## 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 software. Jenkins can be installed through native system packages, Docker, or even run standalone by any machine with a Java Runtime Environment (JRE) installed.

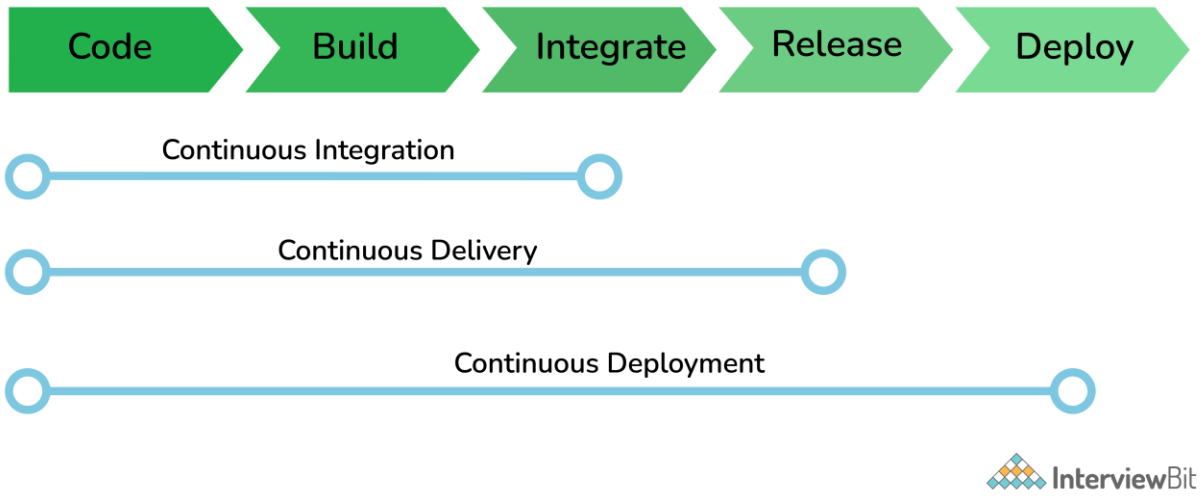
## 2. Tell me something about Continuous Integration, Continuous Delivery, and Continuous Deployment?

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

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

**Continuous Deployment:** A Software Development practice where the continuously integrated (CI) changes are deployed automatically into the target environment after 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 software-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 -

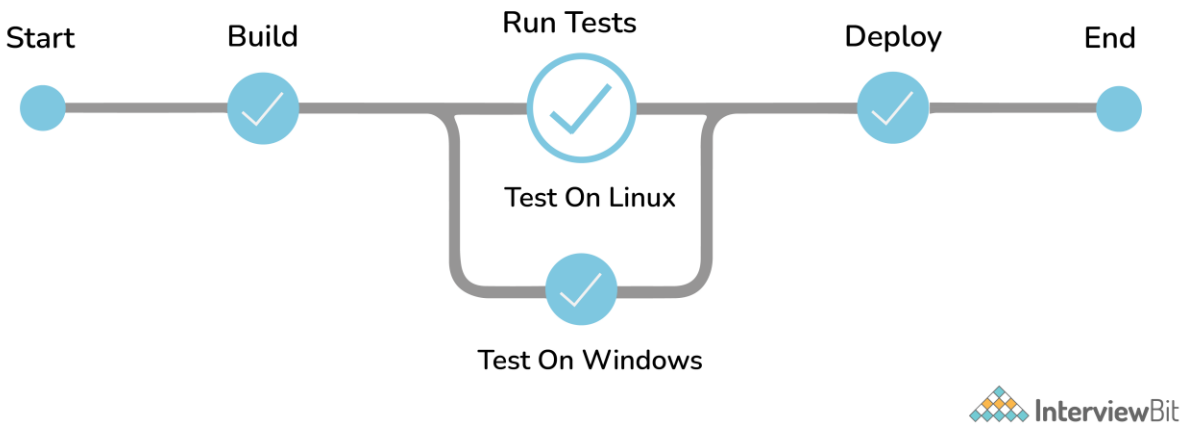
1. Native System Package Manager like - apt (Linux), brew (Mac), etc.
2. Docker (popular docker images for Jenkins is available for different platforms like Unix/Mac/Windows in the docker registry)
3. Kubernetes (available as a helm chart and can be installed on our Kubernetes clusters)
4. Standalone (on any machine with a Java Runtime Environment installed)

## 5. What is a Jenkins job?

A Job/Project is the fundamental unit of a logical work (like a software 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?



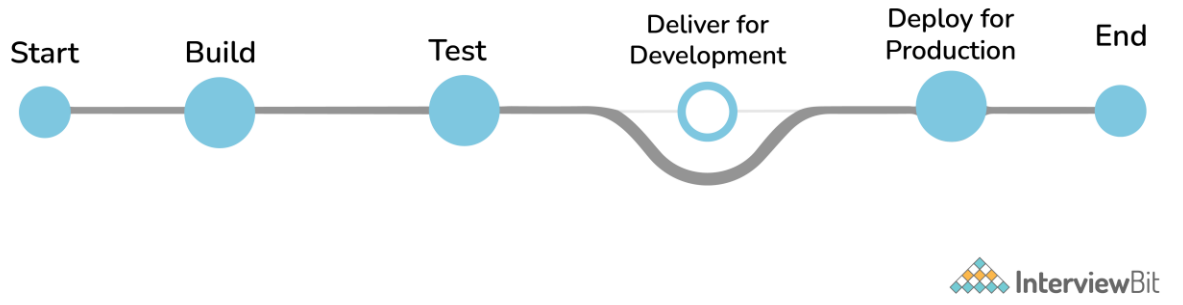

Jenkins Pipeline

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.

### 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.

## Intermediate Questions

### 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.

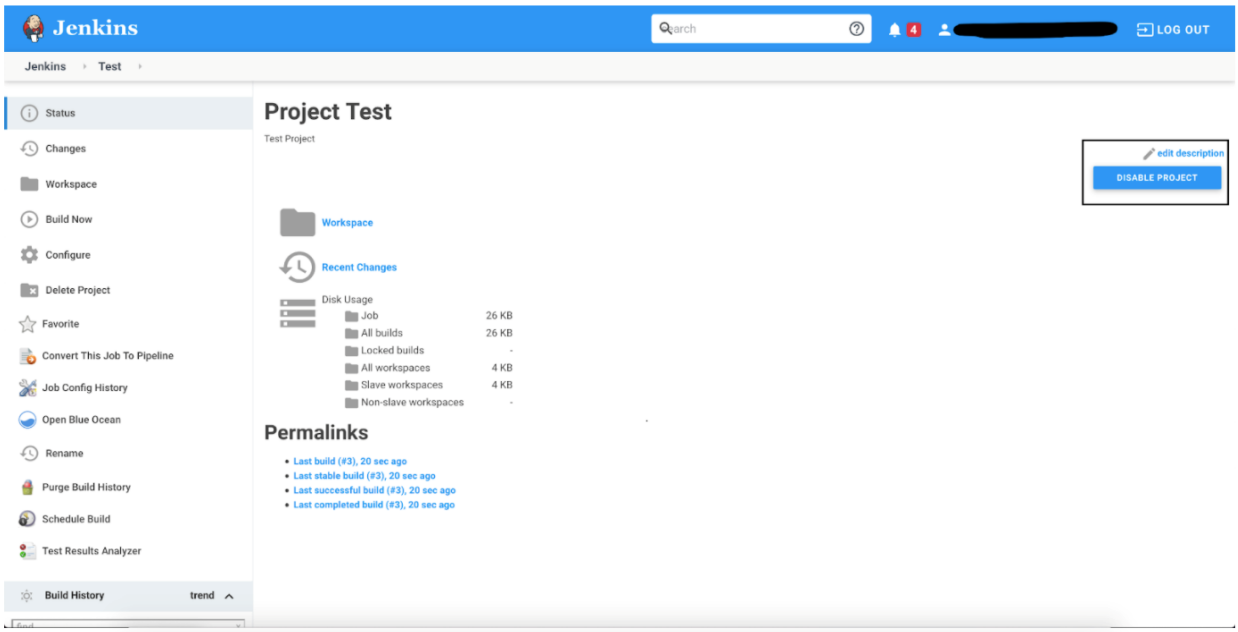
## 20. What is the Blue Ocean?

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.

## Conclusion

Though these are not the complete possibilities of Jenkins, we tried to cover some of the commonly asked interview questions on core Jenkins. We also need to understand that the Jenkins Update Center is enriched with thousands of useful plugins that enhance the supported functionalities of Jenkins.

Before appearing for an interview, make sure to install a Jenkins Server on any of the supported platforms - either locally or on the cloud, install the most common plugins (suggested by Jenkins itself & other commonly used plugins). Try creating & building a normal freestyle project with Git or any other SCM integration plugin and try to execute some code from the connected Git Repository.

Also, try creating a pipeline project with JenkinsFile and a global shared Jenkins library and build the job successfully. This will help us learn how Jenkins actually works with some hands-on issues.