VCS (version controling system) developer
git github

SDLC (Software Development Life cycle)

- Water fall model

1) Info --- Design --- Build --- Test --- Deploy --- Maintainance
(Version1)

2) Info --- Design --- Build --- Test --- Deploy --- Maintainance
(version 2)

- stability more
- remove & reinstall
- maintainance easy

- Agile model

1) Info --- Design --- Build --- Test --- Deploy --- Maintanance.

($V_1 - V_2 - V_3$)

provides agility to the s/w.

- stability less
- updates.
- maintainance difficult.

- DEVOPS

1) To maintain stability and maintainance.
2) To

. project - repository files

Github, create repository

project name.

// thub-devops

README → info about project

- private
- public

} visibility

. Readme file.

Git bash (Works on Linux sub Os)

Profile

↓

Preferences (to allow client to access the files)

User settings

↓

SSH Keys

SSH Key → Securely connect to the remote repositories.

`ssh-keygen -t rsa -b 2048 -C "emailid"`

RSA keygen — ↓ bitwise

Enter passphrase: optional (password)

id-rsa.pub (public key)

location : c/user/Kisho/.ssh/

clone

project (code)

↓

clone

↓

https link (url)

git bash :- git clone url

Github:-

we create

- Repositories ~~creat~~

(directories)

Create a new repository

owner / Repository name

user / [ ] unique for url

- ⊙ private
- ⊙ public

☐ Read me

(Linux)

1) git bash & github connectivity

1) github user name & password

2) ssh

profile

↳ settings

↳

SSH x GPG keys

↳ generating SSH key

↳ generating SSH key x adding it to the ssh-agent

(Windows)

- ssh-keygen -t ed25519 -C "your_email@example.com"

Enter file in which to save the key : Enter.

/c/Users/kishore.g/.ssh.

Enter passphrase : Enter / give any password.

- Give password

pattern.

→ open with notepad.

git client - repository

github, gitlab } deploy complete ci/cd pipeline

---

git bash

↓

command

↓

enter file (location)

↓

enter passphrase

confirm passphrase

↓

pattern.

ssh-keygen -t rsa -b 2048 -C "jayasri3002@gmail.com"

- pwd // location => Key is generated

~~Add new SSH key~~

- pub file
  ↓
  notepad
  ↓
  copy

ssh-rsa

jayasri3002@gmail.com

Add new SSH Key

1) Name, authencation key
2) paste key

Repository

new
↓
owner / Repository name
↓
Description
↓
public/ private. (download & see)
↓
Readme
↓
create.

Invalid key
22/4/23 Error

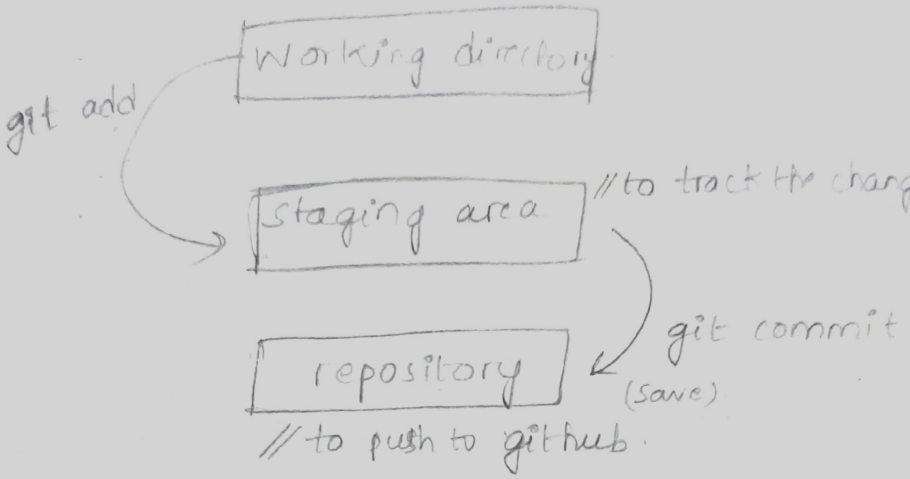
cd dr
Dropbox/ dr-2025/ dr-2025-fn/

cd dr-2025-fn.

ls.-al (.git)

//configuration directory for entire github directory.

(ls -l
ls -la
ls -al)



- git status
- add -- commit -- push -- remote repository -- team

laptop.

github.

(not sync but push & pull)

cat > myfirstcode
My first code ^ //untracked in git status.

git add myfirstcode

git commit enter //Write some message } git commit -m "__"

esc shift+: wq

esc :wq!

git log //all the changes

(tissy)

We cloned repo - manager
We initialize - push (to link)

- collab - team members

branches

1) main // by default
2) jr.dev - features/bug fix
   feature - push
   (crash)

// main has authorization, access

2) bugfix - branch
   features - branch
   security - branch

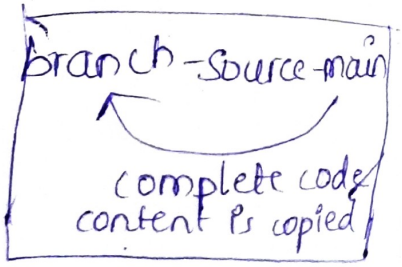
// After completion, request - pull request, merge request.

- approve the request
  approve merge/pull
  bugfix → main
  main → pipeline.

Pipeline.

main branch ⇒
test, build, deploy

⇒ Branch
→ New branch
→ security-patch (Name)
• source



To change branch

git checkout branchname
git checkout security-patch.
~~u~~ ~~to~~ (Error)

- git pull (this branch comes into our system)
- git checkout security-patch

file1 in main & security-patch branch.

- cat >> file1
  This is third line.
- cat file1

  This is third line.

Send to github

git add .
git commit -m "commit all"
git push.

⇒ github : pull request , gitlab : merge request

- commit all

pull request ⇒ check my code and merge with main

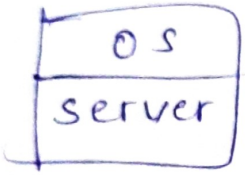
merge pull request
merge confirm

pull request successfully merged & closed.

Docker

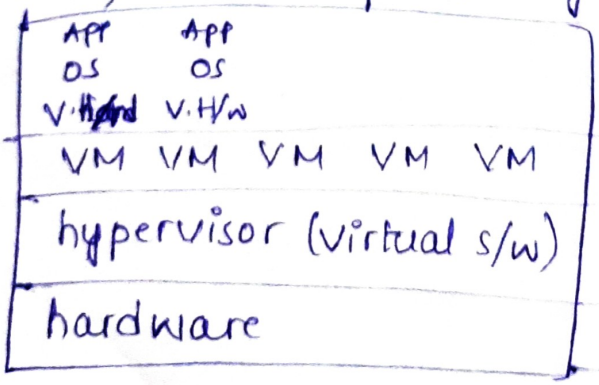
Applications (https, apaches, mail)
OS - traditional
hardware (cpu, memory, storage)

- OS → systematic
  It controls the complete applications, cpu, memory etc.
  8gb, 4 core.
- Server ⇒ 128 gb, 23 core.
  If we install OS, can we control it ?



controlling.

- Not allowing user to control
- Concept of virtualization :- allowing users to control, OS is replaced by hypervisor.



vmware, orcale

virtual genet

- hypervisor gives control to users (system administrators, devops engineers).
- Launching ec2 instance ⇒ virtual hardware
- virtual hardware is extracted from physical hardware by hypervisor.

- Every VM, is dedicated to one OS (Again OS is controlling everything in case V. H/W)
- OS is wasting so many resources
- Websites - apache2 - httpd
  (50mb) (30mb) S/W
  10mb 10mb - webserver

VM ⇒ min 2 cpus, 4gb, 30gb.

* Remove the concept of OS ⇒ containarization. (No OS)

app app app (No VM)
computer runtime (docker).
OS
computer

- Here, no need to host ~~OS~~ app on OS.
- Now, no need of OS and VM,
- OS can be replaced by computer runtime with existing OS.

google - docker hub - sign up - personal id.

Ubuntu server ⇒ apt-get update (redhat/centos ⇒ yum install docker)
apt-get install docker.io (ubuntu)
// apt install docker.io
// snap install docker.io (other flavours)

container image (docker image)
→ application layer
→ alpine Linux (light weight OS) } pack

docker public hub repo of container images (verified images)

OS version is the most complicated. If one developer is using ~~one~~ diff OS with diff versions, code can't be run easily.

- they are easy to move around → container.
  container → same container is used by everyone.

- portable
- versioning
- Light in size.

Before docker, we have container run time cri-o, container_d.

docker → container network
→ container volumes.

- developed on Linux.
- In windows ⇒ docker desktop (Linux subsystem)

nginx
↳ supported tags
version

docker pull nginx
(latest version)

(can't download from google, all these versions are available in docker hub)

- python, redis, nginx

⇒ docker images.
// to see the available versions.

~~nign~~ docker pull nginx:1.25.2-alpine perl

docker --version.

docker run image-name
// run the image in a foreground.

docker run -d image-name
detach
// run the image in background.

docker ps // to see the processes that are running. (containers)

ctrl+c disables the images

docker image ↑ docker hub
docker container ↑ running env for image

docker ps -a // to see the containers that are running and exited ones.

docker stop CONTAINER_ID //stop container to run.
docker start CONTAINER_ID

- docker pull httpd
  docker run httpd ✓

docker run httpd
// doesn't exist ✓
// then it automatically pulls and runs.
(pull & run)

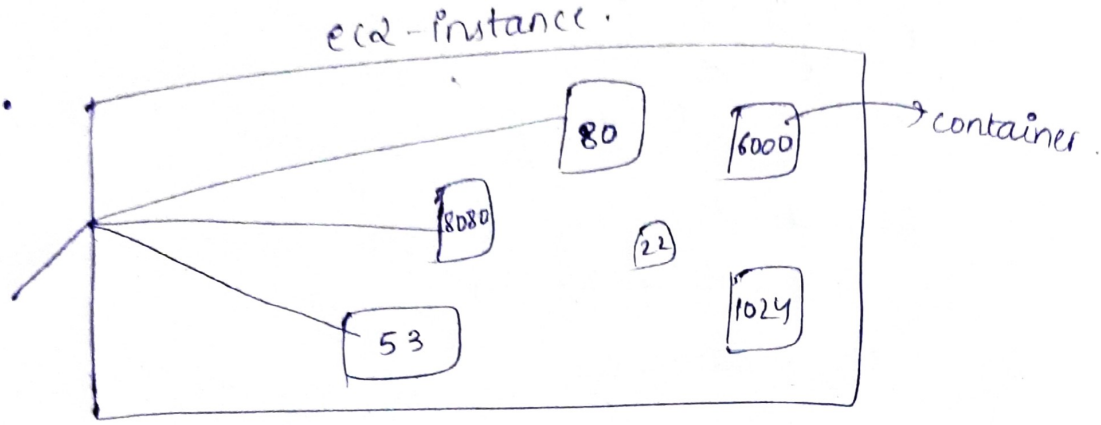
ctrl+c to exit process.

- docker rmi imageID // to remove image.
- docker rmi imageID -f // to remove forcefully.
- docker run -d --name my-app httpd
  flag to name ur container
- docker stats
- docker info

- docker exec -it CONTAINER_ID /bin/bash
  (interactive mode)
- docker env // environment variables (or) configuration variables.

- clear
- ls -l

alpine linux is present in docker



If we want to access the container, we need to map it to its port numbers. (to map we use -p)

- docker run -d -p 80:80 (container) httpd.
  (port no that is coming to ec2 instance)

0.0.0.0 : 8080 → 80 (8080 80)

- 3.89.215.37 : 8080 [chrome]
  port no

(custom TCP) 8080

Q) With container name and ports. http latest (running)
: 1 version before the latest
8081:80

Host a website

laptop → ec2 instance → container
(mobaxtreme).

/usr/local/apache2/htdocs · => from this location the sample page is reflecting

· docker cp => copy content instance to container

docker cp /assets/ container id :/usr/local/apache2/htdocs/

index.html

· ec2 instance 8081-8080 (custom tcp)

session → SFTP (drag & drop)

· Mobaxterm

apt-get update

apt-get install docker.io -y

docker pull httpd (conID-1)

docker pull httpd:2-alpine (conID-2)

docker cp assets/ conID-1:/usr/local/apache2/htdocs/

docker cp index.html conID-1:/usr/local/apache2/htdocs/.

Session
↓
SFTP
↓
public ip (host)
Username ubuntu
port 22
Advanced setting
· private key

laptop window (content)	Ec2 instance window (content)

· docker cp assets/ conID-2:/usr/local/apache2/htdocs/

docker cp index.html conID-2:/usr/local/apache2/htdocs/.

To run ec2 instance

ec2 instance:8080

ec2 instance:8081

· It is httpd image

· We kept some html pages.

· //create my own image

docker commit conID //running container

· docker images

Repository	Tag	
<none>	<none>	// my image.
httpd	latest	conID.

// to tag image

- docker tag Image_ID my-app
- docker images

Repository	tag
my-app	latest
httpd	latest

root@container // exit

- docker stop con_ID
- docker run -d -p 808:80 my-app.

httpd

- Libraries
- binaries
- alpine Linux
- httpd

my-app

- Libraries
- binaries
- alpine Linux
- httpd
- my-website.

// Without changing their version, & configuration we can pull our version and work on the project.

- Light weight
- no need to change version.

Create repository

user id . name

- public
- private

Process automation (pay)

docker tag IMAGE-ID (repository name) username/app:1 (local)

docker login.

username: jaya3002
password: Jaya@#123

docker push repository

Jaya3002/jaya-123

docker commit container ID
docker tag Image ID name (<none>:<none>)
docker stop container ID
docker run -d -p 8081:80 name

.docker login
username:
password:
docker push repository name

Manikanta (Surname, roll no)

container port number

codemind, owlcoder, 1st.

webpage

① web service → run actively (container)
② Port must be assigned
③ Content
④ where content should be deployed.

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

(1 process)

If process running foreground, we can't use any command

So, run in detach mode (background)

docker run -d -it nginx

- ecommerce web site has many pages
- If it exit (stop) we don't know why

nginx (web server)

---

git repo → git bash → pull (no need to drag & drop)

git install
git clone repository URL

where? /home/myselfpance

① docker cp MedPlus-Static containerID:/usr/local/apache2/htdocs.

no such container path
(httpd)

docker cp MedPlus_Static containerID:/usr/local/share/nginx/html.

don't give direct folder (error)
give all files

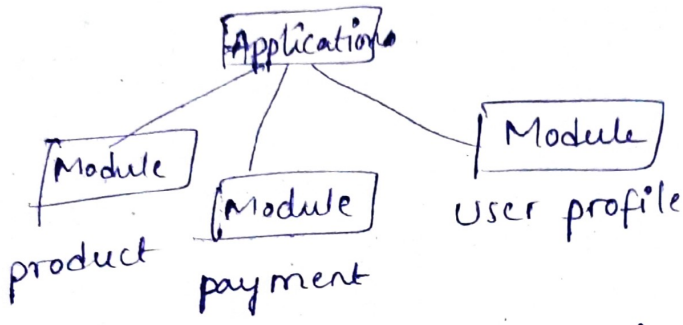
Container :8080/MedPlus_Static/

- /usr/share/nginx
  - ls //MedPlus_Static html
  - rm -rf html
  - ⊙ mv MedPlus_Static html

rename

Kubernetes

⇒ Light weight applications like < 500MB.
⇒ We need to deploy light weight application otherwise 1 MB is also not accessable (single point of failure) once service is down, completely down.
⇒ Application must be light weight and flexible. (microservices)
⇒ Database can't be deployed but database application can be deployed.



- Generally modules are merged but they are not merged in microservices (put in 3 diff ec2 instance)
- If one module is down, still remaining are not down.
- Put in 3 diff containers.

⇒ Single point failure (monolethic)

- Like autoscaling, we use >3 containers.
- AJio ⇒ 10GB

⇒ Kubernetes ⇒ Archestration (managing) container ~~Archetaction~~ tool

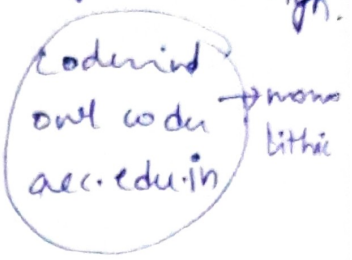
- Container is down (unhealthy), it will automatically it replaces with another container automatically.

- scalable
- flexible
- disaster recovery

Can be deployed in ~~EC2~~ normal machine, VM,
- Primarily developed by Google docker.
  → Maintainer
- CNCF is maintaining
- maintaining x splitting ⇒ system design.

⇒ Pod · (container home)

Container architecture



Application.
Binaries
Libraries
Alpine

Min life span of container → 2 yrs
POD life span → minutes/days

- No stability, no OS to maintain (Abstraction Layer, Enviroment)
- No Physical H/W, no OS.
- We can place multiple containers in POD theoretically · (If we have huge architecture)
  3 containers (suggested) x 1 service.
  1 container (recommanded)
- New IP address on re-creation.
- POD gets private IP address (internal IP add)

Kubernetes Architecture

1) Master Node (Maintainance & Management)
2) ~~Master~~ Worker Node (Actual PODs are here)

docker → container concept
kubernets → POD concept

K8 cluster supports

1) Docker
2) Cri-o
3)

setup

1) cloud.

ec2 - 3

① master

② worker nodes,

. 2 vcpus, 4gb.

If we want replication, min ②
otherwise ①

. free tier ⇒ 1 vcpu.
. t2 medium - not free

2) minikube → K8s (free)

8 gb ram
2 cores - 1 node testing K8s cluster

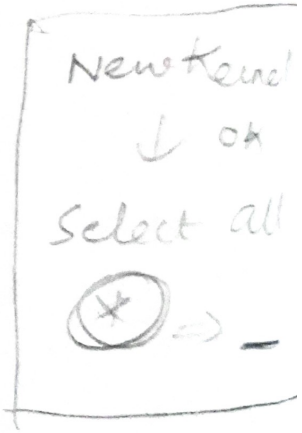
: → management
: → deployment

3) Ubuntu OS ⇒ Ubuntu 22.

1) Ubuntu 22 VMs from AWS or Gcloud with 2 VMs with 2VCPUs 4GB RAM minimum.

* Both have same security groups.
  - Inbound rules => custom TCP (6443)

2) Allow port 6443 on both VMs
3) Connect to VM
4) sudo su
6) apt update (only installed files)
7) apt -y full-upgrade (patches)
8)
9) Connect directly from instance.



Minikube.

1). App
2) ec2 instance.

→ launch instance with 2cpu, 4GB ram (t2 medium)
launch 2 instances.

→ Security groups
custom tcp port → 6443 save changes.

→ connect → SSH → connect

- sudo su
  apt-get update
  curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube_latest_amd64.deb

  dpkg -i minikube_latest_amd64.deb
  apt install docker.io
  minikube start --driver=docker --force
  snap install kubectl --classic
  kubectl get nodes.

↳ alias kubectl="minikube kubectl --"

K8s → pods → have internal IP (called as service)

- pods won't have state. Pods are replaced frequently
  pod → new pod → new IP · (creates problem).

⊙ Kubectl get pods // to know about pods.

⊙ Kubectl get services -o wide // -o gives more info
IP

⊙ Kubectl create deployment my-app --image=httpd

- Abstraction layer ⇒ deployment ⇒ pod.
- file - yaml.

⊙ Kubectl get replicaset
⊙ Kubectl get replicaset -o wide.
⊙ Kubectl edit deployment name.

deployment
pods → replicas
containers

deployment manages
replicaset manages
pod manages
containers

⇒ configuration files - K8s
  yaml
  strict in indentation.
  (online yaml validators/editors)
  ↓
  developers before deployment writes these type of files

api version
kind
- metadata (info about anything like file), name == label
  specifications
  state - K8s

⇒ template => blueprint for container.

⇒ Kubectl -f apply nginix.yaml

Kubectl apply -f nginix.yaml

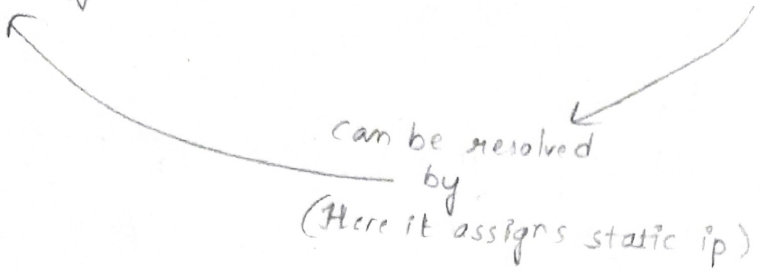
⇒ Kubectl get deployment
  Kubectl get deployment -o wide

- Kubectl get all

match label : app : httpd (wherever app is there we need to have httpd only). - deployment & service same

2 - replicas → 1) 172.10.90.10 (pod created) ← 172.10.90.12 3) something application can't be accessed.
2) 172.10.90.11 (another pod)

- Service (yaml file)



- Container Port of deployment file = Target port of service file.
- Port starting from 30000 (≥ 30000) - 35000 for Kubernetes cluster.

minikube → deployment → container

- static ip, (desired, current & ready)
  2 2 2
- https ⇒ 443/TCP (secure)
- minikube service httpd-service.

Target port	URL
80	http://192.168.49.2:30000 (private)

- can be seen by curl

curl http://192.168.49.2:30000 (just to check
<html><body><h1>It works!</h1></body></html>

- kubectl port-forward --address 0.0.0.0 service/httpd-service 30000:80

3.89.129.145 : 30000 (node port)
public ip

should be allowed in ec2 instance

Yaml files

AWS cloudformations

- repeatable.

* Same config → developers require 5 inst → 5 instances launced → closed

→ again testers require 5 inst → 5 instances launced → closed

(repeated task)

- parameters
  resources
  output

(Meta data)

- YAML validators.

Cloudformation

Iac → Infrastructure as a code service.

- cloud-formation template – ec2

  IAM role. (permission for 1 resource to talk with another service)

  [need to have a role that gives full permission]

⇒ IAM role (create role)
↓
AWS service.
↓
• cloudformation (use case)
↓
next
↓
AWS EC2 Full Access.
AWS VPC Full Access.
↓
Next
↓

↓
role name
↓
Create role.

Policy (getParameters)
↓
create policy
↓
JSON (code)
↓
next
↓
name
↓
next

my-policy

Stacks
↓
create stack. (1 AZ, 1 region). ⇒ stack
(any AZ, any region) ⇒ stackset.
same resources
• can use diff accounts

prepare template
⊙ template is ready
↓
⊙ upload a template file. ⊙ Amazon s3 url.
↓
Json/yaml.
↓
next
↓
stack name
↓
Instance type. //t2.nano
↓
Key name //in repository
↓
roll name

Bind...

↓

⊙ Roll back all stack resources. // error deletes all resources.

⊙ Preserve // ignore errors

↓

next

↓

submit

JSON (Java script object notation).

{ } ⇒ object starting & ending.

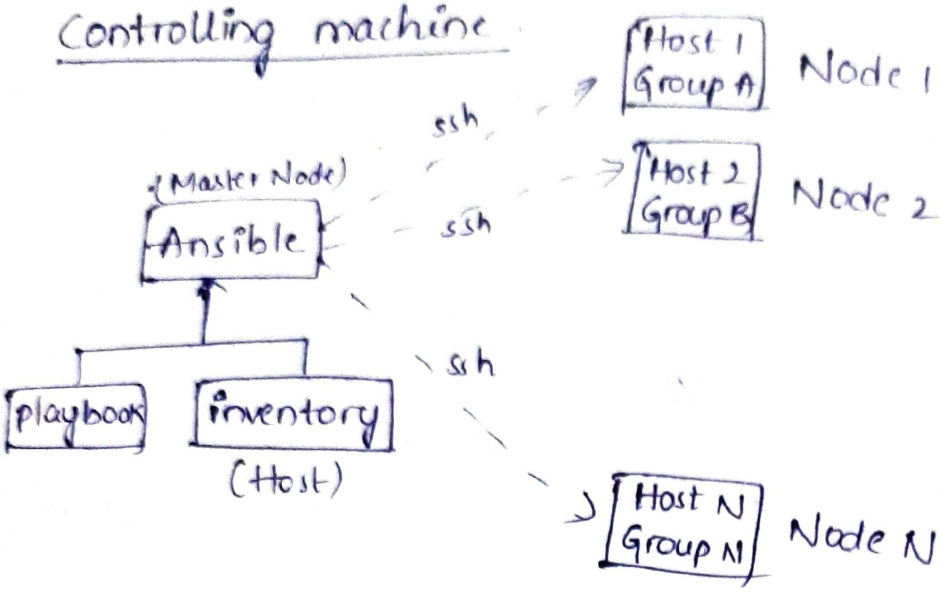
, ⇒ To write about more objects.

* JSON validator

Ansible

* Two types of configurations
  1) H/W ⇒ RAM, ROM, harddisk
  2) S/W ⇒

* Controlling machine



* agentless.
* ~~for~~ kubernetes, Agent is kubectl (to see what's happening inside it)
- Master ⇒ t3.micro (
  nodes ⇒ t3.micro

Super user do (sudo)

Launch Instance

- Open Mobaxtreme (mast node)
- sudo su
- apt ~~get~~ update
- useradd -m devops } create
- passwd devops
- su -devops (switch)
- whoami [to check user]
- sudo apt update
  - not in sudoers file
- exit
- sudo visudo

%sudo ALL=(ALL:ALL) ALL
devops ALL=(ALL:ALL) NOPASSWD:ALL
(system user acts as root user)
ctrl+O enter
ctrl+X

- su - devops
- sudo apt update
- ls
- cd etc
- cd ssh.
- ls (sshd_config)

(i) → INSERT

It is easy to connect user to root

vi /etc/ssh/sshd_config

PermitRootLogin yes (If # , remove.)
PubkeyAuthentication yes
PasswordAuthentication yes.
esc : wq! enter.

- service ssh restart
- systemctl restart sshd.

Master node

- cd/
- come to root
- ssh-keygen

3 enter → /home/devops/.ssh.

(check once)

- cat id_rsa.pub ← id_rsa id_rsa.pub

Copy

Connect with slave node.

Slave

- su - cloud
- pwd
  /home/cloud
- To see ssh folder
  sudo visudo → same pro
- vi authorized_keys
- paste → error → esc :wq!
- sudo touch authorized_keys
- ls -al
  (check)
- sudo chmod 777 authorized_keys
  (read write execute)
- vi authorized_keys
  paste

  esc : save

Ansible (master node)

sudo apt-get update -y
sudo apt-get install software-properties-comma
sudo apt-get

- sudo apt-get install ansible -y
- cd /etc/ansible (Inventory)
- sudo chmod 777 hosts
- vi hosts
  come down
  [dev]
  slave1 ansible_ssh_host = public ip

esc : shift
:wq!

Java-11,17,21
Jenkins
python
Ansible

- ansible dev -m ping
  Unreachable
- host files are located in /etc/~~ssh~~ansible/hosts (Inventory)

⇒ Add node public ip in hosts (vi hosts)

- ansible -m ping dev
- ls
  service.yaml
- vi service.yaml

hosts:dev ⇒ In yaml file

- paste

:q!
:wq!

ssh keys
public ips
private ips
(communica
· no agent

- Run playbook ⇒ ansible-playbook service.yaml

⇒ apache server installed or not?
/var/www ⇒ ls (html)

---

master server

```
sudo apt update -y
sudo apt install python3 (dependencies)
sudo apt-get install software-properties-common
sudo apt-add-repository ppa:ansible/ansible
sudo apt update
sudo apt install ansible
```

python3 interpreter

```
ansible --version
cd .ssh
ls  authorized_keys
ssh-keygen
ls  id_rsa  id_rsa.pub
cat id_rsa.pub
```

cd .ssh
ssh-keygen

/root/.ssh/id_rsa.pub

slave:
sudo apt update
sudo apt install python3

cd .ssh
vi authorized_keys
delete the key
Insert the key we copied.
:wq!

⇒ cd/
cd etc
ls ansible
cd ansible
ls -al
sudo chmod 777 hosts
vim hosts

/etc/ansible/ → hosts.

to change permission for hosts.

last
[test] (created group)
slave1 ansible_ssh_host = slave server public ip.
name
:wq!

cd ~ cd/
ansible -m ping test (group name)

check /var/ls www

- yaml file.
  hosts: test
  put inside master server.
- sudo vi service.yml
  paste yaml file content
- ansible-playbook service.yml

in root

1) install apache2 in nodes

/var/www/html
- sudo rm index.html
- vim index.html
  Welcome page
  :wq!
- file
  src:
  dest:

put webpage into all nodes.

delete in slaves index.html

sudo vi thub.yml

paste template

ansible-playbook thub.yml

2) change masternode & give to every node

Jenkins:-

Monitoring:- Observing the behaviour of servers or load balancers, auto scaling, Vpc

Connect

1) top (task manager in linux server)

(task manager)

- ~~For~~ GUI, monitoring tod - nagios, data dog

  AWS - cloud watch.

  Azure - azure monitor

SNS → simple notification service

Cloud alarm → alerts us.

- Create new dashboard
  ↓
  name

SNS service:-

notification for mobile/email.

create topic (~~name~~)
↓
Next
↓
standard
↓
Next

- Subscription.
  ↓
  create.
  ↓
  protocol (email)
  end point (give email)

Create an alarm (cloud watch).

↓

Alarm notification (mytopic)

↓

Alarm action (stop)

↓

Alarm thresholds.

↓

CPU utilization

80%

5 minutes.

. Average >= 80

→ apt update

apt install stress