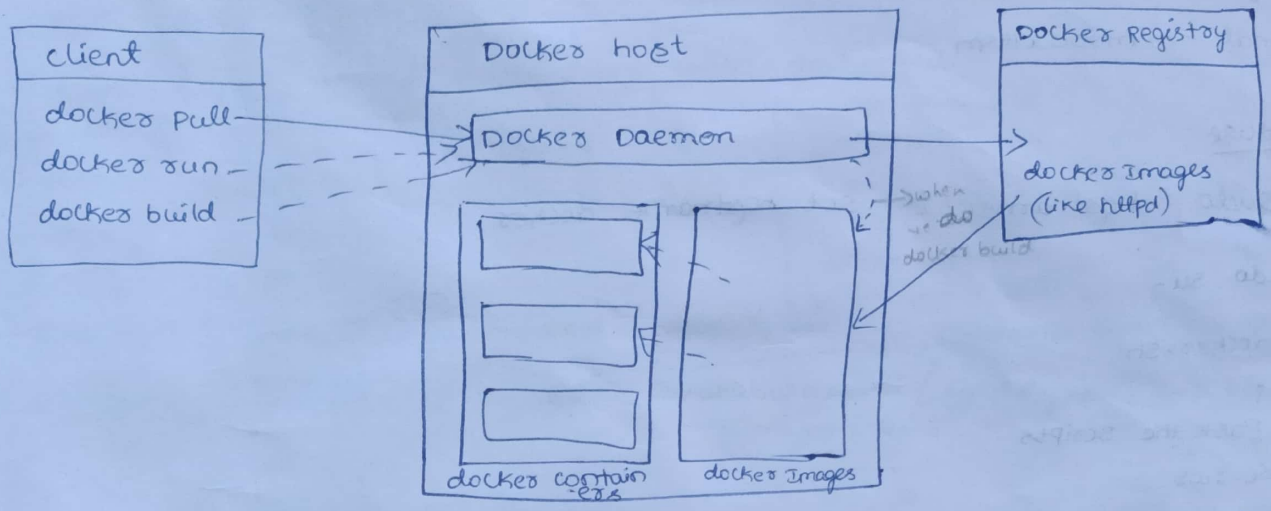
If we done this
=> $git add .

=> we cannot remove the files individually, in this we have only small files, If we want so files that are not to be committed then we cannot remove all those files individually from staging area to workspace so we are using vi.gitignore to specify the files that are not to be committed.

Docker

Docker is a open source container tool which is used to automat the deployment of application in lightweight container.

Docker Architecture



Docker client: Docker client is a command line tool that helps to contrd and manage docker containers and docker images using computer terminal with commands like docker Pull, docker run, docker build => Docker Pull -> to Pull the images from docker registry to docker host
docker run -> It will run the images
docker build -> you are creating a customized image

Docker host: Docker host is an server where the docker Software will be installed and it will create environment to run the container and to store the docker Images

Docker Registry: Docker registry is a repository which contains predefine images like httpd.

Docker Daemon :- Docker Daemon is the background service that managing docker containers & docker images on the docker host

Docker Images :- Docker Images are the template that defines the entire environment to run applications in isolated containers

Container: Containers are the isolated Packages that holds everything needed to run the softwares/applications

Docker containers :- It is a running instance of a docker image where applications can run consistently and applications can run across different environments

Installation of Docker

System Requirement

AMI -> linux2

Instance type -> t2.micro

security-group -> all-traffic

Terminal -> mobaxterm

Procedure

=> # sudo hostnamectl set-hostname docker

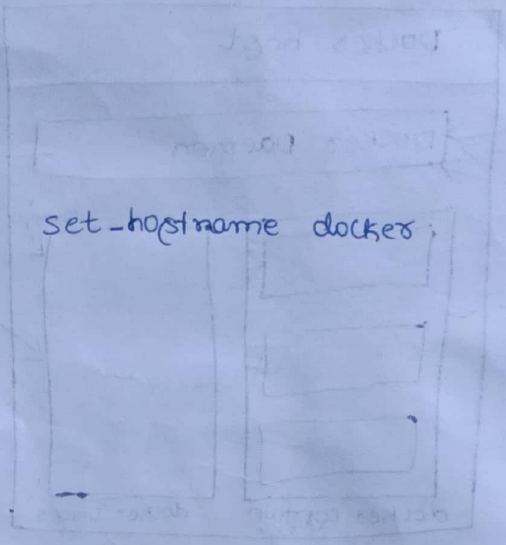
=># sudo su-

=># vi docker.sh

↳ i

Paste the scripts

esc :wq



=> # sh docker.sh -> to install the docker

=> # docker --version -> to check whether the docker is installed or not

=> # docker Images -> to check the images

=> # docker PS -> It will provide running container list

Task1

Pull one httpd Image and run that container

=> # docker Pull httpd

=> # docker images

=> # docker run -it -d httpd -> to create the container

↳ interactive mode ↳ works in the background

=># docker rm Container-id -> to delete the container

we cannot delete the running container first

we need to stop & then delete

=> # docker stop containerid

=> # docker PS -a => It will provide both stopped & running container

=> # docker start container-id => to start the container

=> #docker

=> #docker stop containerid

=> #docker rm containerid

=> #docker rmi httpd -> to remove the images

Task 2

Pull one httpd image and run a container and then take a snapshot [to backup] and push into dockerhub

=> # docker Pull httpd

=> #docker run -it -d httpd

=> # docker PS

=> ~~#docker commit containerid~~ ~~repository name~~

=> #docker commit containerid dockerhubusername/repository name backupfilename

In dockerhub

=> create one repository

Provide name -> demo

select Public

create

dockerhubusername -> sowmyayp8/demo

repository name -> demo

backupfilename -> take snapshot of running container

Note: In a dockerhub we can store docker images we can't store any docker containers

=> #docker Push backupfilename

=> #docker login

username: SowmyayP8

Password: gOWDA2123

note: for first time we cannot Push images to docker because we are not yet login so first we need to login and then Push the images

=> #docker Push backupfilename

Task 3

Pull one web Server Nginx and run a container and it in a a chrome tab

#docker Pull nginx

docker images

docker run -it -d -P 8000:80 nginx

#docker PS

come to instance -> select instance -> copy Public Ip -> open new tab -> paste PublicIp :8000 -Ex: 67.127.11.0:8000

docker stop containerid

docker rm containerid

#docker rmi imagename

Task-4

Pull one amazon linux image and run the containes and install tomcat inside that containes

=> Install docker -> open mobaxterm -> connect to mobaxterm terminal ->
sudo hostnamectl set-hostname docker

=> # sudo su -
=> vi docker.sh
i -> Paste the scripts
esc :wq

=> # docker pull amazonlinux
=> # docker run -it -d amazonlinux
=> # docker PS
=> # docker exec -it containerid /bin/bash -> to get inside that container
=> ls -> we have some predefined directories in linux that will be shown here

=> go to chrome -> search for apache tomcat download -> select version -> in that we have tar.gz -> copy this link address and paste it in wget

=> # wget apachetomcat server url what you have copied
we get error because wget is the unknow command so first
=> #
we need to install that

=> # yum install wget -y

=> # wget apache tomcat linkaddress

=> # ls

=> # yum install tar -y

=> # yum install gzip -y

=> # tar -zxvf filename -> to extract the file server because the file present in zip format
apache-tomcat-9.0.83.tar.gz
we need to extract it

to start the server first we need to go inside apache tomcat folder

=> # ls (extracted file name)

=> # cd apache-tomcat-9.0.83

=> # cd bin/ -> In the server we bin directory we need to go inside that directory

=> # sh startup.sh -> to start the server

↳ The server is not going to started because we don't have java environment

=> # yum install java -y

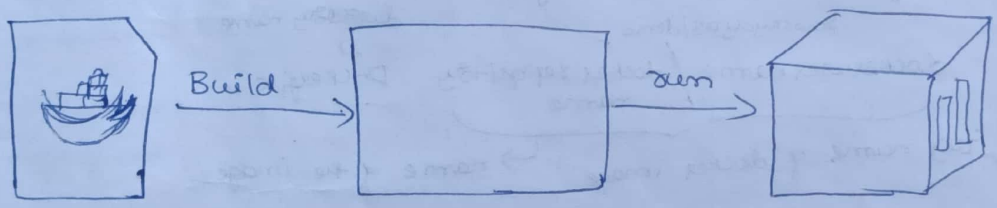
=> # java --version -> to check the version of java

=> # sh startup.sh.

Server is started now

TASK 5

Docker file



Docker file : It is a special text file that contains a set of instruction for building a docker image

=> Docker files are used to automate the process of creating docker images

=> Connect to mobaxterm terminal.

=> # sudo su -

=> # vi docker.sh -> Installation of docker

i-> Paste the scripts

esc :wq

=> # sh docker.sh

=> # mkdir directory name Dockerfile

=> # cd Dockerfile

vi dockerfile → filename

i

```
FROM amazonlinux
MAINTAINER "sowmyayp8@gmail.com"
RUN yum update -y
RUN yum install -y wget
RUN yum install -y tar
RUN yum install -y gzip
RUN wget paste tomcat server url → (chrome → tomcat server version 9 → copy tar.gz link address & Paste it
RUN tar -zxvf file name
        ↳ apache-tomcat-9.0.83.tar.gz
RUN yum install -y java
RUN sh apache-tomcat-9.0.83/bin/startup.sh
```

esc :wq

=> # cd .. → exit from the directory

first create one repository in dockerhub before doing this

=> # docker build -t dockerusername/docker repository name (sowmyayp8/demo) directory name → Dockerfile

↳ tag name of docker image ↳ name of the image

=> # docker run -it -d customized image name

=> # docker PS dockerusername/repository name → ex: sowmyayp8/demo

=> # docker exec -it containerid /bin/bash

=> # ls

=> # cd apache-tomcat-9.0.83

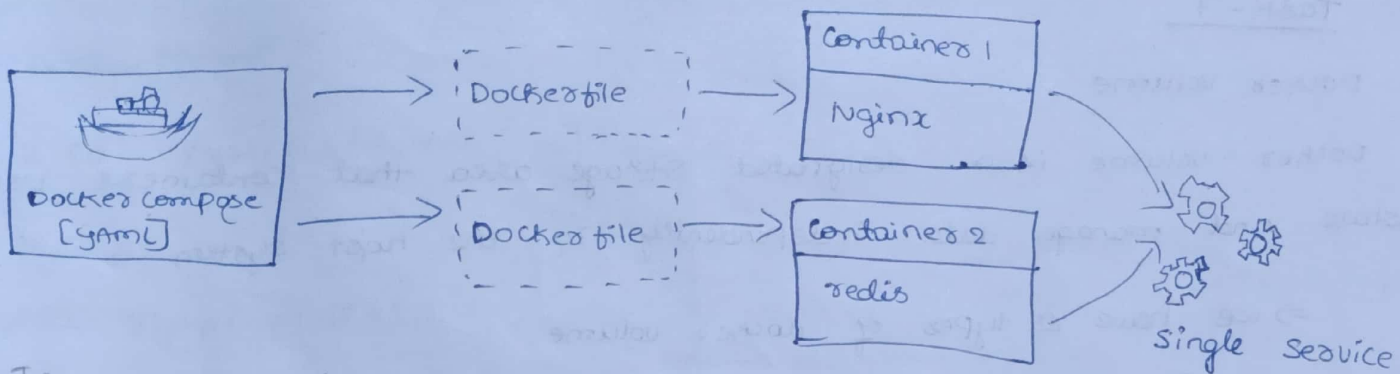
=> # cd bin/

=> # sh startup.sh

server is started.

Docker compose

In web application we have mainly 3 servers



In web application we have mainly 3 servers, in that the data will be collected from web server and the data is stored in data base server, In this case we cannot start both servers at diffe-rent time because in this time anyone can access those, for that we need to start both server at a time, for this we use Docker compose.

Docker compose is a tool which is used to run multiple containers as a single service

Yml script

```
version : '3'
services :
  web :
    image : nginx
    ports :
      - 4000 : 80

  db :
    image : redis
```

indentation: 2 words gap.

=> # vi docker-compose.yml

i -> insert yml script esc :wq

=> # docker-compose up -d -> to pull the images & run the containers

=> # docker-compose down -> to stop & delete the containers

=> # docker rmi nginx redis

=> # rm docker-compose.yml -> remove the file

Task - 7

Docker Volume

Docker volume is a designated storage area that containers share and manage data independently in the host system.

=> we have 2 types of docker volume

(i) Anonymous volume
(ii) named volume

=> # docker volume create volumename → to create volume
sigma

=> # cd /var/lib/docker/

=> # ls → show directories

=> # cd volumes/

=> # ls : Show the named volume

=> # docker volume ls : to check the volumes

=> # docker volume inspect volumename (sigma) → to inspect the named volumes

=> # docker pull httpd

=> # docker run -it -d httpd

=> # docker PS

=> # docker exec -it containerid /bin/bash

=> # ls

=> # cd htdocs/

=> # ls

=> # cat index.html

=> exit

=> # docker stop containerid [running container]

=> # docker rm containerid

=> # duplicate the terminal → right click → duplicate terminal

=> # docker run -it -d -v sigma:/usr/local/apache2/htdocs/ -p 8000:80 httpd

=> # docker PS

=> # docker exec -it containerid /bin/bash

~~=> # cd ..~~

=> # cd /var/lib/docker/volumes

=> # ls

=> # cd sigma/ => # ls

=> # cd -data

=> # touch sample.conf

=> # ls
index.html sample.conf.

Linux

Operating System :- An operating System is an Interface between computer user and computer hardware.

Linux :- Linux is a open source operating system which is a multi user and multithreaded.

It is created by linux torvalds.

Note: Linux OS is a community developed.

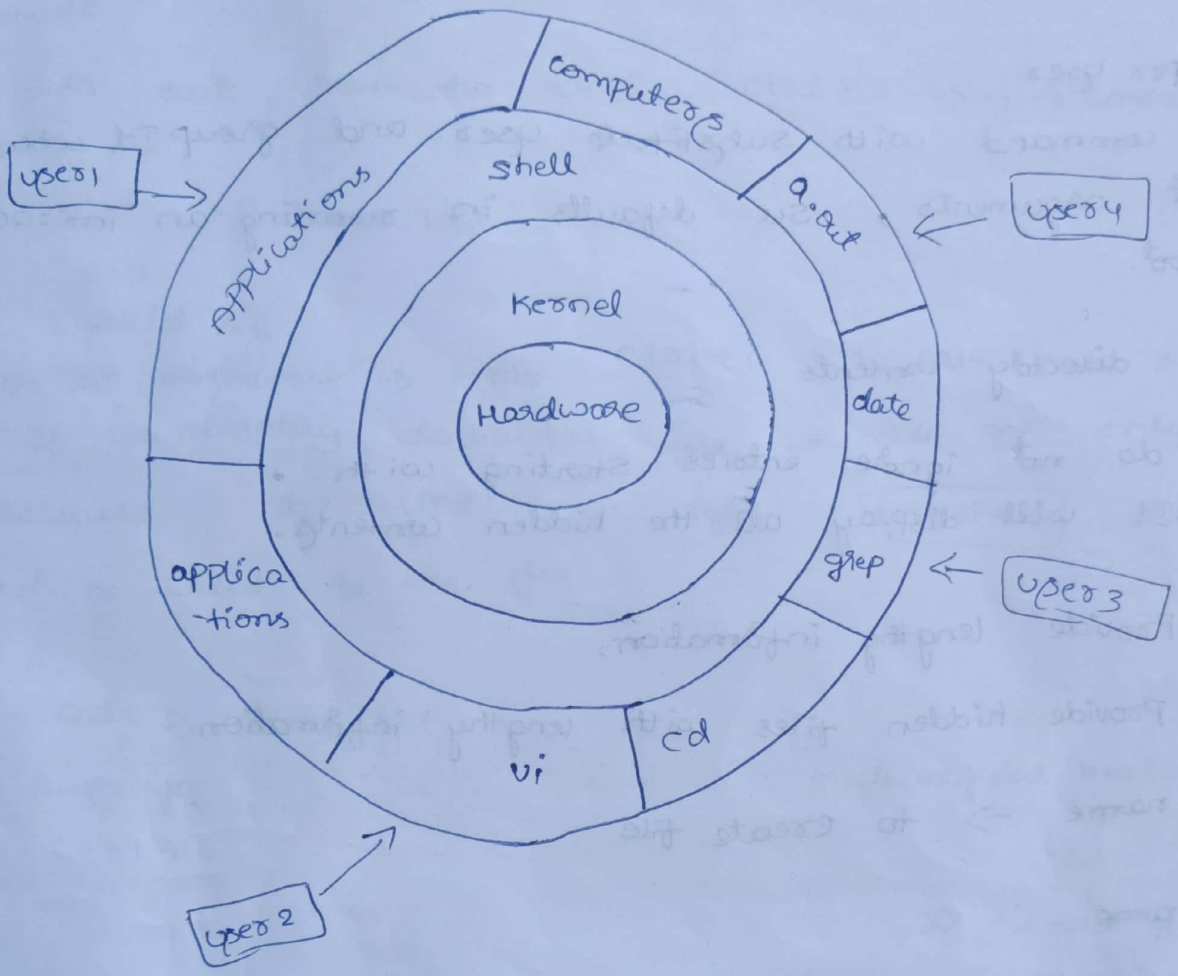
It is used as a foundation for various operating system called linux distributions

1. ubuntu
2. fedora
3. debian
4. centos
5. kali linux
6. RHEL [Red Hat Enterprise Linux]

Difference between windows and linux OS.

windows	linux
* It is an proprietary and requires a license	* It is an open-source and free
* Developed by microsoft, with versions like windows 10 and windows server.	* It is kernel-based, with various distributions like ubuntu, centos and debian
* Graphical user Interface (GUI) centric, with a traditional desktop environment	* Terminal/command-line oriented, offering powerful scripting capabilities
* User Account Control (UAC) for security; but generally considered more vulnerable to malware	* Robust security model with user-based permissions
* widely used for personal computers and dominant in the desktop operating system market	* Excellent stability and performance often used for servers and development environments
* Extensive software support, especially for commercial applications and games.	* supports a wide range of programming languages and development tools

Linux Architecture



Hardware : Hardware consists of all system devices like RAM, ROM CPU etc

Kernel :- Kernel is the core part of linux operating system it establishes communication between devices and softwares.

Shell :- Shell is a command line Interface that allows users to interact with the operating system by commands.

Commands :- Command is a program (or) utility that runs on the command line

Some commands of linux

① sudo su -

su → super user

Run a command with substitute user and group Id when called without arguments, su defaults in running an interactive shell as root.

② ls → list directory contents

③ ls -a → do not ignore entries starting with .
It will display all the hidden contents.

④ ls -l → Provide lengthy information.

⑤ ls -al → Provide hidden files with lengthy information

⑥ touch file name → to create file

⑦ cat file name

⑧ file editors to insert data into file
- ① vi/vim
- ② nano
- ③ cat

⑨ Vim sample.txt

⇒ Vim - Vi improved, a programmer's text editor

⑩ esc :q! → quitting editor without saving that file

esc :q → just quitting from the editor

rm filename → to remove the file

⑫ Pwd → Provide present working directory

⑬ date

⑭ cal → to display the calender

(15) ~~nano~~ nano demo.txt

directly write the text no need of typing i for insert mode

=> to exit from the editor ctrl + x -> y -> enter.

(16) cat >> demo.txt ↵

Hello

world ^C

o/p: cat demo.txt -> Hello ctrl + c -> to exit from the editor.

Note: If we display demo.txt file we will get only Hello because the line we type ctrl + c that line will not be added to the file

(17) cat > demo.txt

supriya

ctrl + c

single angular brace > -> override the existing data

>> => add the data to the previous data

(18) mkdir directory name => to create the directory.

(19) cd directoryname/ => /-> represents as directory.

(20) cd .. -> to come out of the directory.

(21) rmdir directory name -> to remove the empty directory.

(22) rm -rf sample/ -> -rf -> recursively & forcefully removing the directory.

~ -> represents root user as a root directory

(23) ~~mkdir sample~~

24

23 mkdir sample

24 ls

25 cat > jsp.txt

Devops

^c

26 cat jsp.txt

27 cp jsp.txt sample/ → copy jsp.txt to sample directory

28 rm jsp.txt → remove the file

29 mv sample/jsp.txt ./ → to move the file from sample directory to current directory

↳ represents current directory

30 mv jsp.txt 2sp.txt → rename jsp.txt to 2sp.txt

Default linux directories

/bin → core system commands

/etc → configuration files

/home → user home folders

/var → variable data (logs, temp files)

/tmp → Temporary files

/dev → device files

/Proc → Process and System info.

/sys → Kernel-related info

/usr → User-related files

/lib → System libraries

/boot → Boot related files

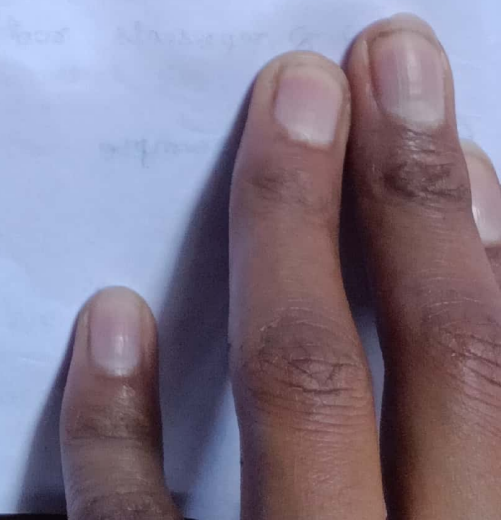
/mnt → Temp mounts

/opt → optional software

/srv → Service data

/root → Root users home

/run → Runtime files.



uname : To get information about my systems and operating system, machine, release, version.

$uname : to know about kernel

$uname -o : to get to know about OS

$uname -r : to know about kernel release

$uname -v : To know about version of kernel

$uname -m : to know about a machine

$uname -a : To get all the information

File permission

=> ls -l

drwx r-x r-x

- rwx rwx rwx

owner group others.

- & d -> file type d -> directory

- -> file

r -> read -> 4

w -> write -> 2

x -> execute -> 1

① rwx - - - - - -

file permission -ion 700.

② rw- rw- rw-

420 420 420

=> 6 6 6

③ -w- -w- -w-

02- 020 020

=> 2 2 2

④ - - x - - x - - x

001 001 001

=> 1 1 1

⑤ r-x r- - - x

401 400 001

=> 5 4 1

chmod : To change the file modes

To set the permission we use chmod

Ex: ① chmod 777 Sample.txt.

change mode (Permission) permissions filename.

② chmod u=rx, g+w, o+x sample.txt

=> ls -l

③ chmod +x sample.txt -> to add execute permission to all users

=> ls -l

④ chmod -x sample.txt => to remove execute permission for all users

=> ls -l

user management.

① # useradd dev ⇒ to create the user.

② # cat /etc/passwd ⇒ To know whether the user is created or not.

③ # passwd dev ⇒ To set the password for the user

New password: ____

④ # su - dev ⇒ To switch from root user to dev user.

exit ⇒ to come out from dev user

⑤ # userdel dev

⑥ # cat /etc/passwd.

group management.

① # groupadd Devops ⇒ To create the group.

② # cat /etc/group

③ # userdel -r dev ⇒ recursively delete the user

③ # useradd dev.

④ # cat /etc/passwd

To add a user into group

⑤ $ usermod -a -G DevOps dev.

groupname ↳ username

Remove a user from group.

⑥ $ gpasswd -d dev DevOps

To delete group.

⑦ $ groupdel DevOps

To display who is a member of a group

⑧ $ getent group DevOps

Compressed file formats.

TO compress.

① tar [tape archive] :- $ tar -cvf <file-tar> <file1 file2 file3>

c: create v: verbosely f: file

② TO extract.

=> $ tar -xvf <file-tar>

ex: extract v: verbosely f: file

③ zip : Package and compress (archive) files

Create : $ zip <file.zip> <file1 file2 file3>

Extract : $ unzip <file-name>

④ gzip : for compressing an individual file.

create : $ gzip <file>

Extract : $ gzip -d <file-gz>

all compressed file commands

=> touch a.txt b.txt c.txt

=> ls

=> tar -cvf jsp.tar

Networking Commands

1. ifconfig : configure a network interface
2. Ping : Send ICMP, ECHO-REQUEST to network hosts.
3. traceroute : Print the route packets trace to network host
4. host : DNS lookup utility
5. nslookup : query Internet name servers interactively
6. ip-address : protocol address management
7. netstat : Print network connections, routing tables, interface statistics
8. ss : another utility to investigate sockets
9. hostname : Show & set the system's host name
10. route : show/ manipulate the IP routing table
11. wget : the non-interactive network downloader
12. curl : transfer a URL

Filter commands

1. $ ifconfig
2. $ ping www.google.com
3. $ traceroute www.google.com
4. $ netstat -tuln
5. $ ss -tuln
6. $ hostname
7. $ route -n
8. $ nslookup flipkart.com
9. $ host www.flipkart.com
10. $ curl -O apache-tomcat.url => same like wget

storage/Disc utility commands.

① fdisk : manipulate disk Partition table.

② lsblk : list block devices.

③ df : report file system disk space usage.

④ du : estimate file space usage.

Commands

$fdisk /dev/xvda1 => volume partitions.

$parted -l

$lsblk.

$du

$df

① $fdisk -d => delete partition

$fdisk -n => add new partition

$fdisk -t => change partition type

$fdisk -i => print information about partition

② $df -h => Information about files in human readable format

③ $du -h => Information about file capacity.

-h -> human readable format

-c => dereference arguments.

④ $uptime => will provide information about server, how much load it is taking, how long it is in running, how much storage it is consuming.

Filter commands

head -> output the first part of files.
tail -> output the last part of files.
uniq -> report & omit repeated lines.
sort -> sort lines of text files.
find => search for files in directory hierarchy.
grep -> Print lines matching a Pattern.
wc -> print newline, word, and byte counts for each file.

① $head Sample.txt => displays first 10 lines of data in the file

② $head -n 5 sample.txt => displays first 5 lines

③ $head -4 Sample.txt => displays first 4 lines.

④ $tail sample.txt => displays last 10 lines of data in the file.

⑤ $tail -n 5 sample.txt

⑥ $tail -4 Sample.txt.

⑦ $sort demo.txt => to sort the contents of file

⑧ $sort -r demo.txt => to sort in reverse order

⑨ Piping commands.

cmd1 | cmd2 | cmd3

⑩ $head -7 Sample.txt | tail -3

⑪ $uniq -c jsp.txt -> count of occurence of duplicates.

⑫ $uniq -u jsp.txt => printing only unique lines.

⑬ $uniq -d jsp.txt => printing only duplicate lines.

⑭ $uniq -cd jsp.txt => Print only the count for duplicate lines.

⑮ $uniq -cu jsp.txt => Print only the count for unique lines.

(16) find command.

① $find -name <file-name> => search for file in user directories

② $find <from directory> -name <file-name> => search for in given directory.
$find ./ -name sample.txt.

③ $find <from-directory> -empty. => search for empty files
$find ./ -empty

④ $find <from-directory> -Perm <num permission => search for file with entered Permission
$find ./ -Perm 664

(17) grep command. => pattern matching.

① $grep "error" filename.log => search for lines containing word "error" in a file.

② $grep -i "error" filename.log => Search for lines containing the word "error" [case -insensitive] in a file.
[case Insensitive Search].

③ $grep -w "error" filename.log => search for lines containing the word "error" as a whole word [not as part of another word] in a file. [match whole words]

④ $grep -v "success" filename.log => search for ~~files~~ lines not containing the word "success" in a file
[Invert match]

⑤ $grep -r "warning" /Path/to/directory/ => Recursively search for lines containing the word "warning" in all files within a directory and its subdirectories.
Ex:- $grep -r "error" ./
[Recursive search]

⑥ $grep -c "hello" file.txt ⇒ count the number of lines containing the word "hello" in a file

⑦ $grep -E "error|warning" filename.log ⇒ search for lines containing either "error" (or) "warning" in a file

⇒ cat ./sample/demo.txt ⇒ to display content of demo.txt in sample directory from current directory

⑱ wc → word count

① wc -l filename.txt ⇒ count the no of lines in a file

② wc -w filename.txt ⇒ count the number of words in a file.

③ wc -c filename.txt ⇒ count the number of characters [including spaces] in a file

④ wc sample.txt

o/p: 1 1 4 sample.txt

1 → no of lines
1 → no of words
4 → no of characters

Shell scripting.

Shell scripting will follow only on indentation.

=># vi Sample.sh

i

#!/bin/bash → default shell in LinuxOS

→ It contains all Linux commands which are present in bin and will be executed in bash cell

```
echo "welcome to devops"
mkdir demo
touch jsp.txt
ls
whoami
uptime
```

#! Shebang => It is a simple command which is used for interpretation.

=> sh Sample.sh => to execute the script.

=> # Ex: user defined variable

① # $vi sample.sh

```
#!/bin/bash
name="Sowmya"
echo "Student name is $name"
echo 'Student name is $name'
```

Types of variables

① User-defined variables

② Environment variables

③ System variables.

Environment Variables

Environment variables are the special variables that holds information about system environment.

① $HOME : This variable represent the home directory of the current user

Ex:- $echo "your home directory is : $HOME"

② $USER : It stores the user name of the current user

Ex:- $echo "Hello, $USER!"

③ $PATH : This variable contains a colon-separated list of directories that the shell searches for executable programs.

Ex:- $echo "your PATH is : $PATH"

④ $PWD: This variable holds the present working directory

Ex:- $echo "you are in : $PWD"

⑤ $SHELL : It contains the path to the current shell interpreter

Ex:- $echo "your shell is located at : $SHELL"

⑥ $TERM : It holds the terminal type being used.

Ex:- $echo "your terminal type is : $TERM"

⇒ vi sample.sh

```
#!/bin/bash
$echo "your home directory is : $HOME"
$ echo "Hello, $USER!"
$echo "your path is : $PATH"
$echo "you are in : $PWD"
$echo "your shell is located at : $SHELL"
$echo "your terminal type is : $TERM"
```

esc :wq

o/p: your home directory is: /root
Hello, root!
your path is: /usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/root/bin
you are in: /root
your shell is located at: /bin/bash
your terminal type is: xterm.

System Variables

System variables are the subset of environment variable that provide information about the system's configuration.

① $HOSTNAME: stores the hostname of the system.
ex:- $echo "hostname: $HOSTNAME"

② $OSTYPE: Represents the operating system type
Ex:- echo "operating system type: $OSTYPE"

③ $MACHTYPE: Indicate the machine architecture
Ex:- echo "machine architecture: $MACHTYPE"

④ $LOGNAME: stores the login name of the user
ex:- echo "your login name is: $LOGNAME"

⇒ vi cat.sh → i

```
#!/bin/bash
$echo "hostname: $HOSTNAME"
$echo "operating system type: $OSTYPE"
$echo "machine architecture: $MACHTYPE"
$echo "your login name is: $LOGNAME"
```

esc :wq

o/p:
hostname: shell-scripting
operating system type: linux-gnu
machine architecture: x86-64-koji-linux-gnu
your login name is: root

methods

syntax :

=> vi method.sh

```
=> #!/bin/bash
   methodname()
   {
        echo "Hello world"
   }
   methodname
```

O/P : Hello world

① vi method1.sh

```
=> #!/bin/bash
   method()
   {
        echo "my name is Soumya"
   }
   method
```

=> sh method1.sh

② vi method2.sh

```
=> #!/bin/bash
   method()
   {
        echo "
             I am from Sira"
   }
   method
```

③ vi method3.sh

```
=> #!/bin/bash
   method()
   {
        echo "currently staying in Banglore"
   }
   method.
```

④ vi method4.sh

=>#!/bin/bash

```
method ()
{
        echo "Done my master degree in MCA at Banglore Institute of
                Technology"
}
method.
```

⑤ vi method5.sh

=>#!/bin/bash

```
method ()
{
    echo "Affiliated Visvesvaiah technological university, Belgavi"
}
method
```

⑥ vi method6.sh

=>#!/bin/bash

```
method ()
{
    echo "done 2 Projects during academics"
}
method
```

⑦ vi method7.sh

=>#!/bin/bash

```
method()
{
    echo "one is mini project and another one is major Project"
}
method
```

(8) vi method8.sh

```
#!/bin/bash
method8()
{
    echo "mini project is based on IOT"
}
method
```

(9) vi method9.sh

```
#!/bin/bash
method9()
{
    echo "mini project title: Solar Tracking System"
}
method9
```

(10) vi method10.sh

```
#!/bin/bash
method10()
{
    echo "major project is based on cloud"
}
method10
```

(11) vi method11.sh

```
#!/bin/bash
method11()
{
    echo "major project title: Global commercial domain designer with reports"
}
method11
```

(12) vi method12.sh

```
#!/bin/bash
method12()
{
    echo "I am interested in sports"
}
method12
```

(13) vi method13.sh

```
#!/bin/bash
method()
{
    echo "willing to learn and grow"
}
method
```

(14) vi method14.sh

```
#!/bin/bash
method()
{
    echo "I love to drink tea"
}
method
```

(15) vi method15.sh

```
#!/bin/bash
method()
{
    echo "learn quickly"
}
method
```

(16) vi method16.sh

```
#!/bin/bash
method()
{
    echo "I know java"
}
method
```

(17) vi method17.sh

```
#!/bin/bash
method()
{
    echo "I know HTML and CSS"
}
method
```

(18) vi method18.sh

```
#!/bin/bash
method()
{
    echo "I know sql"
}
method
```

(19) vi method19.sh

```
#!/bin/bash
method()
{
    echo "I know git"
}
method
```

(20) vi method20.sh

```
#!/bin/bash
method()
{
    echo "I know dockes"
}
method
```

(21) vi method21.sh

```
#!/bin/bash
method()
{
    echo "I know about some AWS services"
}
method
```

(22) vi method22.sh

```
#!/bin/bash
method()
{
    echo "I love trucking"
}
method
```

(23) vi method23.sh

```
#!/bin/bash
method()
{
    echo "I like travelling"
}
method
```

(24) vi method24.sh.

```
#!/bin/bash
method()
{
    echo "I to know IDBC"
}
method
```

(25) vi method25.sh

```
#!/bin/bash
method()
{
    echo "I love. Pets"
}
method
```

(26) vi method26.sh

```
#!/bin/bash
method()
{
    echo "I have participated in Intercollege coding competition"
}
method
```

(27) vi method27.sh

```
#!/bin/bash
method()
{
    echo "I won 2nd Place in district level football sports"
}
method
```

(28) #!/ vi method28.sh

=>
```
#!/bin/bash
method()
{
    echo " Till now I don't have any weakiness"
}
method
```

(29) vi method29.sh

=>
```
#!/bin/bash
method()
{
    echo " If I found in future I will assure"
}
method
```

(30) vi method30.sh

=>
```
#!/bin/bash
method()
{
    echo " It can't affect my career"
}
method
```

write a program to add two numbers by using shell script

=> vi Sum.sh

=>
```
#!/bin/bash
Sum()
{
    a=10
    b=20
    echo "a value is $a"
    echo "b value is $b"
    c=$((a+b))
    echo "Sum is $c"
}
Sum
```

programs for performing arithmetic operation

static input for 2 variables

1) vi sub.sh

```
#!/bin/bash
sub()
{
    a=10;
    b=20
    echo " a is $a"
    echo "b is $b"
    c=$((a-b))
    echo "subtraction is $c"
}
sub
```

2) vi mul.sh

```
#!/bin/bash
mul()
{
    a=10
    b=20
    c=$((a*b))
    echo "multiplication of 2 nos is $c"
}
mul
```

3) vi div.sh

```
#!/bin/bash
div()
{
   a=10
   b=20
   c=$((a/b))
   echo " division is $c"
}
div
```

dynamic Input for 2 variables.

① vi add.sh

```
#!/bin/bash
add()
{
    echo "Enter a value"
    read a
    echo "Enter b value"
    read b
    echo "a & b value is $a $b"
    c=$((a+b))
    echo "Sum is $c"
}
add
```

② vi Sub.sh

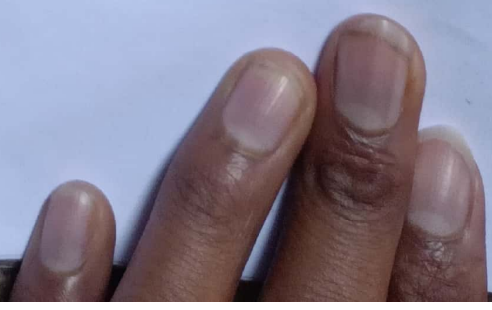
```
#!/bin/bash
sub()
{
    echo "Enter a value"
    read a
    echo "Enter b value"
    read b
    echo "a & b value is $a $b"
    c=$((a-b))
    echo "difference is $c"
}
sub
```

③ vi mul.sh

```
#!/bin/bash
mul()
{
    echo "Enter a value"
    read a
    echo "Enter b value"
    read b
    c=$((a*b))
    echo "Product: $c"
}
mul
```

④ vi div.sh

```
#!/bin/bash
div()
{
    echo "Enter a & b value"
    read a
    read b
    echo "a & b value is $a $
    c=$((a/b))
    echo "division is $c"
}
div
```



Static Input for 3 variables.

① vi add.sh

```
#!/bin/bash
add()
{
    a=10
    b=20
    c=30
    echo " a,b & c value is $a $b & $c"
    d=$((a+b+c))
    echo " sum is $d"
}
add.
```

② vi sub.sh

```
#!/bin/bash
sub()
{
    a=10
    b=20
    c=30
    echo " a, b & c values are $a $b $c"
    d=$((a-b-c))
    echo " difference is $d"
}
sub.
```

③ vi mul.sh

```
#!/bin/bash
mul()
{
    a=10
    b=20
    c=30
    echo " a,b & c value is $a $b $c"
    d=$((a*b*c))
    echo " product is $d"
}
mul
```

④ vi div.sh

```
#!/bin/bash
div()
{
    a=10
    b=20
    c=30
    echo "a,b & c value i
    $a $b $c"
    d=$((a/b/c))
    echo " division is : $d"
}
div.
```

dynamic Input for 3 variables

① #!/bin/bash
```
add()
{
    echo "enter a, b & C values"
    read a
    read b
    read c
    echo "a, b & C value is $a $b $C"
    d=$((a+b+c))
    echo "sum is : $d"
}
add
```

② vi sub.sh
```
#!/bin/bash
sub()
{
    echo "enter a, b & C values"
    read a
    read b
    read c
    echo "a, b & C value is $a $b $C"
    d=$((a-b-c))
    echo "difference is : $d"
}
sub
```

③ vi mul.sh
```
#!/bin/bash
mul()
{
    echo "enter a, b & C values"
    read a
    read b
    read c
    echo "a, b & C values are $a $b $C"
    d=$((a*b*c))
    echo "product is : $d"
}
mul
```

④ vi div.sh
```
#!/bin/bash
div()
{
    echo "enter a, b & C values
    read a
    read b
    read c
    echo "a, b & C values are
        $a $b $C"
    d=$((a/b/c))
    echo "division is $d"
}
div.
```

Note:- Special variables $1, $2, $3 --- these variables are used when we are passing arguments to the method.

static input ⇒ 2 variables by using ↑ passing arguments to the method

① vi add.sh

```
#!/bin/bash
add()
{
  echo "a value is $1"
  echo "b value is $2"
  c=$((($1 + $2))
  echo "the sum of a & b = $c"
}
add 2 3
```

② vi sub.sh

```
#!/bin/bash
sub()
{
  echo "a value is $1"
  echo "b value is $2"
  c=$((($1 - $2))
  echo "difference: $c"
}
sub 10 15
```

③ vi mul.sh

```
#!/bin/bash
mul()
{
    echo "a value is $1"
    echo "b value is $2"
    c=$(($1 * $2))
    echo "Product : $c"
}
mul 10 20
```

④ vi div.sh

```
#!/bin/bash
div()
{
  echo "a value is $1"
  echo "b value is $2"
  c=$(($1/$2))
  echo "division : $c"
}
div 10 2
```

dynamic input for 2 variables by using passing arguments.

① vi add.sh

```
#!/bin/bash
add()
{
    echo "a value is $1"
    echo "b value is $2"
    c=$((($1 + $2))
    echo "addition is $c"
}
echo "Enter 2 numbers"
read a
read b
add a b
```

② vi sub.sh

```
#!/bin/bash
sub()
{
    echo "a value is $1"
    echo "b value is $2"
    c=$((($1 - $2))
    echo "difference is $c"
}
echo "Enter 2 numbers"
read a
read b
sub a b
```

③ vi mul.sh

```
#!/bin/bash
mul()
{
    echo "a value is $1"
    echo "b value is $2"
    c=$((($1 * $2))
    echo "product is $c"
}
echo "Enter 2 nos"
read a
read b
mul a b
```

④ vi div.sh

```
#!/bin/bash
div()
{
    echo "a value is $1"
    echo "b value is $2"
    c=$((($1 / $2))
    echo "division $c"
}
echo "Enter 2 numbers"
read a
read b
div a b
```

static input for 3 variables by using passing arguments to the method

① vi add.sh

```
#!/bin/bash
add()
{
  echo "a value is $1"
  echo "b value is $2"
  echo "c value is $3"
  d=$((($1+$2+$3))
  echo "sum is $d"
}
add 2 3 4
```

② vi sub.sh

```
#!/bin/bash
sub()
{
  echo "a value is $1"
  echo "b value is $2"
  echo "c value is $3"
  d=$((($1-$2-$3))
  echo "difference is $d"
}
sub 10 20 30
```

③ vi mul.sh

```
#!/bin/bash
mul()
{
  echo "a value is $1"
  echo "b value is $2"
  echo "c value is $3"
  d=$((($1*$2*$3))
  echo "Product is $d"
}
mul 10 20 30
```

④ vi div.sh

```
#!/bin/bash
div()
{
  echo "a value is $1"
  echo "b value is $2"
  echo "c value is $3"
  d=$((($1/$2/$3))
  echo "division $d"
}
div 10 20 30
```

dynamic Input

① vi add.sh

```
#!/bin/bash
add()
{
  echo "a value is $1"
  echo "b value is $2"
  echo "c value is $3"
  d=$((($1+$2+$3))
  echo "sum: $d"
}
echo "Enter 3 numbers"
read a
read b
read c
add a b c
sub
mul
div
```

variables

2 types

① local variable

② global variable

```
Pkg = tomcat                    # global variable
install ()
{
        echo " install $Pkg"
        echo "
                function name is $FUNCNAME"
}

config()
{
        Pkg = nginx                     # local variable
        echo " config the $Pkg"
        echo "
                function name is $FUNCNAME"
}

install
config
```

Conditional and Looping statements :-

operators

① -eq → equality condition => [$a -eq $b]

② -ne → not equal => [$a -ne $b]

③ -gt → greater than => [$a -gt $b]

④ -ge → greater than @ equal to => [$a -ge $b]

⑤ -lt → less than => [$a -lt $b]

⑥ -le → less than @ equal to => [$a -le $b]

⑦ if condition

① -eq
equal to

```
#!/bin/bash
a=10
b=10
if [ $a -eq $b]
then
        echo " both are equal"
fi
```

```
#!/bin/bash
echo "enter a value"
read a
echo "enter b value"
read b
if [ $a -eq $b]
then
        echo " both are equal"
fi
```

② -ne
not equal

Static Input

```
#!/bin/bash
a=10
b=20
if [ $a -ne $b]
then
    echo " both are not equal"
fi
```

dynamic Input

```
#!/bin/bash
echo " enter a value"
read a
echo " enter b value"
read b
if [ $a -ne $b]
then
    echo "both are not equal
fi
```

③ -ge [greater than @ equal to]

```
#!/bin/bash
a=100
b=20
if [ $a -ge $b]
then
    echo " a is greater than @ equal
fi          to b"
```

```
#!/bin/bash
echo " enter a value"
read a
echo " enter b value"
read b
if [ $a -ge $b]
then
    echo " a is greater than@
fi          equal to b"
```

④ -gt [greater than]

```
#!/bin/bash
a=12
b=5
if [ $a -gt $b]
then
    echo " a is greater than b
fi
```

```
#!/bin/bash
echo " enter a value"
read a
echo " enter b value"
if [ $a -gt $b]
then
    echo " a greater than b"
fi
```

⑤ lt [less than]

```
#!/bin/bash
a=6
b=15
if [ $a -lt $b]
then
    echo " a is less than b"
fi
```

```
#!/bin/bash
echo "enter a value"
read a
echo "enter b value"
read b
if [ $a -lt $b]
then
    echo " a is lessthan b"
fi
```

⑥ le [less than @ equal to]

```
#!/bin/bash
a=4
b=12
if [ $a -le $b]
then
    echo " a is less than @ equal to b"
fi
```

```
#!/bin/bash
echo "enter a value"
read a
echo "enter b value"
read b
if [ $a -le $b]
then
    echo "a is less than @
    equal to b"
fi
```

II if-else Condition

① eq [equal to]

Static

```
#!/bin/bash
a=10
b=10
if [ $a -eq $b]
then
    echo " a is equal to b"
else
    echo " a is not equal to b"
fi
```

dynamic

```
#!/bin/bash
echo "enter a value"
read a
echo "enter b value"
read b
if [ $a -eq $b]
then
    echo " a is equal to b"
else
    echo " a is not equal to
    b"
fi
```

② -ne [not equal to]

```
#!/bin/bash
a=10
b=20
if [ $a -ne $b]
then
    echo " a is not equal to b"
else
    echo " a is equal to b"
fi
```

```
#!/bin/bash
echo "enter a value"
read a
echo " enter b value"
read b
if [ $a -ne $b]
then
    echo " a is not equal to b"
else
    echo " a is equal to b
```

③ le [less than @ equal to] & ge [greater than @ equal to]

```
#!/bin/bash
a=10
b=20
if [ $a -le $b]
then
    echo " a is less than @ equal to b"
else
    echo " a is not less than @ equal to b"
fi
```

(-ge: greater than)

```
#!/bin/bash.
Echo "enter a & b value"
read a
read b
if [ $a -le $b]
then
    echo " a is less than @ equal to b"
else
    echo "a is not less than @ equal to b"
fi
```

(-ge: greater than)

④ lt [less than] and gt [greater than]

```
#!/bin/bash
a=20
b=100
if [ $a -lt $b]
then
    echo " a is less than b"
else
    echo " a is not less than b"
fi
```

(-gt: greater than)

```
#!/bin/bash
echo "enter a & b value"
read a
read b
if [ $a -lt $b]
then
    echo " a is less than b"
else
    echo " a is not less than b"
```

(-gt: greater than)

III else if condition [else if ladder]

① eq [equal to and not equal to]

```
#!/bin/bash
a=10
b=20
if [ $a -eq $b ]
then
    echo " a is equal to b"
elif [ $a -ne $b ]
then
    echo " a is not equal to b"
else
    echo " invalid "
fi
```

```
#!/bin/bash
echo "enter a & b value"
read a
read b
if [ $a -eq $b ]
then
    echo "a is equal to b"
elif [ $a -ne $b ]
then
    echo " a is not equal to b"
else
    echo "invalid"
fi
```

② gt [greater than] and lt [less than]

```
#!/bin/bash
a=100
b=80
if [ $a -gt $b ]        (-ge @ equal to)
then
    echo " a is greater than b"
elif [ $a -lt $b ]      (-le @ equal to)
then
    echo " a is less than b"
else
    echo " Invalid"
fi
```

```
#!/bin/bash
echo " enter a & b value"
read a
read b
if [ $a -gt $b ]        (-ge @ equal to)
then
    echo " a is greater than b"
elif [ $a -lt $b ]      (-le @ equal to)
then
    echo " a is less than b"
else
    echo " invalid"
fi
```

for loop

① 1 - 10

```
#!/bin/bash
for num in {1..10}
do
    echo $num
done
```

② 10 - 1

```
#!/bin/bash
for num in {10..1}
do
    echo $num
done
```

③ print odd numbers from 1 to 10

```
#!/bin/bash
for num in {1..10..2}
do
    $num
done
```

odd nos from 10 to 1

```
#!/bin/bash
for odd in {9..1..2}
do
    $odd
done
```

④ even nos from 2 to 10

```
#!/bin/bash
for even in {2..10..2}
do
    echo $even
done
```

even nos from 10 to 2

```
#!/bin/bash
for even in {10..2..2}
do
    echo $even
done
```

⑤ Print the multiples of 2.

2 * 1 = 2 to 2 * 10 = 20

```
#!/bin/bash
a=2
for num in {1..10}
do
    echo "$a * $num = $(($a * $num))"
done
```

for loop in single line

```
for i in {1..10} ; do echo "number: $i" ; done
```

Infinite loop

```
while true ; do echo "hii" ; sleep 3 ; done
```

sleep: every second it will be repeated

sleep3 : for every 3 second it will be repeating

while loop

①
```
#!/bin/bash
init=3
while [ $init -le 10 ]
do
    echo "$init"
    ((init++))
done
```

In single line
```
#!/bin/bash
init=1 ;while [ $init -le 10 ];do echo $init ; ((init++));done
```

②
```
#!/bin/bash
init=5
while [ $init -ne 5]
do
    echo $init
    ((init++))
done
```

③
```
#!/bin/bash
init=10
while [ $init -eq 10]
do
    echo $init
    ((init++))
done
```

④
```
#!/bin/bash
a=5
while [ $a -gt 3]
do
    echo $a
    ((init--))
done
```

⑤
```
#!/bin/bash
a=0
while [ $a -lt 3]
do
    echo $a
    ((a++))
done
```

to check files using if condition
```
#!/bin/bash
if [ -f /root/demo.sh]
then
    echo "file exists"
else
    echo "file not exists"
fi
```

to check directories
```
#!/bin/bash
if [ -d "sak/"]
then
    echo "exists"
fi
```

Real time example using looping statement [for & while]

① to fetch list of all files in the directory

```
for file in <directory> /* ; do
    echo "file : $file"
done
```

② to count the files

count the no of files in directory

```
count=0
for file in <directory> /* ; do
    ((count++))
done
    echo "total files : $count"
```

③ to ~~remove~~ rename files

rename all files in directory

```
for file in <directory> /* ; do
  mv "$file" "${file}_new"
done
```

④ copy files

copy all files to another directory

```
destination="<destination-directory>"
for file in <directory> /* ; do
done
```

⑤ delete file

delete all file in the directory

```
for file in <directory> /* ; do
  rm "$file"
done
```

⑥ compress files:

compress all zip files in the directory into a zip archive

```
zip archive.zip <directory>/*
```

⑦ Extract zip files.

Extract all zip file in the directory

```
for file in <directory>/*.zip; do
    unzip "$file" -d <destination directory>
done.
```

⑧ check file types.

check and print the files [mime] of all files

```
for file in <directory>/* ; do
    file -b --mime "$file"
done
```

⑨ calculate file size:

Calculate and print the sizes of all files.

```
for file in <directory>/* ; do
    du -sh "$file"
done.
```

① List files (using ls)

List all files in directory using ls and while loop.

```
ls <directory> | while read file ; do
    echo "file : $file"
done
```

② Count files (using ls)

Count the number of files in directory using ls and while loop

```
count=0
ls <directory> | while read file ; do
    ((count++))
    echo "total count ls : $count"
done
```

③ Rename files.

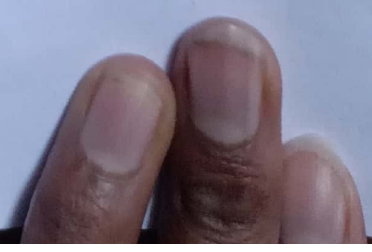
Rename all files in directory using a while loop.

```
ls <directory> | while read file ; do
    mv "$file" $(file)_new
done
```

④ Copy files.

Copy all files to another directory using a while loop.

```
destination = "destination directory"
ls <directory> | while read file
do
    cp "$file" "$destination/"
done
```



⑤ delete files.

delete all files in the directory using while loop

```
ls <directory> | while read file
do
    rm "$file"
done
```

⑥ check file range.

check and print the file types (MIME) of all files using a while loop.

```
ls <directory> | while read file ; do
    file -b --mime "$file"
done.
```

⑦ calculate file sizes.

calculate and print the sizes of all files using a while loop.

```
ls -lh <directory> | while read -r size file ; do
    echo "$size, file : $file"
done
```

Build tool

There are 3 types of build tool

gradle [It can be used for any technology]

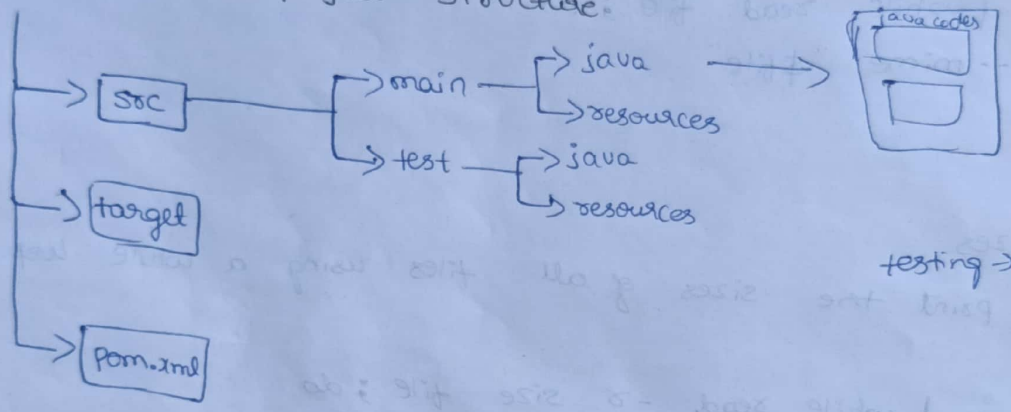
Ant [It is used for java]

maven [maven is a updated version of ant which will be based on Pom.xml]

maven

maven is a open source build management tool which is developed by apache software foundation and it can be used on java technology

Default maven project structure:



testing → unit testing → JUnit done by developers

Pom.xml :- Pom Stands for Project object model.

⇒ xml stands for extensible markup language and it is a heart of maven project without Pom.xml file we can't prepare any builds

⇒ Pom.xml Contains dependencies, Plugins and life cycles

dependencies :- dependencies are the external jar files [java libraries] that are used in projects.

Plugins :- Plugins are used to execute some group of goals.

maven repository :-

maven repositories are the directories of packaged jar files with some meta data

3 types of maven repositories we have

① local repository

② remote repository

③ central repository

① local repository :- maven local repository is located in your local system [laptop @ desktop]

② remote repository :- maven remote repository is located on the company server

③ central repository :- maven central repositories are located on the web.

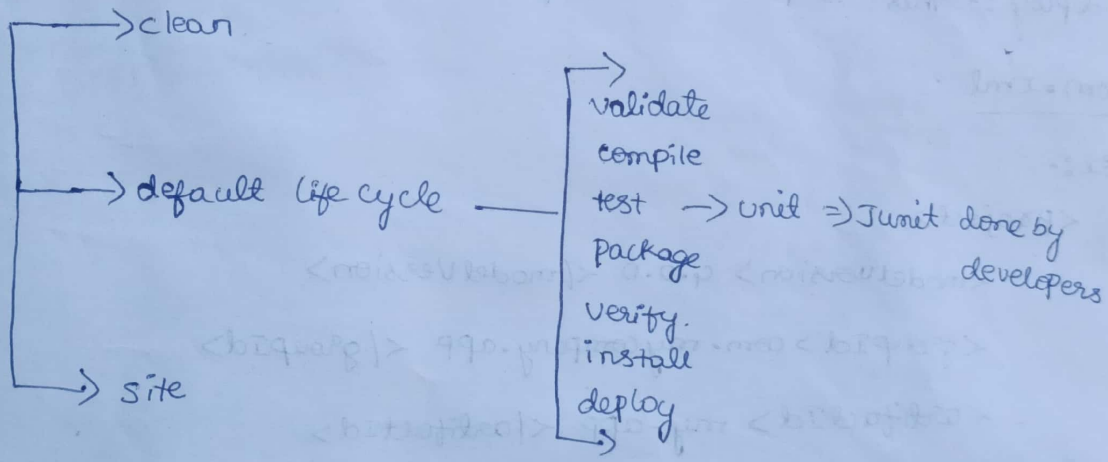
⇒ It is created and managed by apache software foundation

3 types of build life cycles we have

① clean life cycle

② default life cycle

③ site life cycle



Clean life cycle :- To clean the project and remove all the files generated by the previous build.

default life cycle :- It is also called main life cycle and it is responsible for project deployment.

Site life cycle :- To create the project @ documentation

Default life cycle :-

① validate :- This step will validate the maven project structu

⑤ compile :- This step will compile the source code.

③ test :- This step will test the source code by running unit test and it can be run by using Junit framework and this test done by developers.

④ Package :- This step will prepare the build which is in jar ② war format.

⑤ verify :- This step will verify the prepared build.

⑥ install :- This step will install build in the defined machine

⑦ deploy :- This step will deploy the installed build.

Pom.xml

Ex :-

```
<project>
    <modelVersion> 4.0.0 </modelVersion>
    <groupId> com.mycompany.app </groupId>
    <artifactId> my-app </artifactId>
    <version> 1 </version>
</project>
```

groupId :- This is an Id of the projects group generally we can call Organization ② company names.

artifact Id :- This is an Id of the project generally we can call it is a project name

version :- This is the version of the project

arche-type :- an arche-type is a maven-project templating tool kit

Installation of maven

System Requirements

AMI -> amazon linux 2

Instance type -> t2.micro

security group -> all-traffic

terminal -> mobaxterm

procedure to install maven

1. go to chrome -> download maven -> click on apache maven -> select download apache-maven-bin.tar.gz -> right click -> copy link address
2. open mobaxterm terminal
3. #wget Pastelinkaddress
4. # tar -zxvf Pastethe downloaded apache maven
5. #ls
6. #mv apache-maven-3-9.6 Maven
7. $ls
8. $ cd Maven/
9. $ls
10. $cd bin/
11. $Pwd -> copy the Pwd Path
12. cd
13. ls
14. vi .bashrc

:

after fi type

export PATH=/root/maven/bin:$PATH

15. mvn --version -> command not found because it is not active

make to active use the command source .bashrc

16. source .bashrc
17. mvn --version
18. sudo yum install java-11* -y.

19. java --version
20. mvn --version

First task

Prepare a build which is in jar format for the company called wipro and the project name is canara

① $mvn archetype:generate → to get the template

② archetype-2095 : remote → org.apache.maven.archetypes : maven archetype-quickstart (An archetype which contains a sample maven project)

choose 2095 : ~~type~~ type 2095

choose a number 8 : 8

value for property groupId : wipro

artifactId : canara

version : -

Package : -

Y : Y ↵

③ $ls

④ $cd canara/

⑤ $ls

⑥ $cat pom.xml

⑦ $cd src/

⑧ $ls

⑨ $cd main/

⑩ $cd java/

⑪ $ls

⑫ $cd wipro/

⑬ $ls

⑭ cat app.java

note :- all the maven commands should run inside the maven project.

1. $ < root@maven canara > $ maven validate
2. $ mvn clean
3. $ mvn compile
4. $ cd target/
5. $ ls
6. $ cd classes/
7. ls
8. $ cd wipro/
9. $ ls → app.class → after compile the class file is generated.
10. $ pwd. → cd.. come back to canara project
11. $ mvn test
12. $ ls.
13. $ cd target/
14. $ cd surfire-repaits
15. $ cd.. cd..
16. $ mvn Package
17. $ cd target
18. $ ls

    Here the jar file is generated
19. $ cd..
20. $ mvn verify
21. $ mvn install
22. $ cd..
23. $ ls -a
24. $ ls .m2/
25. $ cd repository/
26. $ cd wipro/
27. $ cd canara/
28. $ cd 1.0-SNAPSHOT/
29. $ cd
30. $ ls
31. $ cd canara/
32. $ mvn site
33. $ cd target/
34. $ cd site
35. $ ls
36. cat summary.html

Task 2

generate a war file with company name as test yantra and Project name as test freshers.

① ~~mvn~~ ~~arche-type : generate~~ mvn archetype : generate

choose a number : 2103

choose a no : 7

groupId : test yantra

artifact Id : test ~~yantra~~ freshers

: y

② $ cd testfreshers/

③ $ mvn validate

④ $ mvn clean

⑤ $ mvn compile

⑥ $ mvn test

⑦ $ mvn Package → the war file is generated

⑧ $ cd target/

⑨ $ ls → ~~here~~

⑩ $ cd ..

⑪ $ mvn verify

⑫ $ mvn install

⑬ $ mvn site

⑭ cd target/

⑮ ls

cd site

ls.

Task: generate war file and Push Pom.xml & src file to github repo -try in mobaxterm terminal

Steps: first generate war file

: install git in mobaxterm

⇒ sudo yum install git

⇒ git add pom.xml src ⇒ git commit -m "mi"

⇒ establish the connection to remote repository ⇒ [git remote add origin githuburl]

⇒ It will not be established directly so we need to create tokens for that

⇒ In github → go to profile → setting → developer settings → Personal access tokens → Tokens (classic) → there you have link in first line → click on that → provide name for token in note field → select scopes → select all → Generate token

→ After generating token you will get one Path

copy it paste it in notepad

copy the github repository url and paste in notepad combine both like below

after https:// paste token Path @ github url

⇒ ex :- https://ghp_Dr19 6G402H602uUXP82@github.com/sowmyaup8/maven.git

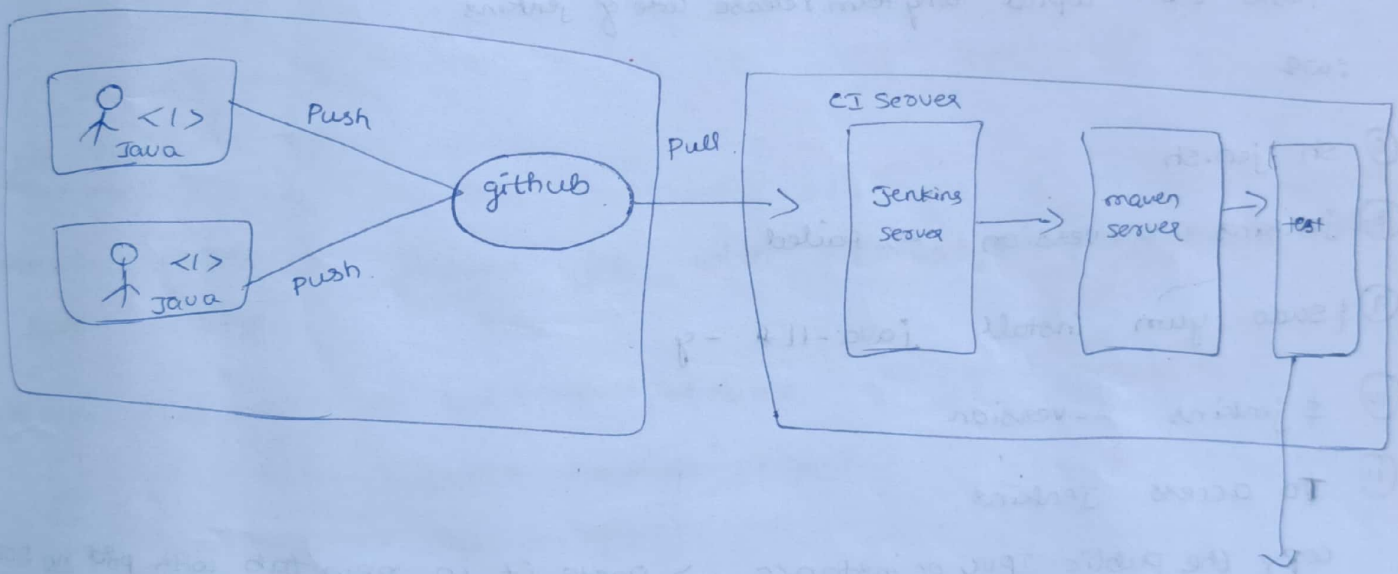
→ ~~Then git add pom.xml src~~ ⇒ git push origin master

Jenkins.

Jenkins is a open source continuous Integration tool that allows continuous development, testing and deployment of newly created codes.

Jenkins a is a opensource automation tool which will be used for software development life cycle.

Jenkins architecture



Installation of Jenkins.

System Requirements

1. AmI → linux 2
2. Instancetype → t2.micro
3. Security group → all-traffic
4. terminal → mobaxterm

Procedure to install Jenkins

① go to chrome → type download jenkins

② click on ~~select~~ second link → Linux

③ select redhat link → click on long term release → copy the entire code in long term release

④ In mobaxterm terminal type

⑤ vi jen.sh
i
Paste the copied long term release code of jenkins
:wq

⑥ sh jen.sh

⑧ jenkins --version → Failed

⑨ $ sudo yum install java-11* -y

⑩ $ jenkins --version

⑪ To access Jenkins
copy the Public IPv4 of instance → paste it in new tab with port no 8080
Ex: 13.44.132.80:8080

Note :- 8080 is a default port number for jenkins

⑫ $ systemctl start jenkins

⑬ come to the tab where you pasted the Public IPv4. there you have the Path of jenkins copy the Path

⑭ $ sudo cat paste the path → /var/lib/jenkins/secrets/initialA
It will generate the password.

⑮ copy the password and paste in the password field in the tab → continue

⑯ click on Install suggested plugins
wait until all get started

(14) Create First Admin User

Username : Admin
password : Admin
confirmPassword : Admin
full name : Admin
Email address : Admin@gmail.com

Save & Continue

Save & finish

(15) click on Start using jenkins

(16)

First Task

TASK 1 :- create a jenkins job with freestyle project to print hello devops.

1. click on new item → provide jobname
   → select freestyle project
   → click on OK

2. click on build steps
   ↳ add build steps
   ↳ Execute shell
   ↳ echo "hello devops"

   click on apply

   save

3. Start Project by clicking build now → there you can see the console output.

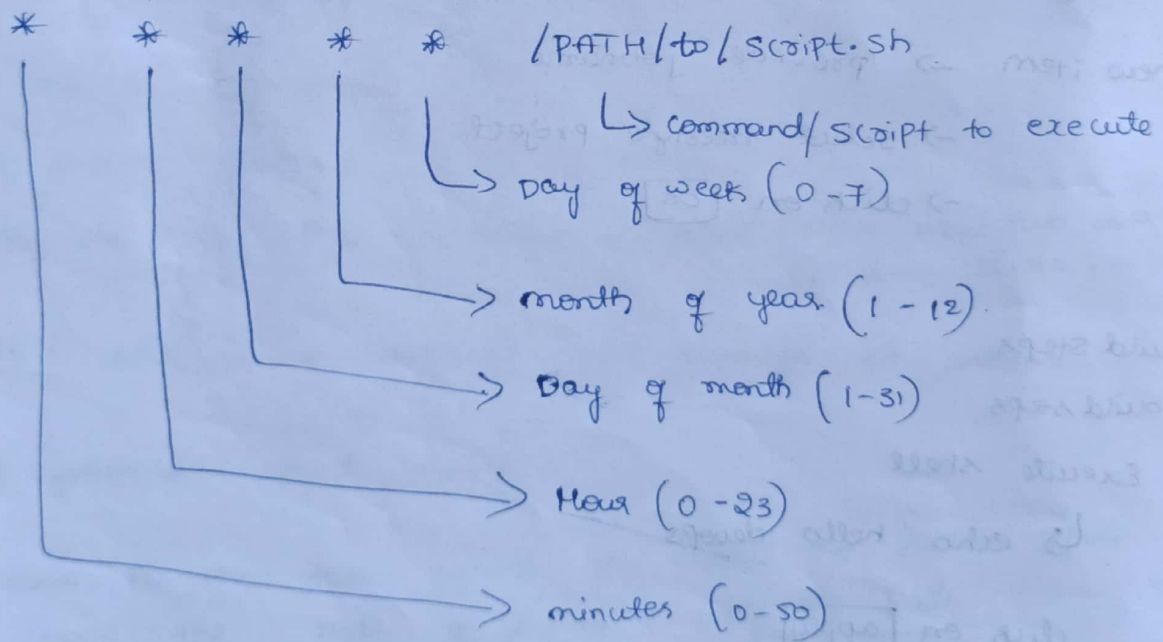
4. If you want to make any changes then
   => click on configure and then build steps then type code whatever you want apply & save

Task 2: create a job with freestyle project to get to know the uptime of server for each minute.

① click on new item
↳ provide job name
↳ select freestyle projects
↳ click on [ok]

② In build triggers
↳ select build periodically
↳ In that type * * * * *

Note:- Build periodically will follow on cron job will schedule the jobs for defined time interval



③ In Build steps type uptime

To stop the execution → click on configure → uncheck the trigger apply & save

Task 3: create a freestyle job to prepare a build.

① load the maven tool to jenkins.

⇒ click on manage jenkins

⇒ click on tools

⇒ maven Installation

add maven

⇒ provide maven name → maven

version → 3.9.6 (default)

check the checkbox

⇒ click on apply and ~~close~~. save

② create a freestyle job with the name maven project.

⇒ click on new item

⇒ select freestyle job

⇒ OK

③ Install git in mobaxterm

⇒ Sudo yum install git

④ In source code management

⇒ select git

⇒ go to github repository where your pom.xml & src are present copy the repository url and paste in the url field

⇒ branch master

⑤ In Build steps

⇒ add Build step.

↳ Invoke top level maven target

⇒ maven name

select the maven tool that you created.

⇒ Goals

type clean package

⇒ apply & save

⑥ click on build now.

In the project workspace the target file is generated it indicates that the build is done.

Task 4 :- Create a free style job to prepare build for every one minute interval when the changes are made in the github repository.

① click on new item => provide job name

=> select freestyle project

=> OK

② In source code management

=> select git

=> paste the github repository url in the url field

=> branch -> master

③ build steps

maven version:

=> select the maven tool which you created

=> ~~version~~ (default which is 3.9.

=> select -> Invoke top level maven target

④ Goals

type -> clean package

⑤ Build triggers

☐ Poll SCM -> This trigger is used to do automatically build when the changes are made in the source code

type * * * * *

apply & save

~~for every~~

after made changes it takes one minute to prepare the build but in web hook at the time when you done changes in the source code it will prepare build immediately

⑥ build now

task-5 :- create a freestyle project job to prepare build when the changes are made in the github repository

① click on new item → provide job name
⇒ select freestyle project
⇒ ok

② In source code management
⇒ select git
⇒ url → copy the url of github repository & paste it
⇒ branch → master

③ In build steps
⇒ select the maven tool which you created
select → Invoke top level maven target

④ Goals
clean package

⑤ build trigger
⇒ select Github hook

apply & save

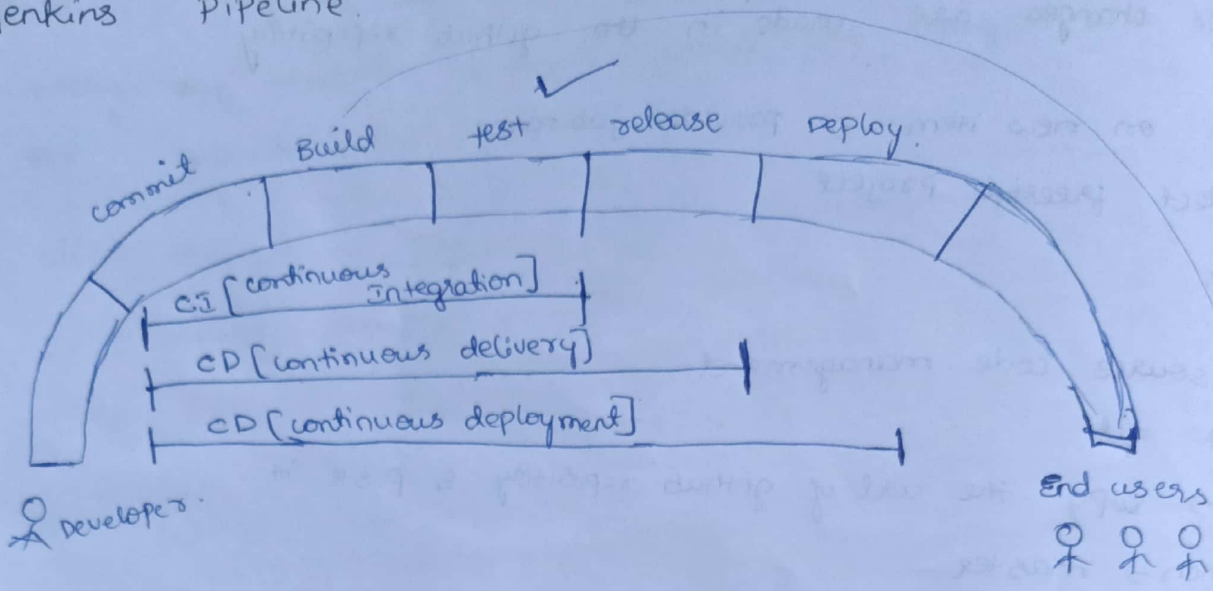
⇒ If we do changes in the github code then it will not prepare the build automatically so we need to configure github repository.

⑥ configuration in github repository
⇒ go inside the repository
⇒ click on settings
⇒ click on web hooks
⇒ click on add webhooks
⇒ In Payload url → paste the jenkins url (search bar)
↳ remove the url what previously was present and then → paste the jenkins tab url / github-webhooks/
Ex:- http://jenkins/github-webhooks/

⑦ Content type
⇒ application/json
add web hooks
build

If you do changes in sourcecode then immediately the next build is prepared.

Jenkins Pipeline



write down about Continuous Integration, Continuous Delivery (CD), continuous deployment

Continuous Integration [CI] :- CI is a development practice that involves regularly merging code changes from different contributors into a shared repository.

⇒ Each merge triggers an automated build and a set of tests to ensure that the integrated code doesn't introduce errors.

Purpose:- CI helps identify and address integration issues early in the development process, promoting collaboration and reducing the risk of defects in the software.

2) Continuous Delivery [CD] :-

CD extends CI by automating the entire delivery process, including testing, deployment and release to production-like environments. The goal is to have a reliable and repeatable process to release software at any time.

Purpose:- CD aims to deliver software changes to production & staging environments quickly and reliably. It ensures that software is always in a deployable state, providing flexibility.

Continuous Deployment

CD takes automation a step further by automatically deploying code changes to production after passing all tests in the staging environment

Task-1

Do the upstream and downstream job

=) new item -> Provide item name -> freestyle project OK

=) Build steps

=) Execute shell

↳ echo "build success"

apply & save

x is upstream project of other project.

upstream project :- after executing x project then the other project should execute

downstream project :- first execute the project then x project will execute

x is downstream project here.

=) new item -> item name -> freestyle project OK

=) Build trigger

☑ Build after other projects are build

Build -> select upstream project name -> build.

⊙ Trigger only if build is stable

=) Build steps

Execute shell

↳ echo "testing success"

apply & save

for build

↓

upstream project -> no upstream project

downstream project : test

for test : upstream project -> build

downstream project -> deploy

for deploy : upstream project -> test

downstream project -> no downstream project.

=) new item -> item name -> freestyle project -> OK

=) Build triggers

↳ Build trigger after other projects are build

-> select upstream project name

⊙ Trigger only if build is stable

Build steps

echo "deployment success"

apply & save

build now

Pipeline

new item → Pipeline project name → select Pipeline → [ok]

⇒ we create Pipeline in 2 ways

i) Pipeline script

ii) Pipeline script from SCM
↳ source code management

⇒ select Pipeline script → in right side → select Hello world
it will generate a sample code

```
Pipeline{
    agent any
    stages {
        stage { "Hello" } }
            steps {
                echo "Hello world"
            }
    }
}
```

[apply & save] build now

Task 2:- create a Pipeline job to clone the github repository and do a compile and test by using maven tool.

manage jenkins → tools → add maven → provide name → version 3-9-6 (default)
apply & save

⇒ new item → project name → select Pipeline → ok.

⇒ Install git in terminal.
↓
sudo yum install git

=> select pipeline script

```
pipeline
{
    agent any
    tools {
        maven 'maven'
    }
    stages {
        stage ('git clone') {
            steps {
                git 'github repository url which has pom.xml & src file'
            }
        }
        stage ('compile') {
            steps {
                sh 'mvn compile'
            }
        }
        stage ('test') {
            steps {
                sh 'mvn test'
            }
        }
    }
}
```

Paste

apply & save

build now

Task 3:- create a pipeline job to execute compile, test and package by using maven tool by defining Pipeline script from scm.

=> new item -> provide project name -> select pipeline -> ok

=> In pipeline -> select pipeline script from scm

=> scm:

select -> git

↳ Repository url:

↳ provide your github url which contains Pom.xml & src

-> provide branch name

-> need a file with the name Jenkinsfile.

↳ go to repository -> click on -> ~~add~~ create new file -> provide name

↓

Jenkinsfile

-> type the code

```
pipeline {
    agent any
    tools {
        maven 'maven'
    }
    stages {
        stage ('compile') {
            steps {
                sh 'mvn compile'
            }
        }
        stage ('test') {
            steps {
                sh 'mvn test'
            }
        }
        stage ('package') {
            steps {
                sh 'mvn package'
            }
        }
    }
}
```

[add maven tool
install git in terminal

commit changes

=) Script Path

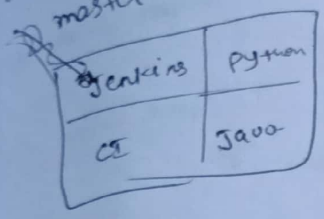
Jenkins file

=) apply & save

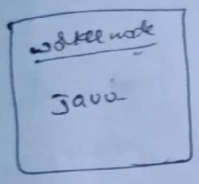
master - slave Jenkins

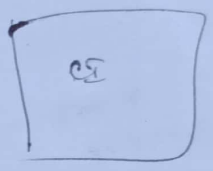
agent

master node



worker nodes





=) Launch 2 instance

=) one is master-node and another one is slave_node

=) connect 2 instance to mobaxterm terminal

=) In masternode terminal

- Install jenkins
- Install java
- update -> yum update -y
- systemctl start jenkins

=) In slave node terminal

- Install java
- update -> yum update -y

=) Connect to Jenkins UI by using master-node IPV4.

IPV4 : 8080

=) Install all plugins.

=> we need to create worker nodes in EC2 server

=> click on manage jenkins

=> click on nodes.

=> click on new node

=> Provide node name

=> select permanent agent

=> create

Note:
=> only in jenkins agent will be allowed

=> Jenkins will work on REST API

=> Name?
Provide name

=> Description
optional

=> no of Executors
2

=> Remote root directory
/home/ec2-user/build -> default directory for master-slave jenkins

=> labels : label
provide name

=> Custom-workdir Path
/home/ec2-user/build.

=> usage
use this node as much as possible

=> Internal data directory.
make it empty -> remove the remoting.

=> select type -> used to establish communication b/w 2 operation securely in live.
web socket

=> Availability
keep this agent online as much as possible

=> click on Save

=) Now the worker node is create but it is connected. X

=) To connect -> click on that node -> Run from agent command line -> select copy the code of this

=) Paste this code in slave-node terminal -> click on enter. It will establish the connection -> to exit press ctrl + C -> after exit then the worker node will become offline

to make it keep it in online mode

=) To keep it in online mode -> paste those commands after "/home/ec2-user/build" space & Jenkins.

=) create one freestyle job

-> new item -> provide name -> select freestyle job -> ok

=) select Restrict where the project is run

Label Expression

=) select provide label-name which you created while creating worker node

=) build steps

-> add build step

uptime

pwd

apply & save

=) build now

To check whether the workspace is created or not in slave-node terminal only -> come back to ec2-user and type ls ->

cd build -> here you will have remoting & workspace ->

In workspace your freestyle project is created present

Now the task is if the master node will get crashed/ destroyed, the entire data will get lost so we need to take the backup of running master node.

how to take backup [In master node terminal]

first stop the jenkins

=) systemctl status jenkins

=) systemctl stop jenkins

=) tar -zcvf jenkinsbackup.tar.gz /var/lib/jenkins/ => default jenkins path. compressing it to tar file. to take backup.

=) store in S3 bucket

=) to this create one S3 bucket. [attach role to master node instance click on actions → modify IAM role → select the role you created

=) create one IAM role → use case → EC2 → access → S3 full access (create role)

=) ~~create role~~ $aws s3 cp jenkinsbackup.tar.gz s3://sowmyayp/jenkinsbackup.tar.gz
bucket name, file name.

=) launch another one instance to restore it

=) launch new instance → connect it mobaxterm terminal → install jenkins → host in chrome.
backup instance terminal

Public IPv4 : 8080.

attach IAM role to backup instance also to restore from S3 bucket

=) cd /var/lib

=) remove jenkins file → rm -rf jenkins.

=) first we need to restore it. and then extract

=) aws s3 cp s3://bucketname/filename
sowmyayp, jenkinsbackup.tar.gz, filename jenkinsbackup.tar.gz

=) ls

=) tar -zxvf jenkinsbackup.tar.gz -C /

=) start jenkins → systemctl start jenkins

=) signin

=) After the jenkins has started you will get the backup of your ~~master~~ node & freestyle project

=) But that node will not be connected

now the task to connect the node to worker node terminal

=> click on that node

=> configure

=> name
keep it as it is

=> no of executors
2

=> remote root directory
/tmp/

=> usage
use this node as much as possible

=> launch method
launch agents via SSH

↳ Host -> Paste master-node -> IPV4 address

↳ Credentials

+add
↳ jenkins

↳ ~~username~~ ~~ec2-user~~ Kind -> SSH username with Private key.

-> ID -> slave @ anything

-> Private key
↳ enter directly -> add -> Paste the Private key file code

-> save

=> Host key verification strategy
-> manually trusted key verification strategy

=> save

Ansible

It is a configuration management tool.

Ansible doesn't allow ~~Jenkins~~ agents

① launch 3 instances

1 -> master node - instance
& 2 -> worker node - instances } connect 3 instances to mobaxterm terminal.

② Install the Ansible in master node.

master node terminal

=> sudo hostnamectl set-hostname master

=> sudo amazon-linux-extras install ansible2 -y

=> ansible --version

=> configuration file = /etc/ansible/ansible.cfg

configuration file

=> $vi /etc/ansible/hosts -> It is a simple path which give information about workers.

after #Ex1 -> type following commands

[demo]

Paste workernode1 private IPv4 address because it is static

Paste workernode2 private IPv4 address

esc :wq

Private Ip -> belongs to Class B

Note: System information is present in hosts

=> $vi /etc/ansible/ansible.cfg

uncomment inventory -> remove #

uncomment sudo_user = root

esc :wq

=> create user in master node

↳ useradd ansible

=> provide password

↳ $passwd ansible

Enter: 1234

confirm: 1234

switch to ansible user.

=> su - ansible

=> $sudo yum install httpd -y

Provide passwd

It will not install so we need to provide all access to ansible user which can be done root user.

workernode1 terminal

=> sudo hostnamectl set-hostname workers1

=> $useradd ansible

=> $passwd ansible

1234
1234

same username & password should be present in all servers.

~~=> vi /etc/~~

=> $vi /etc/ssh/sshd_config

=> uncomment -> password authentication - yes

uncomment -> permit Root login -> yes

comment password authentication -> NO.

=> provide password 1234

=> restart the ssh service

$service sshd restart

then only the worker node will be connected.

=> do configuration in

$visudo also

Ansible ALL=(ALL) NOPASSWD: ALL

workernode2 terminal

=> sudo hostnamectl set-hostname workers2

=> $useradd ansible

=> $passwd ansible

1234
1234

same steps do in this terminal also.

=) come back to root user -> exit

=) visudo -> root permission -> root ALL=(ALL) ALL

after -> ansible ALL=(ALL) NOPASSWD:ALL

space

esc :wq.

=) $su - ansible

=) $sudo yum install httpd -y

=) $sudo which httpd -> checks whether the httpd is installed or not

=) to connect to all worker nodes we use ssh protocol.

=) $ssh paste private IPv4 -> workernode1

It will denied the access -> first exit & come to root user.

=) vi /etc/ssh/sshd_config

uncomment PermitRootLogin -> yes

uncomment PasswordAuthentication -> yes

comment PasswordAuthentication -> NO.

:wq

=) then we need to restart the ssh service

i) $service sshd restart

=) su - ansible

=) $ssh paste private IPv4 of worker node1

it will connect to worker node -> if you done same steps in worker node also means you need to edit the sshd_config file in worker node also & restart the service in worker node.

=) touch sample.txt -> If you create in master node then this file also

=) ls should be present in worker node ~~with~~ of ansible user.

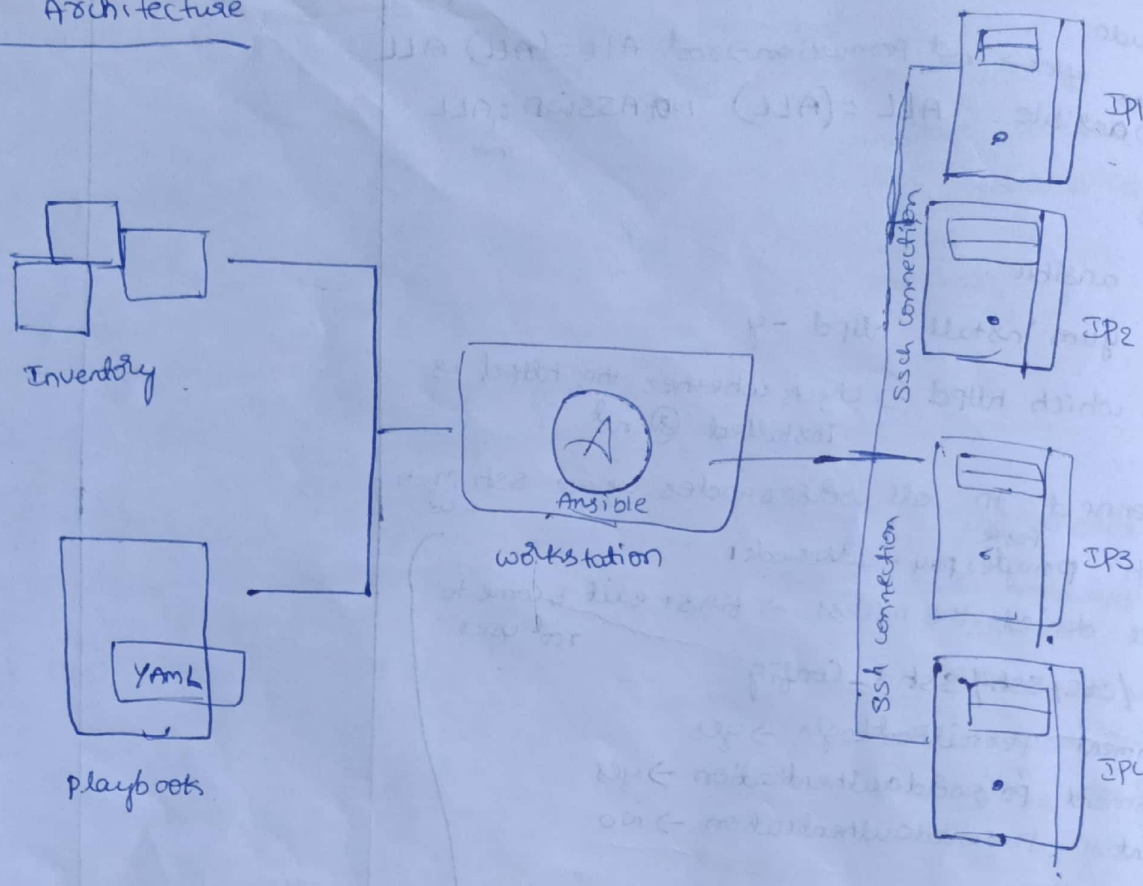
To connect master node to another worker nodes first exit from root connection same steps to be followed for all worker nodes but we need to paste that appropriate private IPv4 address of worker nodes.

Inventory :- It is a file which contains worker node details

Playbook :- execute multiple tasks which are executed in worker node. we write one script in playbook which uses yaml script

Ansible is mainly used to -> we are having multiple worker nodes but one master node, each & every worker node can be connected to master node because we can not do same task in both master node & slave so we are using ansible to do the same thing -> we are connecting master node to worker node & all the operation are done in master node but it is replicated to worker node

Ansible Architecture



Launch 2 instances

one → master node

another one → slave/worker node and connect these 2 to mobaxterm terminal.

master node terminal

=) install ansible

=) $sudo amazon-linux-extras install ansible2 -y

=) $vi /etc/ansible/hosts

after # Ex1

[demo]

Paste PrivateIp of slave node

=) $vi /etc/ansible/ansible.cfg

uncomment inventory

uncomment sudo_user = root

=) create user in master node

$useradd ansible

=) $Passwd ansible

Basic configuration of ansible

This step should be done to do all tasks by using ad-hoc command @ module command @ playbooks => The configuration is needs file => ssh private key slave node

slave node terminal

=) create user

$useradd ansible

=) $Passwd ansible

1234

1234

=) Vi sudo

ansible ALL=(ALL) NOPASSWD:

=) ~~$vi ssh/etc~~

=) $vi /etc/ssh/sshd_config

uncomment PermitRootLogin → yes

uncomment PasswordAuthentication → yes

comment passwordAuthentication → no.

=) $ ~~sshd~~ service sshd restart

=) $visudo permission
after root
ansible ALL = (ALL) NOPASSWD: ALL

esc : wq

=) vi /etc/ssh/sshd_config

uncomment PermitRootLogin -) yes

uncomment PasswordAuthentication -) yes

comment PasswordAuthentication -) no

:wq

=) restart ssh service
$service sshd restart

=) su - ansible -) used to connect to all the worknodes ~~without providing~~ ~~key~~

~~=) ssh - Keygen~~
=) $ssh-keygen
enter
enter
enter

=) $ls -a

=) $cd .ssh/

=) $ssh-copy-id ansible@privateIP of slavenode.

=) exit

=) $ssh privateIP of slavenode.

Three types of commands we will run from ansible master.

① Ad-hoc command

② module command

③ Playbooks command

=) ssh-keygen is compulsory to run the ansible commands like ad-hoc, module & playbooks.

master node terminal.

Ad-Hoc Command

=) Ad-Hoc command is a individual running command which will be execute for only one time

general ansible syntax:-

$ansible <group-name> -a <module> -a <args>

ad-hoc command syntax:-

$ansible <groupname> -a <args>

↳ all Linux commands

Task1 :-

use Ad-Hoc command to check connection.

=) all will be done in ansible user -> master node terminal (ansible user@master1)

=) $ansible demo -m ping

=) $ansible demo -a "pwd" pwd of the worker node will be displayed

=) $ansible demo -a "ls -a"

=) $ansible demo -a "touch sample.txt"

=) $ansible demo -a "rm -rf sample.txt"

=) $ansible demo -a "sudo yum install httpd -y"

=) $ansible demo -a "sudo which httpd"

=) $ansible demo -a "systemctl status httpd"

=) $ansible demo -a "sudo yum remove httpd -y"

=) $ansible demo -b -a "which httpd"

-> become -> to provide sudo privilage permission

module commands

=) ansible demo -a "pwd"

=) ansible demo -m command -a "pwd"

=) ansible demo -m shell -a "ls -a"

Important module commands.

Install >> present/installed

update >> latest

uninstall >> absent/removed

state >> started / stopped / directory / touch / restarted

enabled >> yes

file Path >> Path

Permission >> mode (0777)

First task :- => done in ansible user masternode terminal

Install httpd by using module command

=) ansible demo -b -m yum -a "pkg=httpd state=present"

=) ansible demo -b -m shell -a "systemctl status httpd" (2) // module command

=) ansible demo -b -a "systemctl status httpd" // ad-hoc command

=) ansible demo -b -m service -a "name=httpd state=started"

=) to check whether the instance is allowing httpd @ not paste the Public IPv4 of slave node in chrome -) you get testpage

=) ansible demo -b -m service -a "name=httpd state=stopped"

=) ansible demo -b -m yum -a "name=httpd state=removed"

Task 2 :- To create user by using module command.

To create user :-

ansible demo -b -m user -a "name = Sow state = Present"

To check user :-

ansible demo -b -a "cat /etc/passwd"

To delete user :-

ansible demo -b -m user -a "name = Sow State = absent"

Task 3 :- To create group and delete group by using modu

Commands.

Create a group :-

ansible demo -b -m group -a "name = devops state = present"

To check group :-

ansible demo -b -a "cat /etc/group"

delete a user group :-

ansible demo -b -m group -a "name = devops state = absent"

Task-4 :- To create directory and to delete directory.

To create directory :-

ansible demo -b -m file -a "Path = /home/ansible/demo state = direct

To delete directory :-

ansible demo -b -m file -a "Path = /home/ansible/demo state = absent"

Task-5 :- To create directory based on their permissions

ansible demo -b -m file -a "path=/home/ansible/demo
state=directory mode=0755"

To delete directory with permission 755 :-

ansible demo -b -m file -a "path=/home/ansible/demo state=absent
mode=0755"

Task-6 :- To create file.

create file :- ansible demo -m file -a "path=/home/ansible/jsp.txt
state=touch"

to delete file :- ansible demo -m file -a "path=/home/ansible/jsp.txt.
state=absent"

To create file with permission 755 :-

ansible demo -m file -a "path=/home/ansible/jsp.txt state=touch.
mode=0755"

To delete file with permission 755 :

ansible demo -m file -a "path=/home/ansible/jsp.txt state=absent
mode=0755"

To copy file from master to slave node :-

ansible demo -b -m copy -a "src=/home/ansible/file.txt
dest=/home/ansible"

How to Push your web projects into github.

1st way

=) create new folder -> paste your ~~pro~~ web project -> open gitbash here ->

=) git clone url ->

repository that contain pom.xml & src file

2nd way

Ansible Playbooks.

Playbooks is a file which describes multiple tasks to be executed in the target host.

=) Ansible playbooks will be written in a language called YAML
[Yet another markup language]

Note: Ansible playbooks should be writtened in the format of .yml

=) launch 2 instances -> one - master -> one - slave

=) connect to mobaxterm terminal

=) install ansible in master node terminal and ~~slave~~ and configure it until ssh - keygen

=) ~~same~~ configure the slave node terminal configuration

=) to all execute all the commands in ansible user master node terminal

↳ [ansible@master ~]

In ansible playbooks majorly we have 3 sections

1) target section
2) task section
3) variable section

1) target section :- target section will defines all the worker nodes which is connected to the ansible master node

Ex:- $vi target.yml ⇒ creation of playbooks file

all the filebooks will be written by 3 eithens

```
---
- name: for target check starting with
  hosts: demo
  user: ansible
  become: yes
  connection: ssh
```

after colon -> space

Use

=) $ansible-playbook target.yml --checks -> to checks whether the target.yml script execute correctly not.

=) $ansible-playbook target.yml -> to execute the script

=) It will not execute the script just checking.

Dryrun :- Dryrun will defines to check ansible playbooks scripts without executing.

$ansible-playbook target.yml --checks.

Note :- Each and every play will have target section

③ Task section:- It defines no of tasks to be execute in ansible playbooks.

Task1 :- playbook to install httpd

```
$vi task.yml
---
-name : install httpd
hosts : demo
user : ansible
become : yes
connection : ssh
tasks :
  -name : install httpd in linux server
  command : yum install httpd -y
:wq
```

=> $cat task.yml

=> $ansible-playbook task.yml --check

=> $ansible-playbook task.yml

=> $ansible demo -b -m shell -a "which httpd"

Task2:. write a playbook to execute to install httpd on worker node and need to start that service by using playbook.

```
$ vi task2.yml
---
-name : install httpd
hosts : demo
user : ansible
become : yes
connection : ssh
tasks :
  -name : install httpd in linux server
  command : yum install httpd -y
  -name : start service
  command : systemctl start httpd
:wq
```

Variable section

Installing httpd

=> $vi var.yml

```
---
- name: install httpd
hosts: demo
user: ansible
become: yes
connection: ssh

vars:
  httpd_Package_name: httpd
  httpd_Package_state: present

tasks:
  - name: install httpd package
    yum:
      name: "{{ httpd_Package_name }}"
      State: "{{ httpd_Package_State }}"
```

target section

variable section

tasks section

:wq.

(2)

=> $vi var.yml

```
---
- name: variable section
hosts: demo
user: ansible
become: yes
Connection: ssh
Vars:
  Pkgname: httpd

tasks:
  - name: install HTTPD servers on linux
    action: yum name = "{{ Pkgname }}" State: installed
```

:wq

Handlers and notify.

```
---          #any handlers Playbook
-name:       notify and handler section
hosts: demo
user: ansible
become: yes
Connection: ssh
tasks:
   -name: install HTTPD server on linux
space  action: yum name=httpd state=installed
       notify: restart httpd.

handlers:
   -* name: restart httpd
      action: service name=httpd state=restarted.
```

Loops :-

```
---          # my loops playbook
- name:
   hosts: demo.
   user: ansible
   become: yes
   Connection: ssh
   tasks:
     - name: add list of users in any nodes
       user: name= '{{ item }}' State= present
       with-items:
          - superman
          - batman
          - ironman
          - spiderman.
```

=> $ansible demo -m setup // to Know OS family

To check condition we use 'when'

Ex: If worker node belongs to one OS family then execute some commands @ else execute some other commands

Conditions

=) $vi when.yaml

```
---
# my conditional Playbooks  apt-get = debian  yum = RedHat
- name: install httpd in server
  hosts: demo
  user: ansible
  become: yes
  connection: ssh
  tasks:
    - name: install apache server for debian family
      command: apt-get -y install apache2
      when: ansible_OS_family == "Debian"
    - name: install apache server for RedHat family
      command: yum -y install httpd
      when: ansible_OS_family == "RedHat"
```

To encrypt the file

=) $ansible-vault encrypt filename var.yaml

=) $cat var.yaml

=) $ansible-vault view var.yaml // to view the file in decrypted format

=) $ansible-vault decrypt var.yaml // to decrypt the file

=) $ansible-vault edit var.yaml

write a playbooks to copy files from master node to walker no

=> vi copy.yml

```
---
- name: move files from master node to walker node
  hosts: demo
  user: ansible
  become: yes
  connection: ssh
  tasks:
    - name: move file
      copy:
        src: /home/ansible/demo.txt
        dest: /home/ansible
```

:wq

To

Task :- Launch 4 instances 1 -> master 3 -> slave

① If you need to get test page Paste Public IPV4 of any slave node you

② you need to override the test page and need to get web page content

index.html

```
<html>
  <head> - </head>
    <title> </title>
  </head>
  <body>
    <h1> welcome to devops </h1>
    <h1> I know Ansible </h1>
    <h1> so I last good sleep </h1>
  </body>
</html>
```

=) configure the master node & slave nodes

In master node terminal ansible user [ansible@master ~]

① vi testpage.yml

configuration of master node

```
$ sudo amazon-linux-extras ansible2 -y
$ vi /etc/ansible/hosts
$ vi /etc/ansible/ansible.cfg
$ useradd ansible
$ passwd ansible
  1234
  1234
$ visudo
$ vi /etc/ssh/sshd_config
$ service sshd restart
  su - ansible
  ssh-keygen [enter enter enter]
```

ssh-copy-id ansible@slave1 privatekey
ssh slave1 privatekey
exit
ssh-copy-id ansible@slave2 privatekey
ssh slave2 privatekey
exit
ssh-copy-id ansible@slave3 privatekey
ssh slave3 privatekey
exit
vi testpage.yml

```
---
- name:
  hosts: demo
  user: ansible
  become: yes
  connection: ssh
  tasks:
    - name:
      command: yum install httpd -y
    - name:
      command: systemctl start httpd
```

=) ansible-playbook testpage.yml.

[create one index.html file in master node terminal]

② vi override.yml

```
---
- name:
  hosts: demo
  user: ansible
  become: yes
  connection: ssh
  tasks:
    - name:
      copy:
        src: /home/ansible/index.html
```

configuration of slave node

in all slave nodes

```
$useradd ansible
$ passwd ansible
$visudo
$vi ssh/sshd_config
$service sshd restart
```

~~launch~~

TASK :- you must need to launch 2 instances, if you hit Public IPv4 of any instance in browser you need to get test page and run one docker image and run that container by assigning port number. and if one instance get terminated then the another instance should be launched with same configuration

=) click on launch-template

=) template - name

=) click on -> quickstart

=) AMI -> amazon-linux

Select Linux2 version

=) Provide keypair

=) ~~Provi~~ security group should allow all-traffic

=) Advanced details

user data

```
#!/bin/bash
yum update -y
yum install httpd -y
systemctl start httpd
systemctl enable httpd
sudo yum install docker -y
systemctl start docker
systemctl enable docker
docker pull httpd
docker run -it -d -P 8000:80 httpd
```

launch template

=) go to autoscaling group

=) create autoscaling group

=) provide name

=) select the template that you created -> next -> next

=) desired -> 2

minimum -> 2 maximum -> 2 next -> next

create autoscaling group.

=) copy the same to instance dashboard

=) copy the public IPV4 and paste it in new tab -> you will get testPage

=) publicIPV : 8000 -> you will get -> It works content

Advance task :- now If you paste the public IPV4 :8000 instead of getting "It works" you need to get the content of webpage i.e, welcome to devops
I know docker
So I lost good sleep } index.html

=) connect any one of the instance to terminal

=) docker PS

=) go inside the container

=) $docker exec -it containerid /bin/bash

=) $ cd htdocs

=) $ cat > index.html

<html> <body> <h1> welcome to devops </h1>

<h1> I know docker </h1>

<h1> So lost good sleep </h1> </body> </html>

Next line Press Ctrl+C

=) copy PublicIPV4 Paste PublicIPV4 :8000

now you will get the content

Terraform

Iac :- Infrastructure as a code is a methodology in which we use programming languages to define and manage the setup, configuration and provisioning of infrastructure components.

Iac approach :- In 2 ways we can approach

1. Imperative approach
   1. Aws cloud formation
   2. Azure resource manager
   3. Google cloud deployment manager
2. Declarative approach
   1. Terraform

Terraform :- Terraform is an open-source infrastructure as a code tool developed by Hashicorp

=> It is used to define and provision the complete infrastructure using an easy-to-learn declarative language i.e, Hashicorp language (HCL)

Benefits using terraform

1. Flexibility
2. Portability
3. Speed in automation
4. consistency in all time
5. minimum risks
6. Productive community ecosystem
7. Terraform is not agent-based

Terraform workflow

=) terraform init

=) terraform ~~validate~~ plan

=) terraform ~~plan~~ apply

=) terraform destroy.

terraform important commands

① terraform init :- Initialize terraform configuration

② terraform plan :- preview infrastructure changes

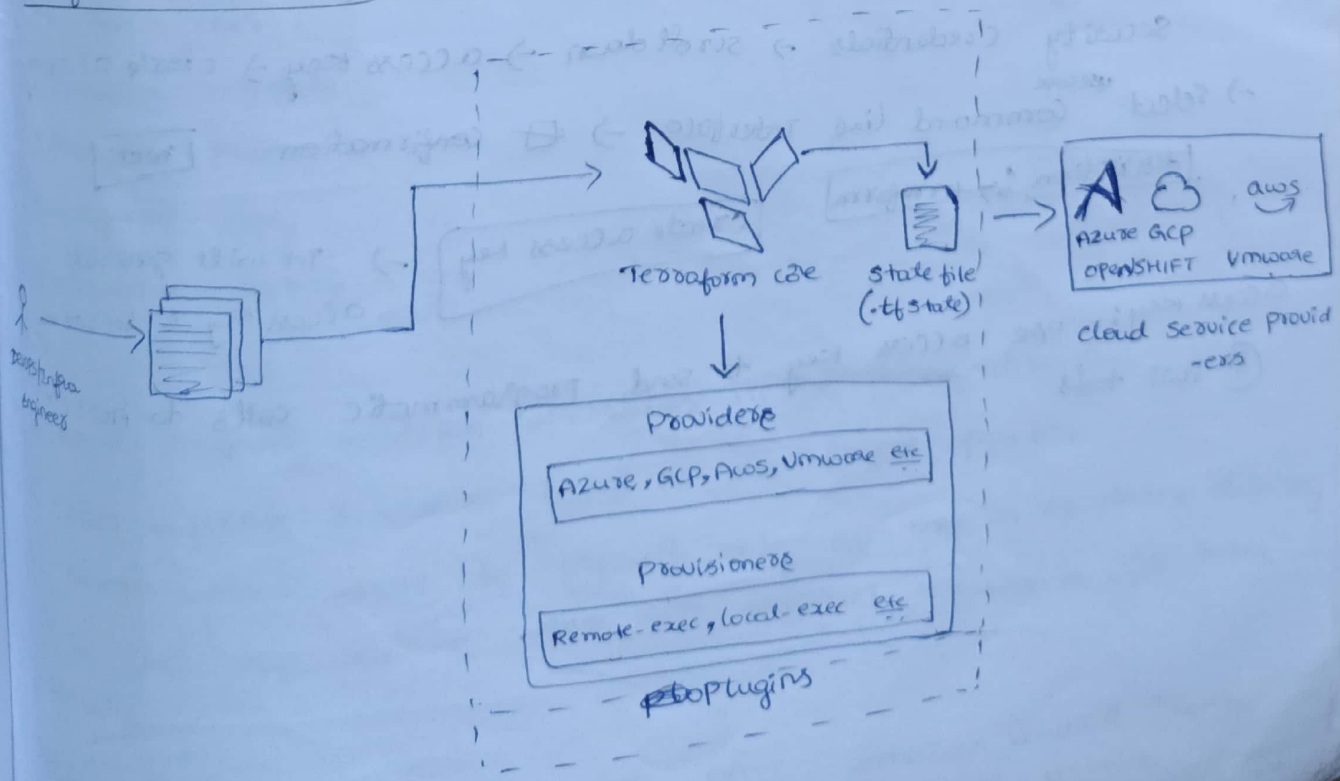
③ terraform apply :- apply defined changes to infrastructure

④ terraform destroy :- destroy created infrastructure

⑤ terraform validate :- validate configuration syntax

Note :- In terraform all the script files will be saved in the extension of .tf

Terraform Architecture



Installation of Terraform

minimum System Requirement :-

AMI :- Amazon linux

Instance type :- t2.micro

Security - group :- allow all traffic

terminal :- mobaxterm

IAm user with administrator access permission

=) launch one instance -> connect to mobaxterm terminal

=) browser search for terraform download

=) click on hashicorp terraform link -> select linux -> In that select amazon linux -> copy the code.

=) $paste in the terminal

=) $terraform --version

=) create one IAm user -> we need to generate access key & secret key

=) after creating IAm user -> click on that user ->

Security credentials -> scroll down -> access key -> create access key

-> select use case Command line interface -> confirmation next

description :- terraform

create access key -> It will generate access key & secret key.

access key :- use access key to send programmatic calls to AWS CLI

(2) AWS tools

task1:- Launch one instance with AMI linux2 and instance type t2.micro in the region of mumbai

```
$mkdir task1
$ cd task1
$vi main.tf
```

```
provider "aws" {
  region = "ap-south-1"
  access-key = " Paste access key that you generated in IAM user"
  secret-key = "  Paste secret key - 11 ------ "
}
resource "aws_instance" "terraform_demo" {
  ami = " ami id of linux" copy&Paste -> it will be present in instance dashboard
  instance-type = "t2.micro"
  key-name = "gowda"
  count = "1"
}
```

```
:wq
$terraform init
$terraform validate
$terraform plan
$terraform apply
$terraform destroy
```

Note :-

1) Don't perform multiple operations in the same directory
2) Create multiple directories based on the project operation
3) Provider blocks should be defined only one time in the same directory
4) We can have multiple files with different resources in the same directory

task2:- Launch 3 instance with AMI amazon linux2 with instance type -> t2.micro in the region of mumbai same as task1 just change count = "3" thats it

TASK 3 :- Launch 2 servers with AMI linux2 with terraform in mumbai region should be configured with docker and run a nginx container in the port number of 6000 should be hosted and override the default nginx page.

```
$mkdir task
$cd task
$vi task.tf
```

```
Provider "aws" {
    access_key = "AKIAUVEYNF7J"
    secret_key = "NSOiIh xle/VotdS2sb8ix"
    region = "ap-South-1"
}
resource "aws_security_group" "allow_all" {
    name = "allow-all-traffic"
    description = "Allow all inbound & outbound traffic"

    ingress {
        from_port = 0
        to_port = 0
        protocol = "-1"
        cidr_blocks = ["0.0.0.0/0"]
    }
    egress {
        from_port = 0
        to_port = 0
        Protocol = "-1"
    } }  cidr_blocks = ["0.0.0.0/0"]
}
resource "aws_instance" "terraform_demo" {
    ami = "ami-0c8481f02b974bc3"
    instance_type = "t2.micro"
    key_name = "gauda"
    count = "2"
    VPC_security_group_ids = [aws_security_group.allow_all.id]

    user_data = <<-EOF
                #!/bin/bash
                sudo yum install httpd -y
                sudo systemctl start httpd
                sudo systemctl enable httpd
                sudo yum update -y
                sudo yum install docker -y
                sudo systemctl start docker
                sudo systemctl enable docker
                docker pull nginx
                docker run -it -d -P 6000:80 nginx
                EOF

    tags = {
        Name = "linux-instance-${count.index+1}"
    }
}
```

Kubernetes (K8s)

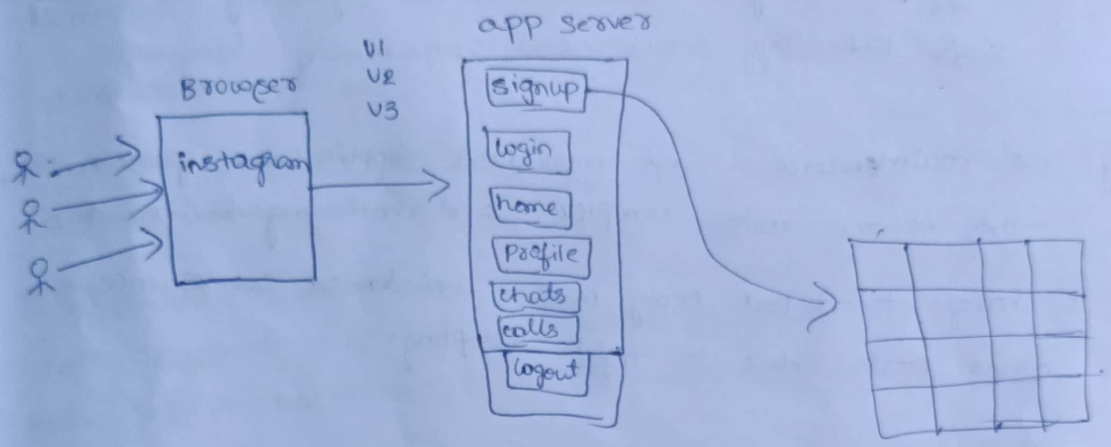
Requirements :-

AMI :- ubuntu

Instance type : t2.medium

version : 1.21

monolithic Architecture.



① Breifly Explain monolithic architecture

⇒ monolithic architecture is a traditional software design approach where an entire application is built as a single, tightly integrated unit.

⇒ In a monolithic architecture, all components and modules of the applicati-on are interconnected and interdependent.

⇒ This means that the entire application, including the user interface, business logic and data access layer is developed, deployed and scaled as a single unit.

characteristics of monolithic architecture include :-

① single codebase :- The entire application is written in a single, codebase usually in a single programming language.

② Tight coupling :- components within the application are tightly coupled, meaning changes to one part may affect other parts of the system.

③ Scalability challenges :- Scaling the application typically involves replicating the entire monolith rather than scaling individual compon-ents independently.

④ Development simplicity :- Development & deployment are generally simpler compared to more distributed architectures as there is only one codebase to manage.

② Drawback of monolithic Architecture

The drawbacks of monolithic architecture are

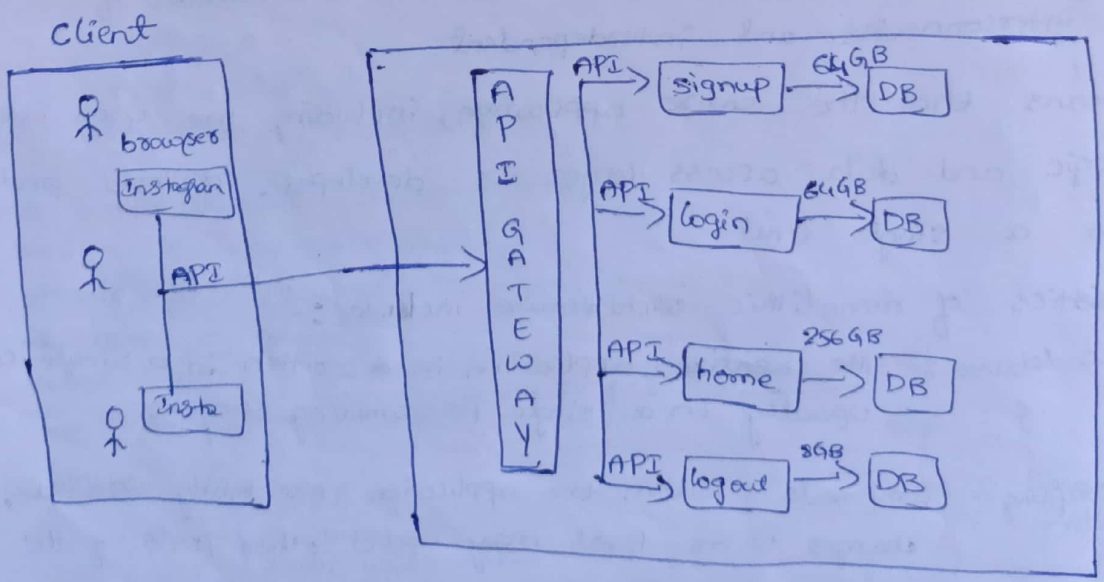
① Scalability challenges :- monolithic applications can be challenging to scale, as the entire application needs to be replicated even if only a specific component requires additional resources.

② Limited technology Flexibility :- monolithics are typically built using single technology stack and programming language.

③ Complexity and maintenance :- As monolithic applications grow in size they become more complex and challenging to maintain
=> changes to one part may have unintended consequences on other parts due to tight coupling.

④ Scaling development :- As the size of the development team grows, coordinating efforts and maintaining a cohesive development process becomes more difficult with a monolithic architecture

micro Services



① write breifly on micro services

⇒ microservices architecture is a design approach for building software applications as a collection of small, independent services that communicate with each other through well-defined API's.

⇒ Unlike monolithic architectures, where the entire application is built as a single, tightly integrated unit,

⇒ microservices break down the application into a set of loosely coupled, independently deployable services.

characteristics of microoperservices architecture are

① modularity :- The application is divided into small, self-contained services, each responsible for a specific business capability.

② Independence :- microoperservices are independent entities, allowing teams to choose different technologies, programming languages and databases for each service based on its specific requirements.

③ Decomposed architecture :- microoperservices architecture breaks down a software application into small, independent services, each focused on a specific business capability

④ ~~Indepen~~ Interconnected with API's :- microoperservices communicate with each other through well-defined API's, enabling loose coupling b/w services. ~~and~~

② Advantages of microservices

Scalability and Independence :- microoperservices architecture allows for independent scaling of individual services.

Flexibility and Technology diversity :- Each microoperservice can be developed deployed and updated independently.

Rapid development and Deployment :- microservices enable autonomous development & deployment of individual services

Installation

~~hostname~~

~~$ sudo apt update -y~~

launch 2 servers

1st one) master node

2nd one) slave node

2 types of installation

minikube

Kubeadm

=> version - V1.21.1

$ sudo hostnamectl set-hostname master

$ sudo apt update -y

$ sudo su -

$ vi kube.sh

Paste script

:wq

$ sh kube.sh

$ sudo kubeadm init -> to create the cluster.

enter

copy those 3 lines

$ mkdir -p $HOME/.kube

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

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

$ paste those 3 lines here & enter.

In worker node

do these things in worker node also after this

$ copy the kube-join token & paste in worker node with sudo permission

$ kubectl get nodes

$ copy the CNI plugins in codeshare & paste it here -> in master node

① what is Kubernetes and features of Kubernetes.

Kubernetes is an open source container orchestration platform that automates the deployment, scaling and management of containerized applns

Features

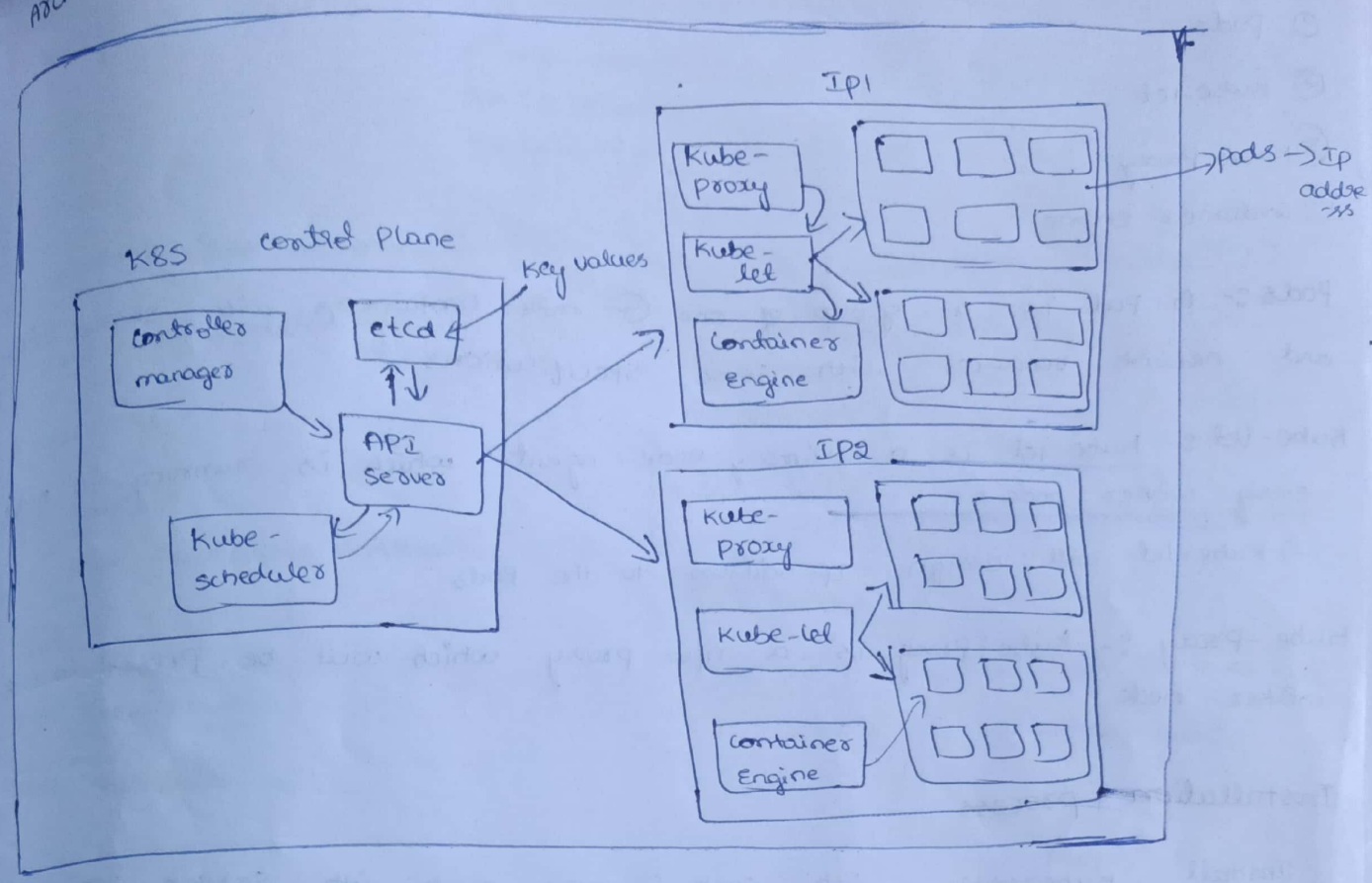
=> Container orchestration

=> Auto-scaling: automatically scale containerized applns & their resources up & down based on usage

=> load balancing

=> persistent storage

Architecture of Kubernetes.



Control plane:- control plane is a master node of kubernetes which controls all the nodes and pods in the cluster.

components in control plane

① Controller manager
② API Server
③ etcd
④ Kube-Scheduler

Controller manager :- It is a main component in the control Plane which controls the processes logically and by reducing complexity and they are all compiled into a single binary and run in a single process.

API Server : The API server is a front-end component in the kubernetes control plane

etcd : Consistant and highly available key-value store used in a kubernetes backing store for all cluster data

Kube-Scheduler :- It is a control plane process which assigns pods to nodes

Worker node components

① Pods
② Kube-let
③ Kube-Proxy
④ Container Engine

Pods :- A Pod is a group of one ② more containers with shared storage and network resources with some specifications

Kube-let :- Kube-let is a primary node agent which is running in each every worker node

⇒ Kube-let will assign IP address to the Pods

Kube-Proxy :- Kube-Proxy is a n/w proxy which will be present in each worker node.

Installation process

Install Kubernetes with each & every node with server specification

Am I → ubuntu
instance-type → t2.medium
security-group → all-traffic
terminal → mobaxterm

Create a Pods in the cluster with ubuntu image [1 Pod with 1 container]

Note:- In Kubernetes we will write script in yaml language which follows with proper indentation.

⇒ Kubernetes is also called object management tool

Task1 :- Create 1 Pod with one container

Script to create 1 Pod with 1 container

```
kind: Pod
apiVersion: v1
metadata:
  name: flipkart
spec:
  containers:
  - name: sign-up
    image: ubuntu
    command: ["/bin/bash", "-c", "while true; do echo 'DevOps'; sleep 5; done"]
    restartPolicy: Never
```

launch 2 servers
$

1 -> master node
2nd one - slave
AmI -> ubuntu
instance type -> t2.medium

master node terminal

$ sudo apt update -y

$ sudo su -

$ vi kube.sh
Paste Kubernetes installation script

:wq

$ sh kube.sh

$ sudo kubeadm init

Copy these 3 lines
$ mkdir -p $HOME/.kube
$ sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
$ sudo chown $(id -u):$(id -g) $HOME/.kube/config

$ Paste here

$ Copy the kube-join 2 lines & paste in worker node

$ kubectl get nodes

$ ~~kubectl get pods~~ Copy & Paste the CNI Plugins from codeshare

$ kubectl get pods
No resources found in default namespace

$ vi pod.yml
Paste script

$ kubectl apply -f pod.yml -> to execute the file
(file name)

$ kubectl get pods

$ kubectl get pods -o wide -> to know overall information of pods

$ kubectl describe pod flipkart -> describing the pod information

$ kubectl logs -f flipkart -> to execute the container

to use ctrl+c

$ kubectl delete -f pod.yml

Slave node terminal

$ sudo apt update -y

$ sudo su -

$ vi kube.sh
Paste Kubernetes installation script
:wq

$ sh kube.sh

$ ~~sudo~~ Paste the kube-join 2 lines here
that's it in worker node

Task : 2

create one Pod with 2 containers

Script :-

Vi Pod.yml

```
kind: Pod
apiVersion: v1
metadata:
  name: flipkart
spec:
  containers:
    - name: signup
      image: ubuntu
      command: ["/bin/bash", "-c", "while true; do echo 'signup'; sleep 5; done"]
    - name: login
      image: ubuntu
      command: ["/bin/bash", "-c", "while true; do echo 'login'; sleep 5; done"]
:wq
```

$ kubectl get pods

$ kubectl apply -f Pod.yml

$ kubectl get pods

$ kubectl get pods -o wide

$ kubectl describe pod flipkart

$ kubectl logs -f flipkart -c signup
ctrl c

$ kubectl logs -f flipkart -c login

$ kubectl delete -f pod.yml

Task 3 :- To assign the port to nginx container

Vi Pod.yml

```
apiVersion: v1
kind: Pod
metadata:
  name: flipkart-signup
spec:
  containers:
    - name: signup
      image: nginx
      ports:
        - containerPort: 80
```

Vi Pod.yml

```
kind: Pod
apiVersion: v1
metadata:
  name: nginx
spec:
  containers:
    - name: signup
      image: ubuntu
      ports:
        - containerPort: 80
```

$ kubectl apply -f pod.yml
$ kubectl get pods -o wide
$ curl pod IP:80

④ label selectors

label selectors are used to filter and select the objects [ex: Pods] based on key values

⇒ label selectors we can use based on 2 ways

① equality based ⇒ to filter based on only one condition
ex: =, ==, !=

② set based ⇒ ex:- [In, NotIn, exists, doesnotexists]

vi pod4.yml

```
kind: Pod
apiVersion: v1
metadata:
  name: testfreshers
  labels:
    department: developer
    batch: sigma
spec:
  containers:
    - name: demo
      image: ubuntu
      command: ["/bin/bash", "-c", "while true; do echo testfreshers; done"]
  restartpolicy: Never
:wq
```

$ kubectl apply -f pod3.yml

$ kubectl get pods

$ kubectl get pods --show-labels

$ kubectl label pods testfreshers name=devops → provide name to label

$ kubectl get pods --show-labels

$ kubectl get pods -l department=developer

$ kubectl get pods -l department!=developer

$ kubectl delete -f pod4.yml

$ kubectl pods -f batch=sigma

$ kubectl delete -f pod3.yml

$alias k='kubectl' instead of using kubectl we can use k in

⑤ ReplicationController

ReplicationController ensures a specified number of pod replicas are always running, helping with fault tolerence & scalability.

Vi rc.yml

```
apiVersion: v1
kind: ReplicationController
metadata:
  name: skillary
spec:
  replicas: 2
  selector:
    app: nginx
  template:
    metadata:
      name: abc
      labels:
        app: nginx
~~spec:~~
    ~~containers:~~
    ~~- name: xyz~~
      ~~image: nginx~~
      ~~ports:~~
        ~~- containerPort: 80~~
    spec:
      containers:
      - name: xyz
        image: nginx
        ports:
        - containerPort: 80
:wq
```

Note: In rc we will be used only equality based label selector

```
$ kubectl apply -f rc.yml
$ kubectl get pods      → here we will get podid.
$ kubectl get rc
$ kubectl delete pod podid
```

) we can scale up & scale down pods in replicationController by using

```
$ kubectl scale --replicas=8 rc -l app=nginx
```

⑥ replicaset

a replicaset is a advanced version of replicationController with more advanced selector options

vi rs.yml

```
kind: ReplicaSet
apiVersion: v1          => you can also use apps : apiVersion: apps
metadata:
  name: testyantra

spec:
  replicas: 2
  selector:
    matchExpressions:
      - {key: myname, operator: In, values: [zspiders, Jspiders, Pyspider]}
      - {key: env, operator: NotIn, values: [skillbay]}

  template:
    metadata:
      name: sak
      labels:
        myname: zspider

    spec:
      containerss:
        - name: abc
          image: ubuntu
          command: ["/bin/bash", "-c", "while true; do echo Hello world;
                                        sleep 5; done"]
```

Note:
ReplicaSet will be allowed both equality based & set based [selector based on matchbyExpression & matchby label]

:wq

$kubectl apply -f rs.yml

$kubectl get rc -> no resources found because we create ReplicaSet not replicationController

$kubectl get rs

$kubectl scale --replicas=4 rs testyantra

$kubectl delete -f rs.yml

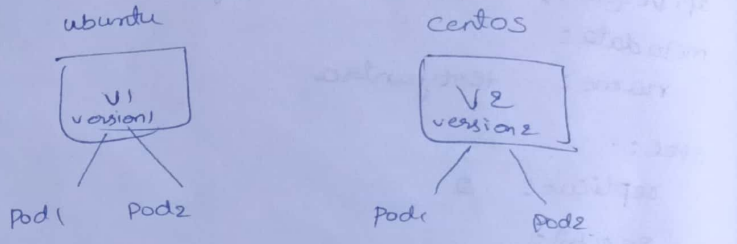
Deployment object : combination of replicaset with Pods
=) used to rollbacks

what is the role of a Deployment in Kubernetes?

A Deployment declaratively manages a set of Pods, facilitating easy updates and rollbacks.

vi dep.yml

```
Kind: Deployment
apiVersion: apps/v1
metadata:
  name: skillary
spec:
  replicas: 2
  selector:
    matchLabels:
      name: deployment
  template:
    metadata:
      name: spider
      labels:
        name: deployment
    spec:
      containers:
        - name: abc
          image: ubuntu
          command: ["/bin/bash", "-c", "while true; do echo skillary; sleep 5; done"]
```



:wq

$ Kubectl apply -f dep.yml
$ kubectl get deploy
$ Kubectl get rs
$ Kubectl get pods
$ Kubectl delete Pod podname → Ex: skillary-7688 - q~~6~~82b
$ Kubectl scale --replicas=4 deploy skillary
$ Kubectl scale --replicas=1 deploy skillary
$ Kubectl logs -f skillary-786488Tb-7688
complete Podname → Kubectl get pods
$ Kubectl exec -it complete Pod name -- cat /etc/os-release
$ Kubectl get pods

$vi dep.yml

same code just change image & echo statement

image: centos
command: ["/bin/bash", "-c", "while true; do echo v2.0; sleep 5; done"]

:wq
$kubectl apply -f dep.yml

$kubectl get deploy

$kubectl get rs

$kubectl get pods

$kubectl scale --replicas=1 deploy skillary

$kubectl get pods

$kubectl logs -f completepodname

$kubectl exec -it completepodname -- cat /etc/os-release

$kubectl rollout status deployment skillary

$kubectl rollout undo deploy skillary

$kubectl get pods

$kubectl logs -f complete pod name

$kubectl exec -it completepodname -- cat /etc/os-release

communication object

communication b/w containers to containers in the same Pod.

vi com.yml

```
kind: Pod
apiVersion: v1
metadata:
  name: flipkart
spec:
  containers:
    - name: c1
      image: ubuntu
      command: ["/bin/bash", "-c", "while true; do echo Hello DevOps; Sleep 5
                                                                    done"]
    - name: c2
      image: httpd
      ports:
      - containerPort: 80
```

```
$kubectl apply -f com.yml
$kubectl get pods
$kubectl exe -it flipkart -c c1 -- /bin/bash
$ ls
$ pwd
$ curl localhost:80
$ apt update -y
$ apt install curl -y
$ curl localhost:80
exit
$ kubectl exec -it flipkart -c c2 -- /bin/bash
$ curl localhost:80
$ apt update -y
$ apt install curl -y
$ curl localhost:80
```

Note :- when we want to communicate from one container to another container in the same Pod we will use a localhost

Communication b/w Pod to Pod

vi pod.yml

```
kind: Pod
apiVersion: v1
metadata:
  name: nginx
spec:
  containers:
  - name: nginx
    image: nginx
    ports:
    - containerPort: 80
```

vi pod2.yml

```
kind: Pod
apiVersion: v1
metadata:
  name: httpd
spec:
  containers:
  - name: httpd
    image: httpd
    ports:
    - containerPort: 80
```

$ $kubectl apply -f .

$kubectl get Pods

$kubectl get Pods -o wide

$curl nginxPodIP:80

$curl httpdPodIP:80

Kubernetes N/w's.

Service :- A service is a method for exposing a network application that is running as one or more pods in the cluster.

Kubernetes services types

there are 3 types of services

1. Cluster IP
2. node port
3. Load Balancer.

Cluster IP :- An cluster IP is a fixed IP, it can be created infront of pod

=) Cluster IP is a default for the service type

=) Internal clients send requests to a stable internal IP address

Note :- TO perform kubernetes service we need deployment with service object.

=) cluster IP is also known as static IP

~~$mkdir cluster~~ =) $cd cluster.

vi deploy.yml

```
kind: Deployment
apiVersion: apps/v1
metadata:
  name: skillary
spec:
  replicas: 1
  selector:
    matchLabels:
      name: deployment
  template:
    metadata:
      name: vip
      labels:
        name: deployment

    spec:
      containers:
        - name: abcd
          image: httpd
          ports:
          - containerPort: 80
```

$kubectl apply -f deploy.yml

$kubectl get deploy

$kubectl get rs

$kubectl get pods

$kubectl get pods -o wide

$curl PodIP:80

vi service.yml

```
apiVersion: v1
metadata:
  name: demoservice
spec:
  ports:
  - port: 80
    targetPort: 80
  selector:
    name: deployment
  type: ClusterIP
```

note :- run both deploy.yml & service.yml script at a time

$kubectl apply -f .

$kubectl get deploy

$kubectl get rs

$kubectl get pods

$kubectl get svc → to know the service object
here we can get cluster IP

$curl clusterIP:80

$kubectl delete -f .

Node port :- It is a type of service where it will build on top of cluster IP by exposing the cluster IP service outside the cluster on high ports

=) The range of node port starts from 30,000 to 32,767

vi node.yml

```
kind: Service
apiVersion: v1
metadata:
  name: demoService
spec:
  ports:
    - port: 80
      targetPort: 80
  Selector:
    name: deployment
  type: NodePort
```

vi deploy.yml

```
kind: Deployment
apiVersion: apps/v1
metadata:
  name: skillary
spec:
  replicas: 1
  selector:
    matchLabels:
      name: deployment
  template:
    metadata:
      name: vip
      labels:
        name: deployment
    spec:
      Containers:
        - name: abcd
          image: nginx
          ports:
          - containerPort: 80
```

$ Kubectl apply -f .

$ kubectl get deploy

$ kubectl get rs

$ kubectl get Pods

$ kubectl get svc

$ came to instance dashboard and copy Public DNS of worker node & port no assigned

$ curl clusterIP:80

Public dns : port

↳ It will be in =) Kubectl get svc

from get svc you will get

Kubernetes Volume

We have 3 types of Kubernetes Volume

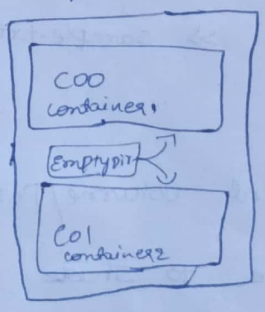
1. Empty Dir => b/w 2 containers we can share the resources
2. Host Path
3. Persistent volume and Persistent Volume claim

Vi Empty.yml

Note: EmptyDir type of Volume will be used when 2 containers needs to share the resources with each other.

```
apiVersion: v1
kind: Pod
metadata:
  name: Volume
spec:
  containers:
  - name: C00
    image: centos
    command: ["/bin/bash", "-c", "sleep 15000"]
    VolumeMounts:
      - name: skillary
        mountPath: "/tmp/2spider"
  - name: C01
    image: centos
    Command: ["/bin/bash", "-c", "sleep 10000"]
    VolumeMounts:
      - name: skillary
        mountPath: "/tmp/jspider"
  Volumes:
  - name: skillary
    emptyDir: {}
```

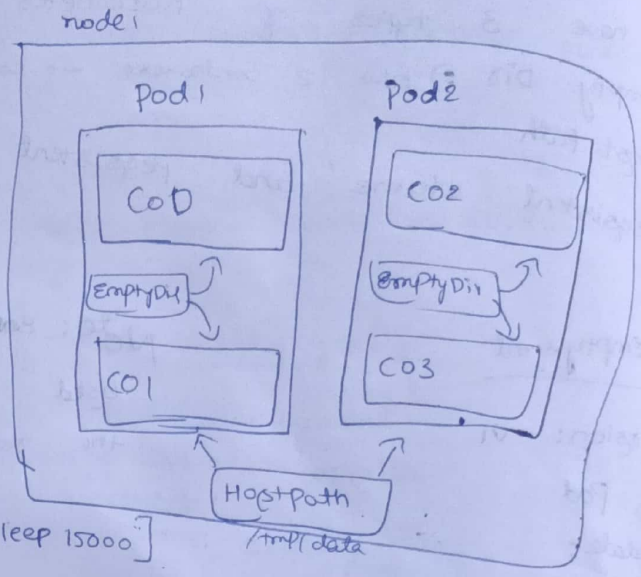
Pod1



Pod2

```
$Kubectl apply -f empty.yml
$kubectl get pods
$ kubectl exec -it Volume -c C00 -- /bin/bash
                  container name
$ cd /tmp/ => $ls
$ cd 2spider/
$ echo "backup" >> sample.txt
$exit
$ kubectl exec -it Volume -c C01 -- /bin/bash
$ cd /tmp/ => $ls => cd jspider => $ls
```

HostPath



Vi host.yml

```
apiVersion: v1
kind: Pod
metadata:
  name: demo
spec:
  containers:
  - image: centos
    name: sample
    command: ["/bin/bash", "-c", "sleep 15000"]
    volumeMounts:
    - mountPath: /tmp/hostPath
      name: skillbay
  volumes:
  - name: skillbay
    hostPath:
      Path: /tmp/data
```

master node terminal

```
$kubectl apply -f host.yml
$kubectl exec -it demo -c sample -- /bin/bash
        (Pod name)
$cd /tmp/ => $ls
$ cd hostPath/
$ echo "checking data" >> sample.txt
$ls
```

slave node terminal

```
$cd /tmp/
$ ls
$cd data)
$ls
```

PV / PVC [Persistent Volume/Persistent Volume claim]

when we have to share the resources from one node to another node then we will use a volume type of PV & PVC

=) PV & PVC can perform in another node which is created in the same cluster

=) That node should be installed with a system called NFS [Network File system]

=) NFS will allow to share the resources from one node to another node

=) we need to create majorly 3 objects

① Deployment object
② Persistent Volume object
③ Persistent Volume claim object.

secret and config map:-

namespace -> it is only a object.