**1. What is Jenkins?**

**Answer:** 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.

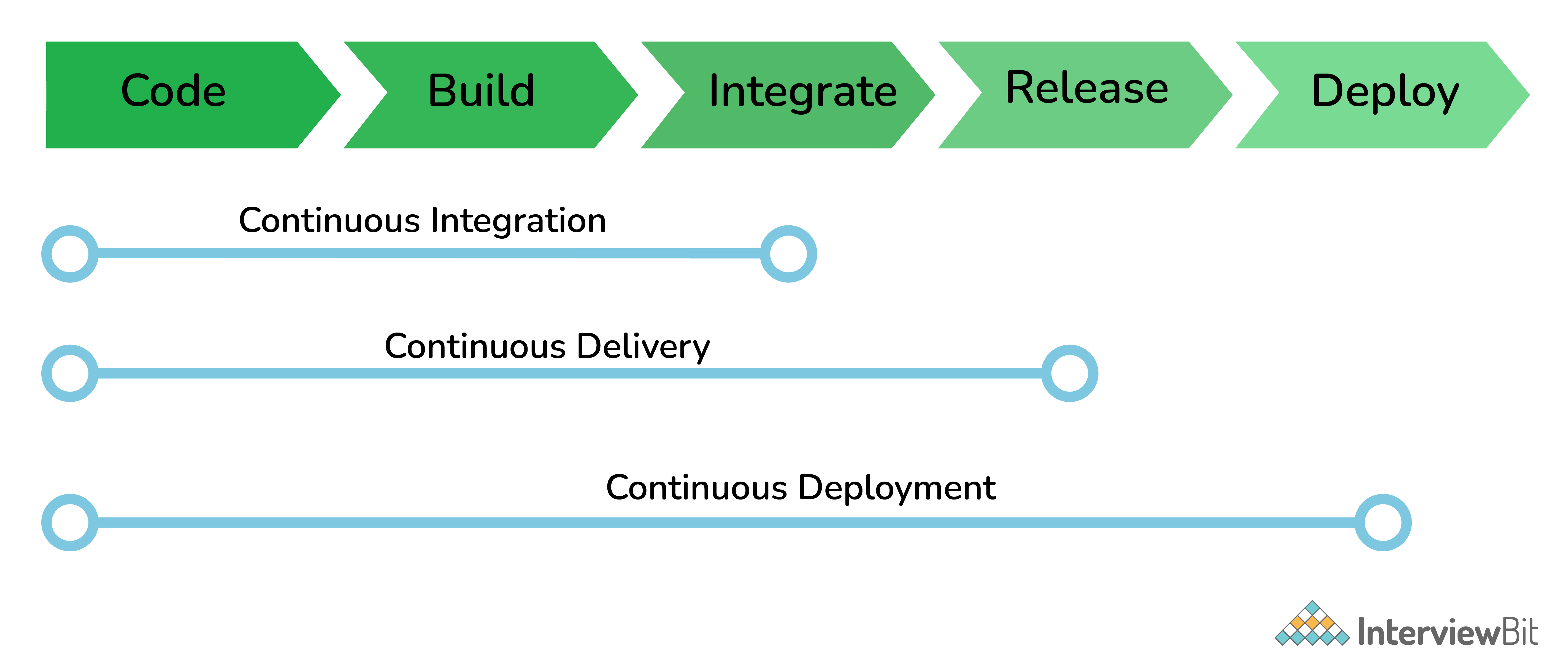
**2. Tell me something about Continuous Integration, Continuous Delivery, and Continuous Deployment?**

**Answer:** 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 –



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

**Answer:** 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?**

**Answer:** Jenkins can be installed using -

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

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

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

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

For more detailed installation instructions refer official documentation

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

**Answer:** 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 are the types of Jenkins pipelines?**

**Answer:** 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.

**7. How can the parameters be defined in Jenkins?**

**Answer:** In Jenkins, a build can take many input parameters to execute.

To define parameters for the job, select the “this project is parameterized” box.

The drop down “Add Parameter” is enabled with the parameter types list. Any number of parameters may be added in the list.

There are several parameter types provided in the list.

**8. How do you store credentials in Jenkins securely?**

**Answer:** 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.

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

**Answer:** 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.

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

**Answer:** 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.

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

**Answer:** 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.

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

**Answer:** 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.

**13. What are the credential types supported by Jenkins?**

**Answer:** 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.

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

**Answer:** 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.

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

**Answer:** 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?**

**Answer:** 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?**

**Answer:** 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?**

**Answer:** 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?**

**Answer:** 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 Artifact Archival & how to do it in Pipelines?**

**Answer:** 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/\*\*/\*'

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

**Answer:** 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

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

**Answer:** 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.

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

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

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

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

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

**Answer:** 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.

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

**Answer:** 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.

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

**Answer:** 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')\_

**28. How can you create a backup and copy files in Jenkins?**

**Answer:** Jenkins stores all the settings, builds scripts, and logs in the home directory.

Then, if you want to create a backup of this Jenkins set up all you have to do is copy this directory.

The job directory may also be copied to clone a job or rename the directory.

**29. How can you deploy a custom build of a core plugin?**

**Answer:** If you wish to deploy a custom build of a core plugin, you follow the following steps:

Stop Jenkins

Then copy the custom HPI to $Jenkins\_Home/plugins

After that, delete the previously expanded plugin directory

Next, make an empty file called <plugin>.hpi.pinned

Finally, start Jenkins

**30. What could be the steps to move or copy Jenkins from one server to another?**

**Answer:** There are multiple ways to move or copy Jenkins from one server to another:

You may move a job from one Jenkins installation to another just by copying the corresponding job directory.

You may make a copy of an already existing job by making a clone of the job directory with an uncommon name.

You may also just rename a current job by renaming a directory.

**31. Explain the process in which Jenkins works?**

**Answer:** Here’s the process in which Jenkins works:

Jenkins checks changes in repositories regularly, and developers must secure their code regularly.

Once the changes are defined, Jenkins detects them and uses them to prepare a new build.

After that, Jenkins will transverse through various stages in its usual pipeline. As one stage completes, the process will move further on to the next stage.

If a stage fails, the Jenkins build will stop there, and the software will email the team using it. When completed successfully, the code implements itself in the proper server so that testing begins.

After the successful testing phase, Jenkins shares the results with the team using it.

**32. What is Jenkins file?**

**Answer:** Jenkins file is a text file that has a definition of a Jenkins pipeline and is checked into the source control repository. It enables code review and iteration on the pipeline. It also permits an audit trail for the pipeline.

**33. What is the process to integrate Git with Jenkins?**

**Answer:** To integrate Git with Jenkins, you can follow the following steps:

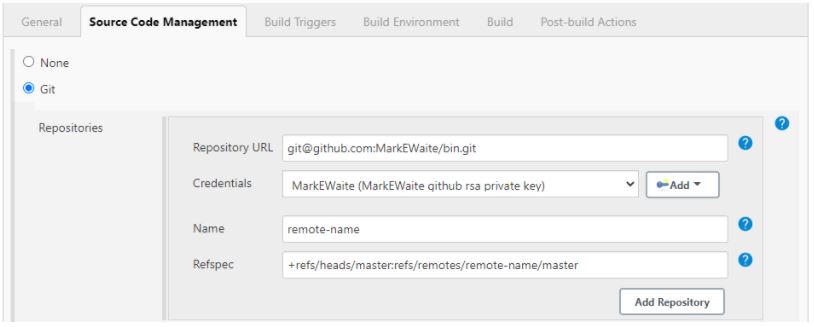
First, create a new Jenkins job and open the Jenkins dashboard.

Now, enter the desired project name and select the job type.

Click on OK.

Then enter the project information.

After that, visit the 'Source Code Management' tab.



If the Git plugin is pre-installed in Jenkins, there will be 'Git'.

If it is not installed, you must reinstall the plugins (GitHub plugin, GitHub Branch Source plugin, GitHub API plugin, Git client plugin, etc.).

After we install the plugins, restart Jenkins.

To check if Git is installed, you can go to Command Prompt and type Git, and you would see various options like usage, version, help, etc.

**34. What is the process to configure Third-party tools in Jenkins?**

**Answer:** The process to configure Third-party tools in Jenkins can be seen in four significant steps:

Install the third-party software

Then install a Jenkins plugin supporting the third-party tool

Now, configure the tool from the Manage Jenkins section

Finally, your plugin is ready to be used.

**35. What are some of the default environmental variables in Jenkins?**

**Answer:** Some of the Jenkins environmental variables are:

$JOB\_NAME - The name that you give your job when it is first set up.

$NODE\_NAME - This is the name of the node on which the current build is running.

$WORKSPACE - Refers to the path of the workspace

$BUILD\_URL - Indicates the URL where the results of the builds can be found.

$JENKINS\_URL - This is set to the URL of the Jenkins master that is responsible for running the build.