Interview Questions-DevOps

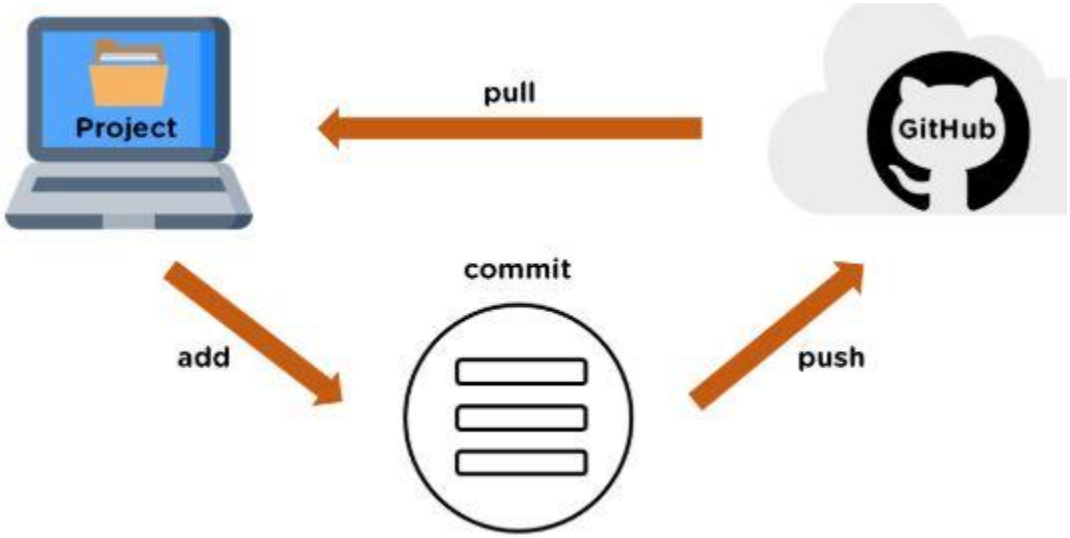
1.How can you initialize a repository in Git?

If you want to initialize an empty repository to a directory in Git, you need to enter the git init command. After this command, a hidden .git folder will appear.

git init - This command helps to create an empty repository while working on a project.

2. What does git pull origin master do?

The git pull origin master fetches all the changes from the master branch onto the origin and integrates them into the local branch.



3. What do you understand about the Git merge conflict?

A Git merge conflict is an event that occurs when Git is unable to resolve the differences in code between the two commits automatically.

Git is capable of automatically merging the changes only if the commits are on different lines or branches.

3. What is Git stash?

Let’s say you're a developer and you want to switch branches to work on something else. The issue is you don’t want to make commits in uncompleted work, so you just want to get back to this point later. The solution here is the Git stash.

Git stash takes your modified tracked files and saves it on a stack of unfinished changes that you can reapply at any time. To go back to the work you can use the stash pop.

4. How do you find a list of files that has been changed in a particular commit?

The command to get a list of files that has been changed in a particular commit is:

git diff-tree –r {commit hash}

- -r flag allows the command to list individual files
- commit hash lists all the files that were changed or added in the commit.

5. How to remove a file from git without removing it from your file system?

One has to be careful during a git add, else you may end up adding files that you didn't want to commit. However, git rm will remove it from both your staging area (index), as well as your file system (working tree), which may not be what you want.

Instead, use git reset:

git reset filename # or

echo filename >> .gitingore # add it to .gitignore to avoid re-adding it

This means that git reset <paths> is exactly the opposite of git add <paths>.

6. How do you squash the last N commits into a single commit?

There are two options to squash the last N commits into a single commit include both of the below-mentioned options in your answer

If you want to write the new commit message from scratch use the following command:

git reset –soft HEAD~N &&git commit

If you want to start editing the new commit message with a concatenation of the existing commit messages then you need to extract those messages and pass them to Git commit for that I will use:

git reset –soft HEAD~N &&git commit –edit -m"$(git log –format=%B –reverse .HEAD@{N})"

7. Differentiate between git pull and git fetch.

git pull	git fetch
This command pulls new changes from the currently working branch located in the remote central repository.	This command is also used for a similar purpose but it follows a two step process: 1. Pulls all commits and changes from desired branch and stores them in a new branch of the local repository. 2. For changes to be reflected in the current / target branch, git fetch should be followed by git merge command.

8. Can you tell something about git reflog?

This command tracks every single change made in the repository references (that can be branches or tags) and also maintains the branches/tags log history that was either created locally or checked out. Reference logs such as the commit snapshot of when the branch was created or cloned, checked-out, renamed, or any commits made on the branch are maintained by Git and listed by the 'reflog' command.

- This recovery of the branch is only possible when the branch was either created locally or checked-out from a remote repository in your local repository for Git to store its reference history logs.
- This command should be executed in the repository that had the lost branch.

9. What are the different kinds of branches that can be created in GIT?

We can make various types of branches for the following purposes in GIT:

- Feature branches: These are utilized for building up a component.
- Release branches: These are utilized for discharging code to create.
- Hotfix branches: These are utilized for discharging a hotfix to generation for an imperfection or crisis settle.

10. What is the meaning of the commands – git status, git log, git diff, git revert <commit>, git reset <file>?

Command	Meaning
git status	Gives a list of which files are staged, unstaged, and untracked
git log	Illustrates the entire commit history by using the default format
git diff	Displays the unstaged changes between index and working directory
git revert <commit>	Undoes all the changes made in <commit> and apply it to the current branch by creating
git reset <file>	Removes <file> from a staging area without overwriting any changes by keeping the worl unchanged

11. Explain the role of the git config command with examples.

The git config command is used to get and set git configuration values on a global or local project level. It uses your username to associate commits with an identity. You can also change your Git configuration, including your username.

For example:

You can give a username and email id to associate a commit with an identity. This will help you know who has made that commit.

git config –global user.name “Your Name”: It will add a username.

git config –global user.email “Your E-mail Address”: It will add an email id.

12. Explain Jenkins’ Master-Slave architecture?

Jenkins supports the owner-slave model, in which a master employs a large number of slaves. Jenkins Distributed Builds is another name for it. It allows you to run jobs on a variety of platforms, including Linux, macOS, and Windows. We may also use Jenkins Distributed Builds to perform a similar project on several conditions in parallel, allowing you to achieve the best results quickly using this distributed approach. The rest of the task results are collected and analyzed on the master hub.

13. Give a simple use case/scenario to explain how Jenkins works.

Let us say a developer is working on some code changes and eventually commits them to the repository.

Jenkins server, which constantly checks for changes in the repository, detects the change and pulls the changes to trigger a build.

The build can fail, in which case the developer is informed with reports.

If the build passes, it is deployed on to the test server.

Once the testing is complete, a test report is generated and sent to the developers. This process continues till all the tests are successful, after which code is deployed to production.

14. What syntax does Jenkins use to schedule build job or SVN polling?

The cron syntax.

Cron syntax is represented using five asterisks each separated by a space. The syntax is as follows – [minutes] [hours] [day of the month] [month] [day of the week]. Example, if you want to set up a cron for every Monday at 11.59 pm, it would be:

59 11 * * 1

15. Explain the blue ocean in Jenkins.

Blue ocean is a modern UI for Jenkins, which helps in a personalized experience with modern design. Through this interface, any user can create, diagnose and visualize Continuous Delivery pipelines. It doesn't need technical skills to create or understand the pipelines as everything is visually presented. Also, detection of automation problems is easy as each step can be easily navigated.

16. What is Pipeline as a Code in Jenkins? Explain various types of Pipeline?

Pipeline-as-a-code is a technique or set of features that help you maintain the CI/CD workflow logic in the source code repository without any additional configurations for each Jenkins branch. This is applicable for projects that have a file named as Jenkinsfile in the root folder of the repo (containing a pipeline script). The types of pipeline syntax are declarative and syntax.

- *Declarative*: These provide an easy way for creating pipelines and has a predefined hierarchy to create Jenkins pipelines. You can control all the aspects of the Pipeline.
- *Scripted*: It runs on the Jenkins master through a lightweight executor and uses few resources for translating the Pipeline into code (commands).

17. What is the use of JENKINS_HOME directory?

All the settings, logs and configurations are stored in the JENKINS_HOME directory.

18. How do you move or copy Jenkins from an old server to a new one?

All the configurations, settings are stored in the JENKINS_HOME (/var/lib/Jenkins) directory. You should copy the entire home directory to the new server. To do so, you can use the command 'rsync'.

19. What is Maven? What is the benefit of integrating maven with Jenkins?

Maven is a build management tool. Using a simple pom.xml, one can configure all the dependencies needed to build, test and run the code. Maven manages the full lifecycle of a test project. Once integrated with Jenkins, the maven Webdriver will build the project and execute all tests efficiently.

20. What is a Jenkins Pipeline?

Jenkins Pipeline (or simply "Pipeline") is a suite of plugins which supports implementing and integrating *continuous delivery pipelines* into Jenkins.

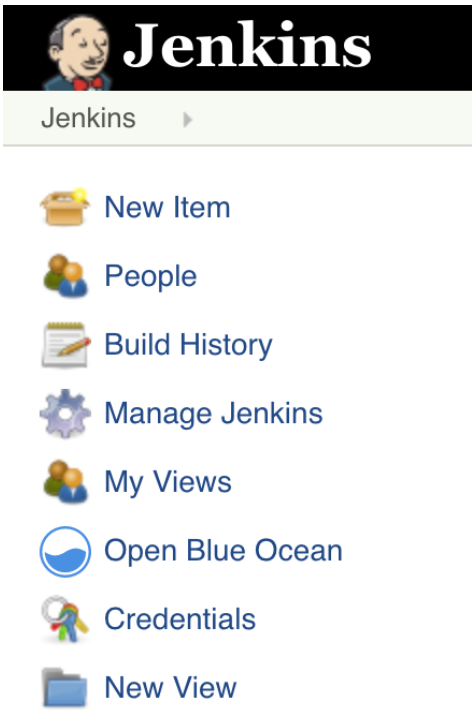
A *continuous delivery pipeline* is an automated expression of your process for getting software from version control right through to your users and customers.

Jenkins Pipeline provides an extensible set of tools for modeling simple-to-complex delivery pipelines "as code". The definition of a Jenkins Pipeline is typically written into a text file (called a Jenkinsfile) which in turn is checked into a project's source control repository. [1]

For more information about Pipeline and what a Jenkinsfile is, refer to the respective Pipeline and Using a Jenkinsfile sections of the User Handbook.

To get started quickly with Pipeline:

1. Copy one of the examples below into your repository and name it *Jenkinsfile*



2. Click the **New Item** menu within Jenkins

3. Provide a name for your new item (e.g. **My-Pipeline**) and select **Multibranch Pipeline**
4. Click the **Add Source** button, choose the type of repository you want to use and fill in the details.
5. Click the **Save** button and watch your first Pipeline run!

You may need to modify one of the example Jenkinsfile's to make it run with your project. Try modifying the sh command to run the same command you would run on your local machine.

After you have setup your Pipeline, Jenkins will automatically detect any new Branches or Pull Requests that are created in your repository and start running Pipelines for them.

Jenkinsfile (Declarative Pipeline)

```
pipeline {
   agent { docker { image 'maven:3.8.4-openjdk-11-slim' } }
   stages {
      stage('build') {
         steps {
            sh 'mvn --version'
         }
      }
   }
}
```

Jenkinsfile (Scripted Pipeline)

```
/* Requires the Docker Pipeline plugin */
node('docker') {
   stage('Build') {
      docker.image('maven:3.8.4-openjdk-11-slim').inside {
         sh 'mvn --version'
      }
   }
}
```

21. How do you store credentials in Jenkins securely?

Credentials can be stored securely in Jenkins using the Credentials plugin, which stores different types of credentials like - Username with a password, SSH username with the private key, AWS Credentials, Jenkins Build Token, Secret File/Text, X509 & other certificates, Vault related credentials securely with proper encryption & decryption as and when required.

22. What are the ways to trigger a Jenkins Job/Pipeline?

There are many ways we can trigger a job in Jenkins. Some of the common ways are as below -

- Trigger an API (POST) request to the target job URL with the required data.
- Trigger it manually from the Jenkins web application.
- Trigger it using Jenkins CLI from the master/slave nodes.
- Time-based Scheduled Triggers like a cron job.
- Event-based Triggers like SCM Actions (Git Commit, Pull Requests), WebHooks, etc.
- Upstream/Downstream triggers by other Jenkins jobs.

23. How code coverage is measured/tracked using Jenkins in a CI environment?

Using language-specific code coverage plugins like JaCoCo, CodeCov, etc or generic tools/plugins like Sonarqube which will add the code coverage data to builds with some minor tweaks in the code and the same can be displayed as a graph in Jenkins.

24. How to download the Console log for a particular Jenkins build programmatically?

Using the Jenkins CLI - console - command

java -jar jenkins-cli.jar console JOB [BUILD] [-f] [-n N]

Produces the console output of a specific build to stdout, as if you are doing 'cat build.log'

- JOB: Name of the job
- BUILD: Build number or permalink to point to the build. Defaults to the last build
- -f: If the build is in progress, append console output as it comes, like tail -f
- -n N: Display the last N lines.

E.g.

ssh -l <ssh_username> -p <port_no> <Jenkins_URL> console <JOB_NAME>

25. What are the credential types supported by Jenkins?

In Jenkins, credentials are a set of information used for authentication with internal/external services to accomplish an action. Jenkins credentials are provisioned & managed by a built-in plugin called - Credentials Binding - plugin. Jenkins can handle different credentials as follows -

- Secret text - A token such as an API token, JSON token, etc.
- Username and password - Basic Authentication can be stored as a credential as well.
- Secret file - A secret file used to authenticate some secure data services & security handshakes.
- SSH Username with a private key - An SSH public/private key pair for Machine to Machine authentication.
- Certificate - a PKCS#12 certificate file and an optional password.
- Docker Host Certificate Authentication credentials.

And as we can guess, this can be extended to several other extensible credential types like - AWS credential, Azure secrets, etc. using commonly available plugins.

26. What is the fully qualified artifact name of maven project?

<groupId>:<artifactId>:<version>

---

27. What is a Maven Archetype?

- Maven Archetype refers to a Maven plugin entitled to create a project structure as per its template.
- These archetypes are just project templates that are generated by Maven when any new project is created.

28. What is POM?

POM stands for Project Object Model. The pom.xml file contains information of project and project configuration.

POM is an acronym for Project Object Model. The pom.xml file contains information of project and configuration information for the maven to build the project such as dependencies, build directory, source directory, test source directory, plugin, goals etc.

Maven reads the pom.xml file, then executes the goal.

Before maven 2, it was named as project.xml file. But, since maven 2 (also in maven 3), it is renamed as pom.xml.

Elements of maven pom.xml file

For creating the simple pom.xml file, you need to have following elements:

Element	Description
Project	It is the root element of pom.xml file.
modelVersion	It is the sub element of project. It specifies the modelVersion. It should be set to 4.0.0.
groupId	It is the sub element of project. It specifies the id for the project group.
artifactId	It is the sub element of project. It specifies the id for the artifact (project). An artifact is something that is either produced or used by a project. Examples of artifacts produced by Maven for a project include: JARs, source and binary distributions, and WARs.
Version	It is the sub element of project. It specifies the version of the artifact under given group.

File: pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">

  <modelVersion>4.0.0</modelVersion>
  <groupId>com.javatpoint.application1</groupId>
  <artifactId>my-app</artifactId>
  <version>1</version>

</project>
```

Maven pom.xml file with additional elements:

Element	Description
packaging	defines packaging type such as jar, war etc.
Name	defines name of the maven project.
url	defines url of the project.
dependencies	defines dependencies for this project.
dependency	defines a dependency. It is used inside dependencies.
Scope	defines scope for this maven project. It can be compile, provided, runtime, test and system.

Here, we are going to add other elements in pom.xml file such as:

File: pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">

  <modelVersion>4.0.0</modelVersion>

  <groupId>com.javatpoint.application1</groupId>
  <artifactId>my-application1</artifactId>
  <version>1.0</version>
  <packaging>jar</packaging>
```

```
  <name>Maven Quick Start Archetype</name>
  <url>http://maven.apache.org</url>

  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.8.2</version>
      <scope>test</scope>
    </dependency>
  </dependencies>

</project>
```

29. Can you tell me the default location of your local repository?

~/m2./repository.

30. Tell me the command to install JAR file in local repository.

mvn install

31. Is there a particular sequence in which Maven searches for dependency libraries?

Following is the search pattern –

- Search for dependency in the local repository, if not found, move to step 2 else do the further processing.
- Search for dependency in the central repository first, if not found and the remote repository is mentioned then move to step 4 else it is downloaded to the local repository for future reference.
- If a remote repository has not been mentioned, Maven simply stops the processing and throws an error (Unable to find dependency).

- Search for dependency in the remote repository first, if found it is downloaded to the local repository for future reference otherwise Maven is expected to stop processing and throws an error.

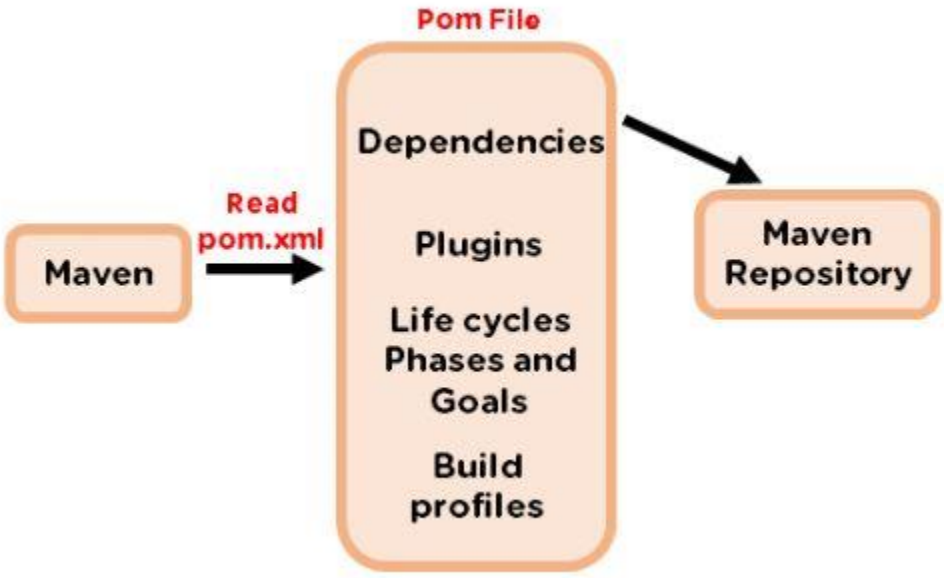
32. What is the command to install JAR files in the Local Repository?

- mvn install is used to install JAR files in the local repository.
- To install the JAR manually into the local Maven repository, the following plugin is used: mvn install:install-file-Dfile=<path to file>.

33. How does Maven Architecture work?

Maven architecture works in three steps, which are as follows:

- The first step is to read the pom.xml file.
- Then, it downloads the dependencies defined in pom.xml into the local repository from the central repository.
- Lastly, it creates and generates a report according to the requirements, and executes life-cycles, phases, goals, plugins, etc.



34. What is the settings.xml file in Maven?

A Maven installation is configured using the settings.xml file. It's comparable to a pom.xml file, but it's either global or user-specific. The Maven settings.xml file provides elements that define the values required to configure Maven's execution in several ways. These values include the location of the local repository, authentication information, and alternate remote repository servers among others.

Let's look at the elements in the settings.xml file that we can change. The settings.xml file's main element, settings, can include up to nine predefined child elements:

```
<settings xmlns = "http://maven.apache.org/SETTINGS/1.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0
https://maven.apache.org/xsd/settings-1.0.0.xsd">
  <localRepository/>
  <interactiveMode/>
  <offline/>
  <pluginGroups/>
  <servers/>
  <mirrors/>
  <proxies/>
  <profiles/>
  <activeProfiles/>
</settings>
```

The following configurations are included:

- Proxy configuration
- Local repository configuration
- Remote repository configuration
- Central repository configuration

35. Where are Maven dependencies stored?

- All the JARS, dependency files, etc. that are downloaded by Maven are saved in the Maven local repository.
- The Maven local repository is a folder location on the local system where all the artifacts are locally stored.

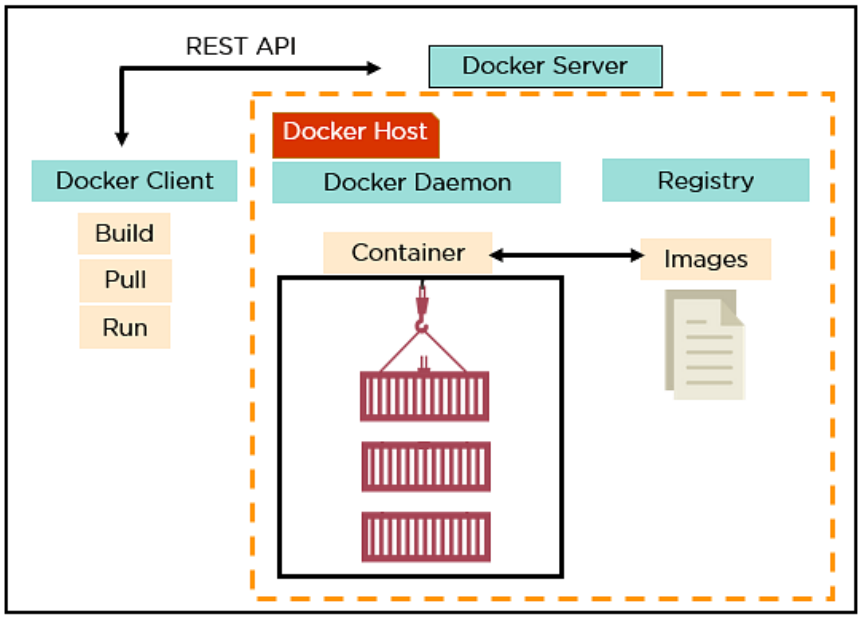
36. How would you refer to a property declared in your pom.xml file?

In order to refer to a property declared in your pom.xml, the property name makes use of the names of the XML components that designate the value, with "pom" being accepted as a synonym for the project (root) element.

So ${pom.name} is the project's name, ${pom.version} is the project's version, ${pom.build.finalName} is the final name of the file generated when the built project is packaged, and so on.

37. Explain the architecture of Docker.

- Docker uses a client-server architecture.
- Docker Client is a service that runs a command. The command is translated using the REST API and is sent to the Docker Daemon (server).
- Docker Daemon accepts the request and interacts with the operating system to build Docker images and run Docker containers.
- A Docker image is a template of instructions, which is used to create containers.
- Docker container is an executable package of an application and its dependencies together.
- Docker registry is a service to host and distribute Docker images among users.

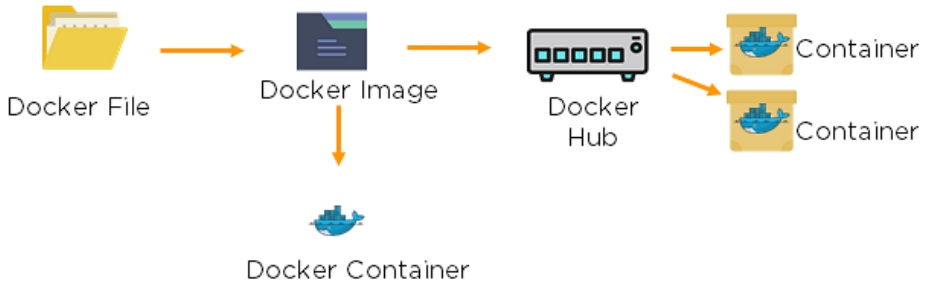


38. How do you run multiple containers using a single service?

- It is possible to run multiple containers as a single service with Docker Compose.
- Here, each container runs in isolation but can interact with each other.
- All Docker Compose files are YAML files.

39. What is a Dockerfile used for?

- A Dockerfile is used for creating Docker images using the build command.
- With a Docker image, any user can run the code to create Docker containers.
- Once a Docker image is built, it's uploaded in a Docker registry.
- From the Docker registry, users can get the Docker image and build new containers whenever they want.



39. Instead of YAML, what can you use as an alternate file for building Docker compose?

To build a Docker compose, a user can use a JSON file instead of YAML. In case a user wants to use a JSON file, he/she should specify the filename as given:

Docker-compose -f Docker-compose.json up

How do you create a Docker container?

Task: Create a MySQL Docker container

A user can either build a Docker image or pull an existing Docker image (like MySQL) from Docker Hub.

Now, Docker creates a new container MySQL from the existing Docker image. Simultaneously, the container layer of the read-write filesystem is also created on top of the image layer.

- Command to create a Docker container: Docker run -t –i MySQL
- Command to list down the running containers: Docker ps

40. What is the purpose of the expose and publish commands in Docker?

Expose

- Expose is an instruction used in Dockerfile.
- It is used to expose ports within a Docker network.
- It is a documenting instruction used at the time of building an image and running a container.
- Expose is the command used in Docker.
- Example: Expose 8080

Publish

- Publish is used in a Docker run command.
- It can be used outside a Docker environment.
- It is used to map a host port to a running container port.
- --publish or –p is the command used in Docker.
- Example: docker run –d –p 0.0.0.80:80

41. What type of applications - Stateless or Stateful are more suitable for Docker Container?

It is preferable to create Stateless application for Docker Container. We can create a container out of our application and take out the configurable state parameters from application. Now we can run same container in Production as well as QA environments with different parameters. This helps in reusing the same Image in different scenarios. Also a stateless application is much easier to scale with Docker Containers than a stateful application.

42. What command can you run to export a docker image as an archive?

This can be done using the docker save command and the syntax is: docker save -o <exported_name>.tar <container-name>

43. What command can be run to import a pre-exported Docker image into another Docker host?

This can be done using the docker load command and the syntax is docker load -i <export_image_name>.tar

44. What command is used to check for the version of docker client and server?

- The command used to get all version information of the client and server is the docker version.
- To get only the server version details, we can run docker version --format '{{.Server.Version}}'

45. Differentiate between COPY and ADD commands that are used in a Dockerfile?

Both the commands have similar functionality, but COPY is more preferred because of its higher transparency level than that of ADD.
COPY provides just the basic support of copying local files into the container whereas ADD provides additional features like remote URL and tar extraction support.

46. Can you tell the differences between a docker Image and Layer?

Image: This is built up from a series of read-only layers of instructions. An image corresponds to the docker container and is used for speedy operation due to the caching mechanism of each step.

Layer: Each layer corresponds to an instruction of the image's Dockerfile. In simple words, the layer is also an image but it is the image of the instructions run.

Consider the example Dockerfile below.

```
FROM ubuntu:18.04

COPY . /myapp

RUN make /myapp

CMD python /myapp/app.py
```

Importantly, each layer is only a set of differences from the layer before it.

- The result of building this docker file is an image. Whereas the instructions present in this file add the layers to the image. The layers can be thought of as intermediate images. In the example above, there are 4 instructions, hence 4 layers are added to the resultant image.

47. What is the purpose of the volume parameter in a docker run command?

- The syntax of docker run when using the volumes is: docker run -v host_path:docker_path <container_name>
- The volume parameter is used for syncing a directory of a container with any of the host directories. Consider the below command as an example: docker run -v /data/app:usr/src/app myapp
  The above command mounts the directory /data/app in the host to the usr/src/app directory. We can sync the container with the data files from the host without having the need to restart it.
- This also ensures data security in cases of container deletion. This ensures that even if the container is deleted, the data of the container exists in the volume mapped host location making it the easiest way to store the container data.

48. Where are docker volumes stored in docker?

Volumes are created and managed by Docker and cannot be accessed by non-docker entities. They are stored in Docker host filesystem at:
/var/lib/docker/volumes/

49. List the most commonly used instructions in Dockerfile?

- FROM: This is used to set the base image for upcoming instructions. A docker file is considered to be valid if it starts with the FROM instruction.
- LABEL: This is used for the image organization based on projects, modules, or licensing. It also helps in automation as we specify a key-value pair while defining a label that can be later accessed and handled programmatically.
- RUN: This command is used to execute instructions following it on the top of the current image in a new layer. Note that with each RUN command execution, we add layers on top of the image and then use that in subsequent steps.
- CMD: This command is used to provide default values of an executing container. In cases of multiple CMD commands the last instruction would be considered.

50. Can you tell the difference between CMD and ENTRYPOINT?

- CMD command provides executable defaults for an executing container. In case the executable has to be omitted then the usage of ENTRYPOINT instruction along with the JSON array format has to be incorporated.
- ENTRYPOINT specifies that the instruction within it will always be run when the container starts.
  This command provides an option to configure the parameters and the executables. If the DockerFile does not have this command, then it would still get inherited from the base image mentioned in the FROM instruction.
  - The most commonly used ENTRYPOINT is /bin/sh or /bin/bash for most of the base images.
- As part of good practices, every DockerFile should have at least one of these two commands.

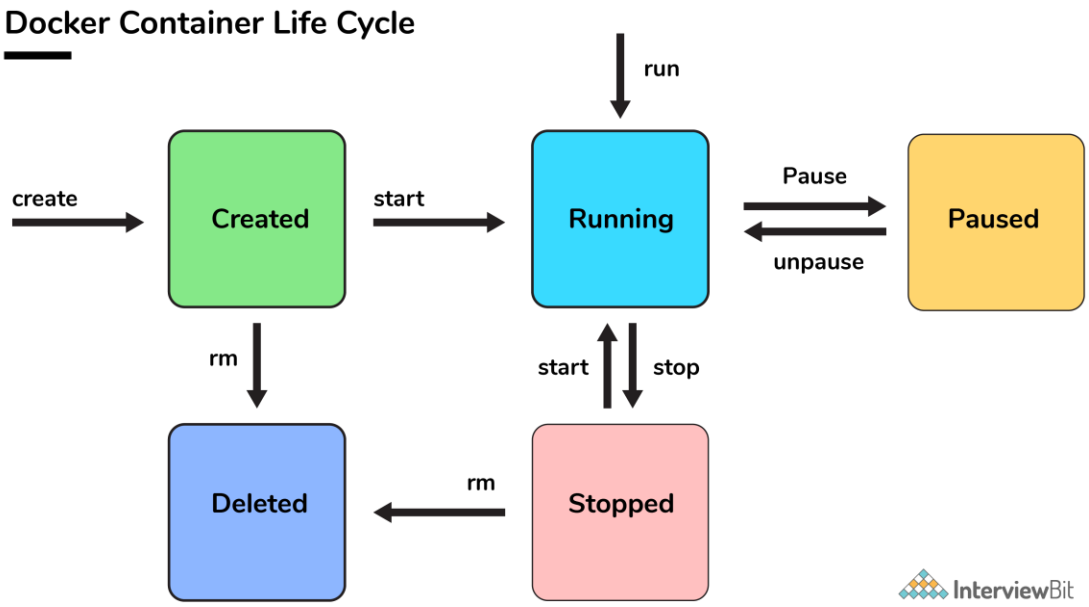
51. How many containers you can run in docker and what are the factors influencing this limit?

There is no clearly defined limit to the number of containers that can be run within docker. But it all depends on the limitations - more specifically hardware restrictions. The size of the app and the CPU resources available are 2 important factors influencing this limit. In case your application is not very big and you have abundant CPU resources, then we can run a huge number of containers.

52. Describe the lifecycle of Docker Container?

The different stages of the docker container from the start of creating it to its end are called the docker container life cycle.
The most important stages are:

- Created: This is the state where the container has just been created new but not started yet.
- Running: In this state, the container would be running with all its associated processes.
- Paused: This state happens when the running container has been paused.
- Stopped: This state happens when the running container has been stopped.
- Deleted: In this, the container is in a dead state.



53. How will you ensure that a container 1 runs before container 2 while using docker compose?

Docker-compose does not wait for any container to be “ready” before going ahead with the next containers. In order to achieve the order of execution, we can use:

The “depends_on” which got added in version 2 of docker-compose can be used as shown in a sample docker-compose.yml file below:

```
version: "2.4"
services:
 backend:
  build: .
  depends_on:
   - db
 db:
  image: postgres
```

The introduction of service dependencies has various causes and effects:

- The docker-compose up command starts and runs the services in the dependency order specified. For the above example, the DB container is started before the backend.
- docker-compose up SERVICE_NAME by default includes the dependencies associated with the service. In the given example, running docker-compose up backend creates and starts DB (dependency of backend).
- Finally, the command docker-compose stop also stops the services in the order of the dependency specified. For the given example, the backend service is stopped before the DB service.

54. Do you know why *docker system prune* is used? What does it do?

$ docker system prune

The above command is used to remove all the stopped containers, all the networks that are not used, all dangling images and all build caches. It's one of the most useful docker commands.

55. Is it better to directly remove the container using the rm command or stop the container followed by remove container?

Its always better to stop the container and then remove it using the remove command.

$ docker stop <container_id>
$ docker rm -f <container_id>

56. Suppose you have an application that has many dependant services. Will docker compose wait for the current container to be ready to move to the running of the next service?

The answer is yes. Docker compose always runs in the dependency order. These dependencies are specifications like depends_on, links, volumes_from, etc.

57. Is it a good practice to run Docker compose in production?

Yes, using docker compose in production is the best practical application of docker compose. When you define applications with compose, you can use this compose definition in various production stages like CI, staging, testing, etc.

58. Write a Docker file to create and copy a directory and built it using python modules?

FROM pyhton:2.7-slim

WORKDIR /app

COPY . /app

docker build –tag

59. Can you implement continuous development (CD) and continuous integration (CI) in Docker?

Yes, you can. You can run Jenkins on Docker and use Docker Compose to run integration tests.

60. What is an Ansible vault?

Ansible vault is used to keep sensitive data, such as passwords, instead of placing it as plain text in playbooks or roles. Any structured data file or single value inside a YAML file can be encrypted by Ansible.

To encrypt the data:

Command: *ansible-vault encrypt foo.yml bar.yml baz.yml*

To decrypt the data:

Command: *ansible-vault decrypt foo.yml bar.yml baz.yml*

61. How to generate encrypted passwords for a user module?

We can do this by using a small code:

```
ansible all -i localhost, -m debug -a "msg={{ 'mypassword' |
password_hash('sha512', 'mysecretsalt') }}"
```

We can also use the Passlib library of Python.

Command: *python -c "from passlib.hash import sha512_crypt; import getpass; print(sha512_crypt.using(rounds=5000).hash(getpass.getpass()))"*

62. Do you have any idea of how to turn off the facts in Ansible?

If you do not need any factual data about the hosts and know everything about the systems centrally, we can turn off fact gathering. This has advantages in scaling Ansible in push mode with very large numbers of systems, mainly, or if we are using Ansible on experimental platforms.

Command:

```
- hosts: whatever
gather_facts: no
```

63. By default, the Ansible reboot module waits for how many seconds. Is there any way to increase it?

By default, the Ansible reboot module waits 600 seconds. Yes, it is possible to increase Ansible reboot to certain values. The syntax given-below can be used for the same:

```
- name: Reboot a Linux system
reboot:
reboot_timeout: 1200
```

64. Can you keep data secret in the playbook?

The following playbook might come in handy if you want to keep secret any task in the playbook when using -v (verbose) mode:

```
- name: secret task
  shell: /usr/bin/do_something --value={{ secret_value }}
  no_log: True
```

It hides sensitive information from others and provides the verbose output.

65. What are tags?

When there's an extensive playbook involved, sometimes it's more expedient to run just a part of it as opposed to the entire thing. That's what tags are for.

66. Speaking of tags, how do you filter out tasks?

You can filter out tasks in one of two ways:

- Use –tags or –skip-tags options on the command line
- If you're in Ansible configuration settings, use the TAGS_RUN and TAGS_SKIP options.

67. What are Ad-hoc commands? Give an example.

Ad-hoc commands are simple one-line commands used to perform a certain task. You can think of Adhoc commands as an alternative to writing playbooks. An example of an Adhoc command is as follows:

ansible host -m netscaler -a "nsc_host=nsc.example.com user=apiuser password=apipass"

68. What is Ansible role and how are they different from the playbook?

Ansible Roles is basically another level of abstraction used to organize playbooks. They provide a skeleton for an independent and reusable collection of variables, tasks, templates, files, and modules which can be automatically loaded into the playbook. Playbooks are a collection of roles. Every role has specific functionality.

Let's understand the difference between Ansible roles and playbook with an example.

Suppose you want your playbook to perform 10 different tasks on 5 different systems, would you use a single playbook for this? No, using a single playbook can make it confusing and prone to blunders. Instead, you can create 10 different roles, where each role will perform one task. Then, all you need to do is, mention the name of the role inside the playbook to call them.

You can create a role using **ansible-galaxy** init command inside /etc/ansible/roles.

$ sudo ansible-galaxy init <role-name>

The directory will look like this:

69. How do I set the PATH or any other environment variable for a task?

The environment variables can be set by using the ‘environment’ keyword. It can be set for either a task or an entire playbook as well. Follow the below code:

```
environment:
  PATH: "{{ ansible_env.PATH }}:/thingy/bin"
  SOME: value
```

70. What is Ansible Inventory and its types?

In Ansible, there are two types of inventory files: Static and Dynamic.

- **Static inventory** file is a list of managed hosts declared under a host group using either hostnames or IP addresses in a plain text file. The managed host entries are listed below the group name in each line. For example

```
[gatewayed]
staging1 ansible_host=10.0.2.1
staging2 ansible_host=10.0.2.2
```

- **Dynamic inventory** is generated by a script written in Python or any other programming language or by using plugins(preferable). In a cloud setup, static inventory file configuration will fail since IP addresses change once a virtual server is stopped and started again. We create a demo_aws_ec2.yaml file for the config such as

```
plugin: aws_ec2 regions:
ap-south-1 filters:
tag:tagtype: testing
```

Now we can fetch using this command

```
ansible-inventory -i demo_aws_ec2.yaml -graph
```

71. What is Ansible Tower and what are its features?

Ansible Tower is an enterprise-level solution by RedHat. It provides a web-based console and REST API to manage Ansible across teams in an organization. There are many features such as

- Workflow Editor - We can set up different dependencies among playbooks, or running multiple playbooks maintained by different teams at once
- Real-Time Analysis - The status of any play or tasks can be monitored easily and we can check what's going to run next
- Audit Trail - Tracking logs are very important so that we can quickly revert back to a functional state if something bad happens.
- Execute Commands Remotely - We can use the tower to run any command to a host or group of hosts in our inventory.

There are other features also such as Job Scheduling, Notification Integration, CLI, etc.

72. What are handlers?

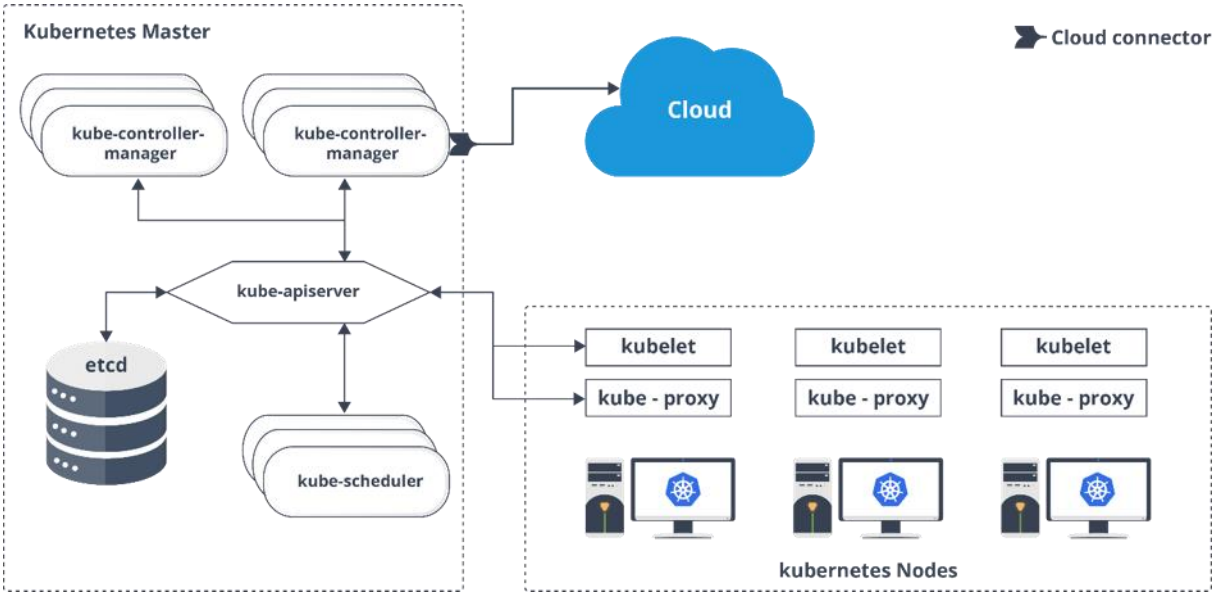
Handlers are like special tasks which only run if the Task contains a "notify" directive.

```
tasks:
 - name: install nginx
   apt: pkg=nginx state=installed update_cache=true
   notify:
    - start nginx
 handlers:
  - name: start nginx
    service: name=nginx state=started
```

In the above example after installing NGINX we are starting the server using a `start nginx` handler.

73. What are the different components of Kubernetes Architecture?

The Kubernetes Architecture has mainly 2 components – the master node and the worker node. As you can see in the below diagram, the master and the worker nodes have many inbuilt components within them. The master node has the kube-controller-manager, kube-apiserver, kube-scheduler, etcd. Whereas the worker node has kubelet and kube-proxy running on each node.



74. What is the difference between a replica set and replication controller?

Replica Set and Replication Controller do almost the same thing. Both of them ensure that a specified number of pod replicas are running at any given time. The difference comes with the usage of selectors to replicate pods. Replica Set use Set-Based selectors while replication controllers use Equity-Based selectors.

- Equity-Based Selectors: This type of selector allows filtering by label key and values. So, in layman terms, the equity-based selector will only look for the pods which will have the exact same phrase as that of the label.

Example: Suppose your label key says app=nginx, then, with this selector, you can only look for those pods with label app equal to nginx.

- Selector-Based Selectors: This type of selector allows filtering keys according to a set of values. So, in other words, the selector based selector will look for pods whose label has been mentioned in the set.
  Example: Say your label key says app in (nginx, NPS, Apache). Then, with this selector, if your app is equal to any of nginx, NPS, or Apache, then the selector will take it as a true result.

75. Consider a multinational company with a very much distributed system, with a large number of data centers, virtual machines, and many employees working on various tasks.

How do you think can such a company manage all the tasks in a consistent way with Kubernetes?

As all of us know that I.T. departments launch thousands of containers, with tasks running across a numerous number of nodes across the world in a distributed system.

In such a situation the company can use something that offers them agility, scale-out capability, and DevOps practice to the cloud-based applications.

So, the company can, therefore, use Kubernetes to customize their scheduling architecture and support multiple container formats. This makes it possible for the affinity between container tasks that gives greater efficiency with an extensive support for various container networking solutions and container storage.

76. How do we control the resource usage of POD?

With the use of limit and request resource usage of a POD can be controlled.

Request: The number of resources being requested for a container. If a container exceeds its request for resources, it can be throttled back down to its request.

Limit: An upper cap on the resources a single container can use. If it tries to exceed this predefined limit it can be terminated if K8's decides that another container needs these resources. If you are sensitive towards pod restarts, it makes sense to have the sum of all container resource limits equal to or less than the total resource capacity for your cluster.

Example:

```
apiVersion: v1
kind: Pod
metadata:
 name: demo
spec:
 containers:
 - name: example1
 image:example/example1
 resources:
   requests:
     memory: "_Mi"
     cpu: "_m"
   limits:
     memory: "_Mi"
     cpu: "_m"
```

77. What are the various K8's services running on nodes and describe the role of each service?

Mainly K8 cluster consists of two types of nodes, executor and master.

Executor node: (This runs on master node)

- Kube-proxy: This service is responsible for the communication of pods within the cluster and to the outside network, which runs on every node. This service is responsible to maintain network protocols when your pod establishes a network communication.
- kubelet: Each node has a running kubelet service that updates the running node accordingly with the configuration(YAML or JSON) file. NOTE: kubelet service is only for containers created by Kubernetes.

Master services:

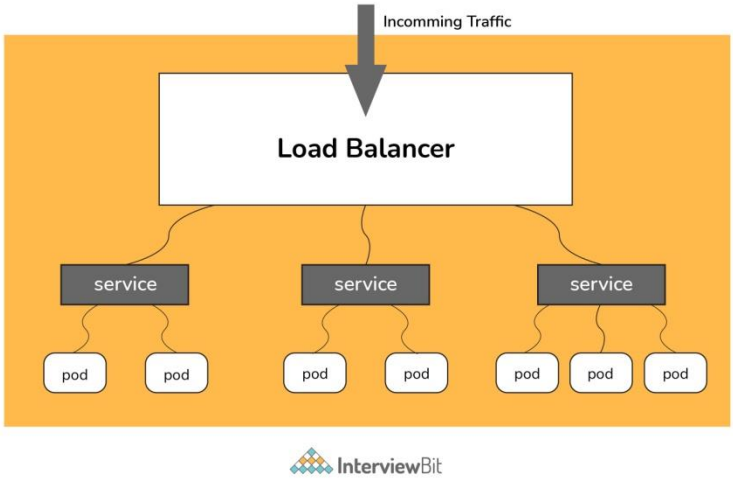
- Kube-apiserver: Master API service which acts as an entry point to K8 cluster.
- Kube-scheduler: Schedule PODs according to available resources on executor nodes.
- Kube-controller-manager: is a control loop that watches the shared state of the cluster through the apiserver and makes changes attempting to move the current state towards the desired stable state

78. What is the role of Load Balance in Kubernetes?

Load balancing is a way to distribute the incoming traffic into multiple backend servers, which is useful to ensure the application available to the users.

In Kubernetes, as shown in the below figure all the incoming traffic lands to a single IP address on the load balancer which is a way to expose your service to outside the internet which routes the incoming traffic to a particular pod (via service) using an algorithm known as round-robin. Even if any pod goes down load balances are notified so that the traffic is not routed to that

particular unavailable node. Thus load balancers in Kubernetes are responsible for distributing a set of tasks (incoming traffic) to the pods



79. What the following in the Deployment configuration file mean?

```
spec:
 containers:
  - name: USER_PASSWORD
    valueFrom:
      secretKeyRef:
        name: some-secret
        key: password
```

Explanation -

USER_PASSWORD environment variable will store the value from the password key in the secret called "some-secret" In other words, you reference a value from a Kubernetes Secret.

80. What is the difference between Docker Swarm and Kubernetes?

Below are the main difference between Kubernetes and Docker:

- The installation procedure of the K8s is very complicated but if it is once installed then the cluster is robust. On the other hand, the Docker swarm installation process is very simple but the cluster is not at all robust.
- Kubernetes can process the auto-scaling but the Docker swarm cannot process the auto-scaling of the pods based on incoming load.
- Kubernetes is a full-fledged Framework. Since it maintains the cluster states more consistently so autoscaling is not as fast as Docker Swarm.

81. How to troubleshoot if the POD is not getting scheduled?

In K8's scheduler is responsible to spawn pods into nodes. There are many factors that can lead to unstartable POD. The most common one is running out of resources, use the commands like kubectl describe <POD> -n <Namespace> to see the reason why POD is not started. Also, keep an eye on kubectl to get events to see all events coming from the cluster.

82. How to run a POD on a particular node?

Various methods are available to achieve it.

- nodeName: specify the name of a node in POD spec configuration, it will try to run the POD on a specific node.
- nodeSelector: Assign a specific label to the node which has special resources and use the same label in POD spec so that POD will run only on that node.
- nodeaffinities: required DuringSchedulingIgnoredDuringExecution, preferredDuringSchedulingIgnoredDuringExecution are hard and soft requirements for running the POD on specific nodes. This will be replacing nodeSelector in the future. It depends on the node labels.

83. What are the different ways to provide external network connectivity to K8?

By default, POD should be able to reach the external network but vice-versa we need to make some changes. Following options are available to connect with POD from the outer world.

- Nodeport (it will expose one port on each node to communicate with it)
- Load balancers (L4 layer of TCP/IP protocol)
- Ingress (L7 layer of TCP/IP Protocol)

Another method is to use Kube-proxy which can expose a service with only cluster IP on the local system port.

$ kubectl proxy --
port=8080 $ http://localhost:8080/api/v1/proxy/namespaces//services/:/

84. How can we forward the port '8080 (container) -> 8080 (service) -> 8080 (ingress) -> 80 (browser)and how it can be done?

The ingress is exposing port 80 externally for the browser to access, and connecting to a service that listens on 8080. The ingress will listen on port 80 by default. An "ingress controller" is a pod that receives external traffic and handles the ingress and is configured by an ingress resource For this you need to configure the ingress selector and if no 'ingress controller selector' is mentioned then no ingress controller will manage the ingress.

Simple ingress Config will look like

```
host: abc.org
http:
paths:
backend:
serviceName: abc-service
servicePort: 8080
Then the service will look like
kind: Service
```

```
apiVersion: v1
metadata:
name: abc-service
spec:
ports:
protocol: TCP
port: 8080 # port to which the service listens to
targetPort: 8080
```

85. If a node is tainted, is there a way to still schedule the pods to that node?

When a node is tainted, the pods don't get scheduled by default, however, if we have to still schedule a pod to a tainted node we can start applying tolerations to the pod spec.

Apply a taint to a node:

kubectl taint nodes node1 key=value:NoSchedule

Apply toleration to a pod:

spec:

tolerations:

- key: "key"

operator: "Equal"

value: "value"

effect: "NoSchedule"

86. What is the difference between a pod and a job?

A Pod always ensure that a container is running whereas the Job ensures that the pods run to its completion. Job is to do a finite task.

Examples:

kubectl run mypod1 --image=nginx --restart=Never

kubectl run mypod2 --image=nginx --restart=onFailure

○ → kubectl get pods

NAME READY STATUS RESTARTS AGE

mypod1 1/1 Running 0 59s

○ → kubectl get job

NAME DESIRED SUCCESSFUL AGE

mypod1 1 0 19s

87. How can containers within a pod communicate with each other?

Containers within a pod share networking space and can reach other on localhost. For instance, if you have two containers within a pod, a MySQL container running on port 3306, and a PHP container running on port 80, the PHP container could access the MySQL one through localhost:3306.

88. Explain the uses of Terraform CLI and list some basic CLI commands?

The Terraform Command-Line Interface (CLI) is used to manage infrastructure and interact with Terraform state, configuration files, providers, etc.

Here are some basic CLI commands:

- terraform init - prepares your working directory for other commands
- terraform destroy - destroys the previously-created infrastructure
- terraform validate - check whether the configuration is valid
- terraform apply - creates or updates the infrastructure
- terraform plan - shows changes needed by the current configuration

89. Give the terraform configuration for creating a single EC2 instance on AWS.

```
provider “aws” {

region = “”}

resource “aws_instance”

“example” {

ami = ""

instance_type = ""

tags {

 Name = "example"}
```

90. Explain State File Locking?

State file locking is Terraform mechanism in which operations on a specific state file are blocked to avoid conflicts between multiple users performing the same process. When one user releases the lock, then only the other one can operate on that state. This helps in preventing state file corruption. This is a backend operation.

91. What do you understand by a Tainted Resource?

A tainted resource is a resource that is forced to be destroyed and recreated on the next apply command. When a resource is marked as tainted, the state files are updated, but nothing changes on infrastructure. The terraform plan out shows that help will get destroyed and recreated. The changes get implemented when the next apply happens.

92. Differentiate between Terraform and Ansible.

Ansible : Ansible is a remarkably straightforward IT automation technology. Configuration management, application deployment, cloud provisioning, ad-hoc task execution, network automation, and multi-node orchestration are all handled by this software. Complex modifications, such as zero-downtime rolling updates with load balancers, are simple using Ansible.

Following table lists the differences between Ansible and Terraform:

Terraform	Ansible
Terraform is a tool for provisioning.	Ansible is a tool for managing configurations.
It uses a declarative Infrastructure as Code methodology.	It takes a procedural method.
It's ideal for orchestrating cloud services and building cloud infrastructure from the ground up.	It is mostly used to configure servers with the appropriate software and to update resources that have previously been

Terraform	Ansible
	configured.
By default, Terraform does not allow bare metal provisioning.	The provisioning of bare metal servers is supported by Ansible.
In terms of packing and templating, it does not provide better support.	It includes complete packaging and templating support.
It is strongly influenced by lifecycle or state management.	It doesn't have any kind of lifecycle management. It does not store the state.

93. What do you mean by a Virtual Private Cloud (VPC)? Which command do you use in Terraform to use a VPC service?

A Virtual Private Cloud (VPC) is a private virtual network within AWS where you can store all of your AWS services. It will have gateways, route tables, network access control lists (ACL), subnets, and security groups, and will be a logical data centre in AWS. When you create a service on a public cloud, it is effectively open to the rest of the world and can be vulnerable to internet attacks. You lock your instances down and secure them from outside threats by putting them inside a VPC. The VPC limits the types of traffic, IP addresses, and individuals who have access to your instances.

This stops unauthorised users from accessing your resources and protects you from DDOS assaults. Because not all services require internet connection, they can be safely stored within a private network. You can then only allow particular machines to connect to the internet.

We use the command aws_vpc to use a VPC Service in Terraform.

94. What do you know about Terraform core? What are the primary responsibilities of Terraform core?

Terraform Core is a binary created in the Go programming language that is statically compiled. The compiled binary is the terraform command line tool (CLI), which is the starting point for anyone who wants to use Terraform. The source code can be found at github.com/hashicorp/terraform.

The primary responsibilities of Terraform core includes:

- Reading and interpolating configuration files and modules using infrastructure as code
- Management of the state of resources
- Resource Graph Construction
- Execution of the plan
- Communication with plugins through RPC