## ****Basic Interview Questions****

### ****Q1. What is the difference between Jenkins and Bamboo?****

|  |  |  |
| --- | --- | --- |
| ****Parameters**** | ****Jenkins**** | ****Bamboo**** |
| ****Open Source**** | Jenkin is open-source | Bamboo is not open source |
| ****Pricing Logic**** | Jenkin is completely free | It charges for the number of build agents required |
| ****Operating System**** | Windows, Ubuntu, Red Hat, Mac OS | Windows, Linux, Solaris |
| ****Browsers**** | Chrome, Firefox, Internet Explorer | Firefox, Chrome, Safari, Edge |
| ****Plugin Support**** | Yes, It supports a lot of plugins | It does not support many plugins as compared to Jenkins |
| ****Support**** | Being open-source, it has a lot of support from communities | It has less support as compared to Jenkins |

### ****Q2. What is Jenkins?****

Jenkins is an open-source automation tool written in Java with plugins built for Continuous Integration purposes. Jenkins is used to build and test your software projects continuously making it easier for developers to integrate changes to the project, and making it easier for users to obtain a fresh build. It also allows you to continuously deliver your software by integrating with a large number of testing and deployment technologies.

### ****Q3. Define the process of Jenkins.****

First, a developer commits the code to the source code repository. Meanwhile, the Jenkins server checks the repository at regular intervals for changes.

Soon after a commit occurs, the Jenkins server detects the changes that have occurred in the source code repository. Jenkins will pull those changes and will start preparing a new build.

If the build fails, then the concerned team will be notified.

If the build is successful, then Jenkins deploys the build in the test server.

After testing, Jenkins generates feedback and then notifies the developers about the build and test results.

It will continue to check the source code repository for changes made in the source code and the whole process keeps on repeating.

### **IMG_256**

### ****Q4. What are the benefits of using Jenkins?****

I will suggest you include the following benefits of Jenkins if you can recall any other benefit apart from the below-mentioned points you can include that as well.

* At the integration stage, you can cache build failures.
* For each change in the source code, you generate an automatic build report notification.
* To notify developers about build report success or failure, Jenkins integrates with the LDAP mail server.
* Achieves continuous integration agile development and test driven development.
* With simple steps, you can automate the maven release project.
* Easy tracking of bugs at an early stage in a development environment than production.

### ****Q5.**** ****What are the pre-requisites for using Jenkins?****

The answer to this is pretty straightforward. To use Jenkins you require:

* A source code repository which is accessible, for instance, a Git repository.
* A working build script, e.g., a Maven script, checked into the repository.

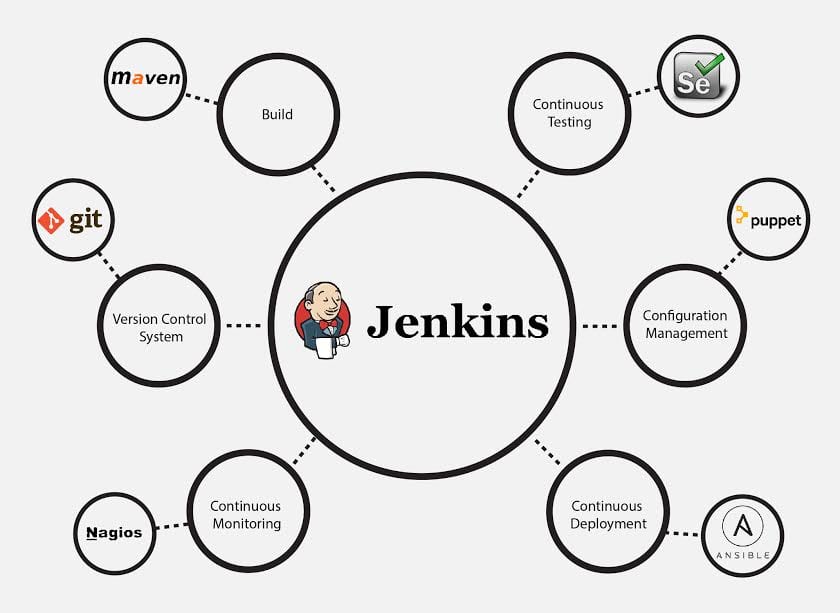
### ****Q6. What is the relation between Hudson and Jenkins?****

You can just say Hudson was the earlier name and version of current Jenkins. After some issues, they renamed the project from Hudson to Jenkins.

### ****Q7.**** ****Mention some of the useful plugins in Jenkins****

Below I have mentioned some important Plugins:

* Maven 2 project
* Git
* Amazon EC2
* HTML publisher
* Copy artifact
* Join
* Green Balls



These Plugins I feel are the most useful plugins, if you want to include any other Plugin that is not mentioned above, you can add that as well, but make sure you first mention the above-stated plugins and then add your own.

### ****Q8. How do you install Jenkins?****

To install Jenkins, you just need to follow these five steps:

****Install Java Version 8**** – Jenkins is a Java based application, hence Java is a must.

****Install Apache Tomcat Version 9**** – Tomcat is essential to deploy Jenkins war file.

****Download Jenkins war File**** – This war is must to install Jenkins.

****Deploy Jenkins war File**** – You deploy Jenkins war file using Tomcat to run Jenkins.

****Install Suggested Plugins**** – Install a list of plugins suggested by Jenkins.

Once the installation is complete, you will be able to see the Jenkins dashboard.

### ****Q9. What are the two components that you can integrate Jenkins with?****

According to me, the integration of Jenkins is possible with the following:

* Version Control system like GIT, SVN.
* Build tools like Apache Maven.

If you have anything else in your mind then mention that as well but make sure you include the above two components in your answer.

### ****Q10. What is Maven? What is the benefit of integrating Maven with Jenkins?****

[Maven](https://maven.apache.org/" \t "https://www.edureka.co/blog/interview-questions/jenkins-interview-questions/_blank) is a build management tool. It uses a simple pom.xml to configure all the dependencies needed to build, test and run the code. Maven manages the full lifecycle of a test project. Once integrated with Jenkins, the maven Webdriver will build the project and execute all tests efficiently.

### ****Q11.**** ****Mention what are the commands you can use to start Jenkins manually.****

For this answer I will suggest you go with the below-mentioned flow:  
To start Jenkins manually open Console/Command line, then go to your Jenkins installation directory. Over there you can use the below commands:

Start Jenkins: ****jenkins.exe start****  
Stop Jenkins: ****jenkins.exe stop****  
Restart Jenkins: ****jenkins.exe restart****

### ****Q12.**** ****Which SCM tools Jenkins supports?****

Here are some of the Source Code Management tools supported by Jenkins:

* AccuRev
* CVS
* Subversion
* Git
* Mercurial
* Perforce
* Clearcase
* RTC

### ****Q13. How will you define Post in Jenkins?****

****Post**** is a section that contains several additional steps that might execute after the completion of the pipeline. The execution of all the steps within the condition block depends upon the completion status of the pipeline. The condition block includes the following conditions – ****changed success, always, failure, unstable and aborted.****

### ****Q14. What are Parameters in Jenkins?****

Parameters are supported by Agent section and they are used to support various use-cases pipelines. Parameters are defined at the top-level of the pipeline or inside an individual stage directive.

### ****Q15. What is Groovy?****

Groovy from Apache is a language designed for the Java platform. It is the native scripting language for Jenkins. Groovy-based plugins enhance Jenkins with great interfaces and build reports that are of dynamic and consistent nature.

**Q16. Which commands can be used to begin Jenkins?**

To start Jenkins, you typically use the following commands, depending on how you have installed Jenkins:

1. Command Line (War File):

   If you are running Jenkins as a standalone application using the .war (Web ARchive) file, open your terminal or command prompt and navigate to the directory where the .war file is located. Then, use the following command:

      “`

   java -jar jenkins.war

   “`

   Replace `jenkins.war` with the actual name of the .war file if it’s different.

1. Docker:

If you have installed Jenkins using Docker, you can start the Jenkins container with the following command:

   “`

   docker run -p 8080:8080 -p 50000:50000 jenkins/jenkins

   “`

   This command will pull the Jenkins Docker image if not already present and start the Jenkins container.

1. Docker Compose:

   If you are using Docker Compose to manage Jenkins, navigate to the directory where your `docker-compose.yml` file is located and use the following command:

   “`

   docker-compose up

   “`

   This command will start Jenkins and any other services defined in your `docker-compose.yml` file.

After running the appropriate command, Jenkins should start, and you can access it by visiting http://localhost:8080 (or the specified port) in your web browser. From there, you will need to follow the initial setup wizard to complete the Jenkins installation and configuration.

**Q17. What are the common use cases Jenkins is used for?**

Jenkins is a popular and versatile automation tool that is widely used in software development and DevOps practices. Some common use cases for Jenkins include:

1. Continuous Integration (CI): Jenkins is primarily known for its role in CI. It automates the process of integrating code changes from multiple developers into a shared repository. It can build, test, and validate code changes automatically, ensuring that new code doesn’t break the existing codebase and that the application remains in a working state at all times.
2. Continuous Delivery and Continuous Deployment (CD): Jenkins facilitates continuous delivery and deployment pipelines, automating the release process. It can automatically deploy application updates to different environments (e.g., staging, production) once they pass the required tests and checks.
3. Automated Testing: Jenkins can execute various types of automated tests, including unit tests, integration tests, and end-to-end tests. It helps ensure code quality by running these tests as part of the CI/CD pipeline.
4. Scheduled Jobs and Cron Jobs: Jenkins can run scheduled jobs or tasks at specific times or on a regular basis. This feature is useful for performing tasks such as data backups, cleanups, or report generation.
5. Artifact Management: Jenkins can manage artifacts generated during the build process, such as compiled binaries, libraries, and documentation. It can store these artifacts and make them easily accessible to developers and other teams.
6. Infrastructure Automation: Jenkins can be used to automate infrastructure-related tasks, such as provisioning virtual machines, configuring servers, or deploying containers.
7. Deployment Orchestration: Jenkins can orchestrate complex deployment processes involving multiple services and applications, ensuring coordinated and consistent releases.
8. Integration with Version Control Systems: Jenkins integrates well with various version control systems like Git, Subversion, and Mercurial. It can automatically trigger builds when code changes are pushed to the repository.
9. Monitoring and Reporting: Jenkins provides insights into build and deployment statistics, allowing teams to monitor the health of their applications and track performance over time.
10. Cross-platform and Cross-language Support: Jenkins supports multiple platforms, languages, and tools, making it adaptable to various development environments.

Overall, Jenkins is an essential tool in the DevOps ecosystem, streamlining development and deployment processes, increasing productivity, and enabling faster and more reliable software delivery. Its flexibility and extensive plugin ecosystem make it suitable for a wide range of use cases in different industries and project types.

**Q18. What is a Jenkins job?**

In Jenkins, a job is a task or a set of tasks that Jenkins will execute as part of the build and automation process. A job in Jenkins represents a specific unit of work, such as building a software project, running tests, deploying applications, or performing any other automated task.

When you create a job in Jenkins, you define the steps or actions that Jenkins needs to perform. These steps can include actions like checking out source code from a version control repository, compiling code, running tests, generating reports, and deploying applications to various environments.

Each job is typically configured with various parameters, settings, and triggers. For example:

1. Job Name: A unique name given to the Jenkins job, allowing you to identify and manage it easily.
2. Source Code Management: The version control system and repository URL from which Jenkins will fetch the source code for building.
3. Build Triggers: Specifies when the job should be triggered, such as on code commits, at a specific time, or when manually triggered.
4. Build Steps: The series of actions that Jenkins will perform during the job execution, such as running build scripts or invoking build tools.
5. Post-build Actions: Actions that are performed after the build is complete, like sending notifications, archiving artifacts, or triggering downstream jobs.

Jenkins jobs can be organized into views and can be configured to run on different build agents (machines) to distribute the build workload efficiently. Once a job is created, Jenkins can execute it automatically based on the defined triggers or manually by users with the appropriate permissions.

Jobs are at the core of Jenkins’ automation capabilities. By creating and configuring jobs, users can define the desired automated processes, ensuring tasks like building, testing, and deploying software are performed consistently and efficiently.

**Q19. How do you store credentials in Jenkins securely?**

Storing credentials securely in Jenkins is crucial to protect sensitive information, such as passwords, API keys, or SSH private keys, from unauthorized access. Jenkins provides several mechanisms to manage and store credentials securely:

1. ****Credentials Plugin:**** Jenkins has a built-in Credentials plugin that allows you to manage and store credentials securely. It provides a user interface to add, update, and delete credentials, which can be used by Jenkins jobs.

   To use the Credentials plugin, go to “Manage Jenkins” > “Manage Credentials” to add your credentials. You can then reference these credentials in your Jenkins jobs, pipeline scripts, or other configurations.

1. ****Secret Text and File:**** In Jenkins, you can use the “Secret Text” and “Secret File” options to store sensitive data securely. These options allow you to specify the secret directly in Jenkins or provide a file containing the secret, respectively. The stored secrets are encrypted and can be referenced in your Jenkins jobs or pipeline scripts.
2. ****Jenkins Credentials Binding Plugin****: This plugin enables you to inject credentials directly into your Jenkins job’s environment. It’s especially useful when you need to pass sensitive data securely to shell scripts or build tools running within your job.
3. ****Environment Variables:**** Jenkins allows you to define environment variables containing sensitive data, such as passwords or API keys. When using environment variables, make sure to mask them in Jenkins job console output to prevent accidental exposure.
4. ****SSH Credentials:**** If you need to use SSH keys to access external servers or repositories, Jenkins provides a dedicated credential type for SSH private keys. This allows you to manage and use SSH credentials securely within your Jenkins jobs.
5. ****Vault Integration:**** Jenkins can integrate with external secret management tools like HashiCorp Vault. This enables you to store and manage sensitive data in Vault and retrieve it securely during Jenkins job execution.
6. ****Restricted Access:**** Limit access to Jenkins and its configurations to authorized personnel only. Use role-based access control (RBAC) to control who can view and modify sensitive credentials.

It’s essential to follow security best practices, such as regularly rotating credentials, using strong encryption algorithms, and restricting access to sensitive data, to ensure the highest level of security in your Jenkins environment. Regularly audit your Jenkins configuration to identify and address any potential security vulnerabilities.

## ****Intermediate Questions****

### ****Q16. How Can You Clone A Git Repository Via Jenkins?****

If you want to clone a Git repository via Jenkins, you have to enter the e-mail and user name for your Jenkins system. Switch into your job directory and execute the “git config” command for that.

### ****Q17.**** ****Explain how you can set up Jenkins job.****

My approach to this answer will be to first mention how to create Jenkins job.

Go to Jenkins top page, select “New Job”, then choose “Build a free-style software project”.

Now you can tell the elements of this freestyle job:

* Optional SCM, such as CVS or Subversion where your source code resides.
* Optional triggers to control when Jenkins will perform builds.
* Some sort of build script that performs the build (ant, maven, shell script, batch file, etc.) where the real work happens.
* Optional steps to collect information out of the build, such as archiving the artifacts and/or recording javadoc and test results.
* Optional steps to notify other people/systems with the build result, such as sending e-mails, IMs, updating issue tracker, etc..

### ****Q18. How to create a backup and copy files in Jenkins?****

The answer to this question is really direct.

To create a backup all you need to do is to periodically back up your JENKINS\_HOME directory. This contains all of your build jobs configurations, your slave node configurations, and your build history. To create a back-up of your Jenkins setup, just copy this directory. You can also copy a job directory to clone or replicate a job or rename the directory.

### ****Q19. How will you secure Jenkins?****

The way I secure Jenkins is mentioned below if you have any other way to do it than mention that:

* Make sure that the global security is on.
* Check if Jenkins is integrated with my company’s user directory with an appropriate plugin.
* Ensure that the matrix/Project matrix is enabled to fine-tune access.
* Automate the process of setting rights/privileges in Jenkins with custom version controlled script.
* Limit physical access to Jenkins data/folders.
* Periodically run security audits on the same.

### ****Q20.**** ****Explain how you can deploy a custom build of a core plugin?****

Below are the steps to deploy a custom build of a core plugin:

* Stop Jenkins.
* Copy the custom HPI to ****$Jenkins\_Home/plugins****.
* Delete the previously expanded plugin directory.
* Make an empty file called ****<plugin>.hpi.pinned****.
* Start Jenkins.

### ****Q21. What you do when you see a broken build for your project in Jenkins?****

There can be multiple answers to this question I will approach this task in the following way:

I will open the console output for the broken build and try to see if any file changes were missed. If I am unable to find the issue that way, then I will clean and update my local workspace to replicate the problem on my local and try to solve it.

If you do it in a different way then just mention that in your answer.

### ****Q22.**** ****What are the various ways in which build can be scheduled in Jenkins?****

You can schedule a build in Jenkins in the following ways:

* By source code management commits
* After completion of other builds
* Can be scheduled to run at a specified time (crons)
* Manual Build Requests

### ****Q23. What is the use of Pipelines in Jenkins?****

Pipeline plugin is used in Jenkins for making the Jenkins Pipeline, which gives us the view of stages or tasks to perform one after the other in the pipeline form. It models a series of related tasks. Pipelines help the teams to review, edit and iterate upon the tasks. Pipelines are durable and it can optionally stop and wait for human approval as well to start the next task. A pipeline is extensible and can perform work in parallel. It supports complex CD requirements.

### ****Q24. Explain the terms Agent, post-section, Jenkinsfile****

****Agent:****It is directive to tell Jenkins to execute the pipeline in a particular manner and order.

****Post-section:****If we have to add some notification and to perform other tasks at the end of a pipeline, post-section will definitely run at the end of every pipeline’s execution.

****Jenkinsfile:****The text file where all the definitions of pipelines are defined is called Jenkinsfile. It is being checked in the source control repository.

### ****Q25. Do you know about cloud computing? How can Jenkins fit into a cloud computing environment? Explain with an example.****

Let us take the example of AWS cloud service. Cloud computing services use the CI/CD model so that they can push their work to the customers and constantly receive feedback. Jenkins is used to automating the CI/CD pipelines. For example, a lot of Jenkins plugins are available for many of the AWS services like Amazon EC2 and ECS.

### ****Q26. What is Kubernetes? How can you integrate Jenkins with Kubernetes?****

[Kubernetes](https://kubernetes.io/" \t "https://www.edureka.co/blog/interview-questions/jenkins-interview-questions/_blank) is a container orchestration tool. With Kubernetes, one can create multiple container instances to achieve more fault tolerance. You can use the Kubernetes deploy plugin to use it with Jenkins for continuous deploy.

### ****Q27. Have you run automated tests on Jenkins? How is it done?****

Yes, this can be done easily. Automated tests can be run through tools like Selenium or maven. Developers can schedule the test runs. Jenkins displays the test results and sends a report to the developers.

### ****Q28.  Let us say, you have a pipeline. The first job was successful, but the second failed.  What should you do next?****

You just need to restart the pipeline from the point where it failed by doing ‘restart from stage’.

### ****Q29. What is the use of JENKINS HOME directory?****

All the settings, logs and configurations are stored in the JENKINS\_HOME directory.

### ****Q30. What is a backup plugin? Why is it used?****

This is a helpful plugin that backs up all the critical settings and configurations to be used in the future. This is useful in cases when there is a failure so that we don’t lose the settings.

### ****Q31. What is a trigger? Give an example of how the repository is polled when a new commit is detected.****

Triggers are used to define when and how pipelines should be executed.

When Jenkins is integrated with an SCM tool, for example, Git, the repository can be polled every time there is a commit.

The Git plugin should be first installed and set up.

After this, you can build a trigger that specifies when a new build should be started. For example, you can create a job that polls the repository and triggers a build when a change is committed.

### ****Q32. How do you define parameters for a build in Jenkins?****

A build can take several input parameters to execute. For example, if you have multiple test suites, but you want to run only one. You can set a parameter so that you are able to decide which one should be run. To have parameters in a job, you need to specify the same while defining the parameter. The parameter can be anything like a string, a file or a custom.

### ****Q33. What are the ways to configure Jenkins node agent to communicate with Jenkins master?****

There are 2 ways to start the node agent –

****Browser**** – if Jenkins node agent is launched from a browser, a JNLP (Java Web Start) file is downloaded. This file launches a new process on the client machine to run jobs.

****Command-line**** – to start the node agent using the command line, the client needs the executable agent.jar file. When this file is run, it simply launches a process on the client to communicate with the Jenkins master to run build jobs.

### ****Q34. How does Jenkins authenticate users?****

#### ****There are 3 ways –****

* The default way is to store user data and credentials in an internal database.
* Configure Jenkins to use the authentication mechanism defined by the application server on which it is deployed.
* Configure Jenkins to authenticate against LDAP server.

### ****Q35.  How can you use a third-party tool in Jenkins?****

Below are the steps used for working with a third-party tool in Jenkins.

* First install the third-party software
* Download the plug-in that supports the third-party tool.
* Configure the third-party tool in the admin console.
* Then use the required plug-in from the Jenkins build job.

For different third-party tools, the procedure may vary slightly, because of the difference in configuration settings.

### ****Q36. What are the types of pipelines in Jenkins?****

There are 3 types –

1. CI CD pipeline (Continuous Integration Continuous Delivery)
2. Scripted pipeline
3. Declarative pipeline

### ****Q37. What syntax does Jenkins use to schedule build job or SVN polling?****

The cron syntax.

Cron syntax is represented using five asterisks each separated by a space. The syntax is as follows – [minutes] [hours] [day of the month] [month] [day of the week]. Example, if you want to set up a cron for every Monday at 11.59 pm, it would be 59 11 \* \* 1

### ****Q38. What is DevOps and in which stage does Jenkins fit in?****

DevOps is a software development practice that blends software development (Dev) with the IT operations (Ops) making the whole development lifecycle simpler and shorter by constantly delivering builds, fixes, updates, and features. Jenkins plays a crucial role because it helps in this integration by automating the build, test and deployment process.

### ****Q39. Do you know any other Continuous Integration tools? How is Jenkins better than any of those?****

There are many other CI tools, and the most prominent ones are –

* TeamCity
* Bamboo
* Perforce
* Circle CI
* Go
* ThoughtWorks
* Integrity
* Travis CI

There are many more. We cannot say if Jenkins is better than each because each has its own unique features. For example, TeamCity offers great .NET support but is complex and costly, Travis CI is free just like Jenkins and has good documentation too. Bamboo too offers efficient and faster builds but is not completely free and so on.

### ****Q40. Name a Jenkins environment variable you have used in a shell script or batch file.****

There are numerous environment variables that are available by default in any Jenkins build job. A few commonly used ones include:

$JOB\_NAME

$NODE\_NAME

$WORKSPACE

$BUILD\_URL

$JOB\_URL

Note that, as new Jenkins plug-ins are configured, more environment variables become available. For example, when the Jenkins Git plug-in is configured, new Jenkins Git environment variables, such as $GIT\_COMMIT and $GIT\_URL, become available to be used in scripts.

### ****Q41. What is Continuous Integration In Jenkins?****

In software development, multiple developers or teams work on different segments of the same web application. So in this case, you have to perform integration testing by integrating all modules. In order to do that an automated process for each piece of code is performed on a daily bases so that all your codes get tested. This process is known as continuous integration.

### ****Q42. How do you achieve continuous integration using Jenkins?****

Here are the steps –

All the developers commit their source code changes to the shared Git repository.

Jenkins server checks the shared Git repository at specified intervals and detected changes are then taken into the build.

The build results and test results are shared to the respective developers

The built application is displayed on a test server like Selenium and automated tests are run.

The clean and tested build is deployed to the production server.

### ****Q43. What is a DSL Jenkins?****

The Jenkins “Job DSL / Plugin” is made up of two parts – first, The Domain Specific Language (DSL) itself that allows users to describe jobs using a Groovy-based language, and second, a Jenkins plugin which manages the scripts and the updating of the Jenkins jobs which are created and maintained as a result.

### ****Q44. How do you create Multibranch Pipeline in Jenkins?****

The Multibranch Pipeline project type enables you to implement different Jenkinsfiles for different branches of the same project. In a Multibranch Pipeline project, Jenkins automatically discovers, manages and executes Pipelines for branches that contain a Jenkinsfile in source control.

### ****Q.45 What are the types of jobs or projects in Jenkins?****

These are the types of jobs/projects in Jenkins –

* Freestyle project
* Maven project
* Pipeline
* Multibranch pipeline
* External Job
* Multi-configuration project
* Github organization

### ****Q46. What is blue ocean in Jenkins?****

It is a project that was started with the purpose to rethink the user experience of Jenkins, modeling and presenting the process of software delivery by surfacing information that’s important to development teams. This is done with as few clicks as possible, while still staying true to the extensibility that is core to Jenkins. While this project is in the alpha stage of development, the intent is that Jenkins users can install Blue Ocean side-by-side with the Jenkins Classic UI via a plugin

**Q50.What is Jenkins Build Cause?**

In Jenkins, a Build Cause refers to the reason or trigger that initiated the execution of a particular build job. It provides information about the event or condition that led to the build process being started. Knowing the build cause can be helpful for understanding why a build was triggered and diagnosing any issues related to the build process.

Jenkins supports various build causes, and some common ones include:

1. Manual Build: The build was manually triggered by a user or an administrator through the Jenkins web interface or API.
2. SCM (Source Code Management) Change: The build was triggered due to changes committed to the version control system (e.g., Git, Subversion) associated with the Jenkins job. This is commonly used for Continuous Integration (CI) to automatically trigger builds when code changes are detected.
3. Timer Trigger: The build was scheduled to run at specific time intervals, such as every hour or daily, using Jenkins’ built-in scheduling mechanism.
4. Upstream Build: The build was triggered as a result of another job (the upstream job) completing successfully and triggering this downstream job.
5. Webhook or External Event: The build was triggered by an external event, such as a webhook from a third-party service or tool, integrating with Jenkins.
6. Pipeline Trigger: For Jenkins Pipeline jobs, the build cause can include various reasons based on the logic and conditions defined in the pipeline script.

Jenkins captures the build cause information and makes it available during the build execution. You can access this information through Jenkins environment variables or by using plugins that provide additional capabilities for processing build cause data.

Understanding the build cause helps in tracing the origin of a build, diagnosing build-related issues, and maintaining a clear audit trail of how and why builds were triggered in a Jenkins environment.

**Q51. How did Jenkins know when to execute a Scheduled job/pipeline and how it is triggered?**

Jenkins can execute scheduled jobs or pipelines based on user-defined time intervals using its built-in scheduling mechanism. Scheduled jobs are also known as periodically executed jobs. Jenkins uses a cron-like syntax to define the schedule for these jobs, allowing users to specify when and how often the job should run.

Here’s how Jenkins knows when to execute a scheduled job or pipeline:

1. Cron-like Schedule: When setting up a scheduled job or pipeline in Jenkins, users define a cron-like schedule using the Jenkins scheduling syntax. The syntax consists of five fields representing minute, hour, day of the month, month, and day of the week, respectively. Each field can be a specific value, a list of values, a range, or an asterisk (\*), representing “every.”

   For example, to run a job every day at 3:00 AM, the schedule would be: `0 3 \* \* \*`.

1. Timer Trigger: Jenkins has a built-in timer that continuously checks the current time against the defined schedule of each job. When the specified time matches the schedule, Jenkins triggers the job to start its execution.
2. Persistence: Jenkins stores the schedule information and job configurations in its database, so the schedule persists across Jenkins restarts.
3. Scheduled Pipeline Syntax: For Jenkins Pipeline jobs, the cron-like schedule can be defined in the Jenkinsfile using the `cron` syntax. For example:

   “`groovy

   pipeline {

       triggers {

           cron(‘0 3 \* \* \*’)

       }

       // Pipeline stages and steps go here

   }

   “`

Once the job or pipeline is triggered, Jenkins starts the build process, and the job’s defined tasks or pipeline stages are executed based on the defined workflow.

Scheduled jobs are useful for automating repetitive tasks, like running tests, performing code analysis, or deploying applications, at specific intervals. They provide a way to regularly execute tasks without manual intervention, ensuring continuous integration, delivery, and deployment processes in a Jenkins environment.

**Q52. What are the Scopes of Jenkins Credentials?**

In Jenkins, credentials have different scopes that define where and how they can be used within the system. The scopes determine the visibility and accessibility of credentials, ensuring that they are appropriately secured and only accessible where needed. There are three main scopes for Jenkins credentials:

1. System: System scope credentials are global credentials available across the entire Jenkins instance. These credentials are not tied to any specific Jenkins job or pipeline. They can be used by any job or plugin within Jenkins, regardless of the folder or project they belong to. System scope credentials are useful for cases where credentials need to be shared and used across multiple jobs or plugins.
2. Global: Global scope credentials are available to all jobs within a specific Jenkins folder or organization. They are visible to all jobs and pipelines within that folder or organization. Global scope credentials are useful when you want to restrict the usage of credentials to a specific group of jobs or projects.
3. Job: Job scope credentials are specific to an individual Jenkins job or pipeline. They are only accessible within the context of that job and are not visible to other jobs or plugins. Job scope credentials are ideal when credentials should be isolated and used only by a specific job or pipeline.

When adding credentials to Jenkins, you can choose the appropriate scope based on the use case and security requirements. It’s essential to carefully manage and secure credentials to prevent unauthorized access and protect sensitive information. By using the correct scope, you can ensure that credentials are available where needed while maintaining a high level of security in your Jenkins environment.

**Q53. What is a Jenkins Shared Library and how it is useful?**

A Jenkins Shared Library is a powerful feature in Jenkins that allows you to define reusable code and functions that can be shared across multiple Jenkins pipelines. It provides a way to centralize and manage common logic, steps, and utilities, making it easier to maintain and standardize Jenkins pipelines.

Shared Libraries are created as Groovy scripts or code that reside in a version control system, such as Git, and can be referenced and used by different Jenkins pipelines. This approach promotes code reuse, reduces duplication, and ensures consistency in your CI/CD workflows. Here’s how a Jenkins Shared Library is useful:

1. Code Reusability: Shared Libraries allow you to write code once and reuse it in multiple pipelines. This avoids duplication of code and simplifies maintenance, as updates and improvements made to the shared code automatically propagate to all pipelines using it.
2. Standardization: By centralizing commonly used steps and utilities in a shared library, you can enforce best practices, coding standards, and security measures consistently across all pipelines in your Jenkins instance.
3. Abstraction: Shared Libraries enable you to abstract complex logic and operations into simple, easy-to-use functions. This abstraction hides the underlying complexity and makes pipelines more readable and maintainable.
4. Encapsulation: With Shared Libraries, you can encapsulate sensitive information, such as credentials, in a secure manner. This helps protect sensitive data and prevents it from being exposed in Jenkins pipeline scripts.
5. Easy Updates: When you need to make changes or enhancements to the shared code, you can do it in one central location. The changes automatically apply to all pipelines using the library, ensuring consistency and reducing the effort required for maintenance.
6. Testing and Validation: Shared Libraries can be thoroughly tested and validated before being used in pipelines, providing an added layer of assurance for code quality.
7. Cross-Team Collaboration: Shared Libraries enable different teams to collaborate and share common functions and tools across projects, enhancing overall efficiency and productivity.

To use a Shared Library in a Jenkins pipeline, you need to define the library in Jenkins’ global configuration. Once configured, you can reference the library and call its functions in your pipeline scripts. This modular approach helps in building complex, reusable, and maintainable CI/CD pipelines and encourages the adoption of best practices across the entire organization.

**Q54. How to get the Jenkins version programmatically in Jobs/Pipelines or nodes other than master?**

To programmatically obtain the Jenkins version from within Jenkins Jobs/Pipelines or nodes other than the master node, you can use Jenkins’ built-in environment variables or access Jenkins API. Here are the two methods you can use:

1. Using Environment Variables:

Jenkins provides several environment variables that you can use in your Jobs/Pipelines to access information about the Jenkins environment. The `JENKINS\_VERSION` environment variable contains the Jenkins version.

In a Jenkins Pipeline, you can access the version like this:

“`groovy

pipeline {

    agent any

    stages {

        stage(‘Get Jenkins Version’) {

            steps {

                script {

                    echo “Jenkins Version: ${env.JENKINS\_VERSION}”

                }

            }

        }

    }

}

“`

1. Using the Jenkins API:

You can also use the Jenkins API to retrieve information about the Jenkins instance, including the version. You can make an HTTP GET request to the Jenkins API endpoint, which will provide you with JSON or XML data containing various details about Jenkins, including the version.

For example, to get the Jenkins version using `curl` in a shell script on a node other than the master, you can use:

“`bash

curl -s “http://JENKINS\_URL/api/json” | jq -r ‘.version’

“`

Replace `JENKINS\_URL` with the URL of your Jenkins instance. The `jq` command is used here to parse the JSON output and extract the version.

Keep in mind that to access the Jenkins API programmatically, you may need to provide authentication credentials, such as an API token or username and password, depending on your Jenkins security settings.

Both of these methods will allow you to programmatically retrieve the Jenkins version from within Jenkins jobs, pipelines, or nodes other than the master node, providing you with the version information for further use or processing in your scripts or automation tasks.

**Q55. What is the Jenkins User Content service?**

In Jenkins, the “User Content” service is a built-in feature that allows users to upload and manage static files and resources that are associated with Jenkins but not directly related to the build process. It offers a convenient way to store files, artifacts, scripts, images, and other resources that Jenkins jobs, pipelines, and plugins can access and use.

The User Content service is particularly useful for storing files that are required for the Jenkins instance itself or for other build jobs, rather than files generated or produced during the build process. Examples of content that can be stored in the User Content directory include:

1. Scripts and Configurations: Users can upload shell scripts, PowerShell scripts, configuration files, or any other scripts required by build jobs or pipelines.
2. Artifacts: Static files, such as binaries, packages, or archives, that are shared among multiple build jobs can be stored in the User Content directory.
3. Shared Libraries: Jenkins Shared Libraries, which contain reusable code and logic, can be stored in the User Content directory and made accessible to Jenkins jobs.
4. Global Configurations: Configuration files or resources that affect the overall Jenkins configuration or behavior can be stored here.
5. Custom Plugins and Extensions: Some Jenkins plugins may require additional files or extensions, which can be placed in the User Content directory.

By using the User Content service, you can organize and manage these static resources separately from the build workspace, making it easier to share and reuse files among different jobs and avoiding duplication of resources.

The User Content service provides URLs that allow direct access to the stored content. For example, if a file is uploaded to the User Content directory with the name “my-script.sh,” it can be accessed using the URL: `JENKINS\_URL/userContent/my-script.sh`, where `JENKINS\_URL` is the URL of your Jenkins instance.

The User Content service is a convenient and practical feature in Jenkins that complements the build process by allowing users to manage and access files and resources that support the continuous integration and delivery workflows in a more organized and efficient manner.

## ****Advanced Questions****

### ****Q47. What is Continuous Testing?****

Continuous Testing is the process where you execute automated tests as part of the software delivery pipeline. This is done so that you get the feedback on the business risks associated with software as early as possible. It consists of evolving and extending test automation to address the increased complexity and pace of modern application development and delivery.

Continuous Testing means that testing takes place on a continuous basis without any disruption of any kind. In a Continuous DevOps process, a software change is continuously moving from Development to Testing to Deployment. The code undergoes continuous development, delivery, testing and deployment.

### ****Q48.**** ****Explain how you can move or copy Jenkins from one server to another?****

I will approach this task by copying the jobs directory from the old server to the new one. There are multiple ways to do that, I have mentioned it below:

You can:

Move a job from one installation of Jenkins to another by simply copying the corresponding job directory.

Make a copy of an existing job by making a clone of a job directory by a different name.

Rename an existing job by renaming a directory. Note that if you change a job name you will need to change any other job that tries to call the renamed job.

### ****Q49. How do you integrate Git with Jenkins?****

These are the steps to integrate [Git](https://bit.ly/1LBvRP9" \t "https://www.edureka.co/blog/interview-questions/jenkins-interview-questions/_blank) with Jenkins –

****1****.Click on the ****Manage Jenkins**** button on your Jenkins dashboard:

****

****2****. Click on ****Manage Plugins.****

****

****3.****In the Plugins Page

1. Select the GIT Plugin
2. Click on ****Install without restart.****The plugin will take a few moments to finish downloading depending on your internet connection, and will be installed automatically.
3. You can also select the option ****Download now and Install after restart.****
4. You will now see a “No updates available” message if you already have the Git plugin installed.

****4.****Once you install the plugins , go to ****Manage Jenkins****on your Jenkins dashboard. You will see your plugins listed among the rest.

### **IMG_260**

### ****Q50. How can you temporarily turn off Jenkins security if the administrative users have locked themselves out of the admin console?****

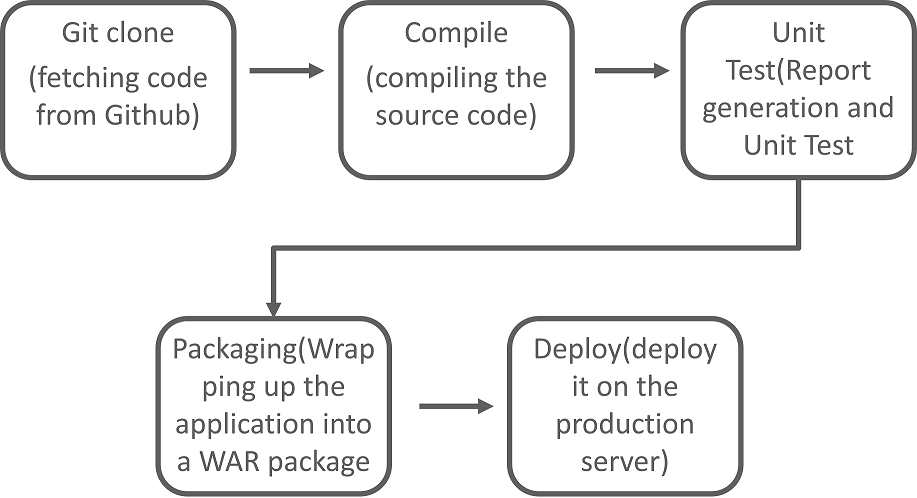
The JENKINS\_HOME folder contains a file named config.xml. When you enable the security, this file contains an XML element named useSecurity that changes to true. If you change this setting to false, security will be disabled the next time Jenkins is restarted.

<useSecurity>false</useSecurity>

However, we must understand that disabling security should always be both a last resort and a temporary measure. Once you resolve the authentication issues, make sure that you re-enable Jenkins security and reboot the CI server.

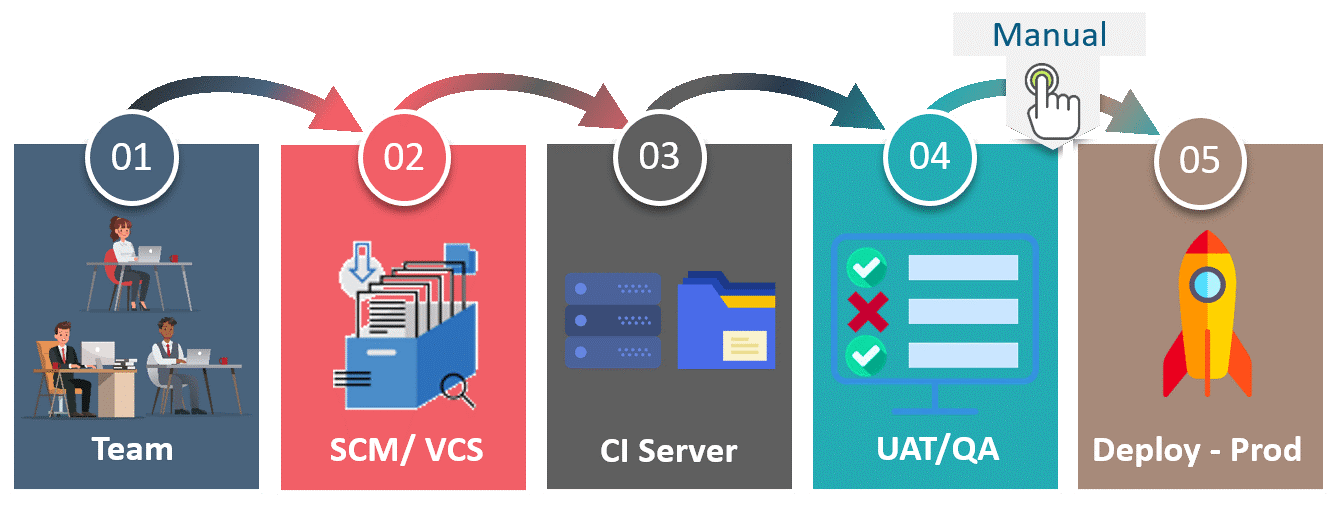
### ****Q51. Can you define a Continuous Delivery Workflow?****

The flowchart below shows the Continuous Delivery Workflow. Hope it will be much easier to understand with visuals.

****

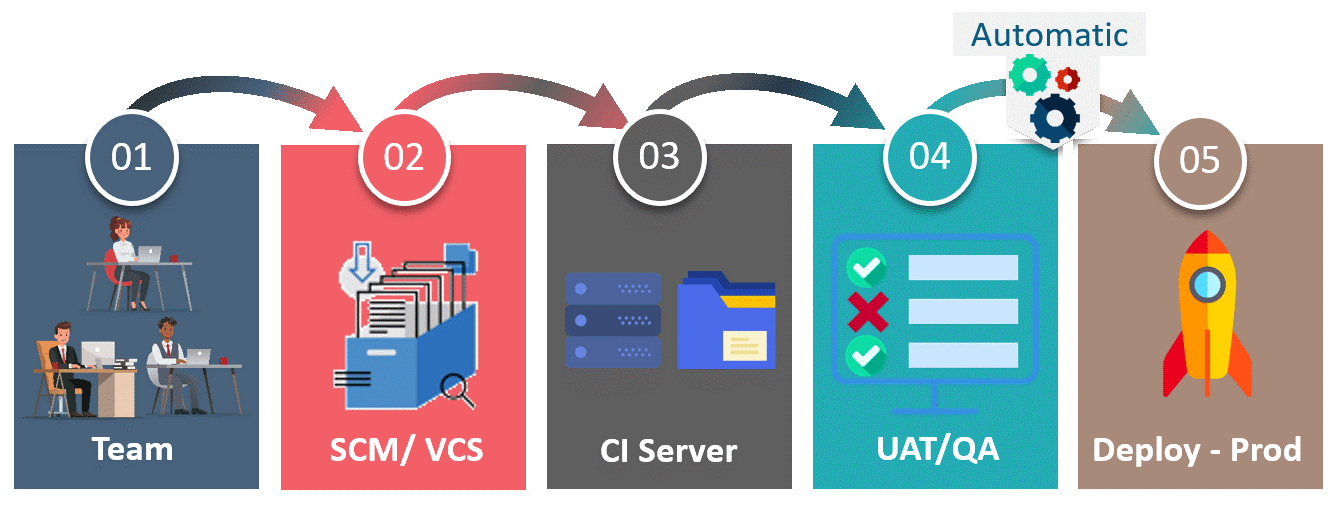
### ****Q52. What is the difference between Continuous Delivery and Continuous Deployment?****

****Continuous Delivery: (Manual Deployment to Production. Does not involve every change to be deployed.)****



Continuous Delivery is a software development practice where you build software in such a way that the software can be released to the production at any time. You achieve Continuous Delivery by continuously integrating the products built by the development team, running automated tests on those built products to detect problems and then push those files into production-like environments to ensure that the software works in production.

****Continuous Deployment: (Automated Deployment to Production. Involves deploying every change automatically)****

Continuous deployment means that every change that you make, goes through the pipeline, and if it passes all the tests, it automatically gets deployed into production. So, with this approach, the quality of the software release completely depends on the quality of the test suite as you have automated everything.

### ****Q53. What do you mean by Pipeline as a Code?****

****Pipeline as Code**** describes a set of features that allow Jenkins users to define pipelined job processes with code, stored and versioned in a source repository. These features allow Jenkins to discover, manage, and run jobs for multiple source repositories and branches — eliminating the need for manual job creation and management.

To use Pipeline as Code, projects must contain a file named Jenkinsfile in the repository root, which contains a “Pipeline script.”

Additionally, one of the enabling jobs needs to be configured in Jenkins:

****Multibranch Pipeline****: build multiple branches of a single repository automatically

****Organization Folders****: scan a ****GitHub Organization**** or ****Bitbucket Team**** to discover an organization’s repositories, automatically creating managed Multibranch Pipeline jobs for them