Day 1 - devops :-

Devops is a combination of pratices, tools and cultural philosophy aimed at improving the collabration b/w the Software development and IT operations.
It help teams deliver the Software faster and more reliably by breaking down barriers b/w the development and operations.

It is like a bridge b/w the Software developers who write code and
the IT operations team who mange and deploy the application to kep them run Smoothly.

How Devops works :-

⇒ Automation :- Tools like Jenkin, docker, kubernets automate repetative task and make the deployment faster and easier.

⇒ CI (continious Integration) :- developers usually Integrate the code changes into the shared repository.

⇒ C/D (continuous deployment) :- Automated process test and deploy the code, ensuring the Quker releases.

⇒ Monitoring :- It is Important to keep track and monitor the System Performance to catch the errors Quickly.

Why devops ??

Before 10 years

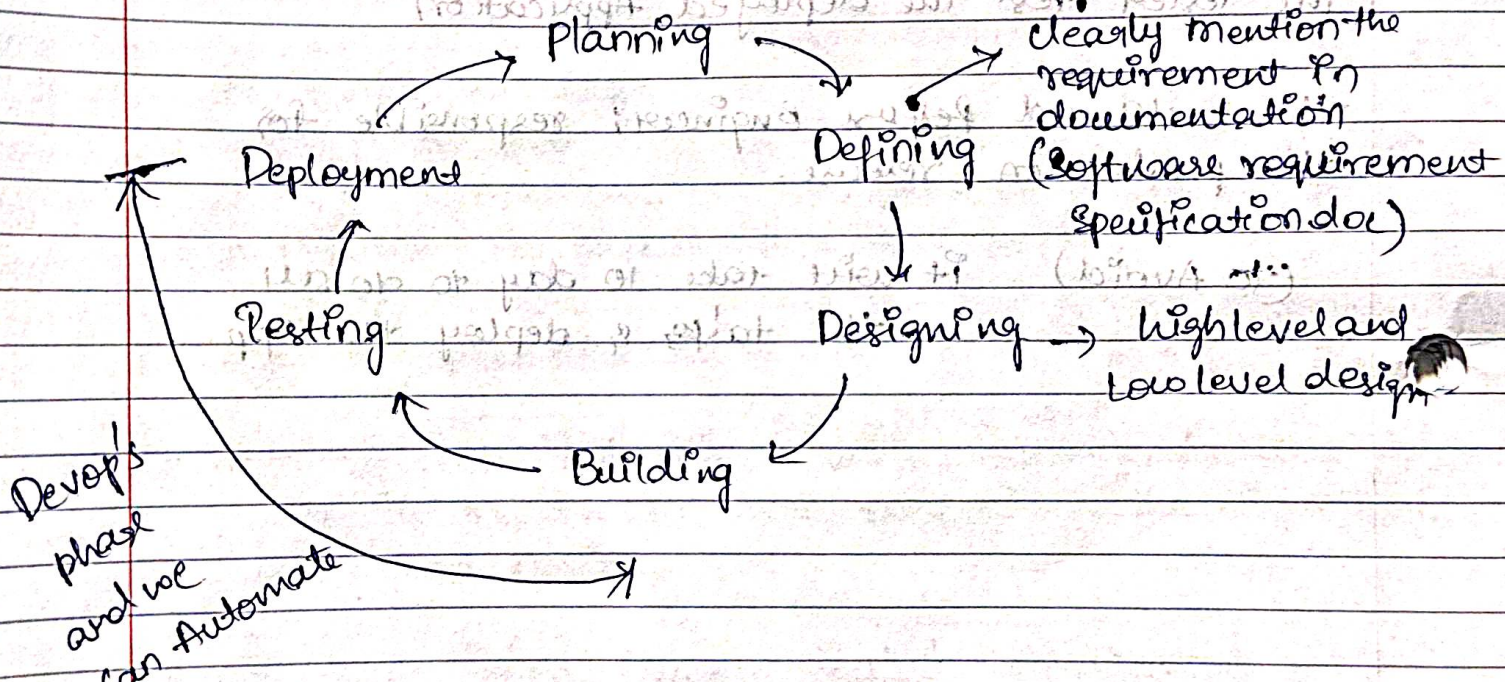
- the developer shared code in centralized location
- the System Admistrator used to create the Server and deploy the Application
- the Tester test the deployed Application
- the Build and Release Engineer; responsible for the production release.

(to Avoid) It will take 10 day to do all the tasks & deploy the App.

Software development life cycle (SDLC)

SDLC is a set a rules followed by the software industries to develop High Quality product

⇒ Design
⇒ develop
⇒ Test



Building → developing the code

Software dev push the code to Git

testing → the code is deployed on the server
is tested by QA team.

Deployment → promote to the production
and release to customer.

Devop's Engineer will Automate the Build, test and deployment phase with out any delay / or human intervention and delivers Quickly & increase the Efficiency.

Concept of Virtual Machines :- Hypervisor

Virtual Machine is like a computer inside a computer. It's a software based simulation of a physical computer that runs it's own operating system and Applications Just like a real computer.

Imagine

No. I have 1 big Physical computer. On this computer, I created Multiple mini-computers (Virtual computers) each with its own operating System files and Apps.

this VM's are Isolated from each other with out effecting other Virtual Machines.

we USE VM's to

- ⇒ Save the Resource.
- ⇒ Experiment Safely
- ⇒ Run different Operating Systems
- ⇒ Easier Maintenance.

Hypervisor :- A Special Software is used to Manage the VM's It Allocates the Memory, CPU and Storage from Physical Computer to VM's

there are so many cloud platforms such as AWS and Azure to get a virtual machine.

But whenever A Devops Engineer get 100 request from team to create 100 VM's.
He cannot make manually one after the other
It will take a lot of time.

Devops is All about Automation and Increasing Efficiency.

We can write script to Automate this process of creating Virtual Machine's through API using

AWS CLI
AWS API
AWS CFT cloud formation templates
AWS CDK cloud development kit.
Terraform
↳ open source
and can Used to handle different cloud platforms.

When you make a API call to create a VM to any cloud platform It will check. (Validate)
Authenticate (UserId & pass)
Authorize (user role to Allow Action)

In AWS EC2 Instance are the Virtual Machines

Operating System Acts like a bridge between the Software And hardware.

Suppose we have hardware (CPU & Memory) and if we want to use Software Like Gaming or Java/python we need a OS to talk b/w the hardware and Software.

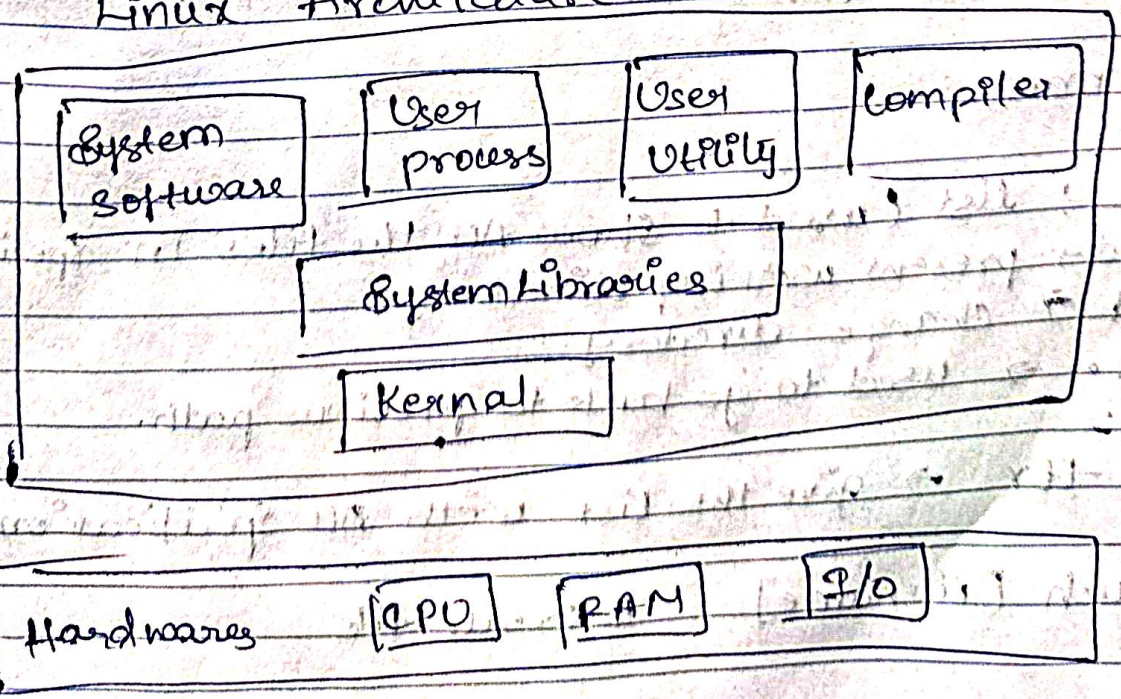
Ex :- Windows, Linux, ~~Ubuntu~~, Mac OS

⇒ Linux is a free OS and open source Software.
⇒ It is Secure and not required to Install Any additional Anti Security Softwares.

⇒ ~~As~~ has lot of distributions (, Ubuntu)

⇒ fast O.S

Linux Architecture



Kernal is a central part of an operating System.

It manages the System resources and Allows Software to communicate with the hardware.

Kernal handle tasks like :-

⇒ Memory Management
⇒ Device Management
⇒ Process Management
⇒ handle System tasks ensuring everything works together efficiently and correctly.

Fundamentals of Shell Scripting :-

Script :- A text file containing a Sequence of commands.
Shell :- A program that take your commands and communicate with the operating System to perform the task.

Commands :-

ls :- list (used to show All the files in Specific directory)
pwd → present working directory.
cd → change directory.
cd .. → used to go back to previous path.
ls -ltr → give the list with All Specifications.

touch [filename] to create file

vi [filename] → to open editor to write into file.

&

click esc and i to shift into insert mode in file.

click ~~esc~~ esc and ~~key~~ :wq! to save the file content

cat [filename] used to directly read the file on terminal

mkdir [directory name] Used to create directory.

rm -r [directory name] Used to remove the directory.

nproc → command used to check the No. of CPU's

free -g → command used to check the Memory

df - h → disc size

top → command used to check CPU, Memory

df → command used to get All the Info.

q → used to quit the process.

man ls

this command Give the manual regarding list (ls)

Vi [file name] → this command can also create and file.

But it is not recommended in Automation because opening too many files on creation will leads to the System crash.

Mostly touch command used to create file in the Automation process.

#! → this is shebang in Shell Scripting.

#! is a special character at the very start of the Shell Scripting that tell the operating System which interpreter to use to execute the script.

If your script is written for bash (#!/bin/bash) makes sure it runs with bash, even if the user's normally uses a different shell like zsh or sh

#!/bin/bash
dash
sh
Ksh
python3

} this are all interpreters.

#!/bin/sh is used as (linking concept) to redirect to bash but recently it is redirected to dash

#!/bin/sh → #!/bin/bash ✗ (not working)

#!/bin/sh → #!/bin/dash ✓ (this works)

echo "hello badri" → it is print statement.

sh <filename>
./<filename>
} this commands used to execute the scripting file.

It can execute all the files (.java, .py, .cs)

before execution the shell check for the permissions. (Security)

chmod [root user permission] [group permission] [All users permission]

Shell use 4 → Read
2 → write
1 → execute

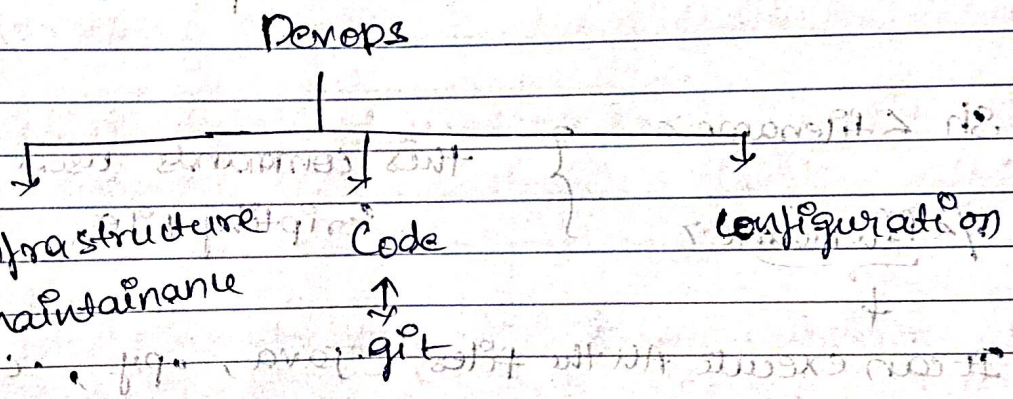
chmod 777 <filename> this will change the permission of the file and remain accessible to everyone.

Chmod 444 <filename> ; will Allow only to Read

history → command to see All the previously used commands.

is used to write a comment.

why shellscripting for devops?



Suppose if a oraganization has 10000 vm's and are making issues. A devops engineer can write a script and automate it to diagnose the vm's and recieve a detailed report as an email.

format of writing a shell script

```
#!/bin/bash

# Author: Badrinadh
# Date  : 08-01-2025
#
# Version : v1
```

Set -x # debug mode → this will clearly display the command on the output and the response of the command.

df -h
free g
nproc

Ps -ef → this command will give All processes running on the System.

Process Status

grep → global regular expression.
to find a specific group of process we use grep

Ps -ef | grep "Amazon"
this give All Processes related to Amazon.

| is a pipe command

Ex ps -ef | grep "amazon

① ②

the pipe send the output of ① to ②

date → this command give the current date

date | echo today is

this only print today is the pipe will not send the data. because the ~~dead~~ date is the shell default command and return to stdin.

suppose namefile.sh has

1 2 3 4

My Name is badrinadh

I am a student

from this we need to (take) print only badrinadh.

grep name namefile | awk -F" " '{print $4}'

this give Column 4 value.

set -e # exit the script when there is an error.

set -o pipefail.

command → set -o pipeline

we can also write All 3 in same command

set -eo pipeline

curl → client url

if we want to get any data from the internet we use curl

curl <link> | grep <condition>

It will take file from server and filter the content based on the condition.

another similar comand

wget <link> <file name>

this will download the file on to the system.

Sudo is used to go the root (main ~~directory~~) user

Sudo su - is used to switch the user

to search some files we use find command

find / -name <filename>

If else in shell

if [condition]
then
else
fi ⟶ this will end if block

trap command. → used to trap signals.

We can Use Amazon CLI to connect to the EC2 instances.
we can also create the EC2 Instance, S3 buckets and many more using Amazon CLI

∴ If need more information → go to Amazon CLI docs.

Q. A Devop's Engineer asked to Send the report of EC2, S3, Lambda ... etc to reporting dashboard.
Devops engineer can write a Shell Script to automate this process.
and if the report should be submitted at a particular time. Like Suppose 6:30 pm every day.

the Devop's Engineer will take the help of "CornJob" to execute the Script at 6:30 pm and Send the reports.

to make the ~~file~~ Script file output formatted use command

./filename.sh | more
↳ this will Beautify Code.

to get a Specific information from JSON data we use Json parsing. → Jq (command)

aws ec2 describe-instances | jq '.Reservations[].Instances[].InstanceId'
↳ this is the flow in JSON

In Reservations there are Instances[] and In Instances[] there is an InstanceId.

Git and GITHUB

Version control System :- is a Software tool that tracks changes made to code over time. It Allows developers to

Manage multiple version :- Software developer can keep multiple versions of the project. So that they can easily update the code for the better User experience.

Collaborate efficiently :- Multiple developer's can work on the same project. Sharing A same code base with out any conflicts.

Rollback changes :- Can easily go back to earlier released version if there is any errors in the current release.

there are 2 types of Version Control System. (VCS)

Centralized VCS	Distributed VCS
* Single central repository	* Each user has a full repository copy
* Repo Location is Server-based	Repo location is local and remote repositories.
Collaboration → Sequential chek-in & Chek-out	Concurrent Branching and Merging.

Git is a distributed Version Control System.
SVN (Sub-Version Software) is the Centralized VCS

Git is Used by the developers for Large Collaborative proje
while SVN is used for Smaller and More Centralized proje

When Someone wants to contribute

they clone the repo
push the code changes.
Gen other can pull the code

If there are multiple changes, developers merge them
to Integrate everyones work.

So If a developer want to make changes to the main
branch with out effecting it
he can create the branch
commit the changes
test
and merge to the main branch.

Understanding GIT

GIT is a distributed Version Control System.

features of GIT :

Creating Repos
tracking changes
committing code ... etc.

Git commands

Git init → initilize the Git

git add . → Add All the files to the remote Server

git commit -m "message" → commit the changes

git push -u origin main → it will push the code to repository

git pull → we can download code from github

git status → we can get the repo status.

this are the basic commands and there are more to learn.

Github and Git are different.

Git → command Line tool for Version Control.

Github → web-based platform that builds on GIT and provides

Additional collaboration features like Issue tracking and project management

Git branching :

It is used for the importance of timely software releases to keep customers happy and ensures frequent delivery of new features.

Branch is created when we want to Add extra feature with out disturbing the functionality of main code (production code).

After testing the Branch code is merged to the main or master branch.

Types of Branches

1) Main / Master Branch
2) feature
3) Release

Kubernetes uses a Structural branching Strategy Allowing effective management of contribution from thousands of developers to deliver timely updates.

Git Interview Q/A

to initilize a local repository

⇒ Git init

to check All the folders and to verify the git is initialized

⇒ ls -a or ls -ltra

↓

means All files including hidden files.

If you have Any Secret passwords and accidently you pushed to Git.
to Avoid Such cases we use (pre-commit)
It will avoid pushing the secret information like passwords.

to check the Status of git

⇒ git Status

to Add the file to the git

⇒ git add <filename> to Add Specific file

⇒ git add . to Add All files in repo folder

to check the changes in the repo

⇒ git diff

⇒ git checkout <filename>

It will remove the changes in the file.

⇒ git status to check the status

⇒ git commit -m "message" // to commit the repo

⇒ git log to get the commits information

⇒ git clone <repo path> // download the remote repo to your local machine.

and git will use the same path to send the changes which you made to in the code.

⇒ ~~git add .~~ we can use the HTTP's or SSH paths

when you use SSH path you need a SSH key (public)

to generate one SSH key

⇒ ssh-keygen -t rsa

this will generate a public key

and after this we need to add the key in github repo settings.

to copy some one's code repository use command

→ git fork

this will create the copy in your Github and you can make changes and fix the bugs.

and make pull Request

then your changes will added to the original repo from which you copied the code.

→ git checkout <branch name>
this is used to switch between the branches.

git checkout -b <branchname>

this will create a new branch

git branch → this command will give All the branches

git log --one line → this will give All the log info in one line.

git log <branch name> this will

git merge <branchname>

* Configuration Management with Ansible. -

+

this is concept of handle Multiple Servers on the cloud

+

Ansible is the tool used to implement the concept

⇒ Puppet
⇒ chefs
⇒ Ansible
⇒ Salt

all are tools.

Ansible works on push mechanism

↳ Agentless
Simple YAML Syntax.

Idempotent ÷ Ensure that task only make changes when necessary.

Scalable ÷ Manage Multiple Servers simultaneously.

Disadvantages ÷

Windows Configuration
Debugging
performance

Ansible Galaxy is used to share the Modules b/w two organisations.

Python language is used by Ansible.

Ansible support Both

Windows → protocal (win RM)
Linux → SSH

Why Ansible over other tools?

Ansible is push mechanism.

from Ansible we can connect to All the cloud Servers like GCP, AWS, Azure.

Using the Public IP Address
SSH or winRM

Practical

connect to EC2 Instance and start linux OS

Enter Sudo apt update → Updates All packages.

then sudo apt install ~~AS~~ ansible

the verify ansible --version

SSh. - key gen to generate a key.

~~SSh~~ - SSh-keygen

/home/ubuntu/.ssh ~~id_rsa~~ at this Address the file will be saved.

copy the public key id_rsa.pub

Now connect to target server from other terminal and generate Secret key.

Now paste the copied public key in authorized_keys file.

your Almost done.

In Ansible files are called playbooks.

Ansible adhoc commands are only 1 or 2 tasks.

Ansible playbooks are used if there are too many tasks.

first we need to create an inventory file.

Save the private Ip address of the Target Servers.

Run command

ansible -i inventory all -m "shell" -a "touch Badri"

inventory → file location

"touch Badri" → this creates file in target server.

If we have too many Server public Addresses in Inventory file we can use the grouping to separate the IP Addresses

[dbserver]
172.83.542.23

[webserver]
184.24.52.67

ansible -i inventory webserver -m "shell" -a "touch badri".

webserver → this is a grouping Name.

Command to Run playbook

Play books are written in .yml files & saved

ansible-playbook -i inventory first-playbook.yml

→ playbookname

⟹ Vim first-playbook.yml

once after opening the file.

write

--- → this indicate the yml file

- name : Install and start nginx. (playbook name)
  hosts : all
  become : root
  tasks :
    - name : Install nginx
      apt :
        name : nginx
        state : present

} this is similar to sudo apt install nginx

    - name : start nginx
      service :
        name : nginx
        state : started

sudo systemctl status nginx. →

this command is to verify the Installation of Nginx.

when ever there are more tasks to implement writing in a one play book will be confusing

So to Avoid that we create roles

⇒ ansible-galaxy role init kubernetes

this will create a Role called kubernetes.

Infrastructure As a Code :-

So if a company is using Mutiple cloud Services like AWS, GCP and Azure. it is hard to write 3 different Scripts to Automate

and the devops Engineer should Learn 3 cloud Services Script writing.

Instead, to Avoid this we Use Terraform. through which we can use one Script for Many cloud Servers. the terraform will take care of the Script changes based on the Server.

Terraform is developed by Hashicorp.

work on the principle of "API As code"

where User writes terraform Scripts; then terraform will convert the Script and generate API to talk to cloud server and return the response via API.

Advantages of Using Terraform

⇒ Manage Any Infrastructure (AWS, GCP, Azure)
⇒ Track your Infrastructure → Terraform maintain State file - this will keep track of deployed Resources.
⇒ Automate changes
⇒ Standardized configurations.
⇒ Collaborate

Terraform life cycle:

1⇒ We need to write the terraform file.
2⇒ dry run to check the errors (Automatic)
3⇒ Apply to connect to the source (AWS, Azure or GCP)

AWS cli
first AWS cli should be Installed

Use the DigitalOcean website to Install terraform in Ubuntu.

write a file .tf it contain the code related to terraform.

after this, write "terraform init"

terraform plan

terraform apply.

So it is a best practice to avoid keeping the terraform.tfstate.

It contain All the sensitive information.

Basically the statefile is stored in Amazon S3 which is a untralized server.
But if 2 users want to Access the teraform at the same time, it will create a error in execution.
to Avoid we use dynamodb in Aws to Allow only one User At a time.

In the remote state we have 1 database and S3 configurations.

Understanding CI/CD

Continious Integration :- Automate the Integration of code changes into the shared repository.

It Automates the proess such as Unit testing, code compilation and static code Analysis.

Continious delivery :- Automates the deployment proess after Sucessful Integration.

After passing All the test's the Application is deployed to a Staging environment. and Await Manual Approval for the production deployment.

Suppose A developer writing a code for Application on his Local Machine (laptop) and the Customer is using the Sam Application in America (some part).

the developer will push the code to the github / gitlab.

CI/CD pipeline trigger's

Unit testing → Verify the funtionality of Individual code Unit.

Static code Analysis : check the code formatting, Syntax and potential Issues.

Code Quality and Vulnerability testing :- Ensure the Security Standards.

Automation testing : perform the functional and end to end testing

Reporting : Generate the report of test coverage and quality reports.

Deployment : deploy the Application to desired Environment.

Jenkins is the open-source Automation Server that orchestrates CI/CD pipelines.

developer write the code → Push → Github/Gitlab → triggers Pipeline → Jenkin does this CI/CD Actions.

Jenkin acts as a controller, integrating multiple roles:

Maven : Build Java Applications
JUnit : Runs Unit testing.
SonarQube → Reporting
Kubernates/Docker → Deployment

Modern CI/CD

Github Actions Event driven Automation Built in Github

⇒ Scales the Resources dynamically
⇒ Zero Infrastructure Cost ; open source
⇒ Support Multiple project pipeline.

Gitlab pipelines :- Integrated CI/CD within GITLAB.

Environment promotion in CI/CD.

⇒ Development Environment :-
⇒ Staging Environment (Pre-Production Stage)
⇒ production Environment → 3 Master Nodes and 30 worker nodes.
→ 3 Master Nodes 5 Worker Nodes

Scalability challenges :-

Jenkin :-
for new project Jenkin require additional worker nodes
Managing Multiple Jenkins become costly and complex.

Physical servers often underutilize the resources like RAM, CPU and hardware.

Hypervisors are used to create the multiple virtual machines on the one physical server.

But even with the virtual machines, the resources are not fully utilized.

containers solve this problem by being light weight and sharing the host operating system resources.

Docker is highlighted as popular containerization platform.

containers can be created on both the physical servers and virtual machines.

Docker simplifies the creation of containers through Docker files which define the environment and dependencies

Life cycle

write the dockerfile
Build the image using "Docker Build"
Run the image using "Docker Run"

Docker Engine is the potential single point of failure; if it goes down all the containers will stop working.

Builder tool is an alternative to the Docker to Address the docker limitations.

- Solve layer Issues In Docker Image.
- Avoid Single point of failure by not relying on a central Engine.
- Integrates well with kubernates, podman.

Logical Isolation in containers :-

the container should have some minimal system dependencies to maintain logical Isolation.

with out the logical Isolation the hacker can access All the containers on a cluster.

container specific files :-

- /bin → binary
- /sbin → System binary
- /etc →
- /lib
- /user
- /var
- /root

Docker Architecture

Docker client → Sends commands via CLI
Docker daemon → Executes commands received from CLI
+
It's the Heart of the Docker, if it goes down, All the containers will stop working.

Docker registry → place to store and share docker images.

For any Project to create a Docker container, we need to write a Dockerfile that configures the Project dependencies and required folders etc.

docker build -t <image name>

docker run -d -p '8080:80 <image name>

Multi-Stage Docker-Build :-

the technique to optimize the docker images by splitting the build process into multiple stages.

this will reduce the final size by excluding unnecessary build dependencies.

Stage1 → Build Stage

Use the rich base image (Ubuntu) to install dependencies and build the Application.

the Stage→2 final Stage.

Use a minimal base image (Python) And copy only the required artifacts from Stage1

this will reduce the size and increase the security.

Distroless image → it is a minimalistic image contain's only runtime environment needed to Run the Application.

it has No shell, curl or wget or other utilities.
Extremely small size

this will eliminate the Vulnerabilities associated with unused packages.

Small size → faster transfer and deploy.