Jenkins is an open-source automation tool built in Java. It has plugins built for continuous integration. Developers use Jenkins to build software projects and test them continuously to integrate the changes in their projects. Jenkins allows consistent delivery of software with its integrated features of testing and deployment. The organizations accelerate their software development with automation because of using Jenkins. In this tutorial, you will learn all the essential concepts of Jenkins. We have covered each concept that starts from the introduction and will explore all its components in detail. Now, let go through the concepts of Jenkins.

Jenkins Tutorial - Table of content

- Introduction to jenkins and continuous integration
- Features of jenkins
- Jenkins architecture
- Installation of Jenkins
- GIT Webhook
- Jenkins Pipelines
- Installing Jenkins on Windows Platform
- Configuration Jenkins on Windows Platform
- Jenkins User Management on Windows Platform
- Building Pipeline on Windows Platform
- Conclusion

1. Introduction to Jenkins and Continuous Integration

Jenkins provides seamless, ongoing development, testing, and deployment of newly created code. The support of continuous integration in Jenkins enables the developers to commit changes to their source code from a shared repository and all the changes to the source code are built continuously.

1.1 What is Continuous Integration?

Continuous Integration is an essential part of DevOps. It is mainly used to integrate various stages of DevOps. It enables the development team to make and implement small changes in the code and version control methods quite frequently. Continuous integration is done when all developers push the code onto a shared repository mostly multiple times a day. It has now become crucial to have such a mechanism in place to integrate and validate the changes made to the code.

1.2 Why use Continuous Integration?

The following advantages of Continuous Integration explain its usages.

Reduction of integration links: The projects need more than one person to develop and it also increases the risk of errors during integration. Depending on the code complexity, possible changes will be made. CI integration helps to alleviate the issues as it allows for regular integration.

High quality: As the risks gradually reduce, the time and manpower can be utilized to create functional-oriented code.

Working of version control: Committing things that break the build immediately triggers a notification thereby preventing anyone from pulling a broken code.

Easy Testing: The testing retains different versions and builds of the code makes the work easy to understand QAs, locate, and trace bugs efficiently.

Decreased deployment time: Automating the process of deployment makes it easy and frees up a lot of time and manpower.

Increased confidence: The absence of a possible failure or breakdown helps developers in delivering greater productivity and higher quality products.

2. Features of Jenkins

Jenkins offers many features for developers.

Easy Installation

Jenkins is a platform-agnostic, self-contained Java-based program, ready to run with packages for Windows, Mac OS, and Unix-like operating systems.

Easy Configuration

Jenkins is easy to set up and configure using its web interface, feature error checks, and a built-in help function.

Available Plugins

There are hundreds of plugins available in the Update Center, integrating with every tool in the CI and CD toolchain.

Extensible

Jenkins is extended with its plugin architecture that provides endless possibilities for what it can do.

Easy Distribution

Jenkins can easily distribute work across multiple machines for faster builds, tests, and deployments across multiple platforms.

Free Open Source

Jenkins is an open-source resource backed by heavy community support.

3. Jenkins Architecture

Standalone Jenkins instances can be an intensive disk and CPU resource-consuming process. You can scale it to avoid this by implementing slave nodes which essentially would help you offload a part of the master node's responsibilities. A slave is a device that is configured to act as an executor on behalf of the master. The master is the base installation of the Jenkins tool and does the basic operations and serves the user interface while the slaves do the actual work. The Developers commit changes to the source code which is found in the repository. The Jenkins CI server checks the repository at regular intervals and pulls any newly available code. The Build Server builds the code into an executable file. If the build fails, then feedback is sent to the developers. Jenkins deploys the build application to the test server. If the test fails, then the developers will be alerted. If the code is free from errors, the tested application will be deployed on the production server.

The files may contain various codes and can be very large that requires multiple builds. A single Jenkins server cannot handle multiple files and builds simultaneously, so to achieve this, a distributed Jenkins architecture is necessary. The below screen displays the Jenkins master is in charge of the UI and the slave nodes are of different OS types.

IMAGE

From the above-illustrated picture, the GitHub Repository is your Remote source code repository. The Jenkins server accesses the master environment and the master environment can push down to multiple other Jenkins Slave environments to distribute the workload. This will let you run multiple builds, tests, and product environment across the entire architecture. Jenkins Slaves can run various build versions of the code in various operating systems and the server master controls how each of the builds operates.

Jenkins includes many slaves working for a master. This Jenkins distributed build architecture can run identical test cases in various environments. The results are collected and combined on the master node for monitoring.

4. Installation of Jenkins

In this module, we cover the steps to install Jenkins.

4.1 Jenkins Master-Slave Installation on AWS

Apply the following steps for creating a master

Steps for creating a Master

1. Create a new EC2 instance with Amazon Linux AMI 2016.03

2. Update the system and install Jenkins with the following commands: $yum update –y$ wget -O /etc/yum.repos.d/jenkins.repo http://pkg.jenkins-ci.org/redhat/jenkins.repo

   $ rpm –import http://pkg.jenkins-ci.org/redhat/jenkins-ci.org.key

   Yum install –y Jenkins

   The user must keep a note that Jenkins installation can be performed either in the root or in the sudo access.
3. Update /etc/sysconfig/Jenkins to authorize Jenkins to access the environment variables used by Jenkins plugins and change the default time zone.
4. Now you need to register and start once the update is performed. Apply the following command.

   $chkconfig jenkins on$service jenkins start

   The following steps will also work for Jenkins installation on CentOS.
5. Jenkins is now enabled and set to run. Visit the site http://SERVER IP :8080. While redirecting to the 'Getting Started' screen, retrieve the password and unlock Jenkins. unlock jenkins'
   IMAGE

You are now running the empty master, let us now configure the slaves.

Steps to Configure the slaves

Let us now configure the slaves and update them in the master configuration.

1. Create a new EC2 instance as you did earlier with the sudo or the root access.
2. Add the base dependencies like Java, Git, Docker, and so on with the following commands: $ yum install -y docker git java-1.8.0-openjdk$ curl -L https://github.com/docker/compose/releases/download/1.6.2/docker-compose-`uname -s`-`uname -m` > /usr/local/bin/docker-compose$ chmod +x /usr/local/bin/docker-compose
3. Create the AMI on AWS. On the EC2 panel, navigate to Launch Instance and click on the slave that is configured and create a new image. Once the image is created, note down the AMI ID.

   IMAGE
4. Configure the master to use the AMI with the following steps: Create AWS credentials. Once chosen, limit the scope to the system and complete the form. For adding AWS as the cloud providers, follow: Manage Jenkins -> configure the system -> add a new cloud -> Amazon EC2 -> complete the form as needed

The master and a slave are now configured successfully!

The below screenshot illustrates the sample setup screen. We may practice Jenkins installation with other specifications of our choice. This is only the default and the generalized format.

IMAGE

4.2 Installing Jenkins Plugins

Installing Plugins is the core feature of Jenkins. This adds functionalities over the core to give us more powerful tools concerning the project. Let's see how to list, add, modify, update, and remove these plugins from Jenkins.

- To list all the plugins supported by Jenkins, visit https://wiki.jenkins-ci.org/display/JENKINS/Plugins
- Once logged in, move to the 'Manage Jenkins' tab on the left-hand side where we would handle all the installed plugins, as well as add or remove new ones.
- Under the Manage Plugins tab, you can search for a plugin or see all the available plugins.
- By selecting a plugin and clicking on Install without restart, you can install the plugin and check its functionality quickly despite waiting to restart Jenkins.
- For uninstalling a plugin, move to the Installed tab, select the plugin that we would like to remove, and then click on Uninstall and restart Jenkins to reflect the changes.

  IMAGE
- In a few cases, we would like to use an older version of a certain plugin. In such circumstances, we have to download the needed plugin from the desired site and then upload it onto Jenkins manually.
- If we have created our own plugins, we have to upload them to the site and help further grow the community base.

  IMAGE

4.3 Creating Junkins Builds

A build is called during the conversion of source code into the usable and runnable form. It allows compiling the code into an executable form.

The build tool handles the process of building. Builds are usually done when we reach a critical standpoint such as the integration of a feature or so on. Jenkins based on CI has a powerful feature that can automate the build process to happen at a particular time or event. This is called 'scheduled builds.'

Creating Scheduled Builds

Apply the below steps to schedule the builds at certain times and triggers.

1. In the “Build Triggers” section, check the “Build periodically” box.
2. In the text box field, enter the scheduling parameters such as date, day, and time. The general syntax is MINUTE (0-59), HOUR (0-23), DAY (1-31), MONTH (1-12), DAY OF THE WEEK (0-7).
   IMAGE

GIT Webhook

Let us now see how to build every time you would like to push the code in. To perform this, install ‘GitLab Hook Plugin’ by following the same procedure which you have done in the previous section. If a project is not created then start a new project and then scroll down to “Source code management” and add the URL of the Git repository, along with required credentials. Now, add the steps that you would like to perform during the build. Go to the Add built setup and add a build step. This step relies on what you would like to do and on the environment. The below screen displays the setup page.

IMAGE

Let us now add a webhook to the repository in GitLab. Apply the below steps.

1. Navigate to the instance, select the “cog” icon, and then choose the webhook.
2. Fill in the URL field with “http:///jenkins /gitlab/build_now/”.
3. Now click on the “Add webhook” button at the end of the page.
4. You can add the required features by checking the boxes as displayed on the below screen.
   IMAGE

5. Jenkins Pipelines

In this section, you will learn what is actually CI/CD, what is CI/CD Jenkin pipelines and how you can create CI/CD pipelines in more detail.

5.1 Continuous Integration and Continuous Delivery (CI/CD)

The CI/CD is a coding practice that allows the development team to make small changes and run version control checks. The main purpose of using CI is to automate the build process and test the applications. It enables frequent changes and better collaboration in the team to ensure high quality. CD is an extension of CI that pitches into the process where CI stops. CD ensures an automated way to push the code changes. It automates all the releases to the infrastructure defined. Alternatively, CD performs the required service calls to servers, databases, and others that may require to be restarted.

IMAGE

5.2 CI/CD Jenkins Pipeline

A CD pipeline is also known as “Deployment pipeline” or “software dev pipeline” is the process where the tasks and jobs of source code transformation into a releasable form are typically integrated into a pipeline where the completion of a task with a successful status would start off another automated process that is next in the sequence. The various parts of the pipeline such as managing, running, monitoring, and reporting are executed by a supervisory application. The real-world implementation varies based on the facts and parameters of source tracking, building, gathering metrics, and testing on the type of project.

The supervisory application manages its own job processes. Job creation performs functions of testing, building, and so on which you have stated before. All the jobs are automated, repeatable, and efficient. Once a job returns a successful status, the supervisory app triggers the next job or task in sequence. The automation can identify errors that occur in each step and fixes them early. This is known as "fail fast", a feature that finds the errors as early as possible and notifies the concerned teams. The benefit of this feature is that the application looks at process history and assigns the error to the respective teams to handle and resolve them.

5.3 Creating CI/CD Jenkins Pipeline

Let us now explore creating a Jenkins CI/CD pipeline from scratch. Jenkins can automate the entire DevOps process with the help of various interfaces and tools. The Dev team commits the code in the GIT repository. Jenkins defines the job or task with the help of a frontend tool. It then pulls the code and moves it to the commit phase. In the build phase, the code is compiled and finally, the code moves on to the staging area with the help of Docker to deploy it.

IMAGE

The following steps are implemented in creating the Jenkins pipeline with the help of Jenkins and Docker.

1. Open the terminal in the VM and run the following commands:
   systemctl start Jenkins
   systemctl enable Jenkins
   systemctl start Docker
2. Create a new job by opening Jenkins on the specified port and click on the new item.
3. Choose a “freestyle” option and enter the item name of your choice.
4. Select “Source code management” and provide the GIT repository. Then, click on “Apply” and “save”.
5. Go to the build and select the execute shell.
6. Provide the shell commands to generate a wat file. The code is then pulled up and installs the package, along with the dependencies, and compiles the application.
7. Repeat steps 3–5 and provide the shell commands. This will start integration and build in Docker.
8. Repeat steps 3-5 again with a different job name and provide the shell commands as before. This will check the Docker container file and will deploy it to the predefined port.
9. Configure the jobs by clicking on Job1 and then on "Post build actions" and "Build other projects", respectively.
10. Provide the project name and then save.
11. Repeat step 9 to configure Job2.
12. Create a pipeline view of the same, click on the plus (+) symbol and select the build pipeline view, and then provide a view name.
13. Select “Run”, that starts the CI/CD process in the system.
14. Once the build is finished, navigate to “localhost:8180/sample.text” to run the application.

6. Installing Jenkins on Windows Platform

The installation prerequisites of Jenkins have the hardware specifications that require a minimum of 256 MB of RAM and a minimum of 1 GB hard drive and software specifications require anyone of Java Development Kit (JDK) or Java Runtime Environment (JRE).

Apply the following steps to know how to download and install Jenkins.

1. Navigate to Jenkins
   The Downloads page displays multiple operating systems which include Docker, macOS, OpenBSD, OpenSUSE, Redhat, Ubuntu, and Windows, etc.
2. On the Downloads page, select the Operating System as Windows.
   The Jenkins zip folder is downloaded.

   IMAGE
3. Save the Jenkins zip folder in the location that you choose.
4. Navigate to the location where the Jenkins zip folder is downloaded and unzip it.
5. Double-click on “Jenkins.msi” as displayed on the below screen.

   IMAGE
   The Jenkins Setup screen is displayed.
6. On the Jenkins Setup screen, click the Next button.

7. Specify the destination folder where you choose the Jenkins to be installed. You can either choose the default location or select another by clicking the Change button and then click the "Next" button as shown below screen.

   IMAGE
8. Click the "Install" button to proceed with the installation.

   IMAGE
9. Click the "Finish" button after completing the installation.

7. Configuration Jenkins on Windows Platform

Configure Jenkins with the following steps as shown below.

1. Open a browser and type http://localhost:8080/
   Note: The default port of Jenkins is 8080. Accessing this URL (http://localhost:8080/) is a sign of the successful installation of Jenkins in your system.
2. Copy the initial Administrator Password from the server file system.
   Note: You can find the initial Administrator password under the Jenkins installation path. If you have chosen the default installation path, then you can find the password in the "C:/Program Files (x86)/Jenkins/secrets" file as shown on the below screen. In case, you have chosen a different path other than the default path, then you need to check that location for the password.

   IMAGE
3. Once the password is copied, paste it into the Administrator Password field in the "Getting Started" screen as shown below.
4. Click the "Continue" button.

   IMAGE
5. Configure the plugins by installing the suggested plugins. Click the "Install suggested plugins" button or click the "Select plugins install" button if you choose to select the plugins of your choice.

   IMAGE
6. The installation of plugins will now begin as shown in the below screenshot.

   IMAGE
7. After plugins are installed, the "Create First Admin User" screen is displayed. On this screen, enter the "User name" and "Password", reenter the password to confirm. Further, enter your "Full name" and "Email", and then click the "Save" and "Continue" buttons.
8. In this step, the default Jenkins URL is displayed Click the "Save" and "Finish" buttons.

   IMAGE

Once Jenkins is installed in your system, click the "Start using Jenkins" button.

IMAGE

Jenkins is up now. To create a new job, click the create new jobs link.

8. Jenkins User Management on Windows Platform

You can create new users, edit the existing user profile, and also delete the users.

Creating a new user

1. On the left panel of the Jenkins Dashboard, click "Manage Jenkins".

   IMAGE
   The Manage Jenkins page is displayed.
2. On the Manage Jenkins page, scroll down to the bottom, and then click "Manage Users".

   IMAGE
   The Users page is displayed.
3. On the Users page, click Create User.

   IMAGE
4. On the Create User page, enter the new username and password, confirm the password, enter the full name of the user, and email ID.
   A new user is created.

Editing user profile

1. On the Users page, click the "cog" icon corresponding to the specific user profile which you would like to edit.

   IMAGE
   The user profile page is displayed.
2. On the User profile page, make the desired changes.
   Note: For instance, you can change details such as user full name, add or change the description, change the email ID, notification URL, password, and time zone, etc.
3. Finally, click the "Save" button at the bottom.
   The user profile is changed.

Deleting user profile

1. On the Users page, click the "cog" icon corresponding to the specific user which you would like to delete.

   IMAGE
   The delete message is displayed.
2. On the delete message, click the "Yes" button if you would like to delete the user.

9. Building pipeline on Windows Platform

Pipelines are a group of plugins or jobs that enables the process of continuous integration. Pipeline jobs are interlinked and also interdependent that are defined by using Jenkins File which enables you to create pipelines for all the branches and also enables you to review code in the Jenkins pipeline. The benefits of the Jenkins pipelines comprise creating multiple automation jobs, automatic resumption of a pipeline during an unexpected server restart, provision to support large projects, facility to pause the pipeline process, provision to edit by multiple users.

Installing Build Pipeline Plugin

1. On the left panel, click "Manage Jenkins" that displays the Manage Jenkins page.
2. On the Manage Jenkins page, scroll down and click the "Manage Plugins" tab that displays the Manage Plugins page.
3. On the Manage Plugins page, click the Installed tab to check if the Build Pipeline plugin is already installed.

   IMAGE
4. If the Build Pipeline plugin is already not installed, click the Available tab, and then install the plugin.

Creating a pipeline plugin

1. On the Dashboard, click the + button next to All.

   IMAGE
   The view name field is displayed.
2. In the "View Name" field, enter a name for the pipeline.

   IMAGE
3. Next, select to build a pipeline view.
4. In the next screen, accept the default settings, and then click OK and Apply.

Conclusion **:**

We have now reached the end of the tutorial. Thus we have learned what Jenkins is, the role of CI/CD, setting up Jenkins and creating masters and slaves, creating builds and scheduling them as per needs, creating CI/CD pipelines, etc. To gain more practical insights, it is now necessary to implement them by creating your own jobs and scheduling them with various parameters.