JENKINS

Continuous Integration is the most important part of DevOps that is used to integrate various DevOps stages. Jenkins is the most famous Continuous Integration tool, I know you are curious to know the reason behind the popularity of Jenkins and I am pretty sure after reading this *What is Jenkins* blog, all your questions will get answered.

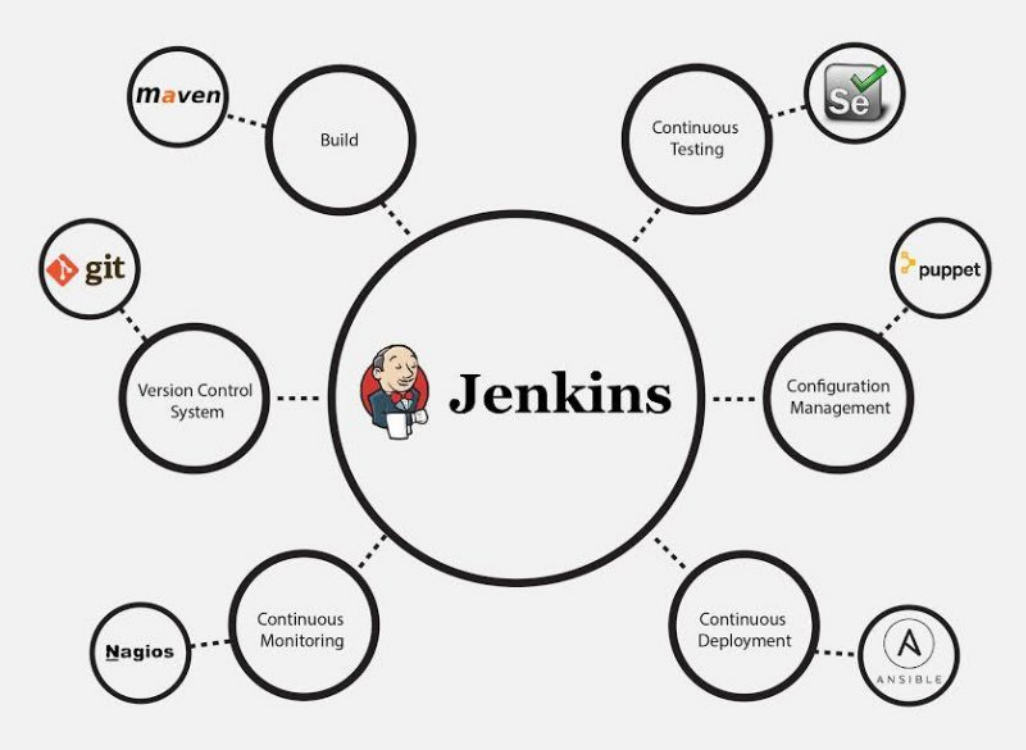
What is Jenkins?

Jenkins is an open source automation tool written in Java with plugins built for Continuous Integration purpose. 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.

With Jenkins, organizations can accelerate the software development process through automation. Jenkins integrates development life-cycle processes of all kinds, including build, document, test, package, stage, deploy, static analysis and much more.

Jenkins achieves Continuous Integration with the help of plugins. Plugins allows the integration of Various DevOps stages. If you want to integrate a particular tool, you need to install the plugins for that tool. For example: Git, Maven 2 project, Amazon EC2, HTML publisher etc.

The image below depicts that Jenkins is integrating various DevOps stages:



Advantages of Jenkins include:

- It is an open source tool with great community support.
- It is easy to install.
- It has 1000+ plugins to ease your work. If a plugin does not exist, you can code it and share with the community.
- It is free of cost.
- It is built with Java and hence, it is portable to all the major platforms.

There are certain things about Jenkins that separates it from other the Continuous Integration tool. Let us take a look on those points.

Jenkins Key Metrics

Following are some facts about Jenkins that makes it better than other Continuous Integration tools:

- Adoption: Jenkins is widespread, with more than 147,000 active installations and over 1 million users around the world.
- Plugins: Jenkins is interconnected with well over 1,000 plugins that allow it to integrate with most of the development, testing and deployment tools.

It is evident from the above points that Jenkins has a very high demand globally. Before we dive into Jenkins it is important to know what is Continuous Integration and why it was introduced.

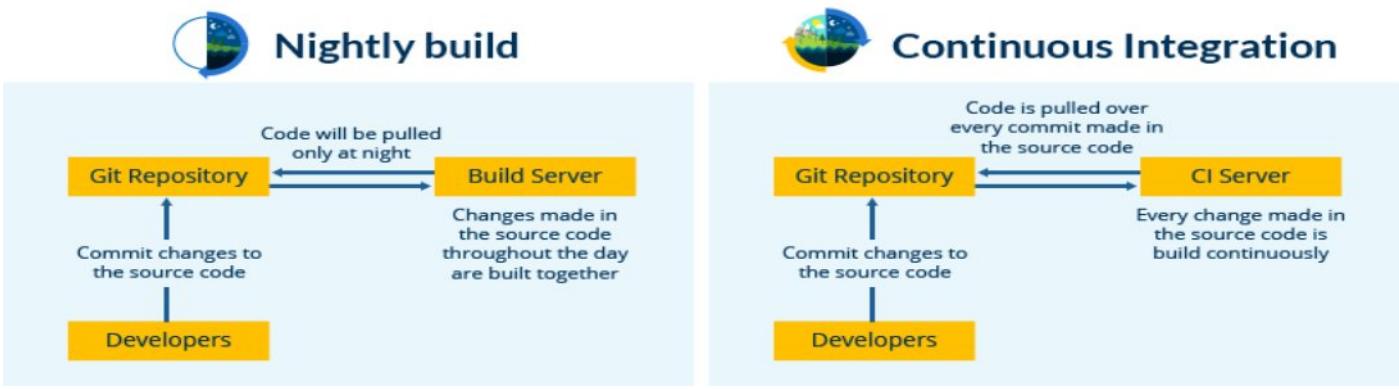
What is Continuous Integration?

Continuous Integration is a development practice in which the developers are required to commit changes to the source code in a shared repository several times a day or more frequently. Every commit made in the repository is then built. This allows the teams to detect the problems early. Apart from this, depending on the Continuous Integration tool, there are several other functions like deploying the build application on the test server, providing the concerned teams with the build and test results etc.

Let us understand its importance with a use-case.

Continuous Integration in Nokia

I am pretty sure you all have used Nokia phones at some point in your life. In a software product development project at Nokia there was a process called Nightly builds. Nightly builds can be thought of as a predecessor to Continuous Integration. It means that every night an automated system pulls the code added to the shared repository throughout the day and builds that code. The idea is quite similar to Continuous Integration, but since the code that was built at night was quite large, locating and fixing of bugs was a real pain. Due to this, Nokia adopted Continuous Integration (CI). As a result, every commit made to the source code in the repository was built. If the build result shows that there is a bug in the code, then the developers only need to check that particular commit. This significantly reduced the time required to release new software.



Now is the correct time to understand how Jenkins achieves Continuous Integration.

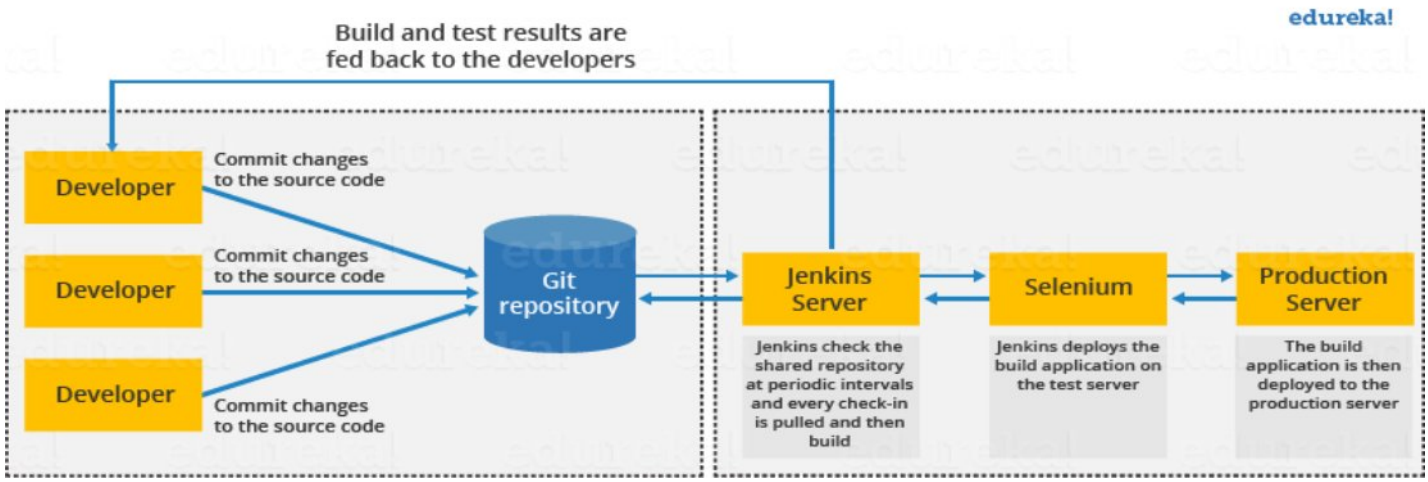
Continuous Integration With Jenkins

Let us imagine a scenario where the complete source code of the application was built and then deployed on test server for testing. It sounds like a perfect way to develop a software, but, this process has many flaws. I will try to explain them one by one:

- Developers have to wait till the complete software is developed for the test results.
- There is a high possibility that the test results might show multiple bugs. It was tough for developers to locate those bugs because they have to check the entire source code of the application.
- It slows the software delivery process.
- Continuous feedback pertaining to things like coding or architectural issues, build failures, test status and file release uploads was missing due to which the quality of software can go down.
- The whole process was manual which increases the risk of frequent failure.

It is evident from the above stated problems that not only the software delivery process became slow but the quality of software also went down. This leads to customer dissatisfaction. So to overcome such a chaos there was a dire need for a system to exist where developers can continuously trigger a build and test for every change made in the source code. This is what CI is all about. Jenkins is the most mature CI tool available so let us see how Continuous Integration with Jenkins overcame the above shortcomings.

I will first explain you a generic flow diagram of Continuous Integration with Jenkins so that it becomes self explanatory, how Jenkins overcomes the above shortcomings:



The above diagram is depicting the following functions:

- 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 built is successful, then Jenkins deploys the built in the test server.
- After testing, Jenkins generates a 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.

You now know how Jenkins overcomes the traditional SDLC shortcomings. The table below shows the comparison between “Before and After Jenkins”.

Before Jenkins	After Jenkins
The entire source code was built and then tested. Locating and fixing bugs in the event of build and test failure was difficult and time consuming, which in turn slows the software delivery process.	Every commit made in the source code is built and tested. So, instead of checking the entire source code developers only need to focus on a particular commit. This leads to frequent new software releases.
Developers have to wait for test results	Developers know the test result of every commit made in the source code on the run.
The whole process is manual	You only need to commit changes to the source code and Jenkins will automate the rest of the process for you.

Jenkins Tutorial

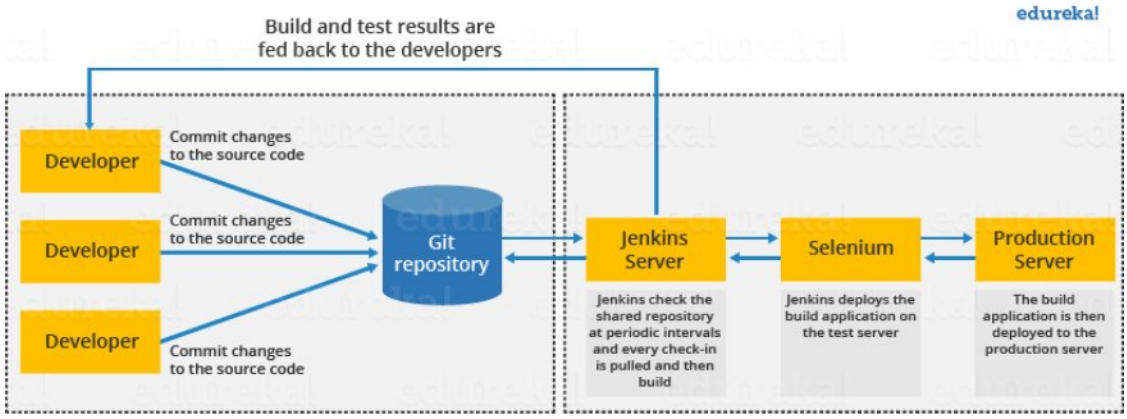
I hope you have read my previous blog on *What is Jenkins*. In this Jenkins Tutorial blog, I will focus on Jenkins architecture and Jenkins build pipeline along with that I will show you how to create a build in Jenkins.

Before we proceed with Jenkins Tutorial, the key takeaways from the previous blog are:

- Jenkins is used to integrate all DevOps stages with the help of plugins.
- Commonly used Jenkins plugins are Git, Amazon EC2, Maven 2 project, HTML publisher etc.
- Jenkins has well over 1000 plugins and 147,000 active installations along with over 1 million users around the world.
- With Continuous Integration every change made in the source code is built. It performs other functions as well, that depends on the tool used for Continuous Integration.
- Nokia shifted from Nightly build to Continuous Integration.
- Process before Continuous Integration had many flaws. As a result, not only the software delivery was slow but the quality of software was also not up to the mark. Developers also had a tough time in locating and fixing bugs.
- Continuous Integration with Jenkins overcame these shortcomings by continuously triggering a build and test for every change made in the source code.

Now is the correct time to understand Jenkins architecture.

Jenkins Architecture



This single Jenkins server was not enough to meet certain requirements like:

- Sometimes you might need several different environments to test your builds. This cannot be done by a single Jenkins server.
- If larger and heavier projects get built on a regular basis then a single Jenkins server cannot simply handle the entire load.

To address the above stated needs, Jenkins distributed architecture was introduced.

Jenkins Distributed Architecture

Jenkins uses a Master-Slave architecture to manage distributed builds. In this architecture, Master and Slave communicate through TCP/IP protocol.

Jenkins Master

Your main Jenkins server is the Master. The Master's job is to handle:

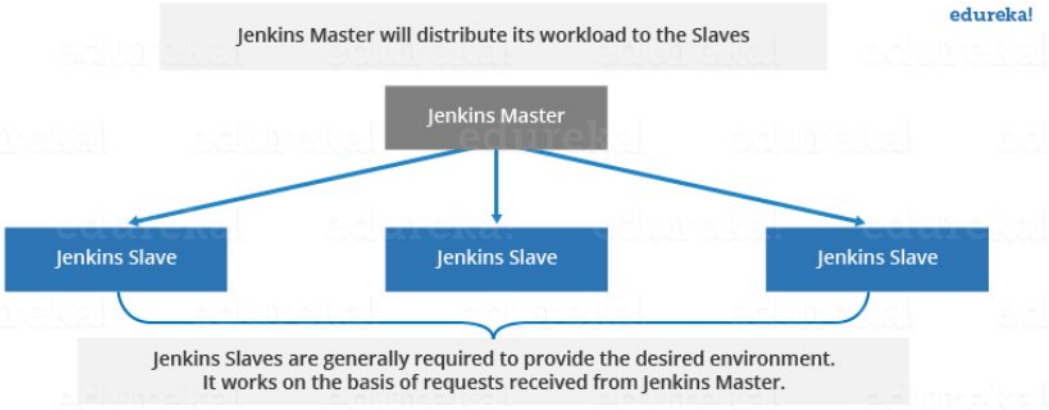
- Scheduling build jobs.
- Dispatching builds to the slaves for the actual execution.
- Monitor the slaves (possibly taking them online and offline as required).
- Recording and presenting the build results.
- A Master instance of Jenkins can also execute build jobs directly.

Jenkins Slave

A Slave is a Java executable that runs on a remote machine. Following are the characteristics of Jenkins Slaves:

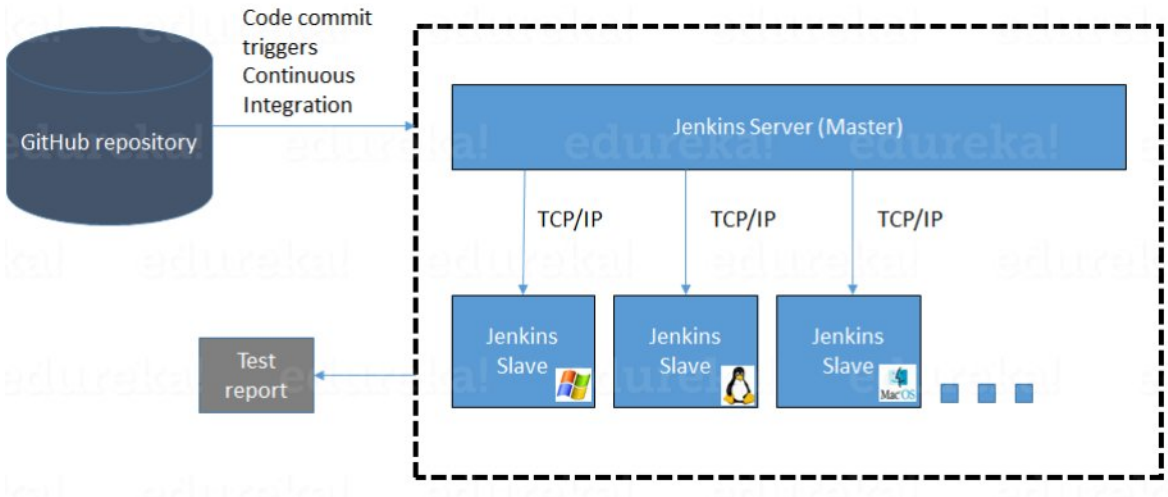
- It hears requests from the Jenkins Master instance.
- Slaves can run on a variety of operating systems.
- The job of a Slave is to do as they are told to, which involves executing build jobs dispatched by the Master.
- You can configure a project to always run on a particular Slave machine, or a particular type of Slave machine, or simply let Jenkins pick the next available Slave.

The diagram below is self explanatory. It consists of a Jenkins Master which is managing three Jenkins Slave.



Now let us look at an example in which Jenkins is used for testing in different environments like: Ubuntu, MAC, Windows etc.

The diagram below represents the same:



The following functions are performed in the above image:

- Jenkins checks the Git repository at periodic intervals for any changes made in the source code.
- Each builds requires a different testing environment which is not possible for a single Jenkins server. In order to perform testing in different environments Jenkins uses various Slaves as shown in the diagram.
- Jenkins Master requests these Slaves to perform testing and to generate test reports.

Jenkins Build Pipeline

It is used to know which task Jenkins is currently executing. Often several different changes are made by several developers at once, so it is useful to know which change is getting tested or which change is sitting in the queue or which build is broken. This is where pipeline comes into picture. The Jenkins Pipeline gives you an overview of where tests are up to. In build pipeline the build as a whole is broken down into sections, such as the unit test, acceptance test, packaging, reporting and deployment phases. The pipeline phases can be executed in series or parallel, and if one phase is successful, it automatically moves on to the next phase (hence the relevance of the name “pipeline”).The below image shows how a multiple build Pipeline looks like.

Hope you have understood the theoretical concepts. Now, let's have some fun with hands-on.

I will create a new job in Jenkins, it is a Freestyle Project. However, there are 3 more options available. Let us look at the types of build jobs available in Jenkins.

Freestyle Project:

Freestyle build jobs are general-purpose build jobs, which provides maximum flexibility. The freestyle build job is the most flexible and configurable option, and can be used for any type of project. It is relatively straightforward to set up, and many of the options we configure here also appear in other build jobs.

Multiconfiguration Job:

The "multiconfiguration project" (also referred to as a "matrix project") allows you run the same build job on different environments. It is used for testing an application in different environments, with different databases, or even on different build machines.

Monitor an External Job:

The "Monitor an external job" build job lets you keep an eye on non-interactive processes, such as cron jobs.

Maven Project:

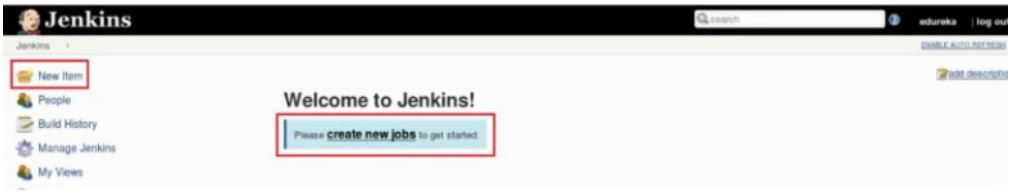
The "maven2/3 project" is a build job specially adapted to Maven projects. Jenkins understands Maven pom files and project structures, and can use the information gleaned from the pom file to reduce the work you need to do to set up your project.

Here is a video on Jenkins tutorial for better understanding of Jenkins. Check out this Jenkins tutorial video.

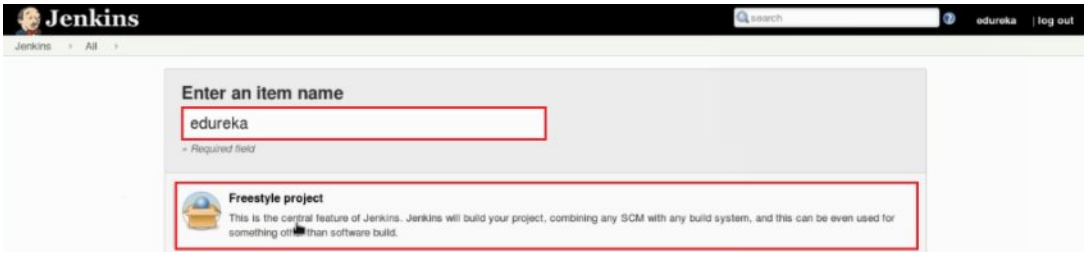
Getting Started With Jenkins | Jenkins and DevOps tutorial | Jenkins for Beginners | Edureka

Creating a Build Using Jenkins

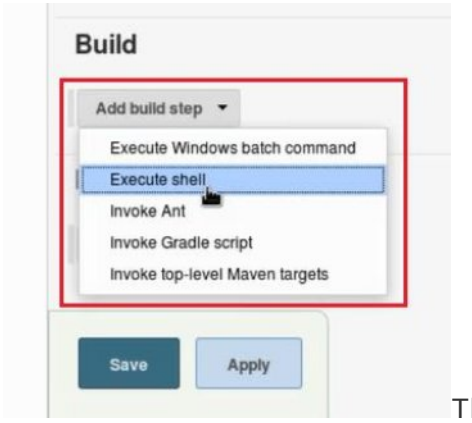
Step 1: From the Jenkins interface home, select New Item.



Step 2: Enter a name and select Freestyle project.

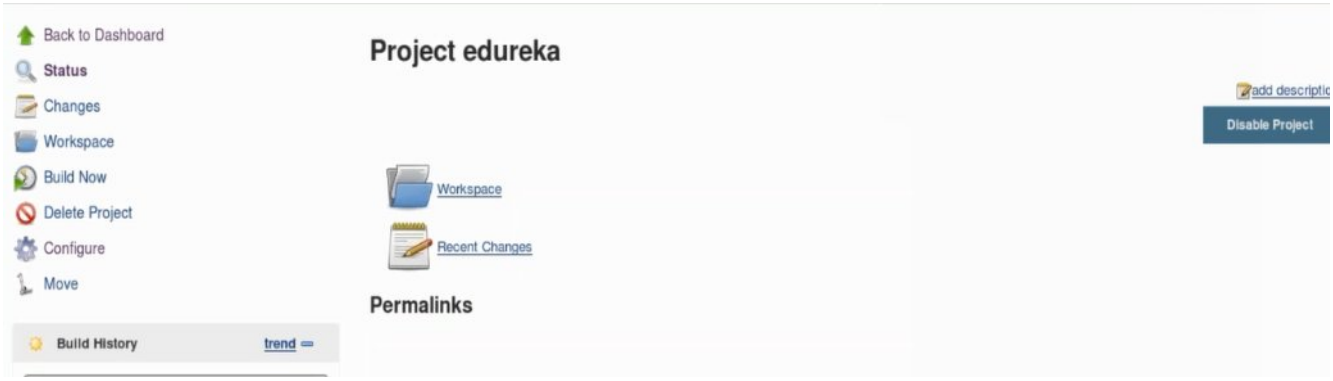


Step 3: This next page is where you specify the job configuration. As you'll quickly observe, there are a number of settings available when you create a new project. On this configuration page, you also have the option to Add build step to perform extra actions like running scripts. I will execute a shell script.



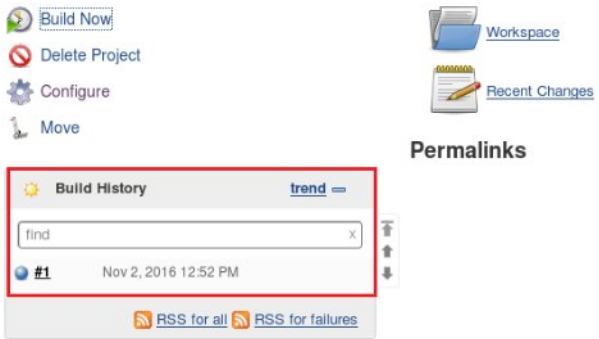
This will provide you with a text box in which you can add whatever commands you need. You can use scripts to run various tasks like server maintenance, version control, reading system settings, etc. I will use this section to run a simple script.

Step 4: Save the project, and you'll be taken to a project overview page. Here you can see information about the project, including its built history.



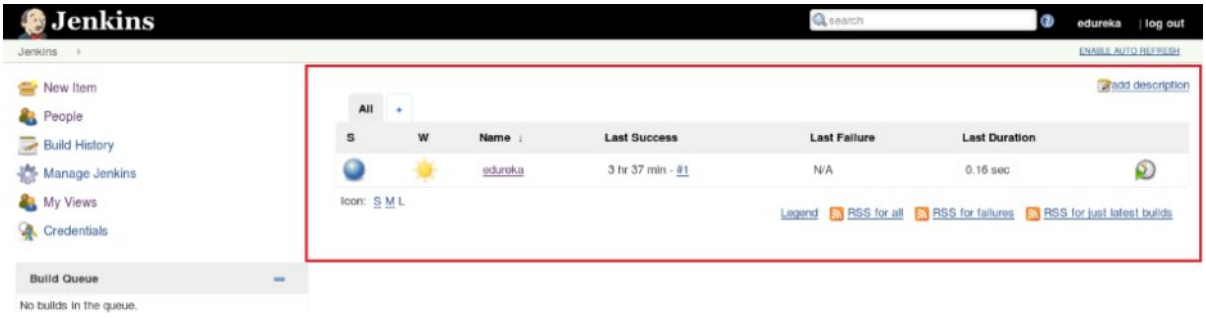
Step 5: Click Build Now on the left-hand side to start the build.

Step 6: To see more information, click on that build in the build history area, whereupon you'll be taken to a page with an overview of the build information.



Step 7: The Console Output link on this page is especially useful for examining the results of the job in detail.

Step 8: If you go back to Jenkins home, you'll see an overview of all projects and their information, including status.



Status of the build is indicated in two ways, by a weather icon and by a colored ball. The weather icon is particularly helpful as it shows you a record of multiple builds in one image.

As you can see in the above image, the sun represents that all of my builds were successful. The color of the ball gives us the status of that particular build, in the above image the color of the ball is blue which means that this particular build was successful.

In this Jenkins Tutorial, I have just given an introductory example. In my next blog, I will show you how to pull and build code from the GitHub repository using Jenkins.

INTERVIEW QUS

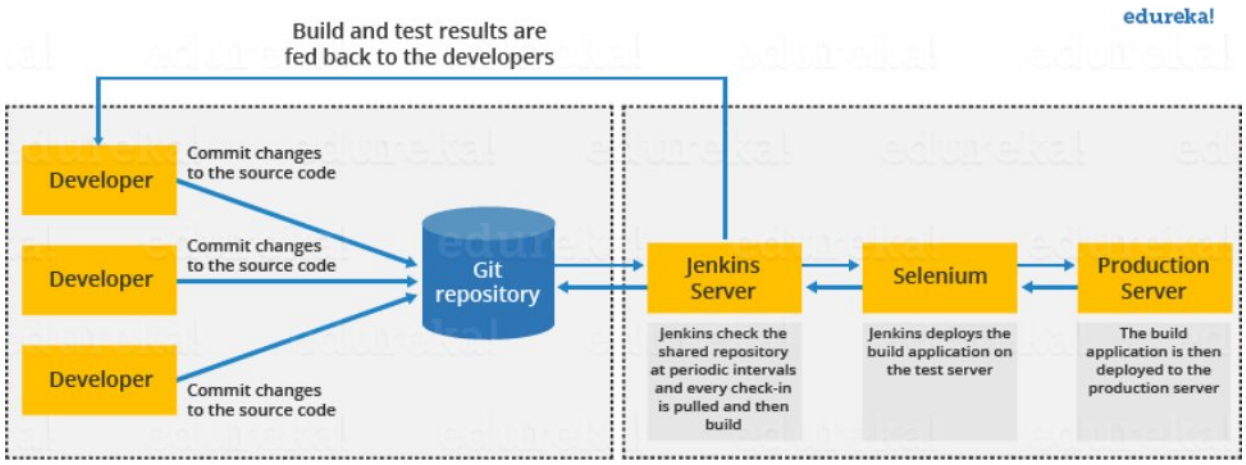
Q1. What is Jenkins?

My suggestion is to start this answer by giving a definition of Jenkins.

Jenkins is an open source automation tool written in Java with plugins built for Continuous Integration purpose. 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.

Once you have defined Jenkins give an example, you can refer the below mentioned use case:

- 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 built is successful, then Jenkins deploys the built in the test server.
- After testing, Jenkins generates a 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.



Interviewer now knows what is Jenkins but why we use it, there are many other CI tools as well, so why Jenkins?, the next question in this Jenkins interview questions will deal with that answer.

Q2. What are the benefits of using Jenkins?

I will suggest you to 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 integration stage, build failures are cached.
- For each change in the source code an automatic build report notification is generated.
- To notify developers about build report success or failure, it is integrated with LDAP mail server.
- Achieves continuous integration agile development and test driven development.
- With simple steps, maven release project is automated.
- Easy tracking of bugs at early stage in development environment than production.

Interviewer: Okay Jenkins looks like a really cool tool, but what are the requirements for using Jenkins?

Q3. What are the pre-requisites for using Jenkins?

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.

Remember, you have mentioned Plugins in your previous answer, so next question in this Jenkins interview questions blog will be regarding Plugins.

Q4. 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.

Q15. Which SCM tools Jenkins supports?

Below are Source code management tools supported by Jenkins:

- AccuRev
- CVS,
- Subversion,
- Git,
- Mercurial,
- Perforce,
- Clearcase
- RTC

Now, the next set of Jenkins interview questions will test your experience with Jenkins.

Q4. Mention what are the commands you can use to start Jenkins manually?

For this answer I will suggest you to 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:

TostartJenkins: jenkins.exe start
TostopJenkins: jenkins.exe stop
To restart Jenkins: jenkins.exe restart

Q6. 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..

Q7. Explain how to create a backup and copy files in Jenkins?

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.

Q8. 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:

- Ensure global security is on.
- Ensure that Jenkins is integrated with my company’s user directory with appropriate plugin.
- Ensure that 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 same.

I hope you have enjoyed the above set of Jenkins interview questions, the next set of questions will be more challenging, so be prepared.

Q9 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.

Q10. What is the relation between Hudson and Jenkins?

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

Q11. 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.

Q12. 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.

Q13. 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 specified time (crons)
- Manual Build Requests

Q14. What is the difference between Maven, Ant and Jenkins?

Maven and Ant are Build Technologies whereas Jenkins is a continuous integration tool.

Q16. What are the two components Jenkins is mainly integrated with?

According to me Jenkins is mainly integrated with the following:

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