**Jenkins Interview Questions**

**1. What is Jenkins?**

Jenkins is a self-contained, open-source automation server that can be used to automate all sorts of tasks related to building, testing, and delivering or deploying so ware. Jenkins can be installed through native system packages, Docker, or even run standalone by any machine with a Java Runtime Environment (JRE) installed.

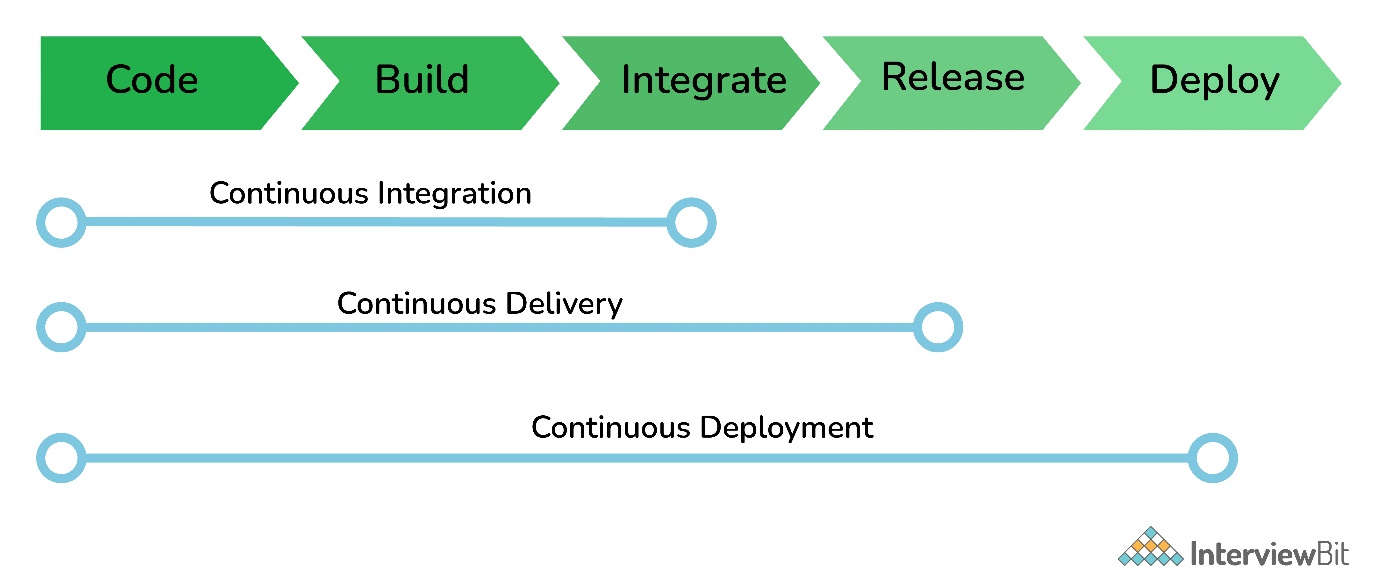
**2. Tell me something about Continuous Integration, Continuous Delivery, and Continuous Deployment?**

**Continuous Integration**: A so ware development process where the changes made to so ware are integrated into the main code as and when a patch is ready so that the so ware will be always ready to be - built, tested, deployed, monitored - continuously.

**Continuous Delivery**: This is a So ware Development Process where the continuously integrated (CI) changes will be tested & deployed continuously into a specific environment, generally through a manual release process, a er all the quality checks are successful

**Continuous Deployment**: A So ware Development practice where the continuously integrated (CI) changes are deployed automatically into the target environment a er all the quality checks are successful

Based on the level of automation, the above three paradigms can be better represented as below -



CI/CD and Continuous Deployment

**3. What are the common use cases Jenkins is used for?**

Jenkins being open-source automation can be used for any kind of so ware-based automation. Some of the common use-cases include but not limited to -

* Software build jobs
* Sanity/Smoke/CI/Regression test jobs
* Web/Data Scraping related jobs
* Code coverage measurement jobs
* General-purpose automation
* Reverse Engineering jobs
* Key Decoding jobs & many other jobs where software automation will be applicable.

**4. What are the ways to install Jenkins?**

Jenkins can be installed using -

1. Native System Package Manager like - apt (Linux), brew (Mac), etc.

2. Docker (popular docker images for Jenkins is available for diﬀerent platforms like Unix/Mac/Windows in the docker registry)

3. Kubernetes (available as a helm chart and can be installed on our Kubernetes clusters)

4. Standalone (on any machine with a Java Runtime Environment installed)

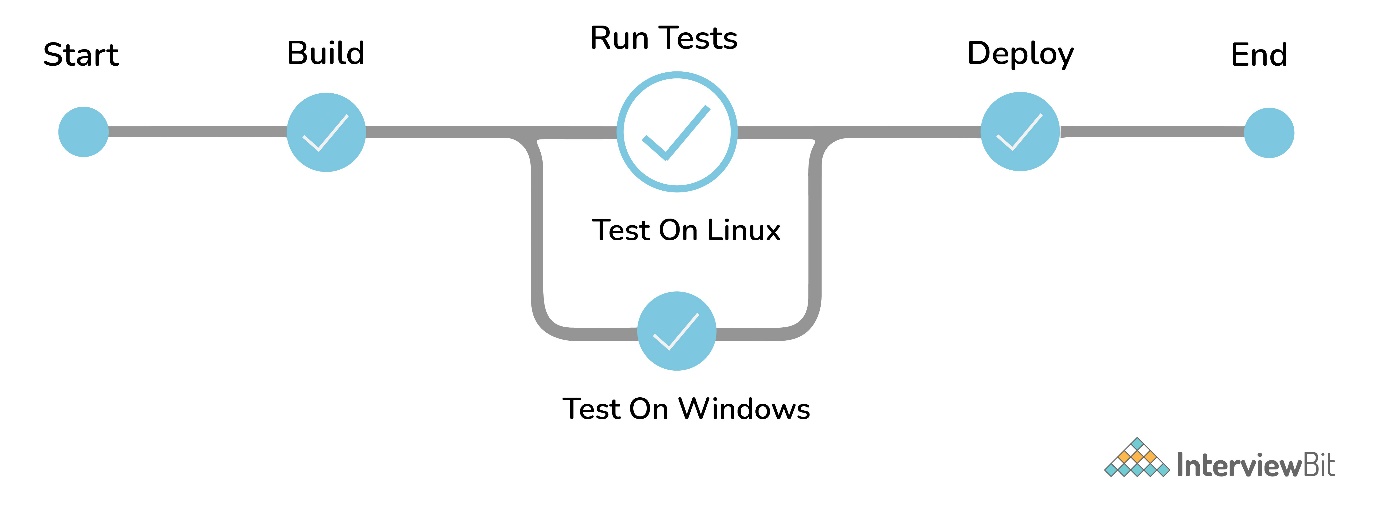
For more detailed installation instructions refer [oﬀicial documentation](https://www.jenkins.io/doc/book/installing/)

**5. What is a Jenkins job?**

A Job/Project is the fundamental unit of a logical work (like a so ware build, an automation task, test execution, etc) using the Jenkins automation server and other required plugins, configurations & infrastructures.

Jobs can be of diﬀerent types like - a freestyle project, a multi-configuration project, a pipeline project, a multi-branch project, etc.

**6. What is a Jenkins Pipeline?**



Jenkins pipeline

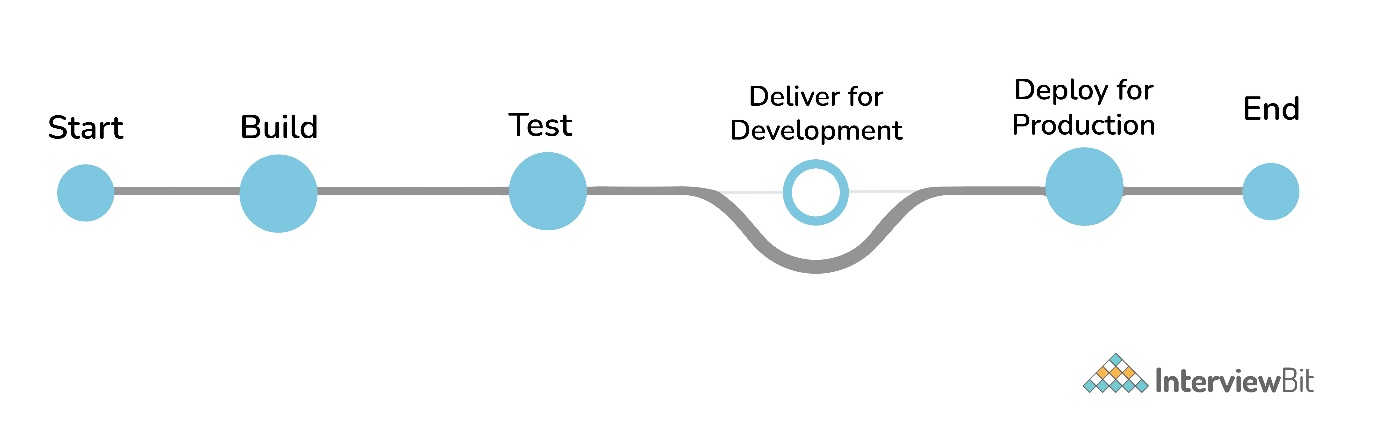
The pipeline is a special type of Jenkins job - simply a sequence of steps controlled by a defined logic - which Orchestrates long-running activities that can span across multiple build agents. It is suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that cannot be easily achieved using a freestyle job.

**7. What are the types of Jenkins pipelines?**

Jenkins Pipelines can be either - a Declarative pipeline or a Scripted Pipeline.

Declarative pipeline makes use of numerous, generic, predefined build steps/stages (i.e. code snippets) to build our job according to our build/automation needs whereas, with Scripted pipelines, the steps/stages can be custom-defined & used using a groovy syntax which provides better control & fine-tuned execution levels.

8. Explain Jenkins Multibranch Pipeline?



Jenkins Multibranch Pipeline

It is a pipeline job that can be configured to Create a set of Pipeline projects according to the detected branches in one SCM repository. This can be used to configure pipelines for all branches of a single repository e.g. if we maintain diﬀerent branches (i.e. production code branches) for diﬀerent configurations like locales, currencies, countries, etc.

**9. 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.Jenkins Interview Questions

**10. How can we stop a scheduled job from being executed temporarily?**

Disable the job from the job details page to temporarily stop all scheduled executions

& other factors/events from triggering the job and enable it back to resume the job schedules/triggers. If a job is not required permanently, we can delete the job from the jobs list view page.

**11. 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

**12. What is Jenkins Build Cause?**

Build Cause is a text attribute that represents what made a job's build to be triggered, say it could be a Jenkins User (from UI), Timer for Scheduled jobs, Upstream jobs for

a job which was triggered by upstream job, etc. This is mainly used to identify the nature of the builds - be it nightly, manual, automated, etc.

**13. How Jenkins knows when to execute a Scheduled job/pipeline and how it is triggered?**

Jenkins master will have the cron entries set up for the jobs as per the scheduled Job's configurations. As and when the time for a particular job comes, it commands agents (based on the configuration of the job) to execute the job with required configurations.

**14. 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.

**15. What are the Scopes of Jenkins Credentials?**

Jenkins credentials can be of one of the two scopes - Global & System

**Global** - the credential will be usable across all the jobs configured in the Jenkins instance (i.e. for all jobs). This is more suited for user Jobs (i.e. for the freestyle, pipeline, or other jobs) to authenticate itself with target services/infrastructures to accomplish the purpose of the job)

**System** - This is a special scope that will allow the Jenkins itself (i.e. the core Jenkins functionalities & some installed plugins) to authenticate itself to external services/infrastructures to perform some defined tasks. E.g. sending emails, etc.

**16. What is a Jenkins Shared Library and how it is useful?**

As an organization starts using more and more pipeline jobs, there is a chance for more and more code being duplicated in every pipeline job, since a part of the build/automation processes will be the same for most of the jobs. In such a situation, every other new upcoming job should also duplicate the same piece of code. To avoid duplications, the Jenkins project brought in the concept of Shared Libraries, to code

- DRY - Don't Repeat Yourself.

Shared libraries are a set of code that can be common for more than one pipeline job and can be maintained separately. Such libraries improve the maintenance, modularity & readability of the pipeline code. And it also speeds up the automation for new jobs.

**17. How Jenkins jobs can be Triggered/Stopped/Controlled programmatically?**

Jenkins Remote Access API can be used to do things like -

* Retrieving information about jobs, views, nodes, builds, etc. from Jenkins for programmatic consumption.
* Trigger a build (both parameterized & non-parameterized), stop/abort a build, enable/disable a Job, group/remove jobs into/from views, etc.
* Create/copy/modify/delete jobs.

and many other programming language-specific functionalities. It has wrappers for main programming languages like - Python, Ruby & Java. It can be triggered via CURL as below -

**Jobs without parameters**

Simply an HTTP POST on JENKINS\_URL/job/JOBNAME/build.

**Jobs with parameters**

Simple example - sending "String Parameters":

curl JENKINS\_URL/job/JOB\_NAME/buildWithParameters  --user USER:TOKEN --data id=123 --data verbosity=high

**18. How to get the Jenkins version programmatically in Jobs/Pipelines or nodes other than master?**

To check the version of Jenkins, load the top-level page or any top-level Remote Access API path like the '.../api/\*' page and then check for the 'X-Jenkins' response header.

This contains the version number of Jenkins, like "1.404". This is also a good way to check if an URL is a Jenkins URL.

**19. What happens when a Jenkins agent is oﬀline and what is the best practice in that situation?**

When a job is tied to a specific agent on a specific node, the job can only be run on that agent and no other agents can fulfill the job request. If the target node is oﬀline or all the agents on that particular node are busy building other jobs, then the triggered job has to wait until the node comes online or an agent from that node becomes available to execute the triggered build request.

As a result, a triggered job may sometimes wait indefinitely without knowing that the target node is oﬀline. So, it is always the best practice to tie the jobs to a group of nodes & agents, referred to with a 'Label'. Once a job is tied to a Label, instead of a specific node/agent, any of the nodes/agents falling under the label can fulfill a build request, when a job is triggered. This way we can reduce the overall turn-around time of the builds.

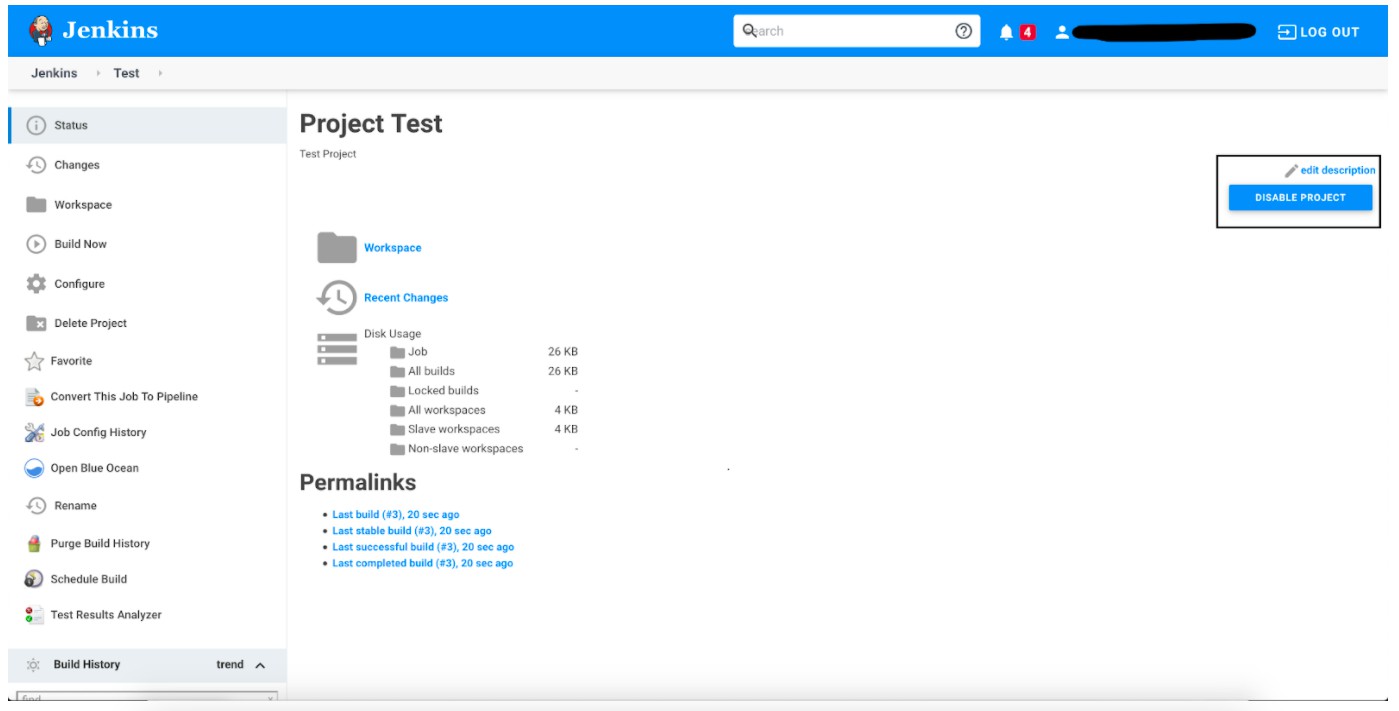
Even then if a job is waiting for more time for the nodes/agents, then it is time to consider adding more nodes/agents.

**20. What is the Blue Ocean?**

Blue Ocean is the redefined user experience for Jenkins. Designed from the ground up for Jenkins Pipeline, it is still compatible with freestyle jobs, Blue Ocean reduces clutter and increases clarity. Blue Ocean’s main features include -

* Sophisticated visualizations of continuous delivery (CD) Pipelines, allowing for fast and intuitive comprehension of your Pipeline’s status.
* Pipeline editor - makes the creation of Pipelines approachable by guiding the user through an intuitive and visual process to create a Pipeline.
* Personalization to suit the role-based needs of each member of the team.
* Pinpoint precision when intervention is needed and/or issues arise. Blue Ocean shows where in the pipeline attention is needed, facilitating exception handling and increasing productivity.
* Native integration for branch and pull requests, enables maximum developer productivity when collaborating on code with others in GitHub, Bitbucket, etc.

Conventional UI - Job Details Page



Conventional UI Jenkins

**21. What is the Jenkins User Content service?**

Jenkins has a mechanism known as "User Content", where administrators can place files inside the $JENKINS\_HOME/userContent folder and these files are served from yourhost/jenkins/userContent

This can be thought of as a mini HTTP server to serve images, stylesheets, and other static resources that you can use from various description fields inside Jenkins.

Advanced Interview Questions

**22. How is continuous integration achieved using Jenkins?**

Continuous integration is a process where a developer’s code changes are constantly integrated into the main code and the same will be tested automatically and the results of the tests will decide whether the change is ready for deployment. In this process -

* Developer Makes a change - commit/pull\_request - in feature/dev branch
* Source Control Management system generates appropriate events
* SCM Specific Jenkins Plugins like Git/SVN will detect those events from the configured repositories and these events will be used to Trigger - build/dependent/test - jobs on Jenkins
* After the Test/Dependent jobs are completed, the change/patch will be labeled according to the status of the test job
* Based on the Status (i.e. readiness of a change to be merged with the main branch), the Continuous Delivery or Continuous Deployment strategy/tool will take it forward.

**23. What is Artifact Archival & how to do it in Pipelines?**

Artifacts are the exportable/storable/archivable results of a specific job build. This can be configured using a plugin called - Copy artifact Plugin. Based on the configured pattern, the files/directories matching the configured patterns will be archived for a Jenkins build, which can be used for future references. In the pipeline, it can be configured as follows -

**archiveArtifacts artifacts: 'output/\*\*/\*'**

**24. How to configure inclusions & exclusions in Artifacts Archival?**

Artifact archival takes in a pattern for matching target files. Similarly, it also takes in a pattern (ANT build system pattern for matching files) for exclusion as well which will be ignored while selecting the files for archival.

For e.g.

archiveArtifacts artifacts: 'output/\*.txt', excludes: 'output/specific\_file.txt'

The above command will archive all the text files from the output folder except specific\_file.txt

**25. How can we share information between diﬀerent build steps or stages in a Jenkins Job?**

Every build step or stage will be running in its process and hence sharing information between two diﬀerent build steps is not so direct. We can use either a File, a Database Entry, an Environment Variable, etc. to share info from one build step to another or a post-build action.

**26. 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.

**27. Default Environment Variables by Jenkins & How to introduce custom environment variables?**

Jenkins provides several environment variables by default like - BRANCH\_NAME, BUILD\_NUMBER, BUILD\_TAG, WORKSPACE, etc.

**28. How can a job configuration be reset to an earlier version/state?**

From the Job details page, we can use Job Config History to - See diﬀ, Review & Revert the Job configs from the history of changes we have made to a particular job. This will be super useful when a job is misconfigured by someone by mistake, it can be reviewed and reverted easily to any of its earlier states.

**29. How to do Global Tools Configuration in Jenkins?**

Global Tools are tools that need to be installed outside the Jenkins environment and need to be controlled from within the Jenkins environment. Hence it needs its corresponding Jenkins plugin as well. Steps to using a Global Tool generally include -

* Install the tool Plugin into the Jenkins instance, to include the global tool into a list of global tools used by Jenkins.
* Install the tool in the Jenkins instance or provide away (maybe a command to download and) install the tool during runtime.
* Go to Manage Jenkins -> Global Tools Configuration and Scroll through the tool list and configure the global tool-specific configurations.
* Make use of the installed global Tool in your job/pipeline

**30. How to create & use a Shared Library in Jenkins?**

Basic requirements for a Jenkins shared library to be used in a Pipeline Code are -

* A Repository with pipeline shared library code in SCM.
* An appropriate SCM Plugin configuration for the Jenkins instance.
* Global Shared Library should be configured in Jenkins Global configuration.
* Include the Shared Library in the Pipeline Code and use the methods defined in the Jenkins Shared Library.

**E.g.**

#!/urs/bin/env groovy  
@Library('fs\_jenkins\_shared\_library@v2.0.7')\_

**31. How to install a Custom Jenkins Plugin or a Version of Plugin Not available in Jenkins Update Center?**

Generally, it is the best practice to use the latest version of a plugin. But there are ways to install custom plugins or outdated versions of a published plugin. Jenkins Plugins are exported using a .hpi file and the same can be installed in multiple ways -

**Using the Jenkins CLI**

java -jar jenkins-cli.jar -s http://localhost:8080/ install-plugin SOURCE ... [-deploy] [-name VAL] [-restart]

The above command Installs a plugin either from a file, an URL or from the update center.

* SOURCE: If this points to a local file, that file will be installed. If this is an URL, Jenkins downloads the URL and installs that as a plugin. Otherwise, the name is assumed to be the short name of the plugin in the existing update center (like "findbugs") and the plugin will be installed from the update center.
* -deploy: Deploy plugins right away without postponing them until the reboot.
* -name VAL: If specified, the plugin will be installed as this short name (whereas normally the name is inferred from the source name automatically).
* -restart: Restart Jenkins upon successful installation.

**Advanced Installation - via - Web UI**

* Assuming a .hpi file has been downloaded, a logged-in Jenkins administrator may upload the file from within the web UI:
* Navigate to the Manage Jenkins > Manage Plugins page in the web UI.
* Click on the Advanced tab.
* Choose the .hpi file under the Upload Plugin section.
* Upload the plugin file.
* Restart the Jenkins instance

**Advanced Installation - via - On the master**

Assuming a .hpi file has been explicitly downloaded by a systems administrator, the administrator can manually place the .hpi file in a specific location on the file system.

Copy the downloaded .hpi file into the JENKINS\_HOME/plugins directory on the Jenkins controller (for example, on Debian systems JENKINS\_HOME is generally /var/lib/jenkins).

The master will need to be restarted before the plugin is loaded and made available in the Jenkins environment

**32. 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>

**33. What is Jenkins Remote Access API?**

Jenkins provides remote access API to most of its functionalities (though some functionalities are programming language-dependent). Currently, it comes in three flavors -

* XML
* JSON with JSONP support
* Python

Remote access API is offered in a REST-like style. That is, there is no single entry point for all features, and instead, they are available under the ".../api/" URL where the "..." portion is the data that it acts on.

For example, if your Jenkins installation sits at interviewbit.com, visiting /api/ will show just the top-level API features available – primarily a listing of the configured jobs for this Jenkins instance.

Or if we want to access information about a particular build, e.g. https://ci.jenkins.io/job/Infra/job/jenkins.io/job/master/lastSuccessfulBuild/, then go to https://ci.jenkins.io/job/Infra/job/jenkins.io/job/master/lastSuccessfulBuild/api/ and you’ll see the list of functionalities for that build.

**34. What is In-process Script Approval and how it works?**

Jenkins, and several plugins, allow users to execute Groovy scripts in Jenkins. To protect Jenkins from the execution of malicious scripts, these plugins execute user- provided scripts in a Groovy Sandbox that limits what internal APIs are accessible.

This protection is provided by the Script Security plugin. As soon as an unsafe method is used in any of the scripts, the "In-process Script Approval" action should appear in "Manage Jenkins" to allow Administrators to make a decision about which unsafe methods, if any, should be allowed in the Jenkins environment.

This in-process script approval inherently improves the security of the overall Jenkins ecosystem.

**35. Can we monitor Jenkins using common Observability tools?**

Common monitoring platforms like **DataDog, Prometheus, JavaMelody** & few others - have their corresponding Jenkins plugin, which when configured, sends Metrics to the corresponding Monitoring platform, which can then be Observed with the latest tools & technologies. The same can be configured with Alarms & Notifications for immediate attention when something goes wrong.

**36. What is a Ping Thread in Jenkins and how it works?**

Jenkins installs "ping thread" on every remote connection, such as Controller/Agent connections, regardless of its transport mechanism (such as SSH, JNLP, etc.). The lower level of the Jenkins Remoting Protocol is a message-oriented protocol, and a ping thread periodically sends a ping message that the receiving end will reply to. The ping thread measures the time it takes for the reply to arrive, and if it’s taking excessive time (currently 4 minutes and configurable), then it assumes that the connection was lost and initiates the formal close down.

This is to avoid an infinite hang, as some of the failure modes in the network cannot be detected otherwise. The timeout is also set to a long enough value so that a temporary surge in the load or a long garbage collection pause will not trip oﬀ the close-down.

Ping thread is installed on both controller & agent; each side pings the other and tries to detect the problem from their sides.

The ping thread time out is reported through java.util.logging. Besides, the controller will also report this exception in the agent launch log. Note that some agent launchers, most notably SSH agents, writes all stdout/stderr outputs from the agent JVM into this same log file, so you need to be careful.

**Docker Interview Questions**

**Introduction to Docker:**  
Docker is a very popular and powerful open-source containerization platform that is used for building, deploying, and running applications. Docker allows you to decouple the application/software from the underlying infrastructure.

**1.What is a Container?**  
A container is a standard unit of software bundled with dependencies so that applications can be deployed fast and reliably b/w different computing platforms.

* Docker can be visualized as a big ship (docker) carrying huge boxes of products (containers).
* Docker container doesn’t require the installation of a separate operating system. Docker just relies or makes use of the kernel’s resources and its functionality to allocate them for the CPU and memory it relies on the kernel’s functionality and uses resource isolation for CPU and memory, and separate namespaces to isolate the application’s view of the OS (operating system).

Diagram

Description automatically generated

**2.Why Learn Docker?**  
Application development is a lot more than just writing code! They involve a lot of behind-the-scenes things like usage of multiple frameworks and architectures for every stage of its lifecycle which makes the process more complex and challenging. Using the nature of containerization helps developers to simplify and efficiently accelerate the application workflow along with giving them the liberty to develop using their own choice of technology and development environments.

* All these aspects form the core part of DevOps which becomes all the more important for any developer to know these in order to improve productivity, fasten the development along with keeping in mind the factors of application scalability and more efficient resource management.
* Imagine containers as a very lightweight pre-installed box with all the packages, dependencies, software required by your application, just deploy to production with minimal configuration changes.
* Lots of companies like PayPal, Spotify, Uber, etc use Docker to simplify the operations and to bring the infrastructure and security closer to make more secure applications.
* Being portable, Containers can be deployed on multiple platforms like bare instances, virtual machines, Kubernetes platform etc. as per requirements of scale or desired platform.

**3.Can you tell something about docker container?**

* In simplest terms, docker containers consist of applications and all their dependencies.
* They share the kernel and system resources with other containers and run as isolated systems in the host operating system.
* The main aim of docker containers is to get rid of the infrastructure dependency while deploying and running applications. This means that any containerized application can run on any platform irrespective of the infrastructure being used beneath.
* Technically, they are just the runtime instances of docker images.

**4.What are docker images?**

They are executable packages(bundled with application code & dependencies, software packages, etc.) for the purpose of creating containers. Docker images can be deployed to any docker environment and the containers can be spun up there to run the application.

**5.What is a DockerFile?**

* It is a text file that has all commands which need to be run for building a given image.

A picture containing logo

Description automatically generated

**6.Can you tell what is the functionality of a hypervisor?**

A hypervisor is a software that makes virtualization happen because of which is sometimes referred to as the Virtual Machine Monitor. This divides the resources of the host system and allocates them to each guest environment installed.

Graphical user interface

Description automatically generated

This means that multiple OS can be installed on a single host system. Hypervisors are of 2 types:  
  
1. **Native Hypervisor:** This type is also called a Bare-metal Hypervisor and runs directly on the underlying host system which also ensures direct access to the host hardware which is why it does not require base OS.  
2. **Hosted Hypervisor:** This type makes use of the underlying host operating system which has the existing OS installed

**7.What can you tell about Docker Compose?**

It is a YAML file consisting of all the details regarding various services, networks, and volumes that are needed for setting up the Docker-based application. So, docker-compose is used for creating multiple containers, host them and establish communication between them. For the purpose of communication amongst the containers, ports are exposed by each and every container.

**8.Can you tell something about docker namespace?**

A namespace is basically a Linux feature that ensures OS resources partition in a mutually exclusive manner. This forms the core concept behind containerization as namespaces introduce a layer of isolation amongst the containers. In docker, the namespaces ensure that the containers are portable and they don't affect the underlying host. Examples for namespace types that are currently being supported by Docker – PID, Mount, User, Network, IPC.

**9.What is the docker command that lists the status of all docker containers?**

In order to get the status of all the containers, we run the below command: **docker ps -a**

**10.On what circumstances will you lose data stored in a container?**

The data of a container remains in it until and unless you delete the container.

**11.What is docker image registry?**

* A Docker image registry, in simple terms, is an area where the docker images are stored. Instead of converting the applications to containers each and every time, a developer can directly use the images stored in the registry.
* This image registry can either be public or private and Docker hub is the most popular and famous public registry available.

**12.How many Docker components are there?**

There are three docker components, they are - Docker Client, Docker Host, and Docker Registry.

* **Docker Client:** This component performs “build” and “run” operations for the purpose of opening communication with the docker host.
* **Docker Host:** This component has the main docker daemon and hosts containers and their associated images. The daemon establishes a connection with the docker registry.
* **Docker Registry:** This component stores the docker images. There can be a public registry or a private one. The most famous public registries are Docker Hub and Docker Cloud.

Graphical user interface, application, Teams

Description automatically generated

**13.What is a Docker Hub?**

* It is a public cloud-based registry provided by Docker for storing public images of the containers along with the provision of finding and sharing them.
* The images can be pushed to Docker Hub through the **docker push** command.

**14.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>**

**15.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**

**16.Can a paused container be removed from Docker?**

No, it is not possible! A container MUST be in the stopped state before we can remove it.

**17.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}}'**

**18. Differentiate between virtualization and containerization.**

The question indirectly translates to explaining the difference between virtual machines and Docker containers.

| **Virtualization** | **Containerization** |
| --- | --- |
| This helps developers to run and host multiple **OS** on the hardware of a single physical server. | This helps developers to deploy multiple **applications** using the same operating system on a single virtual machine or server. |
| **Hypervisors** provide overall virtual machines to the guest operating systems. | **Containers** ensure isolated environment/ user spaces are provided for running the applications. Any changes done within the container do not reflect on the host or other containers of the same host. |
| These virtual machines form an **abstraction of the system hardware** **layer**this means that each virtual machine on the host acts like a physical machine. | Containers form **abstraction of the application** **layer** which means that each container constitutes a different application. |

### 19.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.

**20.Can a container restart by itself?**

* Yes, it is possible only while using certain docker-defined policies while using the docker run command. Following are the available policies:  
    
  1. **Off:** In this, the container won’t be restarted in case it's stopped or it fails.  
  2. **On-failure**: Here, the container restarts by itself only when it experiences failures not associated with the user.  
  3. **Unless-stopped:** Using this policy, ensures that a container can restart only when the command is executed to stop it by the user.  
  4. **Always:** Irrespective of the failure or stopping, the container always gets restarted in this type of policy.  
    
  These policies can be used as:  
  docker run -dit — restart [restart-policy-value] [container\_name]

**21.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.

**22.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 tore the container data.

**23.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/

**24.What does the docker info command do?**

The command gets detailed information about Docker installed on the host system. The information can be like what is the number of containers or images and in what state they are running and hardware specifications like total memory allocated, speed of the processor, kernel version, etc.

**25.Can you tell the what are the purposes of up, run, and start commands of docker compose?**

* Using the up command for keeping a docker-compose up (ideally at all times), we can start or restart all the networks, services, and drivers associated with the app that are specified in the docker-compose.yml file. Now if we are running the docker-compose up in the “attached” mode then all the logs from the containers would be accessible to us. In case the docker-compose is run in the “detached” mode, then once the containers are started, it just exits and shows no logs.
* Using the run command, the docker-compose can run one-off or ad-hoc tasks based on the business requirements. Here, the service name has to be provided and the docker starts only that specific service and also the other services to which the target service is dependent (if any).  
  - This command is helpful for testing the containers and also performing tasks such as adding or removing data to the container volumes etc.
* Using the start command, only those containers can be restarted which were already created and then stopped. This is not useful for creating new containers on its own.

**26.What are the basic requirements for the docker to run on any system?**

Docker can run on both Windows and Linux platforms.

* For the Windows platform, docker atleast needs Windows 10 64bit with 2GB RAM space. For the lower versions, docker can be installed by taking help of the toolbox. Docker can be downloaded from <https://docs.docker.com/docker-for-windows/> website.
* For Linux platforms, Docker can run on various Linux flavors such as Ubuntu >=12.04, Fedora >=19, RHEL >=6.5, CentOS >=6 etc.

**27.Can you tell the approach to login to the docker registry?**

Using the docker login command credentials to log in to their own cloud repositories can be entered and accessed.

**28.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

**29 can you differentiate between Daemon Logging and Container Logging?**

* In docker, logging is supported at 2 levels and they are logging at the Daemon level or logging at the Container level.
* **Daemon Level:** This kind of logging has four levels- Debug, Info, Error, and Fatal.  
  - Debug has all the data that happened during the execution of the daemon process.  
  - Info carries all the information along with the error information during the execution of the daemon process.  
  - Errors have those errors that occurred during the execution of the daemon process.  
  - Fatal has the fatal errors that occurred during the execution.
* **Container Level:**  
  - Container level logging can be done using the command: sudo docker run –it <container\_name> /bin/bash  
  - In order to check for the container level logs, we can run the command: sudo docker logs <container\_id>

**30.What is the way to establish communication between docker host and Linux host?**

This can be done using networking by identifying the “ipconfig” on the docker host. This command ensures that an ethernet adapter is created as long as the docker is present in the host.

**31.What is the best way of deleting a container?**

We need to follow the following two steps for deleting a container:  
- docker stop <container\_id>  
- docker rm <container\_id>

**32.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.

**33.Can we use JSON instead of YAML while developing docker-compose file in Docker?**

Yes! It can be used. In order to run docker-compose with JSON, docker-compose -f docker-compose.json up can be used.

**34.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.

**35.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.

Diagram

Description automatically generated

### 36. How to use docker for multiple application environments?

* Docker-compose feature of docker will come to help here. In the docker-compose file, we can define multiple services, networks, and containers along with the volume mapping in a clean manner, and then we can just call the command “docker-compose up”.
* When there are multiple environments involved - it can be either dev, staging, uat, or production servers, we would want to define the server-specific dependencies and processes for running the application. In this case, we can go ahead with creating environment-specific docker-compose files of the name “docker-compose.{environment}.yml” and then based on the environment, we can set up and run the application.

### 37. 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.

**GIT Interview Questions**

### What is Git and why is it used?

* Git is the most popular, open-source, widely used, and an example of distributed version control system (DVCS) used for handling the development of small and large projects in a more efficient and neat manner.
* It is most suitable when there are multiple people working on projects as a team and is used for tracking the project changes and efficiently supports the collaboration of the development process.
* With the help of the versioning system, the developer can identify who has made what changes and then run tests and fix bugs if any and then do necessary feature implementation. In case of any unforeseen circumstances, the code can be reverted to any of the previously working versions thereby saving huge efforts.

A picture containing diagram

Description automatically generated

### ****Scope of Git:****

* Due to a well-established version control system and the support for collaborative work, git has garnered wide popularity not just amongst the software developers, but also among the people who do other tasks like documentation or any other collaborative work. It can seem challenging at first, but once we get the hang of git, we find that it makes our lives much simpler.
* It has an amazing branching system that supports nonlinear development along with keeping the developers accountable for their code. This helps in making the development process efficient and faster.

### 1. What is a version control system (VCS)?

A VCS keeps track of the contributions of the developers working as a team on the projects. They maintain the history of code changes done and with project evolution, it gives an upper hand to the developers to introduce new code, fixes bugs, and run tests with confidence that their previously working copy could be restored at any moment in case things go wrong.

### 2. What is a git repository?

A repository is a file structure where git stores all the project-based files. Git can either stores the files on the local or the remote repository.

### 3. What does git clone do?

The command creates a copy (or clone) of an existing git repository. Generally, it is used to get a copy of the remote repository to the local repository.

### 4. What does the command git config do?

The git config command is a convenient way to set configuration options for defining the behavior of the repository, user information and preferences, git installation-based configurations, and many such things.   
  
For example:  
To set up your name and email address before using git commands, we can run the below commands:

* git config --global  
  user.name  
  “<<your\_name>>”
* git config --global user.email “<<your\_email>>”

### 5. Can you explain head in terms of git and also tell the number of heads that can be present in a repository?

* A head is nothing but a reference to the last commit object of a branch.
* For every repository, there will always be a default head referred to as “master” or now “main” (as per GitHub) but there is no restriction to the count of heads available. In other words, it can have any number of heads.
* **Usages:**  
    
  - To go or checkout to 1 commit before the latest commit, we use git checkout HEAD~1  
    
  - To uncommit the last 3 commits without losing the changes, we first run git reset HEAD~3. Then we can see the changes made in the last 3 commits and then update it manually and commit it finally.  
    
  - In order to uncommit the last 3 commits and also remove the changes, we can run the command: git reset --hard HEAD~3. This command will completely remove all the changes.  
    
  - To look into the changes made in the last 3 commits, we can run git diff HEAD~3  
    
  - To make a new commit by reverting the last 3 commits, we can run the command: git revert --no-commit HEAD~3...HEAD

### 6. What is a conflict?

* Git usually handles feature merges automatically but sometimes while working in a team environment, there might be cases of conflicts such as:  
    
  1. When two separate branches have changes to the same line in a file  
  2. A file is deleted in one branch but has been modified in the other.
* These conflicts have to be solved manually after discussion with the team as git will not be able to predict what and whose changes have to be given precedence.

Diagram, schematic

Description automatically generated

### 7. What is the functionality of git ls-tree?

This command returns a tree object representation of the current repository along with the mode and the name of each item and the SHA-1 value of the blob.

### 8. What does git status command do?

git status command is used for showing the difference between the working directory and the index which is helpful for understanding git in-depth and also keep track of the tracked and non-tracked changes.

### 9. Define “Index”.

Before making commits to the changes done, the developer is given provision to format and review the files and make innovations to them. All these are done in the common area which is known as ‘Index’ or ‘Staging Area’.

Diagram

Description automatically generated

In the above image, the “staged” status indicates the staging area and provides an opportunity for the people to evaluate changes before committing them.

### 10. What does git add command do?

* This command adds files and changes to the index of the existing directory.
* You can add all changes at once using git add . command.
* You can add files one by one specifically using git add <file\_name> command.
* You can add contents of a particular folder by using git add /<folder\_name>/ command.

## Intermediate GIT Interview Questions

### 11. Why is it considered to be easy to work on Git?

With the help of git, developers have gained many advantages in terms of performing the development process faster and in a more efficient manner. Some of the main features of git which has made it easier to work are:

* **Branching Capabilities:**  
    
  - Due to its sophisticated branching capabilities, developers can easily work on multiple branches for the different features of the project.  
  - It also has an easier merge option along with an efficient work-flow feature diagram for tracking it.
* **Distributed manner of development:**  
    
  - Git is a distributed system and due to this nature, it became easier to trace and locate data if it's lost from the main server.  
  - In this system, the developer gets a repository file that is present on the server. Along with this file, a copy of this is also stored in the developer’s system which is called a local repository.  
  - Due to this, the scalability of the project gets drastically improved.
* **Pull requests feature:**  
    
  - This feature helps in easier interaction amongst the developers of a team to coordinate merge-operations.  
  - It keeps a proper track of the changes done by developers to the code.
* **Effective release cycle:**  
    
  - Due to the presence of a wide variety of features, git helps to increase the speed of the release cycle and helps to improve the project workflow in an efficient manner.

### 12. How will you create a git repository?

* Have git installed in your system.
* Then in order to create a git repository, create a folder for the project and then run git init.
* Doing this will create a .git file in the project folder which indicates that the repository has been created.

### 13. Tell me something about git stash?

Git stash can be used in cases where we need to switch in between branches and at the same time not wanting to lose edits in the current branch. Running the git stash command basically pushes the current working directory state and index to the stack for future use and thereby providing a clean working directory for other tasks.

### 14. What is the command used to delete a branch?

* To delete a branch we can simply use the command git branch –d [head].
* To delete a branch locally, we can simply run the command: git branch -d <local\_branch\_name>
* To delete a branch remotely, run the command: git push origin --delete <remote\_branch\_name>
* Deleting a branching scenario occurs for multiple reasons. One such reason is to get rid of the feature branches once it has been merged into the development branch.

### 15. What differentiates between the commands git remote and git clone?

git remote command creates an entry in  git config that specifies a name for a particular URL. Whereas git clone creates a new git repository by copying an existing one located at the URL.

### 16. What does git stash apply command do?

* git stash apply command is used for bringing the works back to the working directory from the stack where the changes were stashed using git stash command.
* This helps the developers to resume their work where they had last left their work before switching to other branches.

### 17. 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.  current 2. For changes to be reflected in the current / target branch, git fetch should be followed by git merge command. |

git pull = git fetch + git merge

### 18. Can you give differences between “pull request” and “branch”?

| **pull request** | **branch** |
| --- | --- |
| This process is done when there is a need to put a developer’s change into another person’s code branch. | A branch is nothing but a separate version of the code. |

### 19. Why do we not call git “pull request” as “push request”?

* “Push request” is termed so because it is done when the target repository requests us to push our changes to it.
* “Pull request” is named as such due to the fact that the repo requests the target repository to grab (or pull) the changes from it.

Diagram

Description automatically generated

### 20. Can you tell the difference between Git and GitHub?

| **Git** | **GitHub** |
| --- | --- |
| This is a distributed version control system **installed on local machines** which allow developers to keep track of commit histories and supports collaborative work. | This is a **cloud-based source code repository** developed by using git. |
| This is maintained by “The Linux Foundation”. | This was acquired by “Microsoft” |
| SVN, Mercurial, etc are the competitors | GitLab, Atlassian BitBucket, etc are the competitors. |

* GitHub provides a variety of services like forking, user management, etc along with providing a central repository for collaborative work.

### 21. What do the git diff and git status commands do?

| **git diff** | **git status** |
| --- | --- |
| This shows the changes **between commits, working trees,** etc. | This shows the difference **between the working directory and index** that is essential in understanding git in depth. |

* git diff works in a similar fashion to git status with the only difference of showing the differences between commits and also between the working directory and index.

### 22. What has to be run to squash multiple commits (last N) into a single commit?

Squashing multiple commits to a single one overwrites the history which is why it is recommended to be done using full caution. This step can be done by running the command: git rebase -i HEAD~{{N}} where {{N}} represents the number of commits needed to be squashed.

### 23. How would you recover a branch that has already pushed changes in the central repository but has been accidentally deleted from every team member’s local machines?

We can recover this by checking out the latest commit of this branch in the reflog and then checking it out as a new branch.

### 24. 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.

### 25. What consists of a commit object?

A commit object consists of the following components:

* A set of files that represents the state of a project at a given point in time.
* Reference to parent commit objects.
* A 40 character string termed as SHA-1 name uniquely identifies the commit object.

### 26. Explain the levels in git config and how can you configure values using them?

* In order to make git work, it uses a set of configurations that are pre-defined by default by means of configuration files (or config files). We can change the default behavior of git by just modifying these files which are basically text files. In order to do this, it is important to understand how git identifies these files. It does so by following the below steps:  
    
  - Firstly, git searches for the config values in the system-wide gitconfig file stored in <<installation\_path>>/etc/gitconfig file that has settings defined and applied to **every user** of the system and all their repos.  
       - In case you want git to search from this particular file and read/write on it, we can pass the option --system to git config command.  
    
  - Next, git searches for the ~/.gitconfig file or ~/.config/git/config that has the scope specific to the user.  
     - Git can be made to read/ write from this file specifically bypassing --global to the git config command.  
    
  - Lastly, git searches for the config values in the git directory of the local repository that we are currently working on.  
     - These config values are specific to that particular repository alone and can be accessed by passing --local to the git config command.This is the default config file that gets accessed and modified upon in case we do not specify any levels.

### 27. What is a detached HEAD and what causes this and how to avoid this?

Detached HEAD indicates that the currently checked-out repository is not a local branch. This can be caused by the following scenarios:

* When a branch is a read-only branch and we try to create a commit to that branch, then the commits can be termed as “free-floating” commits not connected to any branch. They would be in a detached state.
* When we checkout a tag or a specific commit and then we try to perform a new commit, then again the commits would not be connected to any branch. When we now try to checkout a branch, these new commits would be automatically placed at the top.  
    
  In order to ensure that detached state doesn't happen, =instead of checking out commit/tag, we can create a branch emanating from that commit and then we can switch to that newly created branch by using the command: git checkout -b <<new\_branch\_name>>. This ensures that a new branch is checkout out and not a commit/tag thereby ensuring that a detached state wouldn't happen.

### 28. What does git annotate command do?

* This command annotates each line within the given file with information from the commit which introduced that change. This command can also optionally annotate from a given revision.
* Syntax: git annotate [<options>] <file> [<revision>]
* You can get to learn more about this command from the official git documentation [here](https://git-scm.com/docs/git-annotate).

### 29. What is the difference between git stash apply vs git stash pop command?

* git stash pop command throws away the specified stash (topmost stash by default) after applying it.
* git stash apply command leaves the stash in the stash list for future reuse. In case we wanted to remove it from the list, we can use the git stash drop command.

git stash pop = git stash apply + git stash drop

## Advanced GIT Interview Questions

### 30. What command helps us know the list of branches merged to master?

* git branch --merged helps to get the list of the branches that have been merged into the current branch.
* Note: git branch --no-merged lists the branches that have not been merged to the current branch.

### 31. How will you resolve conflict in Git?

* Conflicts occur whenever there are multiple people working on the same file across multiple branches. In such cases, git won't be able to resolve it automatically as it is not capable of deciding what changes has to get the precedence.
* Following are the steps are done in order to resolve git conflicts:  
  1. Identify the files that have conflicts.  
  2. Discuss with members who have worked on the file and ensure that the required changes are done in the file.  
  3. Add these files to the staged section by using the git add command.  
  4. Commit these changes using the git commit command.  
  5. Finally, push the changes to the branch using the git.

### 32. What is best advisable step in cases of broken commit: Create an additional commit OR amend an existing commit?

* It is always advisable to create an additional commit rather than amending the existing commit due to the following reasons:  
  - Doing the amend operation destroys the previously saved state of that commit. If only the commit message gets changes or destroyed, it's acceptable but there might be cases when the contents of the commits get amended. This results in the loss of important information associated with the commit.  
  - Over usage of git commit --amend can have severe repercussions as the small commit amend can continue to grow and gather unrelated changes over time.

### 33. How to revert a bad commit which is already pushed?

There can be cases where we want to revert from the pushed changes and go back to the previous version. To handle this, there are two possible approaches based on the situations:

* **Approach 1**: Fix the bad changes of the files and create a new commit and push to the remote repository. This step is the simplest and most recommended approach to fix bad changes. You can use the command: git commit -m "<message>"
* **Approach 2**: New commit can be created that reverts changes done in the bad commit. It can be done using git revert <name of bad commit>

### 34. What is the functionality of “git cherry-pick” command?

This command is used to introduce certain commits from one branch onto another branch within the repository. The most common use case is when we want to forward- or back-port commits from the maintenance branch to the development branch.

### 35. Explain steps involved in removing a file from git index without removing from the local file system?

* Sometimes we end up having certain files that are not needed in the git index when we are not being careful while using the git add command. Using the command git rm will remove the file from both the index and the local working tree which is not always desirable.
* Instead of using the git rm command we can use the git reset command for removing the file from the staged version and then adding that file to the .gitignore file to avoid repeating the same mistake again.

**git reset <file\_name> # remove file from index**

**echo filename >> .gitingore # add file to .gitignore to avoid mistake repetition.**

### 36. What are the factors involved in considering which command to choose among: git merge and git rebase?

Both these commands ensure that changes from one branch are integrated into another branch but in very different ways. Git rebasing can be thought of as saying to use another branch as a new base for the work.

* Whenever in doubt, it is always preferred to use the git merge command.  
    
  Following are some factors that tell when to use merge and rebase commands:
* In case our branch gets contributions from other developers outside the team as in open-source or public repositories, then rebase is not preferred.  
  - This is because rebase destroys the branch and it results in broken and inconsistent repositories unless the git pull --rebase command is used.
* Rebase is a very destructive operation. If not applied correctly, it results in loss of committed work which might result in breaking the consistency of other developer’s contribution to the repository.
* If the model of having branches per feature is followed, rebasing is not a good idea there because it keeps track of related commits done by the developers. But in case the team follows having branches per developer of the team, then the branch has no additional useful information to be conveyed. In this model, rebasing has no harm and can be used.
* If there is any chance where there might be a necessity to revert a commit to previous commits, then reverting a rebase would be almost impossible as the commit data would be destroyed. In such cases, the merge can be used.

### 37. How do you find a commit which broke something after a merge operation?

* This can be a time-consuming process if we are not sure what to look at exactly. Fortunately, git provides a great search facility that works on the principle of binary search as git-bisect command.
* The initial set up is as follows:

git bisect start # initiates bisecting session

git bisect bad # marks current revision as bad

git bisect good revision # marks last known commit as good revision

* Upon running the above commands, git checks out a revision that is labeled as halfway between “good” and “bad” versions. This step can be run again by marking the commit as “good” or “bad” and the process continues until the commit which has a bug is found.

### 38. What are the functionalities of git reset --mixed and git merge --abort?

* git reset --mixed command is used for undoing changes of the working directory and the git index.
* git merge --abort command is used for stopping the merge process and returning back to the state before the merging occurred.

### 39. Can you tell the differences between git revert and git reset?

| **git revert** | **git reset** |
| --- | --- |
| This command is used for creating a new commit that undoes the changes of the previous commit. | This command is used for undoing the local changes done in the git repository |
| Using this command adds a new history to the project without modifying the existing history | This command operates on the commit history, git index, and the working directory. |

## DevOps

DevOps stands for Development and Operations. It is a software engineering practice that focuses on bringing together the development team and the operations team for the purpose of automating the project at every stage. This approach helps in easily automating the project service management in order to aid the objectives at the operational level and improve the understanding of the technological stack used in the production environment.

This way of practice is related to agile methodology and it mainly focuses on team communication, resource management, and teamwork. The main benefits of following this structure are the speed of development and resolving the issues at the production environment level, the stability of applications, and the innovation involved behind it.

Timeline

Description automatically generated with low confidenceDevOps

**DevOps Tools**

DevOps is a methodology aimed at increased productivity and quality of product development. The main tools used in this methodology are:

* Version Control System tools. Eg.: git.
* Continuous Integration tools. Eg.: Jenkins
* Continuous Testing tools. Eg.: Selenium
* Configuration Management and Deployment tools. Eg.:Puppet, Chef, Ansible
* Continuous Monitoring tool. Eg.: Nagios
* Containerization tools. Eg.: Docker

Chart, sunburst chart

Description automatically generatedDevOps Tools

Organizations that have adopted this methodology are reportedly accomplishing almost thousands of deployments in a single day thereby providing increased reliability, stability, and security with increased customer satisfaction.

### 1. Who is a DevOps engineer?

A [DevOps engineer](https://www.interviewbit.com/blog/devops-engineer/) is a person who works with both software developers and the IT staff to ensure smooth code releases. They are generally developers who develop an interest in the deployment and operations domain or the system admins who develop a passion for coding to move towards the development side.

In short, a DevOps engineer is someone who has an understanding of SDLC (Software Development Lifecycle) and of automation tools for developing CI/CD pipelines.

### 2. Why DevOps has become famous?

These days, the market window of products has reduced drastically. We see new products almost daily. This provides a myriad of choices to consumers but it comes at a cost of heavy competition in the market. Organizations cant afford to release big features after a gap. They tend to ship off small features as releases to the customers at regular intervals so that their products don't get lost in this sea of competition.

Customer satisfaction is now a motto to the organizations which has also become the goal of any product for its success. In order to achieve this, companies need to do the below things:

* Frequent feature deployments
* Reduce time between bug fixes
* Reduce failure rate of releases
* Quicker recovery time in case of release failures.
* In order to achieve the above points and thereby achieving seamless product delivery, DevOps culture acts as a very useful tool. Due to these advantages, multi-national companies like Amazon and Google have adopted the methodology which has resulted in their increased performance.

### 3. What is the use of SSH?

SSH stands for Secure Shell and is an administrative protocol that lets users have access and control the remote servers over the Internet to work using the command line.

SSH is a secured encrypted version of the previously known Telnet which was unencrypted and not secure. This ensured that the communication with the remote server occurs in an encrypted form.

SSH also has a mechanism for remote user authentication, input communication between the client and the host, and sending the output back to the client.

### 4. What is configuration management?

Configuration management (CM) is basically a practice of systematic handling of the changes in such a way that system does not lose its integrity over a period of time. This involves certain policies, techniques, procedures, and tools for evaluating change proposals, managing them, and tracking their progress along with maintaining appropriate documentation for the same.

CM helps in providing administrative and technical directions to the design and development of the appreciation.

The following diagram gives a brief idea about what CM is all about:

Diagram

Description automatically generatedDevOps Configuration Management

### 5. What is the importance of having configuration management in DevOps?

Configuration management (CM) helps the team in the automation of time-consuming and tedious tasks thereby enhancing the organization’s performance and agility.

It also helps in bringing consistency and improving the product development process by employing means of design streamlining, extensive documentation, control, and change implementation during various phases/releases of the project.

### 6. What does CAMS stand for in DevOps?

CAMS stands for Culture, Automation, Measurement, and Sharing. It represents the core deeds of DevOps.

### 7. What is Continuous Integration (CI)?

Continuous Integration (CI) is a software development practice that makes sure developers integrate their code into a shared repository as and when they are done working on the feature. Each integration is verified by means of an automated build process that allows teams to detect problems in their code at a very early stage rather than finding them after the deployment.

Diagram

Description automatically generatedContinuous Integration (CI)

Based on the above flow, we can have a brief overview of the CI process.

* Developers regularly check out code into their local workspaces and work on the features assigned to them.
* Once they are done working on it, the code is committed and pushed to the remote shared repository which is handled by making use of effective version control tools like git.
* The CI server keeps track of the changes done to the shared repository and it pulls the changes as soon as it detects them.
* The CI server then triggers the build of the code and runs unit and integration test cases if set up.
* The team is informed of the build results. In case of the build failure, the team has to work on fixing the issue as early as possible, and then the process repeats.

### 8. Why is Continuous Integration needed?

By incorporating Continuous Integration for both development and testing, it has been found that the software quality has improved and the time taken for delivering the features of the software has drastically reduced.

This also allows the development team to detect and fix errors at the initial stage as each and every commit to the shared repository is built automatically and run against the unit and integration test cases.

### 9. What is Continuous Testing (CT)?

Continuous Testing (CT) is that phase of DevOps which involves the process of running the automated test cases as part of an automated software delivery pipeline with the sole aim of getting immediate feedback regarding the quality and validation of business risks associated with the automated build of code developed by the developers.

Using this phase will help the team to test each build continuously (as soon as the code developed is pushed) thereby giving the dev teams a chance to get instant feedback on their work and ensuring that these problems don’t arrive in the later stages of SDLC cycle.

Doing this would drastically speed up the workflow followed by the developer to develop the project due to the lack of manual intervention steps to rebuild the project and run the automated test cases every time the changes are made.

### 10. What are the three important DevOps KPIs?

Few KPIs of DevOps are given below:

* Reduce the average time taken to recover from a failure.
* Increase Deployment frequency in which the deployment occurs.
* Reduced Percentage of failed deployments.

## Intermediate Interview Questions

### 11. Explain the different phases in DevOps methodology.

DevOps mainly has 6 phases and they are:

**Planning:**

This is the first phase of a DevOps lifecycle that involves a thorough understanding of the project to ultimately develop the best product. When done properly, this phase gives various inputs required for the development and operations phases. This phase also helps the organization to gain clarity regarding the project development and management process.

Tools like Google Apps, Asana, Microsoft teams, etc are used for this purpose.

**Development:**

The planning phase is followed by the Development phase where the project is built by developing system infrastructure, developing features by writing codes, and then defining test cases and the automation process. Developers store their codes in a code manager called remote repository which aids in team collaboration by allowing view, modification, and versioning of the code.

Tools like git, IDEs like the eclipse, IntelliJ, and technological stacks like Node, Java, etc are used.

**Continuous Integration (CI):**

This phase allows for automation of code validation, build, and testing. This ensures that the changes are made properly without development environment errors and also allows the identification of errors at an initial stage.

Tools like Jenkins, circleCI, etc are used here.

**Deployment:**

DevOps aids in the deployment automation process by making use of tools and scripts which has the final goal of automating the process by means of feature activation. Here, cloud services can be used as a force that assists in upgrade from finite infrastructure management to cost-optimized management with the potential to infinite resources.

Tools like Microsoft Azure, Amazon Web Services, Heroku, etc are used.

**Operations:**

This phase usually occurs throughout the lifecycle of the product/software due to the dynamic infrastructural changes. This provides the team with opportunities for increasing the availability, scalability, and effective transformation of the product.

Tools like Loggly, BlueJeans, Appdynamics, etc are used commonly in this phase.

**Monitoring:**

Monitoring is a permanent phase of DevOps methodology. This phase is used for monitoring and analyzing information to know the status of software applications.

Tools like Nagios, Splunk, etc are commonly used.

### 12. How is DevOps different than the Agile Methodology?

DevOps is a practice or a culture that allows the collaboration of the development team and the operations team to come together for successful product development. This involves making use of practices like continuous development, integration, testing, deployment, and monitoring of the SDLC cycle.

DevOps tries to reduce the gap between the developers and the operations team for the effective launch of the product.

Agile is nothing but a software development methodology that focuses on incremental, iterative, and rapid releases of software features by involving the customer by means of feedback. This methodology removes the gap between the requirement understanding of the clients and the developers.

Diagram

Description automatically generatedAgile Methodology

### 13. Differentiate between Continuous Deployment and Continuous Delivery?

The main difference between Continuous Deployment and Continuous Delivery are given below:

| **Continuous Deployment** | **Continuous Delivery** |
| --- | --- |
| The deployment to the production environment is fully automated and does not require manual/ human intervention. | In this process, some amount of manual intervention with the manager’s approval is needed for deployment to a production environment. |
| Here, the application is run by following the automated set of instructions, and no approvals are needed. | Here, the working of the application depends on the decision of the team. |

Diagram

Description automatically generated with medium confidenceContinuous Deployment and Continuous Delivery

### 14. What can you say about antipatterns of DevOps?

A pattern is something that is most commonly followed by large masses of entities. If a pattern is adopted by an organization just because it is being followed by others without gauging the requirements of the organization, then it becomes an anti-pattern. Similarly, there are multiple myths surrounding DevOps which can contribute to antipatterns, they are:

* DevOps is a process and not a culture.
* DevOps is nothing but Agile.
* There should be a separate DevOps group.
* DevOps solves every problem.
* DevOps equates to developers running a production environment.
* DevOps follows Development-driven management
* DevOps does not focus much on development.
* As we are a unique organization, we don’t follow the masses and hence we won’t implement DevOps.
* We don’t have the right set of people, hence we cant implement DevOps culture.

### 15. Can you tell me something about Memcached?

Memcached is an open-source and free in-memory object caching system that has high performance and is distributed and generic in nature. It is mainly used for speeding the dynamic web applications by reducing the database load.

**Memcached can be used in the following cases:**

* Profile caching in social networking domains like Facebook.
* Web page caching in the content aggregation domain.
* Profile tracking in Ad targeting domain.
* Session caching in e-commerce, gaming, and entertainment domain.
* Database query optimization and scaling in the Location-based services domain.

**Benefits of Memcached:**

* Using Memcached speeds up the application processes by reducing the hits to a database and reducing the I/O access.
* It helps in determining what steps are more frequently followed and helps in deciding what to cache.

**Some of the drawbacks of using Memcached are:**

* In case of failure, the data is lost as it is neither a persistent data store nor a database.
* It is not an application-specific cache.
* Large objects cannot be cached.

### 16. What are the various branching strategies used in the version control system?

Branching is a very important concept in version control systems like git which facilitates team collaboration. Some of the most commonly used branching types are:

**Feature branching**

* This branching type ensures that a particular feature of a project is maintained in a branch.
* Once the feature is fully validated, the branch is then merged into the main branch.

**Task branching**

* Here, each task is maintained in its own branch with the task key being the branch name.
* Naming the branch name as a task name makes it easy to identify what task is getting covered in what branch.

**Release branching**

* This type of branching is done once a set of features meant for a release are completed, they can be cloned into a branch called the release branch. Any further features will not be added to this branch.
* Only bug fixes, documentation, and release-related activities are done in a release branch.
* Once the things are ready, the releases get merged into the main branch and are tagged with the release version number.
* These changes also need to be pushed into the develop branch which would have progressed with new feature development.

The branching strategies followed would vary from company to company based on their requirements and strategies.

### 17. Can you list down certain KPIs which are used for gauging the success of DevOps?

KPIs stands for Key Performance Indicators. Some of the popular KPIs used for gauging the success of DevOps are:

* Application usage, performance, and traffic
* Automated Test Case Pass Percentage.
* Application Availability
* Change volume requests
* Customer tickets
* Successful deployment frequency and time
* Error/Failure rates
* Failed deployments
* Meantime to detection (MTTD)
* Meantime to recovery (MTTR)

### 18. What is CBD in DevOps?

CBD stands for Component-Based Development. It is a unique way for approaching product development. Here, developers keep looking for existing well-defined, tested, and verified components of code and relieve the developer of developing from scratch.

### 19. What is Resilience Testing?

Resilience Testing is a software process that tests the application for its behavior under uncontrolled and chaotic scenarios. It also ensures that the data and functionality are not lost after encountering a failure.

### 20. Can you differentiate between continuous testing and automation testing?

The difference between continuous testing and automation testing is given below:

| **Continuous Testing** | **Automation Testing** |
| --- | --- |
| This is the process of executing all the automated test cases and is done as part of the delivery process. | This is a process that replaces manual testing by helping the developers create test cases that can be run multiple times without manual intervention. |
| This process focuses on the business risks associated with releasing software as early as possible. | This process helps the developer to know whether the features they have developed are bug-free or not by having set of pass/fail points as a reference. |

### 21. Can you say something about the DevOps pipeline?

A pipeline, in general, is a set of automated tasks/processes defined and followed by the software engineering team. DevOps pipeline is a pipeline which allows the DevOps engineers and the software developers to efficiently and reliably compile, build and deploy the software code to the production environments in a hassle free manner.

Following image shows an example of an effective DevOps pipeline for deployment.

Timeline

Description automatically generated

The flow is as follows:

* Developer works on completing a functionality.
* Developer deploys his code to the test environment.
* Testers work on validating the feature. Business team can intervene and provide feedback too.
* Developers work on the test and business feedback in continuous collaboration manner.
* The code is then released to the production and validated again.

### 22. Tell me something about Ansible work in DevOps

It is a DevOps open-source automation tool which helps in modernizing the development and deployment process of applications in faster manner. It has gained popularity due to simplicity in understanding, using, and adopting it which largely helped people across the globe to work in a collaborative manner.

| **Ansible** | **Developers** | **Operations** | **QA** | **Business/Clients** |
| --- | --- | --- | --- | --- |
| **Challenges** | Developers tend to focus a lot of time on tooling rather than delivering the results. | Operations team would require uniform technology that can be used by different skillset groups easily. | Quality Assurance team would require to keep track of what has been changed in the feature and when it has been changed. | Clients worry about getting the products to the market as soon as possible. |
| **Need** | Developers need to respond to new features/bugs and scale the efforts based on the demand. | Operation team need a central governing tool to monitor different systems and its workloads. | Quality Assurance team need to focus on reducing human error risk as much as possible for bug-free product. | Clients need to create a competitive advantage for their products in the market. |
| **How does Ansible help?** | Helps developers to discover bugs at an earlier phase, and assists them to perform faster deployments in a reliable manner. | Helps the Operations team to reduce their efforts on shadowing IT people and reduce the times taken for deployment. Also, Ansible assists them to perform automated patching. | Helps QA team to establish automated test cases irrespective of the environments for achieving more reliable and accurate results. Helps to define identical security baselines and helps them reduce the burden of following traditional documentation. | Helps the Business team to ensure the IT team is on the right track. Also helps them to optimize the time taken for project innovation and strategising. Helps teams to collaborate in an effective manner. |

### 23. How does Ansible work?

Ansible has two types of servers categorized as:

* Controlling machines
* Nodes

For this to work, Ansible is installed on controlling machine using which the nodes are managed by means of using SSH. The location of the nodes would be specified and configured in the inventories of the controlling machine.

Ansible does not require any installations on the remote node servers due its nature of being agentless. Hence, no background process needs to be executed while managing any remote nodes.

Ansible can manage lots of nodes from a single controlling system my making use of Ansible Playbooks through SSH connection. Playbooks are of the YAML format and are capable to perform multiple tasks.

### 24. How does AWS contribute to DevOps?

AWS stands for Amazon Web Services and it is a well known cloud provider. AWS helps DevOps by providing the below benefits:

* **Flexible Resources:** AWS provides ready-to-use flexible resources for usage.
* **Scaling:** Thousands of machines can be deployed on AWS by making use of unlimited storage and computation power.
* **Automation:** Lots of tasks can be automated by using various services provided by AWS.
* **Security:** AWS is secure and using its various security options provided under the hood of Identity and Access Management (IAM), the application deployments and builds can be secured.

### 25. What can be a preparatory approach for developing a project using the DevOps methodology?

The project can be developed by following the below stages by making use of DevOps:

* **Stage 1: Plan:** Plan and come up with a roadmap for implementation by performing a thorough assessment of the already existing processes to identify the areas of improvement and the blindspots.
* **Stage 2: PoC:** Come up with a proof of concept (PoC) just to get an idea regarding the complexities involved. Once the PoC is approved, the actual implementation work of the project would start.
* **Stage 3: Follow DevOps:** Once the project is ready for implementation, actual DevOps culture could be followed by making use of its phases like version control, continuous integration, continuous testing, continuous deployment, continuous delivery, and continuous monitoring.

## DevOps Interview Questions For Experienced

### 26. Can you explain the “Shift left to reduce failure” concept in DevOps?

In order to understand what this means, we first need to know how the traditional SDLC cycle works. In the traditional cycle, there are 2 main sides -

* The left side of the cycle consists of the planning, design, and development phase
* The right side of the cycle includes stress testing, production staging, and user acceptance.

In DevOps, shifting left simply means taking up as many tasks that usually take place at the end of the application development process as possible into the earlier stages of application development. From the below graph, we can see that if the shift left operations are followed, the chances of errors faced during the later stages of application development would greatly reduce as it would have been identified and solved in the earlier stages itself.

A picture containing timeline

Description automatically generatedShift Left To Reduce Failure

The most popular ways of accomplishing shift left in DevOps is to:

* Work side by side with the development team while creating the deployment and test case automation. This is the first and the obvious step in achieving shift left. This is done because of the well-known fact that the failures that get notices in the production environment are not seen earlier quite often. These failures can be linked directly to:
  + Different deployment procedures used by the development team while developing their features.
  + Production deployment procedures sometimes tend to be way different than the development procedure. There can be differences in tooling and sometimes the process might also be manual.
* Both the dev team and the operations teams are expected to take ownership to develop and maintain standard procedures for deployment by making use of the cloud and the pattern capabilities. This aids in giving the confidence that the production deployments would be successful.
* Usage of pattern capabilities to avoid configurational level inconsistencies in the different environments being used. This would require the dev team and the operation team to come together and work in developing a standard process that guides developers to test their application in the development environment in the same way as they test in the production environment.

### 27. Do you know about post mortem meetings in DevOps?

Post Mortem meetings are those that are arranged to discuss if certain things go wrong while implementing the DevOps methodology. When this meeting is conducted, it is expected that the team has to arrive at steps that need to be taken in order to avoid the failure(s) in the future.

### 28. What is the concept behind sudo in Linux OS?

Sudo stands for ‘superuser do’ where the superuser is the root user of Linux. It is a program for Linux/Unix-based systems that gives provision to allow the users with superuser roles to use certain system commands at their root level.

### 29. Can you explain the architecture of Jenkins?

Jenkins follows the master-slave architecture. The master pulls the latest code from the GitHub repository whenever there is a commitment made to the code. The master requests slaves to perform operations like build, test and run and produce test case reports. This workload is distributed to all the slaves in a uniform manner.

Jenkins also uses multiple slaves because there might be chances that require different test case suites to be run for different environments once the code commits are done.

Diagram

Description automatically generatedJenkins Architecture

### 30. Can you explain the “infrastructure as code” (IaC) concept?

As the name indicates, IaC mainly relies on perceiving infrastructure in the same way as any code which is why it is commonly referred to as “programmable infrastructure”. It simply provides means to define and manage the IT infrastructure by using configuration files.

This concept came into prominence because of the limitations associated with the traditional way of managing the infrastructure. Traditionally, the infrastructure was managed manually and the dedicated people had to set up the servers physically. Only after this step was done, the application would have been deployed. Manual configuration and setup were constantly prone to human errors and inconsistencies.

This also involved increased cost in hiring and managing multiple people ranging from network engineers to hardware technicians to manage the infrastructural tasks. The major problem with the traditional approach was decreased scalability and application availability which impacted the speed of request processing. Manual configurations were also time-consuming and in case the application had a sudden spike in user usage, the administrators would desperately work on keeping the system available for a large load. This would impact the application availability.

IaC solved all the above problems. IaC can be implemented in 2 approaches:

* Imperative approach: This approach “gives orders” and defines a sequence of instructions that can help the system in reaching the final output.
* Declarative approach: This approach “declares” the desired outcome first based on which the infrastructure is built to reach the final result.

### 31. What is ‘Pair Programming’?

Pair programming is an engineering practice where two programmers work on the same system, same design, and same code. They follow the rules of “Extreme Programming”. Here, one programmer is termed as “driver” while the other acts as “observer” which continuously monitors the project progress to identify any further problems.

### 32. What is Blue/Green Deployment Pattern?

A blue-green pattern is a type of continuous deployment, application release pattern which focuses on gradually transferring the user traffic from a previously working version of the software or service to an almost identical new release - both versions running on production.

The blue environment would indicate the old version of the application whereas the green environment would be the new version.

The production traffic would be moved gradually from blue to green environment and once it is fully transferred, the blue environment is kept on hold just in case of rollback necessity.

Diagram

Description automatically generated

In this pattern, the team has to ensure two identical prod environments but only one of them would be LIVE at a given point of time. Since the blue environment is more steady, the LIVE one is usually the blue environment.

### 33. What is Dogpile effect? How can it be prevented?

It is also referred to as cache stampede which can occur when huge parallel computing systems employing caching strategies are subjected to very high load. It is referred to as that event that occurs when the cache expires (or invalidated) and multiple requests are hit to the website at the same time. The most common way of preventing dogpiling is by implementing semaphore locks in the cache. When the cache expires in this system, the first process to acquire the lock would generate the new value to the cache.

### 34. What are the steps to be undertaken to configure git repository so that it runs the code sanity checking tooks before any commits? How do you prevent it from happening again if the sanity testing fails?

Sanity testing, also known as smoke testing, is a process used to determine if it’s reasonable to proceed to test.  
Git repository provides a hook called pre-commit which gets triggered right before a commit happens. A simple script by making use of this hook can be written to achieve the smoke test.

The script can be used to run other tools like linters and perform sanity checks on the changes that would be committed into the repository.

The following snippet is an example of one such script:

#!/bin/sh

files=$(git diff –cached –name-only –diff-filter=ACM | grep ‘.py$’)

if [ -z files ]; then

exit 0

fi

unfmtd=$(pyfmt -l $files)

if [ -z unfmtd ]; then

exit 0

fi

echo “Some .py files are not properly fmt’d”

exit 1

The above script checks if any .py files which are to be committed are properly formatted by making use of the python formatting tool pyfmt. If the files are not properly formatted, then the script prevents the changes to be committed to the repository by exiting with status 1.

### 35. How can you ensure a script runs every time repository gets new commits through git push?

There are three means of setting up a script on the destination repository to get executed depending on when the script has to be triggered exactly. These means are called hooks and they are of three types:

* **Pre-receive hook:** This hook is invoked before the references are updated when commits are being pushed. This hook is useful in ensuring the scripts related to enforcing development policies are run.
* **Update hook:** This hook triggers the script to run before any updates are actually made. This hook is called once for every commit which has been pushed to the repository.
* **Post-receive hook:** This hook helps trigger the script after the updates or changes have been accepted by the destination repository. This hook is ideal for configuring deployment scripts, any continuous integration-based scripts or email notifications process to the team, etc.

## Conclusion

DevOps is a culture-shifting practice that has and is continuing to help lots of businesses and organizations in a tremendous manner. It helps in bridging the gap between the conflict of goals and priorities of the developers (constant need for change) and the operations (constant resistance to change) team by creating a smooth path for Continuous Development and Continuous Integration. Being a DevOps engineer has huge benefits due to the ever-increasing demand for DevOps practice.