1. What is DevOps?

➔ Implementing automation at each and every stages. The objective of DevOps is to shorten the SDLC.

  DevOps is a methodology that allows a single team to manage the entire application development life cycle i.e. development, testing, deployment and operations.

2. Why DevOps is needed?

➔

- Fast Delivery
- Higher Quality
- Less Capex (capital expenditure) + Opex (Operational expenditure)
- Reduced Outages

3. What is build?

➔ Software artifact generated after converting the source code into executable code that can be run on a computer.

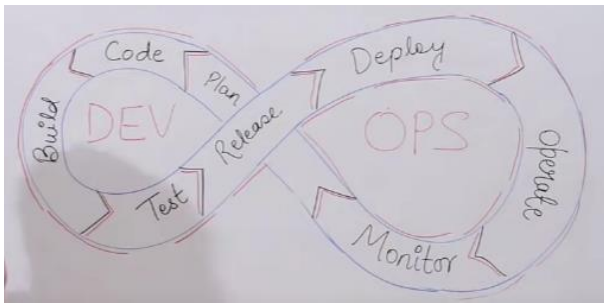
4. What are DevOps Stages?

➔ There are mainly 4 stages:-

- Version Control – Maintain different version of code
- Continuous Integration – Compile, Validate, Code Review, Unit Testing, Integration Testing
- Continuous Delivery – Deploying build app to test servers
- Continuous Deployment – Deploying the test app on the production server for release

5. What are the stages of DevOps Lifecycle?

➔ Integration of
  Continuous Development – Plan and Code
  Continuous Testing – Build and Test
  Continuous Deployment - Deploy and Operate
  Continuous monitoring - Monitor



6. What is IaaS?

➔ Infrastructure as a service (IaaS) is a type of cloud computing service that offers essential compute, storage and networking resources on demand, on a pay-as-you-go basis.

7. What is Cloud? What are the terms in AWS?

➔ "The cloud" refers to servers that are accessed over the Internet - software and databases that run on those servers, on a pay-as-you-go basis.

Region – Country
Availability Zone – Data Centers in a city

8. What are the feature of Linux?

➔ Open Source, Secure, Light Weight, Multiuser – Multitask, Simplified updates for all installed software, Multiple distribution – Redhat, Fedora, Debian

9. Explain File System Hierarchy of Linux.

➔ / - Top level root directory
/root – Home directory for root user
/home – Home directory for other users
/boot – It contains bootable files for Linux (To identify issues and make files in active states)
/etc – It contains all configuration files (Processor, Hardware, Software Information)
/usr – By default all software are installed in this directory
/bin – It contains commands used by all users including root user
/sbin – It contains commands used by root user
/opt – Optional application software packages
/dev – Essential device files. This includes terminal devices, USB or any other device attached to the system

10. How to create a file in Linux?

➔

- **CAT** :- Cat command is one of the most universal tools, yet all it does is copy standard Input to standard Output.

➢ Create file - Creates a single file
Cat > file_name (ctrl + d)

➢ Concatenate file – To add more than one file into a single file
Cat >> file_name (ctrl + d)

➢ Copy file – To copy the content of FILE1 to FILE2
Cat file1 file2 > file_name

➢ TAC – To see the content in bottom to top

Limitations:-

➢ We can't be able to edit the created file using CAT command.

- **TOUCH** :- Touch command is used to update/modify access time and to create empty files.

➢ Create an empty file
Touch file_name

➢ Create multiple empty files
Touch file1 file2 file3

➢ Change all timestamp of a file
Touch file_name (file_name already present)

Stat file_name – will give all 3 information mentioned below

▪ Access Time – Last time when a file was access will be changed to current time
Touch -a

▪ Modify Tine – Last time when a file was modified will be changed to current time
Touch -m

▪ Change Time – Last time when file metadata (file info) was changed will be changed to current time.

➢ Update only access time of a file, modify time of a file

Limitations:-

➢ Touch command creates empty files and not allowed to write when created

- **VI** :- VI is standard and programmer text editor. It can be used to edit all kind of plain text, it is specially useful for editing programs, mainly used for UNIX programs.
  Esc + :w – To save
  Esc + :wq or :x – To save and quit
  Esc + :q – Quit
  Esc + :!q – Force quit without saving

- **NANO** :- Nano is a text editor.
  Nano file_name (ctrl + x and Y for saving)
  Ctrl + o – over writing

11. How to create directory in Linux?

➔

- **Mkdir** :- To create a directory in Linux
  mkdir dir1 or mkdir dir1/dir2/dir3 (subdirectory)
- **CD** :- To Change Directory
- **PWD** :- Present Working Directory

12. What are different operations to be performed on file/directory?

➔

- To Create hidden file – touch .file1
- To Create hidden directory – mkdir .dir1
- To copy a file – cp source destination
- To move/ cut and paste – mv source destination
- To rename a file – mv old_name new_name
- To remove/delete a empty directory – rmdir dir_name or rm -r dir_name
- To remove/delete recursive directory – rmdir -p dir_name or rm -rp dir_name
- To remove/delete directory with verbrose – rmdir -pv dir_name or rm -rf dir_name
- To display first 10 lines – head file_name (q quit)
- To display last 10 line – tail file_name (q quit)
- To display next set of data – more file_name (q quit)

13. List few Linux commands.

➔

- Hostname – Details of currently logged in machine
- Hostname -I – To display only Machine IP address
- Ifconfig – To check Machine IP details
- Cat /etc/os-release – To display present OS version

- Yum install package_name – To install any package using yum (yellowdog updater modified)
- Yum remove package_name – To remove/un-install any package
- Yum update package_name – To update any installed package
- Yum list install – To list all installed packages
- Service package_name start – To start a service after installation
- Service package_name status – To check the status of a service
- Chkconfig package_name on – To start service automatically after machine start
- Chkconfig package_name off - To stop service from starting automatically after machine start
- Which package_name - To check package is installed or not
- Whoami – To display current logged in user
- Echo – To display message to other users (in script or in terminal)
- Grep file_name location– To display files/directory using pattern matching
- Sort – To display file content in ascending order (q quit)
- Ln – To create hard link (backup – in sync and accessible after original is deleted)
- Ln -s – To create soft link (shortcut – in sync and not accessible after original is deleted)

14. List Linux commands for User creation.

➔

- Useradd user_name – To create user
- Cat /etc/passwd – To list created users
- Groupadd group_name – to create group
- Cat /etc/group – To list created user
- Gpasswd -a user_name group_name – To add user in a group
- Gpasswd -M user1,user2,user3 group_name – To add multiple user in a group

15. List Linux commands for zipping files.

➔

- Tar -cvf destination source – tar is an archival tool used to combine multiple files into one
- Tar -xvf dir_name – To extract files from the combined file
- Gzip dir_name – gzip is a compression tool used to reduce the size of a file
- Gunzip - gzip is a de-compression tool used to reduce the size of a file
- Wget URL – It is the non-interactive network downloader

C – create
V – verbrose
F – forcefully
X - extract

16. List Linux commands for file/directory Access and Permissions.

➔

- Chmod – Used to change the access mode of a file or directory
- Chown – Change the owner of file or directory
- Chgrp – Change the group of file or directory

d	rwx	r-x	r--	1	Root	Root	0	March 20 08 10	Dirc1

File/dirc	Owner/root	group	others	link	User	Group	File size in bytes	Date/time	Director name

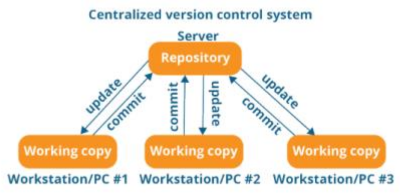
R → 4 → Only to display the content of file | To list the content of directory
W → 2 → To modify the content of file | To create or remove file/directory
X → 1 → To execute the file | To enter into directory

17. What is centralized version control system (CVCS)?

➔ Centralized version control systems are based on the idea that there is a single "central" copy of your project somewhere (probably on a server), and programmers will "commit" their changes to this central copy.

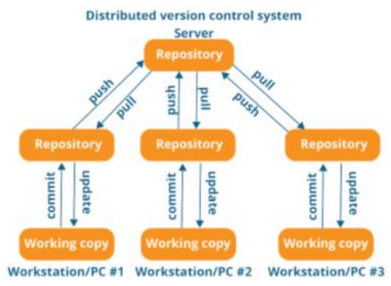
Drawbacks:-

- It is not locally available means you always need to be connected to a network to perform any action.
- Since everything is centralized, if central server gets failed you will lose entire data.
- E.g :- SVC Tool



18. What is distributed version control system (DVCS)?

➔ In Distributed version control system every contributor has a local copy or "clone" of the main repository i.e. everyone maintains a local repository of their own which contains all the files and metadata present in the main repository.



19. CVCS vs DVCS?

➔

Sr	CVCS	DVCS
1	In CVCS, client need to get local copy of source into local machine, do the changes and commit those changes to central source	In DVCS, each client can have a local branch as well and complete history of it. Client need to push the changes to branch which will then be pushed to server repo

2	CVCS is easy to learn and setup	DVCS system are difficult for beginners. Multiple commands needs to be remembered
3	Working on branches is difficult in CVCS. Developer faces many conflicts.	Working on branches is easy in DVCS. Developer faces less conflicts.
4	CVCS do not provide offline access.	DVCS are working fine on offline mode as a client copies the entire repo into their local machine.
5	CVCS is slower as every command needs to be communicated with server.	DVCS is faster as mostly users deals with local copy without hitting server every time.
6	If CVCS server is down, developers cannot work.	If DVCS server is down, developer can work using their local copy.

20. What is the difference between git and github?

➔ *Git is just a version control system that manages and tracks changes to your source code whereas GitHub is a cloud-based hosting platform that manages all your Git repositories.*

In CVCS, developers generally makes modifications and commit there changes directly to the repository but git uses a different strategy. Git does not track each and every modified files. Whenever you do commit an operation, git looks for the files present in the staging area. Only those files present in the staging area are considered for commit and not all the modified files.

21. What are the stages of git?

➔ There are 3 stages of git:-

Working directory - Working directory is the area where you are currently working. It is where your files live on your computer

Staging area – The staging area is like a rough draft space, it's where you can git add the version of a file or multiple files that you want to save in your next commit

Local Repository – The Local Repository is everything in your .git directory. Mainly what you will see in your Local Repository are all your checkpoints or commits. It is the area that saves everything (so don't delete it)

22. Explain various git terminology.

➔

Commit :-

- It is 40 alpha-numeric characters.
- It uses SHA-1 checksum concept.
- Store changes in local repository and commit-id gets created.
- Even if one data changes, commit id will get change.
- It helps to track the changes.
- Commit is also known as SHA1 hash.

Commit-ID/Version-ID/Version :-

- Reference to identify each change.
- To identify who changed the file.

Tags :-

- Tags assign a meaningful name with a specific version in the repository.
- Once a tag is created for a particular save even if you create a new commit it will not be updated.

Snapshots :-

- Represents some data of particular time.
- It is always incremental i.e it stores changes only not entire copy.

Push :-

- Push operations copies changes from a local repository instance to a remote or central repo.
- This is used to store the changes permanently into the git repository.

Pull :-

- Pull operation copies changes from a remote repository to a local machine.
- The pull operation is used for synchronization between two repo.

Branch :-

- Product is same, so one repository but different task.
- Default branch is master.
- Each task has one separate branch.
- Finally merge all branches.
- Useful when team wants to work parallelly.
- Can create one branch based on another branch.
- Changes are personal to that particular branch.
- File created in workspace will be visible in any of the branch workspace until you commit. Once you commit then that file belongs to that branch.

23. What are the advantages of git?

➔

- Free and open source
- Fast and small as most of the operations are performed locally
- Git is secured as it uses SHA1 checksum concept
- No need of powerful hardware
- Easier branching

24. What are the different repository?

➔

Bare repository (central repository) :-
- Store and share only
- All central repositories are bare repository

Non-bare repository (Local repository):-

- Where you can modify the files
- All local repositories are non-bare repository

25. What are the various git commands?

➔

- Git - -version – Displays the git version
- Git config - -global user.name “gaurav” – To configure username in git
- Git config - -global user.email “Prajapati.gaurav016@gmail.com” - to configure email id in git
- Git config - -list – Displays user configured details
- Git init – Converts local directory into repository
- Git status – Displays the repository status (untrack File not moved to staging area)
- Git add . – Moves all contents of working directory to staging area
- Git commit -m “commit” – commits all contents of staging area into local repository
- Git log – Displays all user and user action details who performs commit
- Git log -1 – Displays only details of last committed changes
- Git log - -oneline – Displays details of all committed changes in 1 line
- Git log - -grep “abc” – Displays details of committed changes which contains commit message as “abc”
- Git show <commit-id> - Displays the committed code
- Git remote add origin <central git URL> - To add/link central repository for code push
- Git push -u origin master – All user code will be pushed to master branch
- Git pull -u origin master – Pull all master code into working repository
- Git add .gitignore – To ignore some files (file format added in .gitignore) while committing
- Git clean -n – To delete untrack files with warning
- Git clean -f - To delete untrack files forcefully

26. What is the best branch strategy?

➔ Use feature branches for all new features and bug fixes. Merge feature branches into the main branch using pull requests. Keep a high quality, up-to-date main branch.

By developing branches for features, its not only possible to work on both of them in parallel, but it also keeps the main master branch free from error.

27. What are the basics command used for git branches?

➔

- Git branch – To list all available branches and current working branch will have a * mark
- Git branch branch_name – To create a branch
- Git checkout branch1 – To move from current working branch to branch1
- Git branch -d branch_name – To delete a branch
- Git branch -D branch_name – To delate a branch forcefully
- Git merge branch_name – To merge two branches

28. What is git stashing and what are git stash commands?

➔ Git stash temporarily shelves (or stashes) changes you've made to your working copy so you can work on something else, and then come back and re-apply them later. To stash an item (Only applies to modified file not on new file)

- Git stash – To stash an item (working file will be moved to temp location)
- Git stash list – To list stashed items
- Git stash apply stash@{0} – To copy stash items to working workspace and then add, commit
- Git stash clear – To clear the stash items

29. What is git reset?

➔ Git reset is a powerful command that is used to undo local changes to the state of a git repo. (before commit)

  To reset staging area :-
  Git reset file_name – To reset/remove file changes from staging area
  Git reset . – To reset/remove all changes from staging area

  To reset changes from both staging area and working directory at a time:-
  Git reset - -hard

30. What is git revert?

➔ The revert command helps to undo an existing commit (after commit). It does not delete any data in the process instead git creates a new commit with the included files reverted to their previous state. (so your version control history moves forward while the state of your file moves backward)

  Git revert commit_id

31. What is tag in git?

➔ Tag operation allows giving meaningful names to a specific version in the repository.

  Git tag -a tag_name -m <message> <commit_id>
  Git tag – To display the list of tags
  Git show tag_name – To display commit content of specific tag
  Git tag -d tag_name – To delete a tag

32. What is github clone?

➔ It creates a local repo automatically in Linux machine with the name as in github account.

  Git clone <URL of github>

33. What is docker?

➔

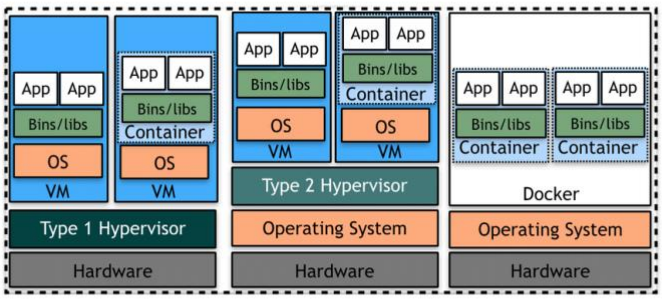
- Docker is an open source centralized flatform designed to create, deploy and run applications.
- Docker uses container on the host OS to run applications.
- It allows applications to use the same Linux kernel as a system on the host computer rather than creating a whole virtual OS.
- We can install docker on any OS but docker engine runs natively on Linux distribution.
- Docker is support only windows 10 pro and enterprise.
- Docker is written in “go” language.
- Docker is a tool that performs OS level virtualization also know as containerization.

- Before docker, many user faces the problem that a particular code is running in the developer's system but not on the user's system.
- **Docker** is a set of platforms as a service that **uses OS level virtualization** whereas **VMware uses hardware level virtualization**.

34. Virtualization vs Containerization?

➔ **Virtualization** – In virtualization, VM's utilizes the Hardware resources and once created VM's can't deallocate the resources, they need to be deleted in order to release the hardware resources.

Containerization – In containerization, containers utilize the OS resources (which utilizes the hardware resources) and containers can deallocate the resources if not required.



35. Does container have OS? → Yes. It has only 5% OS supporting files which are negligible that why most of the people says No.

36. Can a docker container run on any OS?

➔ No. They cannot. Containers are using the underlying operating system resources and drivers, so Windows containers can run on Windows only (2nd env), and Linux containers can run on Linux only (2nd env).

In case of windows base OS, Container can run Linux but docker tool needs to be installed on base windows OS so that the tool can provide the required Linux files to container.

37. What are the advantages of docker?

➔

- No pre-allocation of RAM
- Less cost
- It is light in weight
- It can run on physical hardware or virtual hardware or on cloud
- You can re-use the image
- It takes very less time to create container
- CI Efficiency – Docker enables you to build a container image and use that same image across every step of the deployment.

38. What are the disadvantages of docker?

➔

- Docker is not a good solution for application that requires rich GUI.
- Difficult to manage large amount of containers.

- Docker does not support cross-platform that means if an application is designed to run in a docker container of windows then it cannot on Linux or vice versa.
- Docker is suitable when the development OS and testing OS are same if the OS is different, we should use VM.
- No solution for data recovery and backup.

39. What is docker image?

➔ A Docker image is a read-only template that contains a set of instructions for creating a container that can run on the Docker platform. It provides a convenient way to package up applications and preconfigured server environments, which you can use for your own private use or share publicly with other Docker users.

Docker images are read-only (cannot be modified) templates used to build containers. Containers are deployed instances created (can be modified) from those templates.

40. Why is container called as layered file system?

➔ Because each container has its own thin writable container layer, and all changes are stored in this container layer, this means that multiple containers can share access to the same underlying image and yet have their own data state.

41. What are the components of docker container?

➔ The core of the docker consists of –

- **Docker Engine** - The Docker engine is a part of Docker which create and run the Docker containers. Docker Engine is a client-server-based application with following components –
  - A server which is a continuously running service called a daemon process.
  - A REST API which interfaces the programs to use talk with the daemon and give instruct it what to do.
  - A command line interface client.

- **Docker Containers** - The docker container is a live running instance of a docker image. Container held the entire packages that is needed to run the application. Container is like VM.

- **Docker Images** – A Docker image is a read-only template that contains a set of instructions for creating a container that can run on the Docker platform. It provides a convenient way to package up applications and preconfigured server environments to run a program/container, which you can use for your own private use or share publicly with other Docker users.

  3 Ways to create an image in local machine-
  - Take image from docker hub and use.
  - Create image from docker file.
  - Create image from existing docker container.

- **Docker Client** – Docker user interacts with docker daemon through a client (CLI). Docker client uses command and rest API's to communicate with the docker daemon. When a client runs any server command on the docker client terminal, the client terminal sends these docker commands to the docker daemon. It is possible for docker client to communicate with more than one daemon.

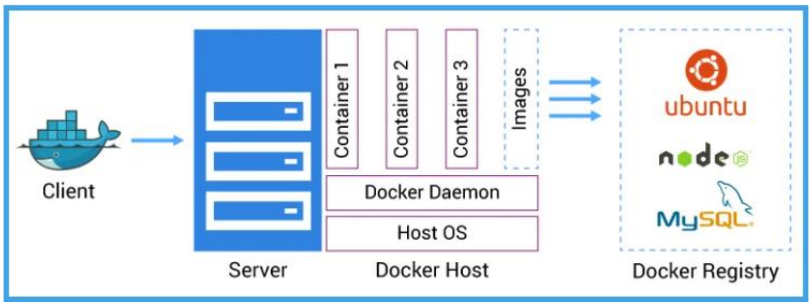
- **Docker Daemon** – Docker daemon runs on the host OS. The docker daemon process is used to control and manage the containers. It also communicates with other daemons to manage Docker services. The Docker daemon listens to only Docker API requests and handles Docker images, containers, networks, and volumes.

- **Docker Host** – Docker host is used to provide an environment to execute and run applications. It contains the docker daemon, images, containers, networks and storages.

- **Docker Hub/Registry** – Docker Hub/Registry manages and stores the docker images. There are two types registries in the docker-
  - Public Registry – It is also called as docker hub.
  - Private Registry – It is used to share images within the enterprise.
-

42. Explain docker architecture.

➔ Docker uses a client-server architecture. The Docker client talks to the Docker daemon, which does the heavy lifting of building, running, and distributing your Docker containers. The Docker client and daemon can run on the same system, or you can connect a Docker client to a remote Docker daemon.



43. What are the various docker commands used?

➔

- To start docker service → service docker start
- To stop docker service → service docker stop
- To check service is start or not → service docker status
  - To see docker info in details → docker info

- To see all the images present in your local machine → docker images
- To find out images in docker hub → docker search image_name
- To download images from docker hub to local machine → docker pull image_name
- To see all containers → docker ps -a
- To see only running containers → docker ps (Process Status)
- To check container details → docker container inspect container_name
- To create container → docker container create -it - -name container_name image_name
- To create and run a default container → docker run -it image_name /bin/bash
  - It creates container after downloading latest image from docker hub by default if not present in local

- To create and run container with name→ docker run -it - -name container_name image_name /bin/bash
  → i- interactive mode
  → t- terminal

- To start container → docker start container_name
- To stop container → docker stop container_name
  → Docker container stops automatically when we exit from container

- To go inside a container → docker attach container_name
- To come outside of a container → exit
- To delete container → docker rm container_name
- To create image from container → docker commit container_name new_image_name
- To see the changes done on base image → docker diff container_name

```
O/p → C /root                  (Change)
      A /root/.bash-history    (Append)
      C /tmp                   (Change)
      D /tmp/myfile            (Delete)
```

44. What is Dockerfile? What are the components of dockerfile?

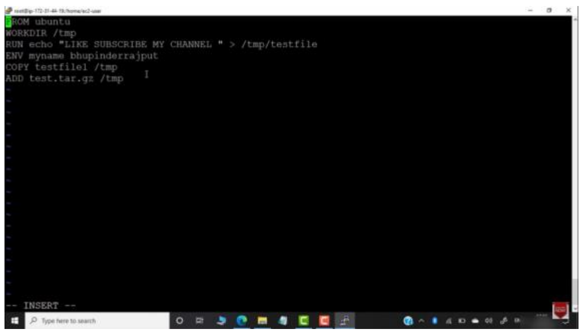
→ Dockerfile is basically a text file which contains some set of instructions to automate creation of docker image.

Note:- Docker file created can only be named as "Dockerfile" with "D" as capital. Below mentioned components/instruction must be in capital letters only.

Dockerfile components are listed below:-

- **FROM** :- for base image. This command must be on top of the dockerfile.
- **RUN** :- to execute commands, it will create a layer in image.
- **MAINTAINER** :- Author/Owner/Description
- **COPY** :- copy files from local system we need to provide source and destination.
  Note:- we cant download files from internet and any remote repo.

- **ADD** :- similar to copy but it provides a feature to download files from internet also extract files at docker image side.
- **EXPOSE** :- to expose ports such as 8080 for tomcat.
- **WORKDIR** :- to set working directory for a container.
- **CMD** :- execute commands but during container creation.
- **ENTRYPOINT** :- similar to CMD but has higher priority over CMD, firstly commands will be executed by ENTRYPOINT only.
- **ENV** :- to set environment variables in container.
- **ARG** :- similar to ENV, to set build-time variables.



```
FROM ubuntu
WORKDIR /tmp
RUN echo "LIKE SUBSCRIBE MY CHANNEL " > /tmp/testfile
ENV myname bhupinderrajput
COPY testfile1 /tmp
ADD test.tar.gz /tmp
~
-- INSERT --
```

45. How to create docker images in local machine?

➔ There are 3 ways to create docker images in local machine:-

1. **Creating image from docker hub (2 ways) :-**

   Steps:-
   - Create image in local machine by pulling from docker hub
     Docker pull image_name

   - Create image in local machine while creating container
     docker run -it - -name container_name image_name /bin/bash

2. **Creating image from container :-**

   Steps:-
   - Create container from base image
     docker run -it - -name container_name image_name /bin/bash

   - Make changes into container
     Cd /tmp
     Touch myfile

   - To check the differences
     Docker diff container_name

   - Create new image of the container
     Docker commit container_name new_image_name

   - Create container from new image
     docker run -it - -name new_container_name new_image_name /bin/bash

   - Check the changes done are reflecting in new container
     Cd /tmp
     Ls

   O/P :- myfile

3. **Creating image from Dockerfile :-**

Steps:-

- Create a file named Dockerfile
  Vi Dockerfile

- Add instructions in Dockerfile
  FROM ubuntu
  RUN echo "Technical guftgu" > /tmp/testfile

- Build Dockerfile to create image
  Docker build -t image_name . (to build image from current directory with tag_name)

- Run image to create container
  Docker run -it - -name container_name image_name /bin/bash

46. What is docker volume. Explain?

➔ Docker volumes are directories and files that exist on the host file system outside of the Docker container. These volumes are used to persist data and share data between Docker containers.

- Firstly we have to declare directory as volume and then share the volume.
- Even if we stop container, still we can access volume.
- Volume will be created in one container.
- You can declare a directory as a volume only while creating container.
- You cant create volume from existing container.
- You can share volume across any number of containers.
- Volume will not be included when you update an image.
- You can map volume in 2 ways:-
  Container to Container
  Host to Container

47. What are the benefits of docker volume?

➔

- Volume can be shared among multiple containers.
- On deleting container volume does not get deleted.
- Volume can be attached to containers.
- Decoupling container from storage.

48. How to create volume and share volume between containers?

➔ There are 2 ways to create volume:-

1. **Creating volume from Dockerfile:-**

   Steps:-

   - Create a dockerfile
     FROM ubuntu
     VOLUME ["/volume_name"]

- Create image from dockerfile
  Docker build -t image_name .

- Create and run container from the image
  Docker run -it - -name container_name image_name /bin/bash

- Share the created volume with another container
  Docker run -it - -name new_container_name - - privileged=true - -volumes-from container_name image_name /bin/bash

- To check the shared volume in new container
  Docker attach new_container_name
  Ls /volume_name

2. **Creating volume using Command:-**

   Steps:-
   - Create a container with volume
     Docker run -it - -name container_name -v volume_name image_name /bin/bash

   - Share volume with another container
     Docker run -it - -name new_container_name - -privileged=true - -volumes-from container_name image_name /bin/bash

   - To check the shared volume in new container
     Docker attach new_container_name
     Ls /volume_name

Note:-

➔ Volume should be declared while creating container. (1.1)
➔ Volume should be used while creating new container. (1.4) or (2.2)
➔ Container1 volume files will be visible in Container2 volume after sharing. Also, if any files are added in Container2 volume will be visible in Container1 volume.

49. How to create volume and share volume from host to container?
➔
Steps:-
- Verify files in host
  /home/dir_name
  Ls

- Create container
  Docker run -it - -name container_name -v /home/dir_name:/volume_name privileged=true image_name /bin/bash

- Create file in container and verify in host
  Cd /container_name

Touch file_name
exit

- Create file in host and verify in container
  Cd /dir_name
  Touch file_name

50. What are the various commands used for volume?

➔

- To list all volume → docker volume ls
- To create volume → docker volume create volume_name
- To delete volume → docker volume rm volume_name
- To remove all volumes that are not in use → docker volume prune
- To check volume details → docker volume inspect volume_name

51. How to expose docker port?

➔

Steps:-

- Sudo su
- Yum update -y
- Yum install docker -y
- Service docker start
- Docker run -td - -name container_name -p 80:80 image_name
  d- daemon
  p- port or publish
  80- host
  80- container
- Docker ps
- Docker port container_name → display ports details which are mapped with container
- Docker exec -it container_name /bin/bash → to go inside a container as new process
- Apt-get update
- Apt-get install apache2 -y
- Cd /var/www/html
- Echo “abc” > index.html
- Service apache2 start
- Docker run -td - -name jenkinscontainer -p 80:80 jenkins → Jenkins example

52. What is the difference between docker attach and docker exec?

➔

Docker exec:-

- Docker exec creates a new process in the container environment.
- Docker exec is specifically for running new things in a already started container, be it shell or some other process.

Docker attach:-

- Docker attach just connect the standard input/output of the main process inside the container to corresponding standard input/output error of current terminal.

53. What happens when you use docker expose and docker publish?

➔

1. **Neither specify expose nor -p:-**
   If you specify neither expose nor -p then the service in the container will only be accessible from inside the container itself.

2. **Only specify expose:-**
   If you expose a port, the service in the container is not accessible from outside docker but from inside other docker containers, so this is good for inter-container communication.

3. **Specify expose and -p:-**
   If you expose and -p a port, the service in the container is accessible from anywhere even outside docker.

4. **Only specify -p:-**
   If you do -p and do not expose, docker does an implicit expose. This is because, if the port open to the public, it is automatically also opens to the other docker containers. Hence -p includes expose.

54. How to push docker image in dockerhub?

➔

Steps:-

- Docker login with credentials using CLI
  Docker login

- Give tag to docker image
  Docker tag image_name dockerid/new_image_name

- Push image to dockerhub
  Docker push dockerid/new_image_name

- To pull image from dockerhub
  Docker pull dockerid/new_image_name

- To run container from new image
  Docker run -it - -name container_name dockerid/new_image_name /bin/bash

55. Important docker commands?

➔

- ➢ To stop all running containers → docker stop $(docker ps -a -q)
- ➢ To delete all stopped containers → docker rm $(docker ps -a -q)
- ➢ To delete all images → docker rmi -f $(docker images -q)

56. What is continuous integration?

➔ Continuous integration refers to the build and unit testing stages of the software release process. Every revision that is committed triggers an automated build and test. It’s a methodology not tool.
Continuous integration = Continuous Build + Continuous Testing

57. How Jenkins works?

➔ Jenkins is an open-source project written in java that runs on windows, macOS and other Unix-like operating system. It is free, community supported and might be your first choice tool for CI. Jenkins automate the entire SDLC.

- Jenkins was originally developed by SUN microsystem in 2004 under the name Hudson. The project was later named as Jenkins when oracle bought microsystems.
- It can run on any major platform without any compatibility issues.
- Whenever developer write code, we integrate all that code of all developers at that point of time and we build, test and deliver/deploy to the client. This process is called CI/CD. Jenkins helps us to achieve this.
- Because of CI, now bugs will be reported fast and get rectified fast. So the entire software development happens fast.

58. Explain Jenkins workflow?

➔

Developer → Github → Jenkins → Maven → Selenium → QA → Deploy/Deliver → Nexus/S3

- We can attach git, maven, selenium and Artifactory plugins to Jenkins.
- Once developer puts code in github, Jenkins pull that code and sends to maven for build.
- Once build is done, Jenkins pull that code and sends to selenium for testing.
- Once testing is done then Jenkins will pull that code and sends to Artifactory as per requirement and so on.
- We can also deploy with Jenkins.
- A **plugin** is a **software add-on** that is installed on a program, enhancing its capabilities.

59. What are the advantages of Jenkins?

➔

- It has lots of plugins.
- You can write your own plugin.
- You can use community plugin.
- Jenkins is not just a tool. It is a framework i.e. you can do whatever you want. All you need is plugins.
- We can attach slaves(nodes) to Jenkins master. It instructs others(slaves) to do job. Its slaves are not available, Jenkins itself does the job.
- Jenkins also behave as crone server replacement i.e. can do scheduled task.
- It can create labels.

Build

Compile → Code Review → Unit Testing → Integration Testing → Packaging (War/Jar)

60. What are various plugins used and path setup in Jenkins?

➔

- ➢ Maven → maven integration
  Path – Manage Jenkins → Global tools configuration → Add Maven → Provide name and local path
- ➢ Jenkins User Management → Role based Authorization strategy

All created users will have Admin rights by default which can be restricted with the help of plugin

➢

61. How to schedule task and why SCM is used in Jenkins?

➔ **Scheduled task**:- To avoid manual intervention of triggering Jenkins build, jobs are scheduled as task so that after a period of time build should be triggered automatically.

Steps-

- Goto Project → Configure → Build triggers → Build periodically
- Provide input as * * * * * (Min, Hr, Day of month, Month, Day of week)→ save
- Can see automatic builds after every 1 min
- You can manually trigger build as well

SCM:- Scheduled task utilizes CPU and memory periodically i.e. it checks Repo every 1 min and start build, so to overcome utilization issue SCM is used. SCM triggers only when some changes occur in repo and then build.

Steps-

- Goto Project → Configure → Build triggers → Poll SCM
- Provide input as * * * * * (Min, Hr, Day, Month, Week)→ save
- After changes donein Git source code, build will start automatically after 1 min
- You can manually trigger build as well

62. Upstream Linked Project vs Downstream Linked roject?

➔

Upstream Linked Project:-

- First project will do its work and then inform Second project to resume.
  Post Build Actions → Post-build activity in Jenkins

Downstream Linked Project:-

- Second project will observe First project for work completion and then will resume its work
  Build trigger → Build after other projects are built

63. Java Project Structure

➔

Source Code
Test Code
Project Structure (assets, directories, resources)
Dependencies/Library
Configuration
Task Runner – build, test, run
Reporting

64. What is Maven and why we use it?

➔ Maven is an automation and project management tool developed by Apache software foundation. It is based on POM (Project Object Model)

- It was initially released on 13th July 2004.
- Maven is written in Java.
- Maven can build any number of projects into desired output such as .jar, .war, metadata.
- Mostly used for Java based project.
- Meaning of maven is “Accumulator of knowledge”.
- Maven helps in getting the right jar file for each project as these may be different version of separate packages.
- To download dependencies, it is no more needed to visit the official website of each software. It could now be easily done by visiting “mvnrepository.com”

  Dependencies- it refers to the java libraries that one needed for the project.
  Repositories- refer to the directories of packaged jar files.
  Build tools-
    Java- Ant, Maven, Gradle
    .Net- Visual Studio
    C,C++- Make File

65. What problems may rise if not using maven?

➔

- **Adding set jars in each project:-** In case of struts, spring, we need to add jar files in each project. It must include all the dependencies of jars also.
- **Creating the right project structure:**- We must create the right project structure in servlet, struts, etc. otherwise it will not be executed.

  For e.g:- .war file layout

-

66.

Notes:-

- Linux is developed based on MINIX and not UNIX
- Linux is a kernel (Not an OS)
- GNU is set of free software built during free software movement
- Operating System is built by combining Kernel and GNU
- Windows and Linux architecture difference is OS used in Windows where as Kernel used in Linux for H/W interactions
- $ - User and # - root
- Repository- It is a place where you have all your codes or kind of folder on server.