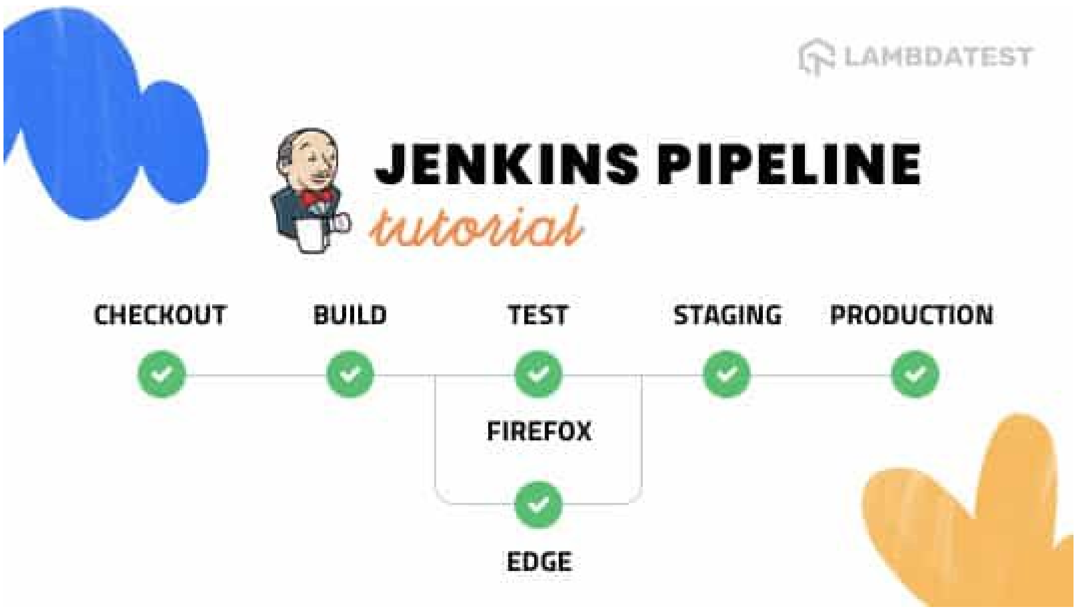
Ritesh Shetty Follow

Sep 22 · 19 min read · Listen

Best Jenkins Pipeline Tutorial For Beginners [Examples]



Jenkins CI/CD has always been the goto option for DevOps professionals and beginners. It has more than 16,000 stars on GitHub and 6,500 forks. Being one of the oldest players in the CI/CD market, Jenkins has huge community support with more than 1500 plugins to help professionals ship faster through their Jenkins pipeline.

Whether you are just starting off your CI/CD journey or are looking for a quick recap over Jenkins Pipeline, then you're at the right place!

This Jenkins pipeline tutorial will provide all the information that you need to set up a Jenkins pipeline and a detailed explanation of the underlying concepts. You will also learn how to perform automation testing using Selenium in Jenkins Pipeline through an online Selenium Grid.

you are already familiar with it, feel free to skip this section.

What Is CI (Continuous Integration)?

Continuous Integration is a development approach in which members work on the same project coordinate to integrate their work more frequently. The code is integrated into a shared repository, and any integration is tested by automated test cases or sequences to look for an error. Each team member is expected to integrate their code at least once a day or more as and when required.

CI's definition was first developed almost two decades ago to prevent "integrating chaos," which occurs when integration is postponed to the end of production. Once the code is committed, and the software build process is successfully completed, the CI tool immediately tests the code. If the test is passed, the CI tool deploys the code and pushes it into the production process. This entire process of committing, building, testing, and deploying goes on continuously in software development, and thus it is known as "Continuous Integration."

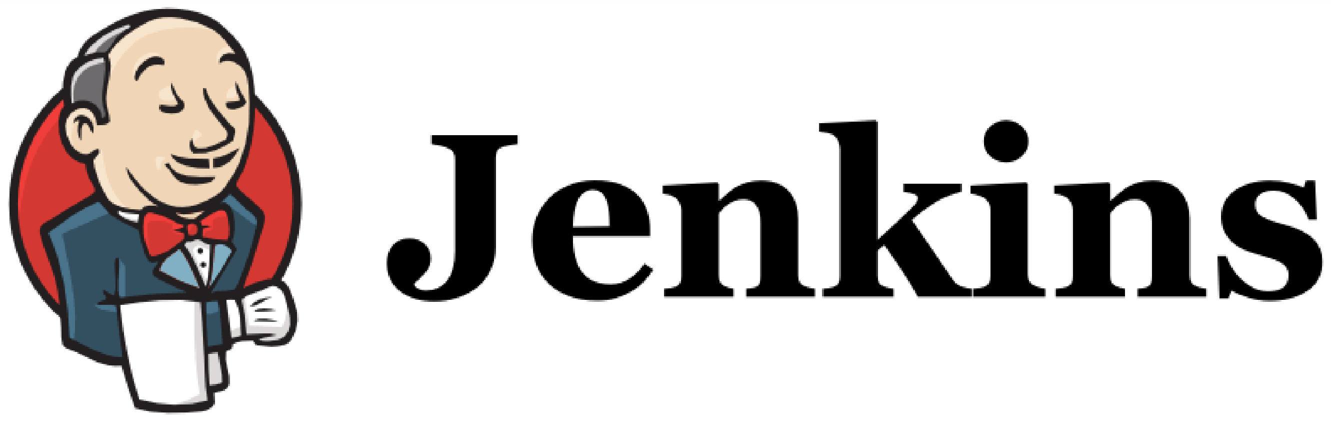
The CI process consists of four key elements which help in the execution:

- Hosted CI Tool Solution
- Virtual Machine
- Version Control System
- Tools

It was always a major challenge for firms to set up, configure, and manage a constant workflow. Owing to rapid changes over the last few years, software development has taken a big step forward ever since the introduction of Continuous Integration & Continuous Delivery (CI/CD). Now, that you understand what role CI plays in software development, we can move on to this Jenkins pipeline tutorial's central objective.

What Is Jenkins?

Get unlimited access Open in app



Jenkins CI/CD is a continuous integration open-source tool. It has been developed using Java. This allows real-time monitoring and recording of discrete improvements to a more comprehensive codebase. It lets developers easily identify, fix bugs in their codebase, and simplify their builds' validation. Jenkins is the most popular CI/CD tool because it closely monitors repetitive jobs and assists in automated execution during a project's production.

Jenkins CI/CD is a cloud-based system that can work as a standalone application running on a server of its own. Alternatively, it can also utilize a web server like Glassfish, JBoss, or WebSphere. Using Jenkins, developers can speed their product creation cycle because Jenkins can simplify the build and test instantly. Jenkins CI/CD promotes the entire software development cycle (SDLC) by designing, testing, monitoring, deploying, and other phases of the cycle.

Advantages Of Jenkins CI/CD

- An open community operates Jenkins. They host public meetings each month and seek feedback from customers on the progress of the Jenkins initiative.
- Jenkins also embraces cloud-based design to facilitate cloud-based platforms.
- Jenkins offers over 1500+ extensions in its plugins folder. Through plugins, Jenkins CI/CD is much more efficient and mature in comparison to other tools.
- Jenkins CI/CD offers fantastic support for parallel test execution. This helps

Integrate Jenkins with LambdaTest and
Bring all Testing to Cloud
Get Started

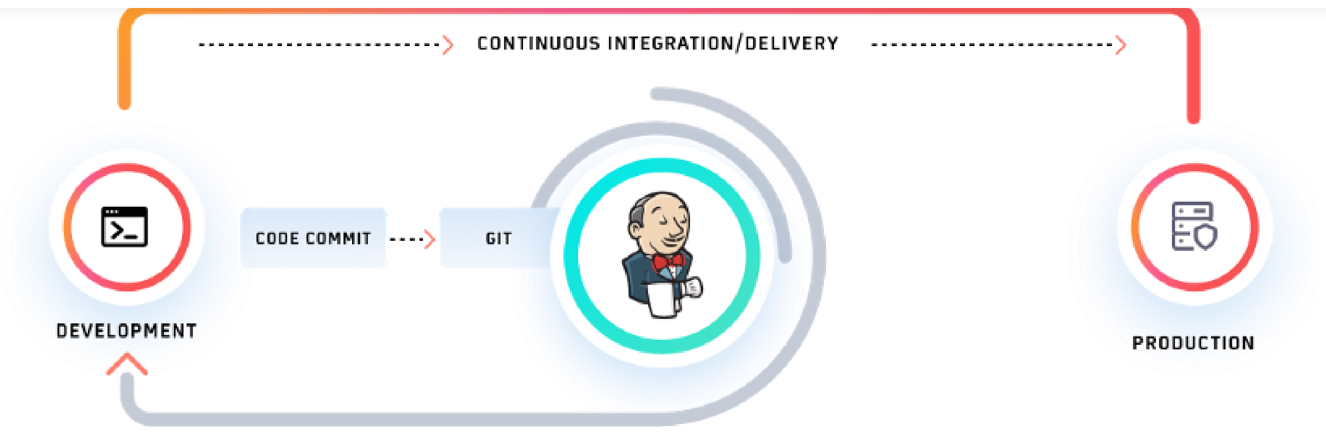
While Jenkins CI/CD offers multiple benefits for developers, it is not perfect. Some of the major pain points of using Jenkins are complicated installation & customization, outdated layout, and the need for some prior experience in project management.

Yet these points are not significant enough to stop developers from adopting Jenkins because it still offers few of the best features in the whole CI/CD universe. The next section of this Jenkins pipeline tutorial will explain why Jenkins CI/CD is still the first choice for developers worldwide.

For more information, you can refer to our previous article on What Is Jenkins?

This Jenkins Tutorial for beginners and professionals will help you learn how to use Jenkins, one of the most popular CI/CD tools used in DevOps.

What Is Jenkins Pipeline?



Inside Jenkins CI/CD, a pipeline is defined as a series of events or tasks which are interconnected in a particular order. In simple terms, Jenkins pipeline is a set of modules or plugins which enable the implementation and integration of Continuous Delivery pipelines within Jenkins.

The Jenkins pipeline has an expandable automation system for building basic or complicated 'template' distribution pipelines via the Domain-specific language (DSL) used in the pipeline. There are four states of Continuous Delivery in Jenkins pipeline-

- Build
- Deploy
- Test
- Release

We will discuss these states in detail in the next sections of this Jenkins pipeline tutorial. For now, let's talk more about why you need the Jenkins pipeline.

Why Use Jenkins Pipeline?

As explained above in this Jenkins pipeline tutorial, Jenkins CI/CD plays a significant role in delivering high-quality applications or products. We are now aware that Jenkins has proved to be a specialist in Continuous Integration, Continuous Testing, and Continuous Delivery. It uses a feature called Jenkins pipeline for Continuous Delivery, which is basically the ability to release apps

To dive deeper into this Jenkins pipeline tutorial, we need to understand the advantages offered by the pipeline of Jenkins CI/CD.

Advantages Of Jenkins Pipeline

The highlight of Jenkins pipeline is that it offers the feature to define the complete deployment flow-through configuration and code. It states that all the standard Jenkins jobs can be written manually as an entire script and can be managed with a version control system.

It's essentially following the discipline of 'pipeline as code.' Hence instead of creating multiple jobs for each process, it allows us to code the entire workflow and place it in a Jenkins file. Below are some of the reasons that one might consider before using the Jenkins pipeline for Jenkins test automation with Selenium.

- By using Groovy DSL (Domain Specific Language), it models easy to complex pipelines as code.
- The code is stored in the form of a text file called 'Jenkinsfile' that can be scanned into Source Code Management.
- It supports complex pipelines by adding conditional loops, forks, or joining operations and allowing parallel execution tasks.
- It improves user experience by integrating user feedback into the pipeline.
- It's resilient in terms of Jenkins' master unplanned restart.
- It can resume from checkpoints saved.
- It can incorporate multiple additional plugins and add-ins.

Next up, we learn about Jenkinsfile in this Jenkins pipeline tutorial.

What Is Jenkinsfile?

To move ahead with our Jenkins pipeline tutorial, we need to understand the role of Jenkinsfile. It is a text file that stores the whole process as code in our local

The Jenkinsfile is written using the Groovy Domain-Specific Language and can be generated using a text editor or the Jenkins instance configuration tab.

There are two different types in which the Jenkins pipeline can be constructed. These are the syntaxes-

- Declarative pipeline syntax
- Scripted pipeline syntax

The Declarative Pipelines is a relatively new feature that supports the concept of code pipeline. It enables the reading and writing of the pipeline code. This code is written within a Jenkinsfile, which can be tested into a tool such as Git for source control.

The **Scripted** pipeline is a typical method of code writing. The Jenkinsfile is written on the Jenkins user interface instance in this pipeline.

While both of these pipelines are Groovy-based, the scripted pipeline uses more strict Groovy-based syntaxes. This is because it was the first groovy foundation pipeline that was created for use. As this Groovy script was not usually suitable to all users, it introduced the declarative pipeline to provide a simpler and more flexible Groovy syntax. The declarative pipeline is defined within a 'pipeline' block, while the scripted pipeline is defined within a 'node' block.

If you are looking for automated testing tools, you can check out this blog: 30 Top Automation Testing Tools In 2022 https://www.lambdatest.com/blog/automation-testing-tools/

How To Install & Run Jenkins On Windows?

It's time to get to the best part of this Jenkins pipeline tutorial and start the whole set up process. First of all, we need to install Jenkins. Jenkins can be installed on either Windows, Unix, or macOS systems. In this Jenkins pipeline tutorial, we're just going to concentrate on Windows installation.

Hardware Specifications:

- You require a total of 512 MB of RAM to run Jenkins on your device or laptop.
- You require at least 1 GB of storage on your hard drive.

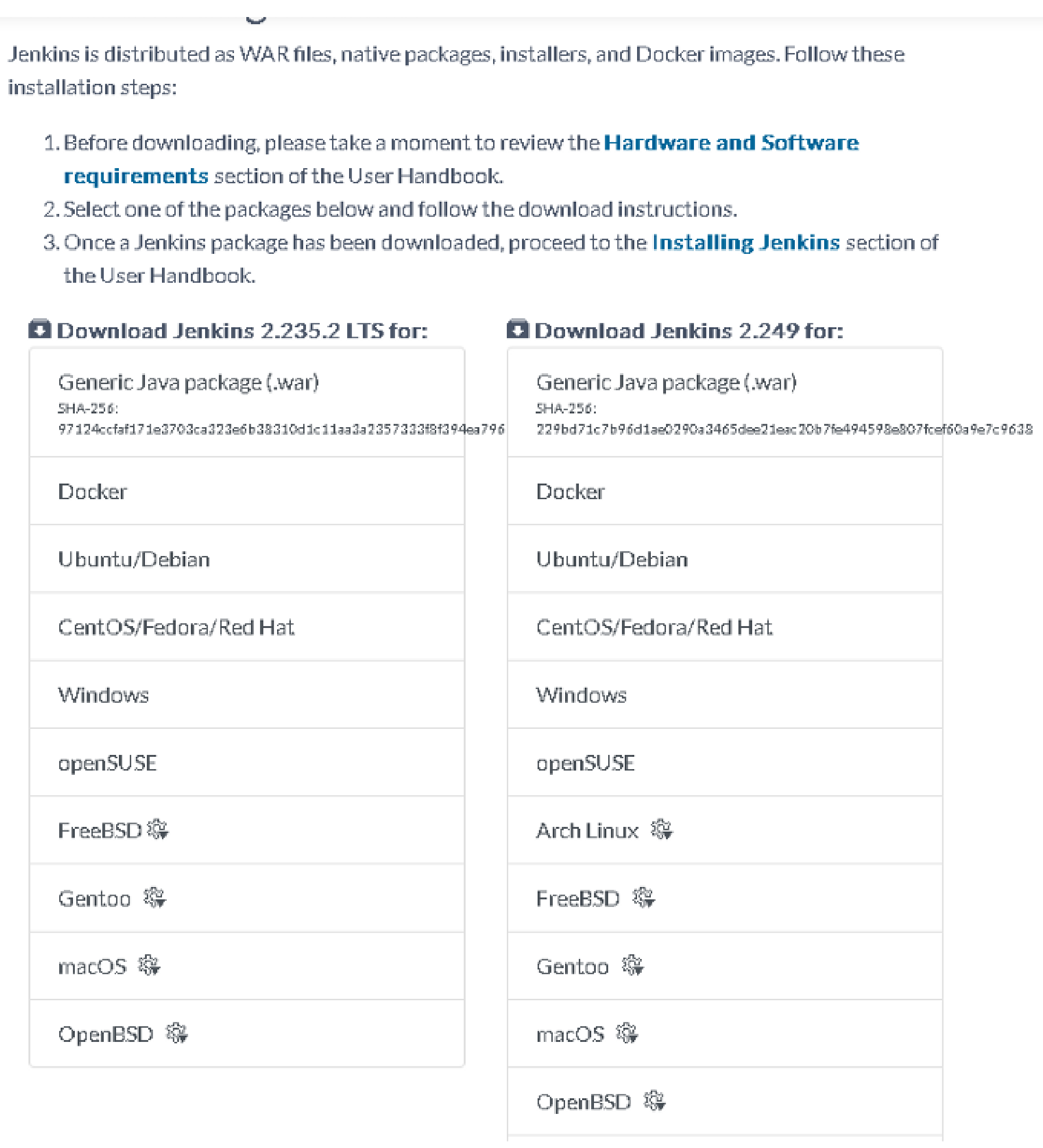
Software Specifications:

- Because Jenkins is based on Java, you need either the latest Java Development Kit (JDK) or Java Runtime Environment (JRE) versions based on your system configuration 32-bit or 64-bit.

Read — How To Set Jenkins Pipeline Environment Variables?

Steps For Installation

Step 1: Once all the prerequisites are met, download Jenkins from this web address https://jenkins.io/download/



Jenkins is distributed as WAR files, native packages, installers, and Docker images. Follow these installation steps:

1. Before downloading, please take a moment to review the **Hardware and Software requirements** section of the User Handbook.
2. Select one of the packages below and follow the download instructions.
3. Once a Jenkins package has been downloaded, proceed to the **Installing Jenkins** section of the User Handbook.

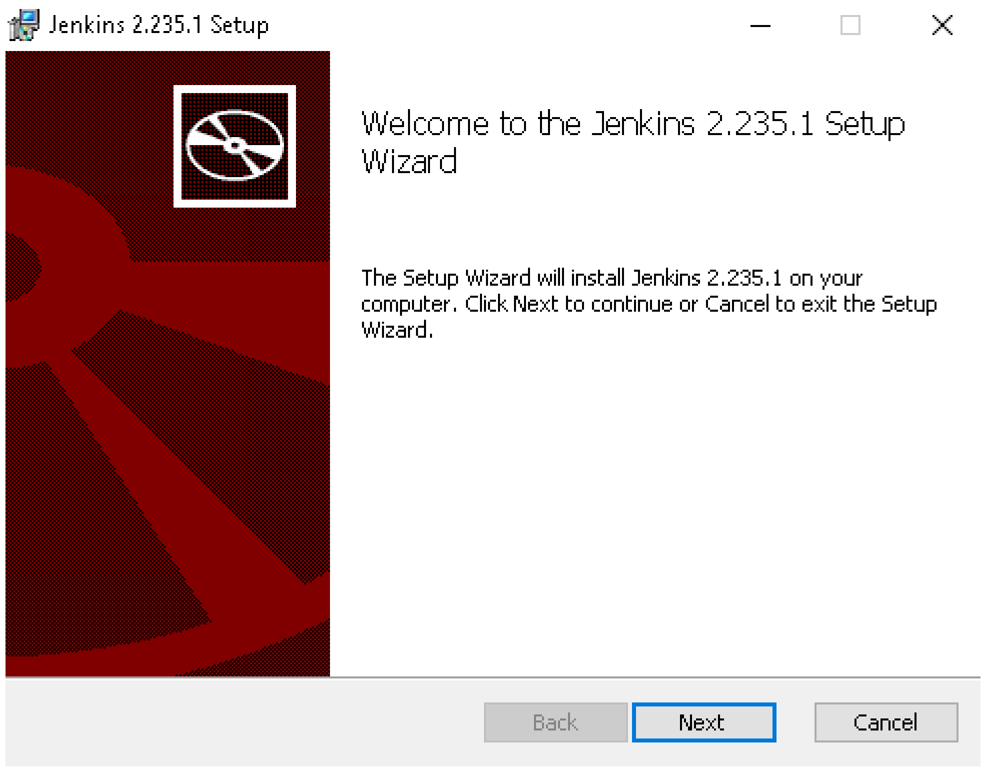
Download Jenkins 2.235.2 LTS for:	Download Jenkins 2.249 for:
Generic Java package (.war) SHA-256: 97124ccfaf171e3703ca323e6b38310d1c11aa3a2357333f8f394ea796	Generic Java package (.war) SHA-256: 229bd71c7b96d1ae0290a3465dee21eac20b7fe494598e807fcef60a9e7c9638
Docker	Docker
Ubuntu/Debian	Ubuntu/Debian
CentOS/Fedora/Red Hat	CentOS/Fedora/Red Hat
Windows	Windows
openSUSE	openSUSE
FreeBSD	Arch Linux
Gentoo	FreeBSD
macOS	Gentoo
OpenBSD	macOS
	OpenBSD

Step 2: Once you open the Jenkins download page, you will be prompted to select the platform you intend to download Jenkins. Click on Windows (or other options based on your system). For this Jenkins pipeline tutorial, we will be choosing Windows.

jenkins-2.235.1.zip

http://mirror.serverion.com/jenkins/windows-stable/jenkins-2.235.1.zip

Unzip the downloaded kit. Double-click the unzipped jenkins.msi file. You may also use the .war file to install Jenkins as a Java Package. This is recommended if you are focused on Jenkins test automation with Selenium in non-headless mode.

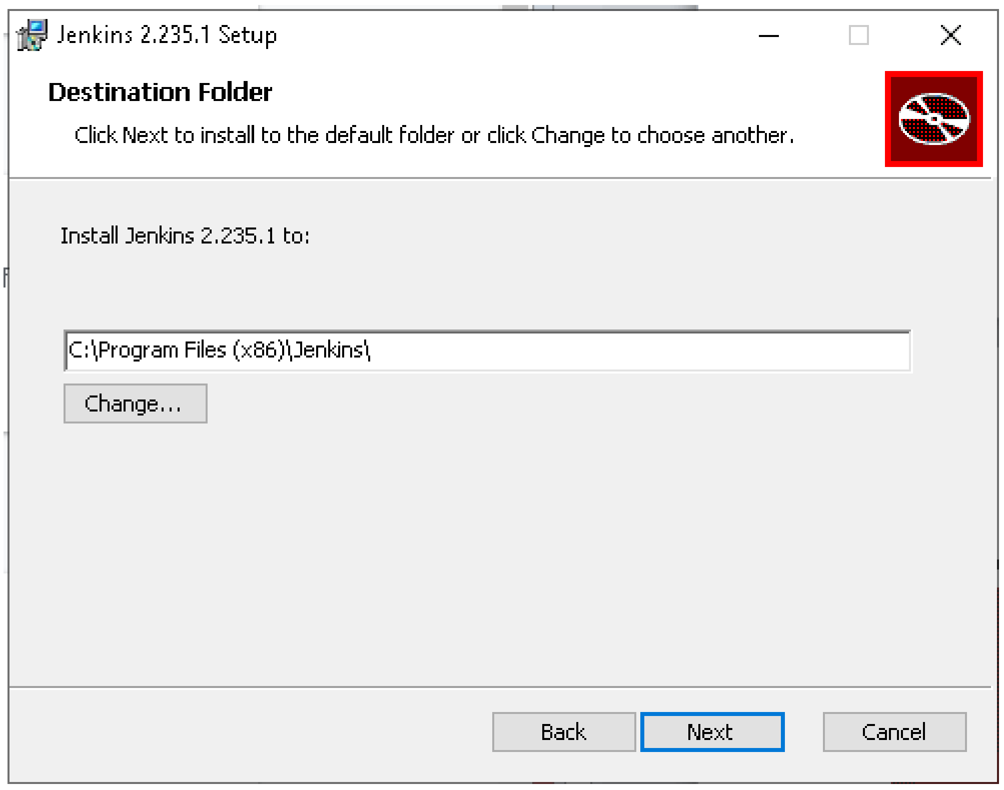


Jenkins 2.235.1 Setup

Welcome to the Jenkins 2.235.1 Setup Wizard

The Setup Wizard will install Jenkins 2.235.1 on your computer. Click Next to continue or Cancel to exit the Setup Wizard.

Step 4: Select the directory where you'd like Jenkins version to be installed (the default setting is C:\Program Files (x86)\Jenkins), then press the '**Next**' tab.



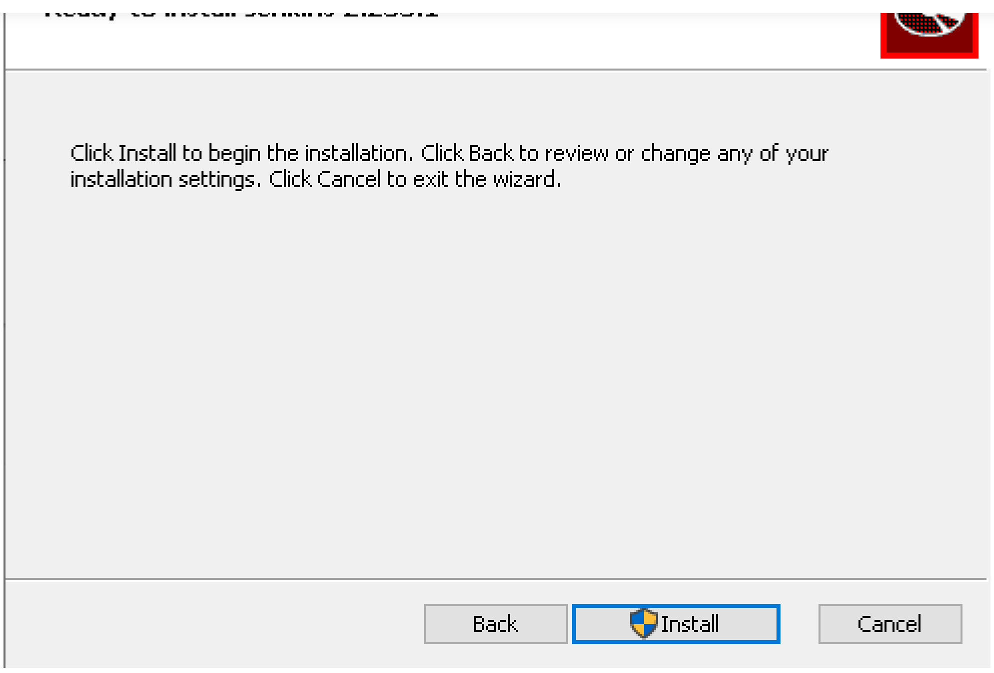
Jenkins 2.235.1 Setup

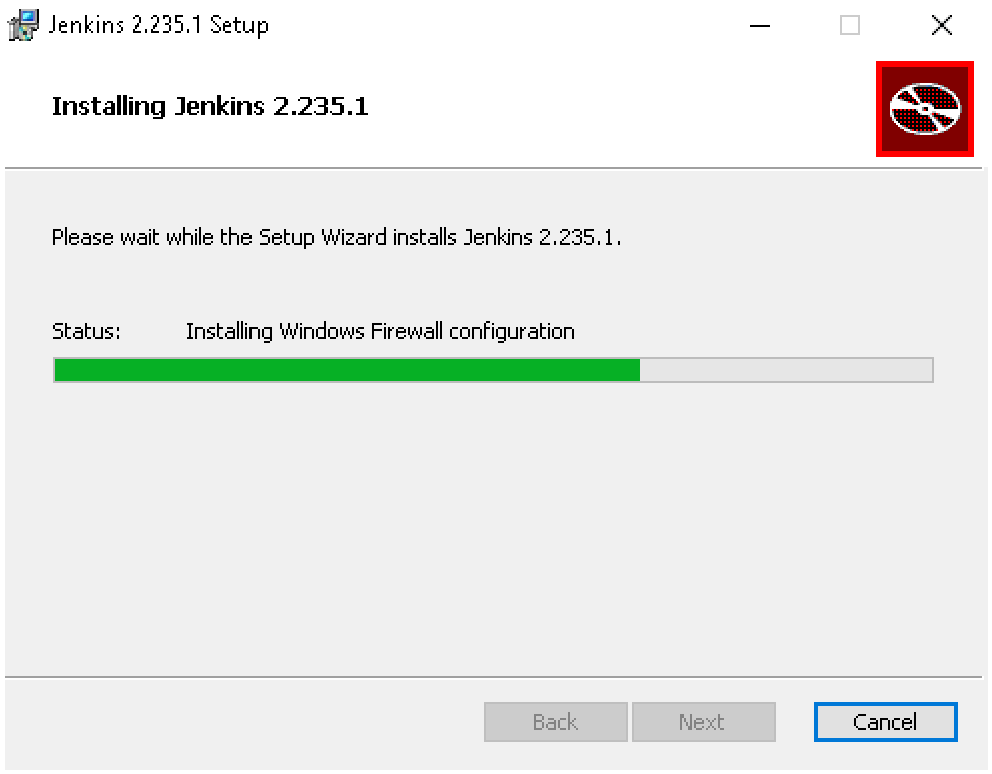
Destination Folder

Click Next to install to the default folder or click Change to choose another.

Install Jenkins 2.235.1 to:

C:\Program Files (x86)\Jenkins\

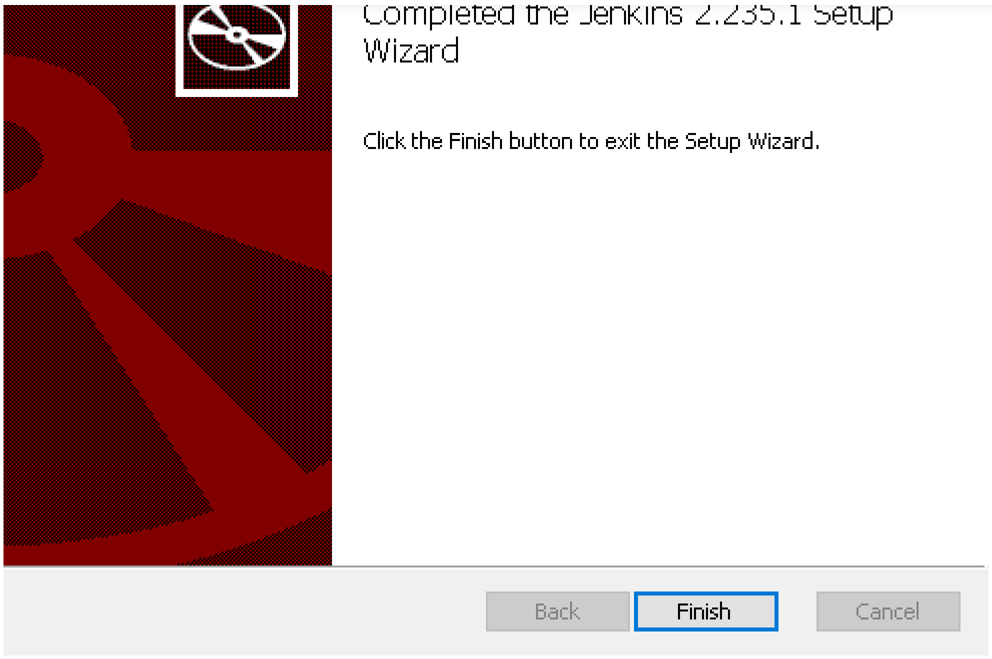

Click Install to begin the installation. Click Back to review or change any of your installation settings. Click Cancel to exit the wizard.


Jenkins 2.235.1 Setup

Installing Jenkins 2.235.1

Please wait while the Setup Wizard installs Jenkins 2.235.1.

Status: Installing Windows Firewall configuration


Completed the Jenkins 2.235.1 Setup Wizard

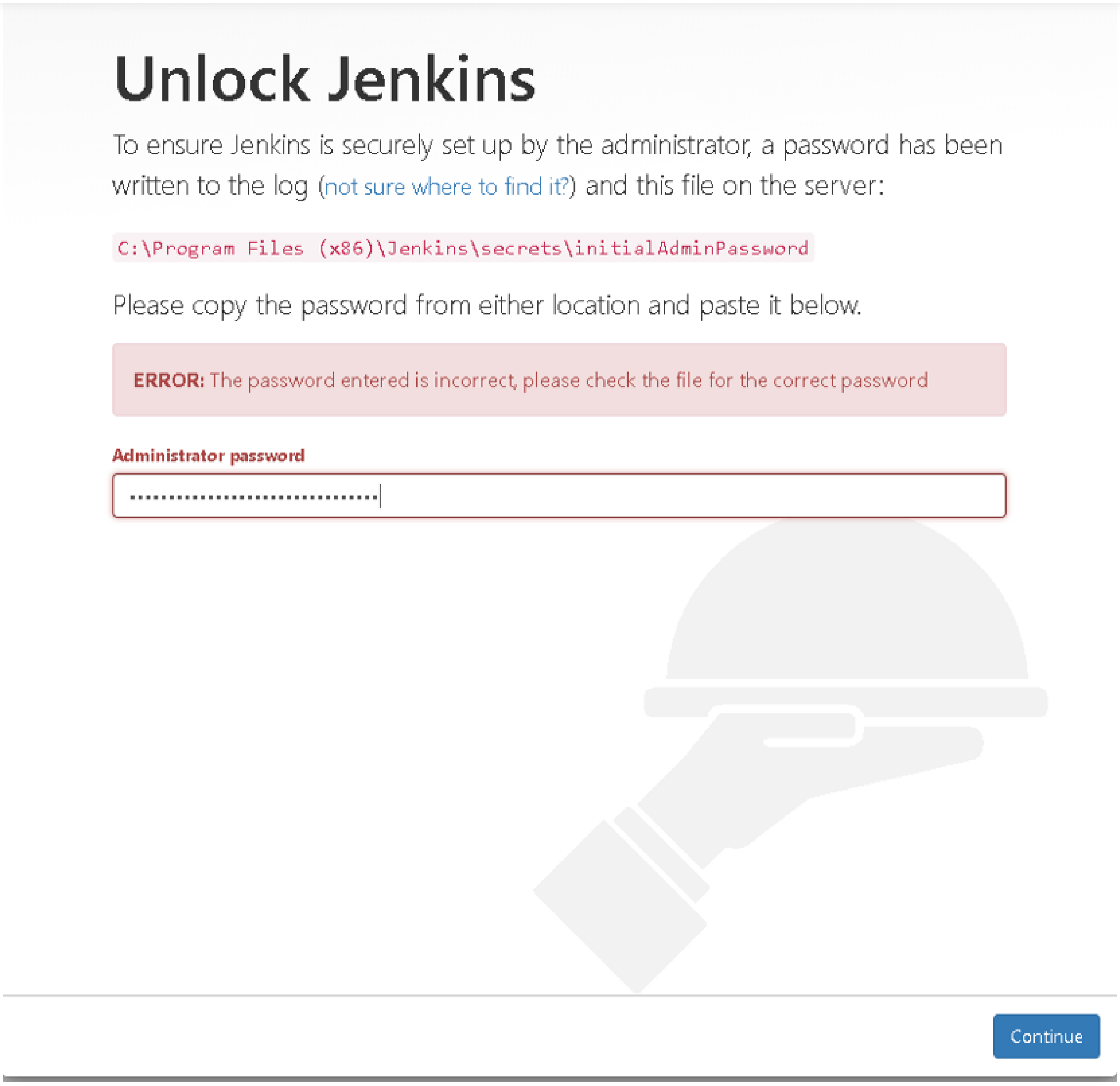
Click the Finish button to exit the Setup Wizard.

Step 6: Upon finishing the Jenkins setup process, you can continue further and start the setup. The next steps should instruct you on how to unblock the Jenkins submission. Upon completion of the Jenkins installation phase, a window tab will show up requesting for the Administrator's initial password.

Step 7: You have to proceed to the below path in your web browser to access Jenkins.

localhost:8080

Once you can navigate the URL above, it will indicate that Jenkins has been properly installed on the system.



Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log (not sure where to find it?) and this file on the server:

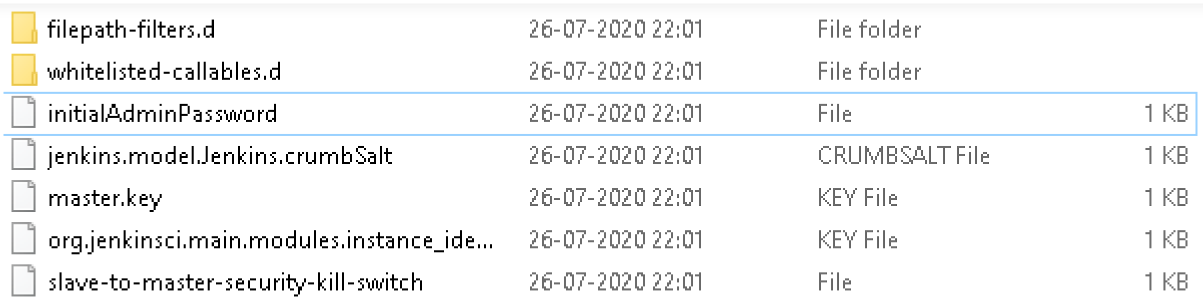
```
C:\Program Files (x86)\Jenkins\secrets\initialAdminPassword
```

Please copy the password from either location and paste it below.

ERROR: The password entered is incorrect, please check the file for the correct password

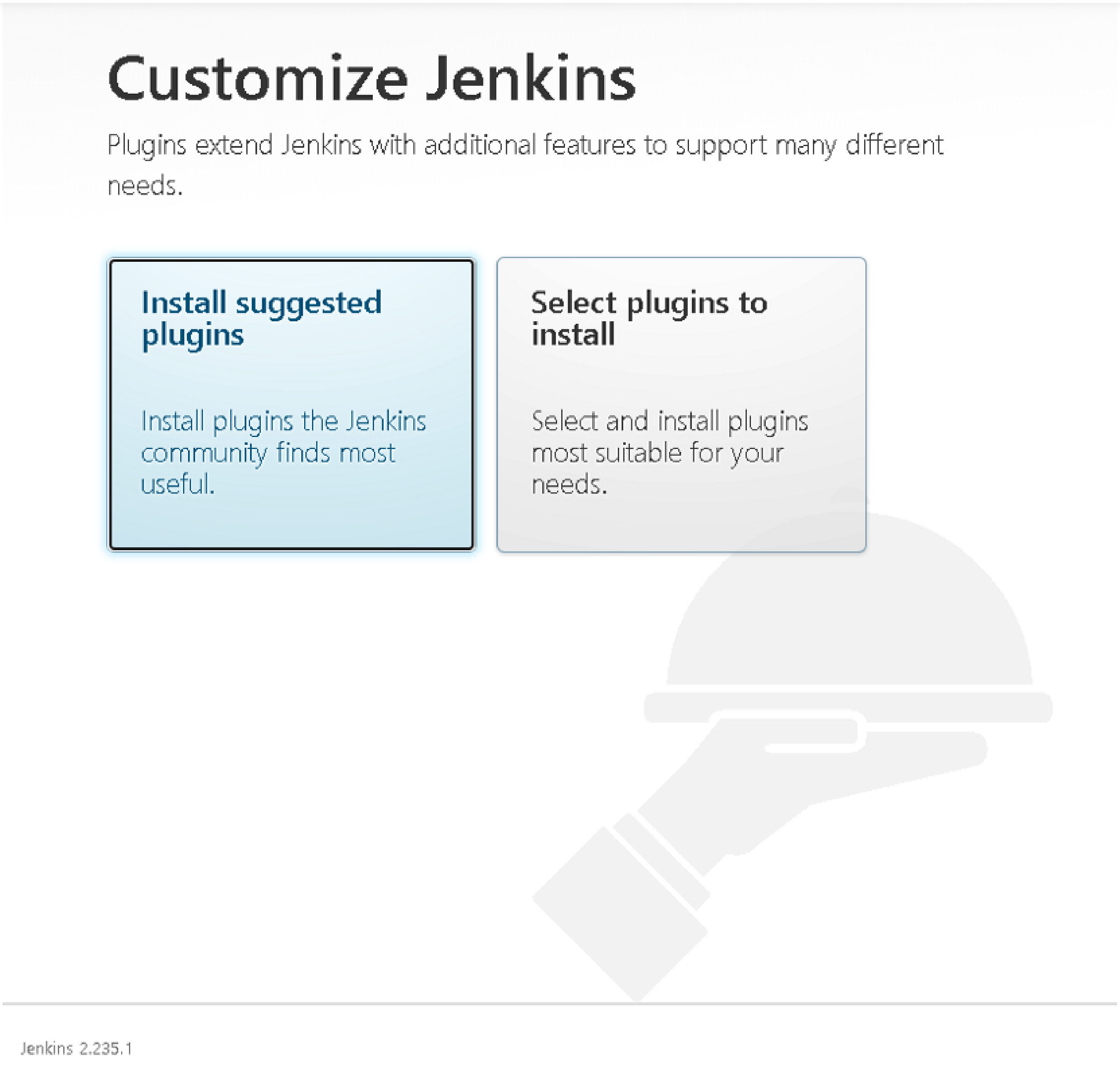
Administrator password

Step 8: The initial Administrator password can be located in the Jenkins setup path. A file named Initial Admin Password can be located at `installation-directory >\Jenkins\secrets\ initialAdminPassword`



filepath-filters.d	26-07-2020 22:01	File folder	
whitelisted-callables.d	26-07-2020 22:01	File folder	
initialAdminPassword	26-07-2020 22:01	File	1 KB
jenkins.model.Jenkins.crumbSalt	26-07-2020 22:01	CRUMBSALT File	1 KB
master.key	26-07-2020 22:01	KEY File	1 KB
org.jenkinsci.main.modules.instance_ide...	26-07-2020 22:01	KEY File	1 KB
slave-to-master-security-kill-switch	26-07-2020 22:01	File	1 KB

Step 9: Once you have entered the password on the Jenkins login page, you will now be prompted to choose the requisite plugins for your personalized project. You can select the '**Install suggested plugin**' for proceeding with the default list of plugins, or you can choose the plugins you want by clicking on '**Select plugins to install.**'



Customize Jenkins

Plugins extend Jenkins with additional features to support many different needs.

Install suggested plugins

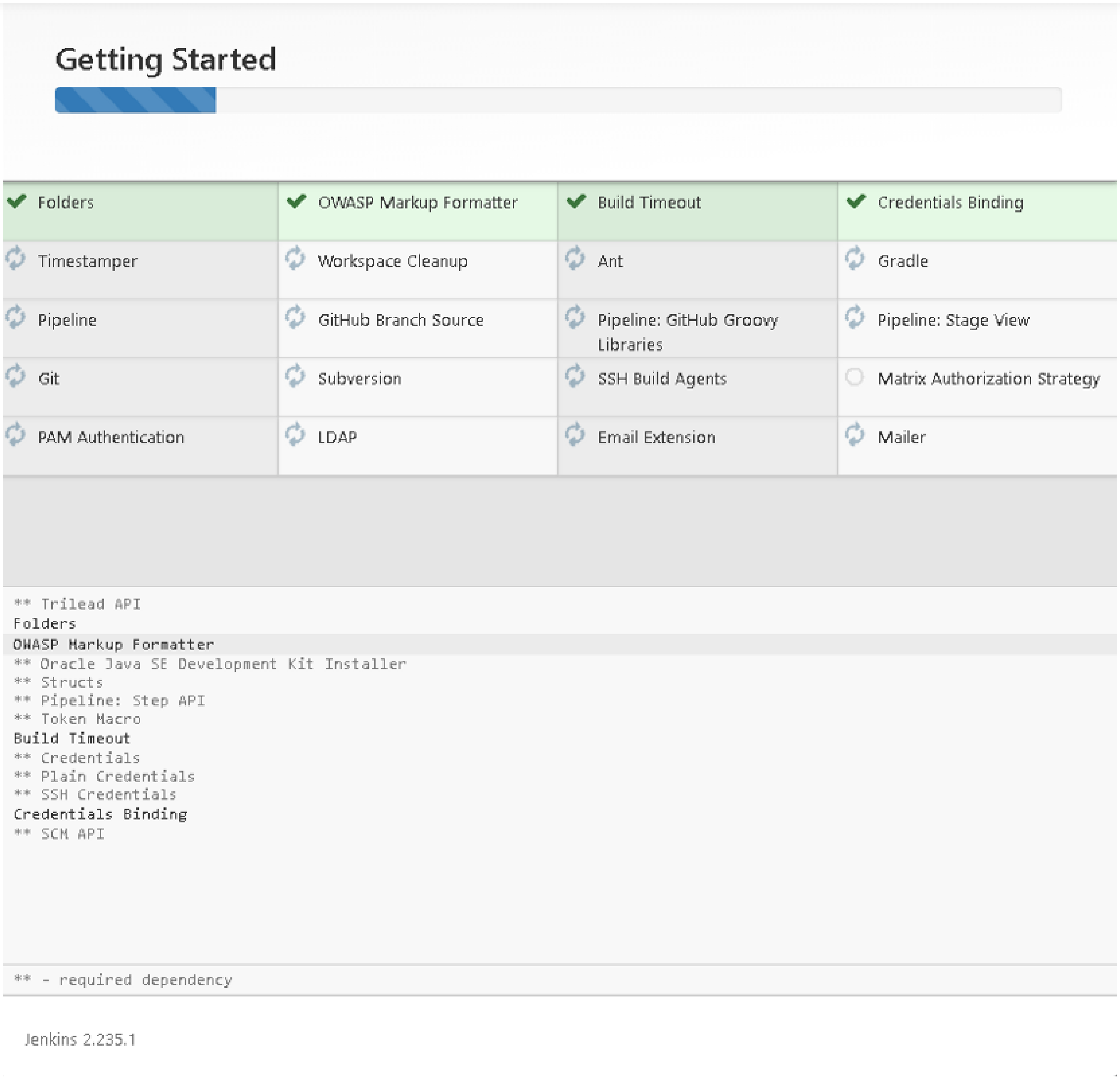
Install plugins the Jenkins community finds most useful.

Select plugins to install

Select and install plugins most suitable for your needs.

Jenkins 2.235.1

You will then see a **Getting Started page** that will install all the necessary plugins required to get you started with Jenkins CI/CD.



Getting Started

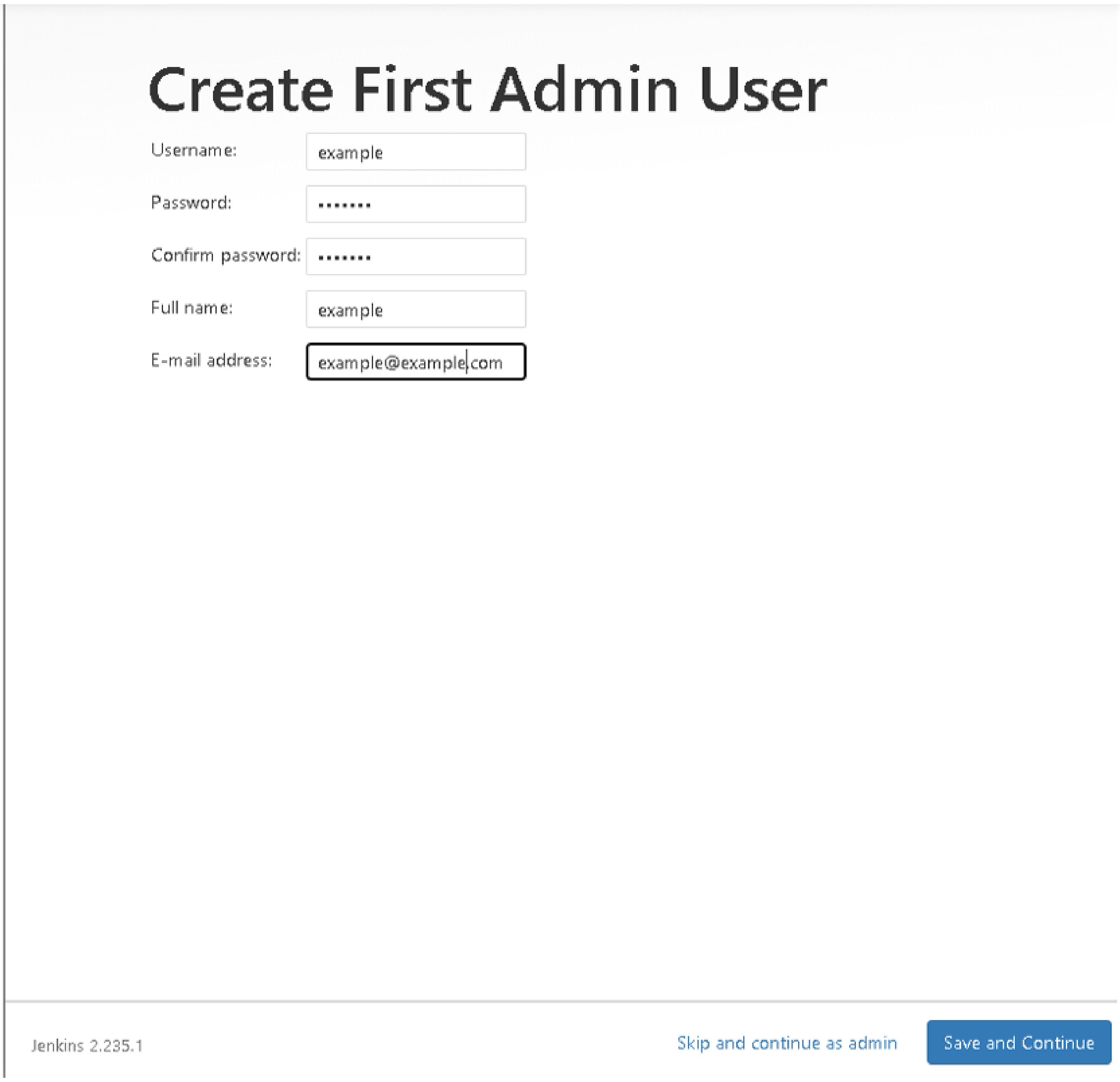
✓ Folders	✓ OWASP Markup Formatter	✓ Build Timeout	✓ Credentials Binding
Timestamper	Workspace Cleanup	Ant	Gradle
Pipeline	GitHub Branch Source	Pipeline: GitHub Groovy Libraries	Pipeline: Stage View
Git	Subversion	SSH Build Agents	Matrix Authorization Strategy
PAM Authentication	LDAP	Email Extension	Mailer

```
** Trilead API
Folders
OWASP Markup Formatter
** Oracle Java SE Development Kit Installer
** Structs
** Pipeline: Step API
** Token Macro
Build Timeout
** Credentials
** Plain Credentials
** SSH Credentials
Credentials Binding
** SCM API
```

```
** - required dependency
```

Jenkins 2.235.1

Step 10: Once all the recommended plugins have been downloaded, the '**Create First Admin User**' tab will appear. Fill all boxes with the account info you like and press the '**Save and Continue**' option.


Create First Admin User

Username:	example
Password:	•••••••
Confirm password:	•••••••
Full name:	example
E-mail address:	example@example.com

Jenkins 2.235.1

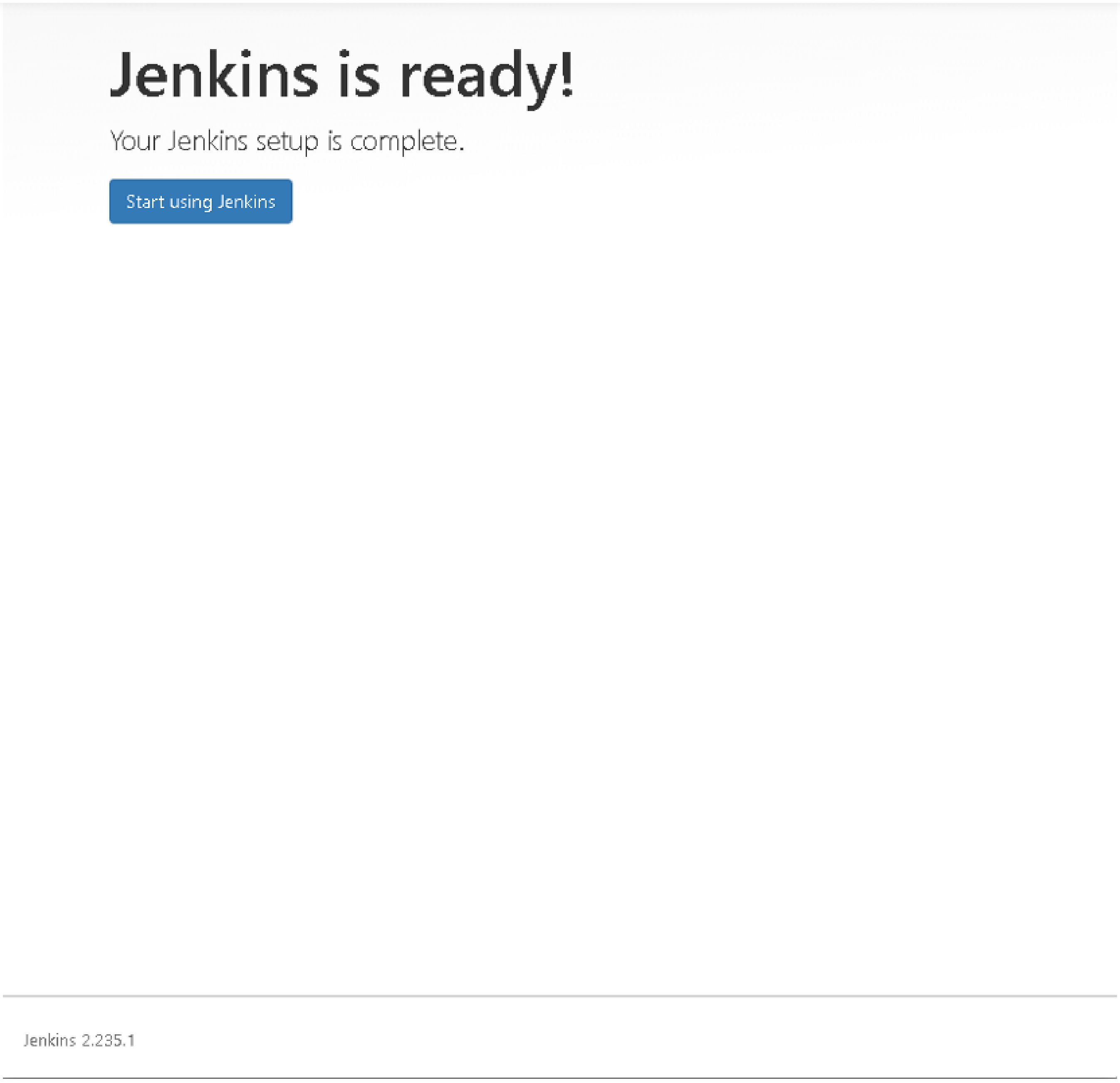
Step 11: Once this information is duly filled, a webpage may ask you for URL data. This URL is utilized to set the default configuration path for Jenkins. Keep it as it is to prevent any confusion later on. Nevertheless, if another program is still utilizing an 8080 port, you can either stop that program from using the port and make Jenkins' space and then save the settings.

Or you can change the port number, which we recommend in this Jenkins pipeline tutorial. To change the port, go to `< Jenkins-installation-folder >\Jenkins\jenkins.xml`

You might be prompted to have administrator rights for implementing this port change. Follow this-

```
<service>
  <id>Jenkins</id>
  <name>Jenkins</name>
  <description>This service runs Jenkins automation server.
</description>
  <env name="JENKINS_HOME" value="%BASE%"/>
  <!--
    if you'd like to run Jenkins with a specific version of Java
version, specify a full path to java.exe.
    The following value assumes that you have java in your PATH.
  -->
  <executable>%BASE%\jre\bin\java</executable>
  <arguments>-Xrs -Xmx256m -
Dhudson.lifecycle=hudson.lifecycle.WindowsServiceLifecycle -jar
"%BASE%\jenkins.war" --httpPort=9090 --webroot="%BASE%\war"
</arguments>
```

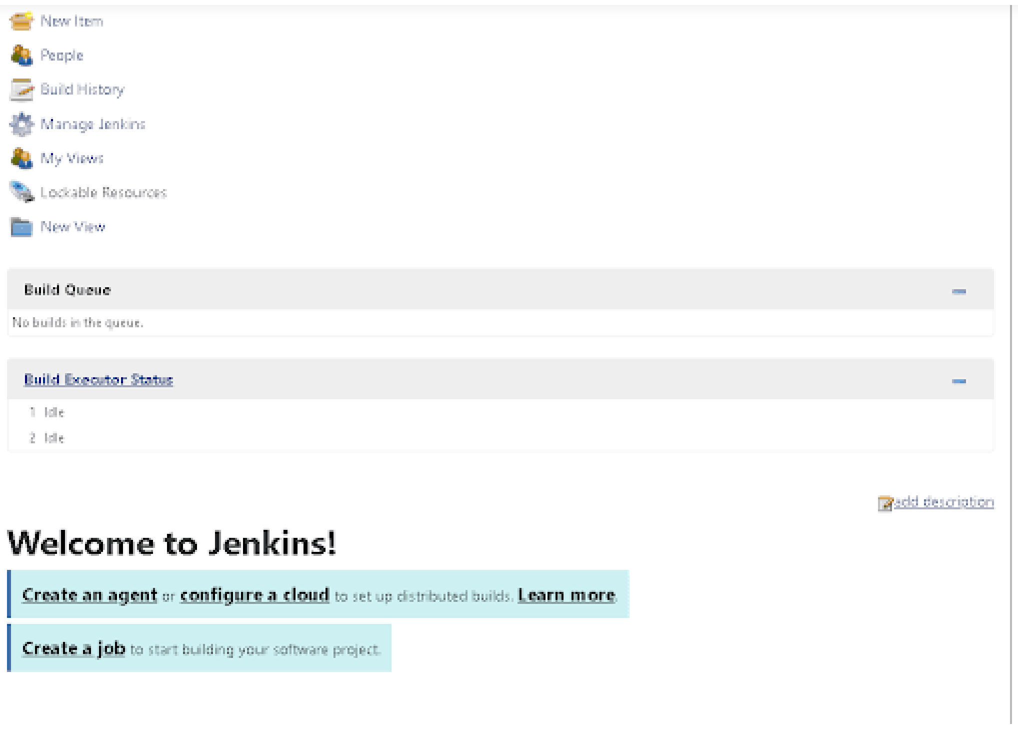
Once you do this, you're through with Jenkins deployment. After this step, the Jenkins account will be set up and is ready to use.



Step 12: Jenkins is enabled, and we're almost finished. Before we switch to build as per this Jenkins pipeline tutorial, we will need to customize Jenkins to recognize other resources like Java, Maven, etc.

Click on '**Manage Jenkins**.' After this, select '**Global Tool Configuration**' and then select '**JDK Installation**.' Include the JDK version's name in the JDK Name line and access the Java route in the '**JAVA HOME**' portion.

Uncheck the '**Install Automatically**' radio option, because there might be a risk that Selenium may not help if the latest Java version is modified. Save the changes and



Check this out: A Complete End to End (E2E) Testing Tutorial- Comprehensive Guide With Examples and Best Practices https://www.lambdatest.com/learning-hub/end-to-end-testing

Jenkins Pipeline Tutorial Concepts

We hope that Jenkins is up and running in your Windows system, following the steps mentioned above in this Jenkins pipeline tutorial. Before moving ahead with Jenkins test automation with Selenium, it is time to get accustomed to some of this Jenkins pipeline tutorial's crucial concepts.

Here they are-

Pipeline

It is a user-defined framework that includes all the processes like create, check, deploy, etc. In a Jenkinsfile, it's a list of all the levels. All of the stages and steps within this block are described. This is the fundamental block to the syntax of a declarative pipeline.

```
pipeline {
```

A node is a system running a complete workflow. It’s an integral part of the syntax of the scripted pipeline.

```
node  {

}
```

Some standard sections are available to both declarative and scripted pipelines. These are:

Agent

An agent is described as a directive that can run multiple builds using just one Jenkins instance. This feature helps spread the workload to various agents and execute multiple projects with [illegible]nce. It instructs Jenkins to assign the builds to an executor.

A single agent may be defined for a whole Jenkins pipeline, or different agents may be assigned to execute each stage within a pipeline. Some of the most commonly used Agent parameters are:

1. Any

Runs the stage pipeline on any available agent.

2. None

This parameter is added to the root of the pipeline. It means that there is no global agent for the entire pipeline, and each stage must define its own agent.

3. Label

Performs on the labeled agent the pipeline/stage.

4. Docker

image can now be used to run multiple commands as an execution environment.

```
pipeline {
    agent {
        docker {
            image  'ubuntu'
                }
            }
        }
```

Stages

This section includes all of the work that needs to be completed. The work is defined in the form of stages. Within this Directive, there may be more than one level. Each stage executes a particular task.

```
pipeline {
agent any
    stages {
        stage ('Build') {

        }
        stage ('Test') {

        }
        stage ('QA') {

        }
        stage ('Deploy') {

        }
        stage ('Monitor') {

        }

    }
 }
```

Steps

```
pipeline {
    agent any
        stages {
            stage ('Build') {
                steps {
                    echo
                    'Running build phase. '
                }
            }
        }
    }
```

Subscribe to Coding Jag and get the best news around the testing world delivered to your inbox every Thursday morning.

Integrating Jenkins Pipeline With Cloud Selenium Grid

This Jenkins pipeline tutorial's point was not just to get you acquainted with Jenkins CI/CD or the pipeline itself. Yes, we wanted to help you set up a Jenkins pipeline but then you must be wondering what's the purpose of this entire setup. As pointed out earlier in this Jenkins pipeline tutorial, Jenkins pipeline is quite useful in executing Selenium. For implementing this, Jenkins provides various plugins available for integration with various tools and utilities while also monitoring productivity.

To get started with Jenkins test automation with Selenium, LambdaTest Jenkins plugin is one such plugin to help you speed up automated cross browser testing. With the LambdaTest Jenkins plugin, we can easily automate our test scripts by connecting the Jenkins CI instance to the LambdaTest Selenium grid.

LambdaTest Selenium grid offers us a great range of 3000 + browsers and browser versions to achieve higher test coverage while using the Selenium test suite to

- Configuring the credentials for LambdaTest and the job scheduled with Jenkins.
- Performing a setup and tearing down the binary LambdaTest Tunnel file to conduct automated cross browser testing even on your local web applications.
- Providing all test data, such as video logs, network logs, and screenshots, in each of the steps performed at various stages with your Jenkins job via LambdaTest.

For Jenkins test automation with Selenium, you would require access to the following-

- An account at LambdaTest. You can register here for LambdaTest Account.
- An existing server with the Jenkins CI (version 2.138.2+)
- A root-access Jenkins Client.

Check this out: Automated Testing Platform- Accelerate your release velocity with blazing fast test automation on cloud https://www.lambdatest.com/automation-testing

Configuring LambdaTest Jenkins Plugin

To move ahead with the most exciting section of this Jenkins pipeline tutorial, we would need to download the LambdaTest Jenkins plugin as explained below in order to run our Selenium tests on LambdaTest using Jenkins. Before proceeding with installing Jenkins, it is required to provide access to the administrator level.

Before moving ahead with the installation of the LambdaTest Jenkins plugin, make sure you have no active build jobs in execution or queue.

Step 1: First, we click on **Manage Jenkins** and then **Manage Plugins**.

Step 2: Then, we navigate and click on the **Available** tab.

Step 3: Search for LambdaTest in the filter search box on the tab.

Step 4: You will find a list of plugins on which proceed and select **LambdaTest**.

Step 6: After the successful installation of the plugin, you will be able to find the LambdaTest Jenkins plugin under your plugins installed.

Sometimes, your newly installed plugins fail to synchronize with the Jenkins plugins you have available. This is quite common. If you fail to find LambdaTest under your available plugins section, you can do this. Refresh the Jenkins plugin list by simply clicking the **Check Now** button, and you will get a list with new plugins from the Jenkins update center.

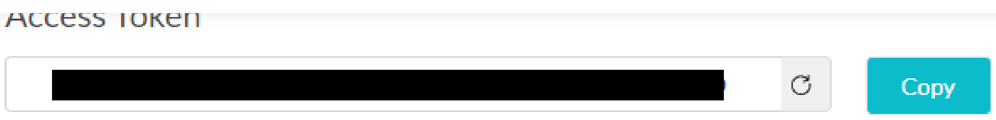
Once the LambdaTest Jenkins plugin installation is done, we would need to integrate our Selenium WebDriver tests with Jenkins by configuring the LambdaTest credentials in the Jenkins CI server. This step is essential if we want to move ahead with Jenkins test automation using Selenium or even to perform automated cross browser testing.

Check this out: Automation Testing Platform- Accelerate your release velocity with blazing fast test automation on cloud https://www.lambdatest.com/automation-testing

Configuring LambdaTest Credentials

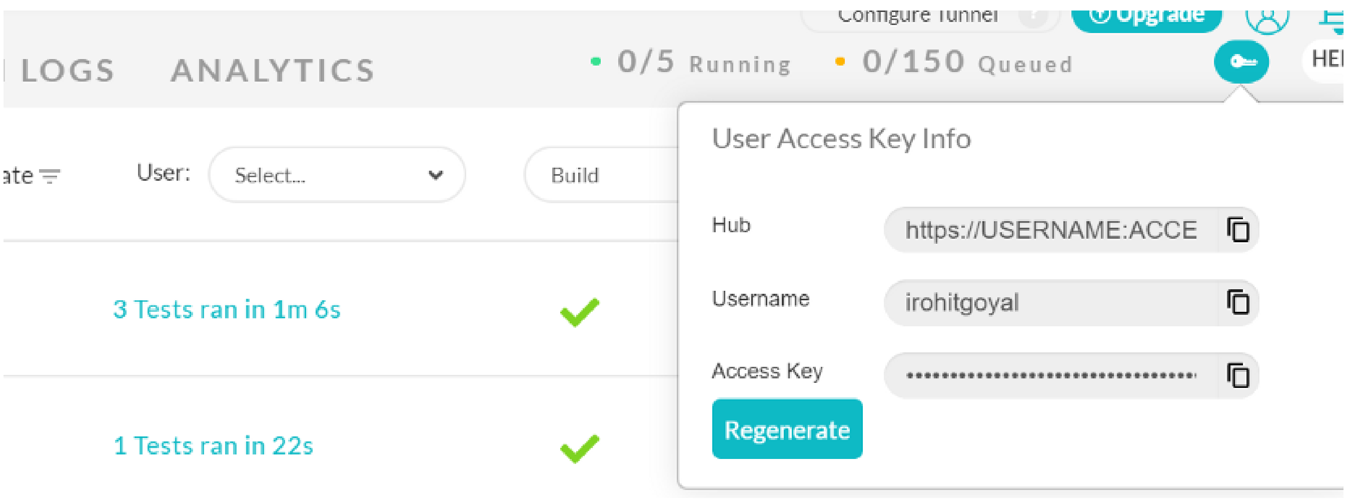
To proceed with this Jenkins pipeline tutorial configuration, we would need to work with the Jenkins UI. We don't have to prepare the script separately from scratch for executing the Lambda Test Platform test. We can reuse our local test script and just add some of the minor configuration changes that will enable us to connect to the cloud platform.

One of the most important and the most vital things for us to connect to the Lambda Test platform is to get our access key token. The access key tokens are a set of private keys that enable us to connect to the cloud platform and execute automated tests on the platform. This access key is unique for everyone and should not be shared with other users. You can fetch and regenerate it from the Profile section of your user account, as shown below.


Access Token

[redacted]

Alternatively, we can also fetch the access key, username, and hub details from the Automation dashboard. You'll need to click on the key-shaped icon near the Help button, as shown in the image screenshot below.



User Access Key Info

Hub: https://USERNAME:ACCE

Username: irohitgoyal

Access Key: ••••••••••••••••••••••••••••••••••

This can be set as below in the script-

```
$ export LT_USERNAME= {your lambdatest username}

$ export LT_ACCESS_KEY= {your lambdatest access_key}

$ set LT_USERNAME= {your lambdatest username}

$ set LT_ACCESS_KEY= {your lambdatest access_key}
```

Once we have done that, we need to follow the steps below in order to configure it in Jenkins-

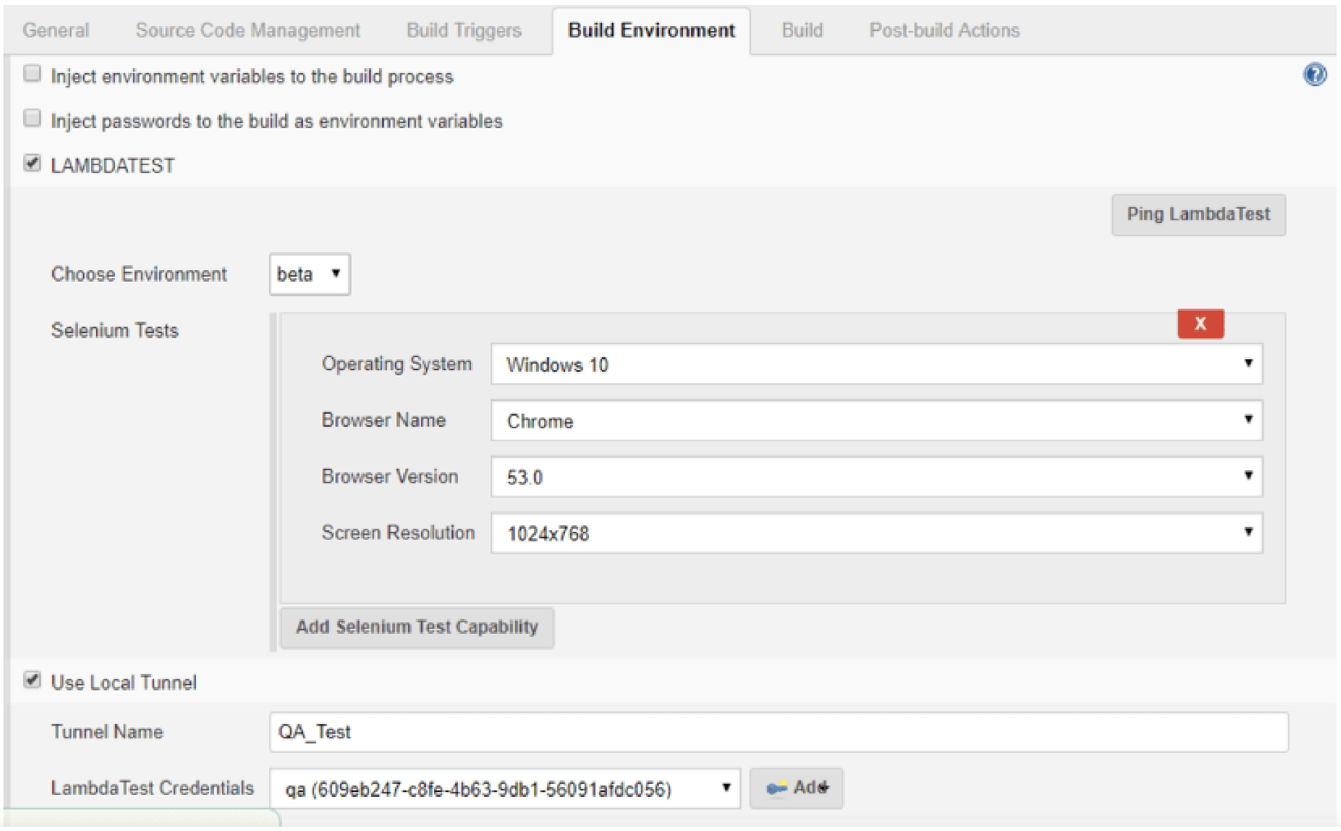
- Click Credentials on Jenkins Home Page.
- Select System under the Credentials tab to display the System page.

- Click on **Add Credentials** to open the credentials page where we add the access details.
- Enter your relevant data in all the fields and click **Verify Credentials**.
- Then click the **Ok** button after verification.
- Jenkins generates the ID that appears on the Credential page, and we save the changes.

After successfully attaching your credentials, Jenkins creates an ID. To get this ID for using LambdaTest credentials, you'd need to go to Jenkins's homepage and click on the **Credentials** on the navigation screen's left side.

From the home page of Jenkins, click on **Credentials** from the menu on the left. LambdaTest credentials can be copied from this ID.

Next, we would need to use the LambdaTest Jenkins plugin to enable LambdaTest Tunnel to perform cross browser testing on locally hosted web pages or web files.



☐ Inject environment variables to the build process
☐ Inject passwords to the build as environment variables
☑ LAMBDATEST

Choose Environment	beta
Operating System	Windows 10
Browser Name	Chrome
Browser Version	53.0
Screen Resolution	1024x768

☑ Use Local Tunnel

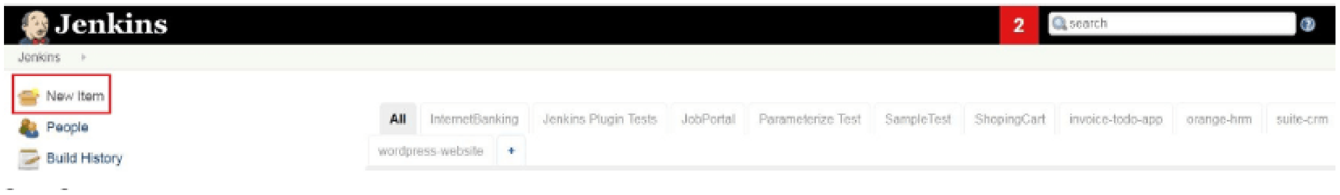
Tunnel Name	QA_Test
LambdaTest Credentials	qa (609eb247-c8fe-4b63-9db1-56091afdc056)

Creating A Jenkins Pipeline & Running Our First Test

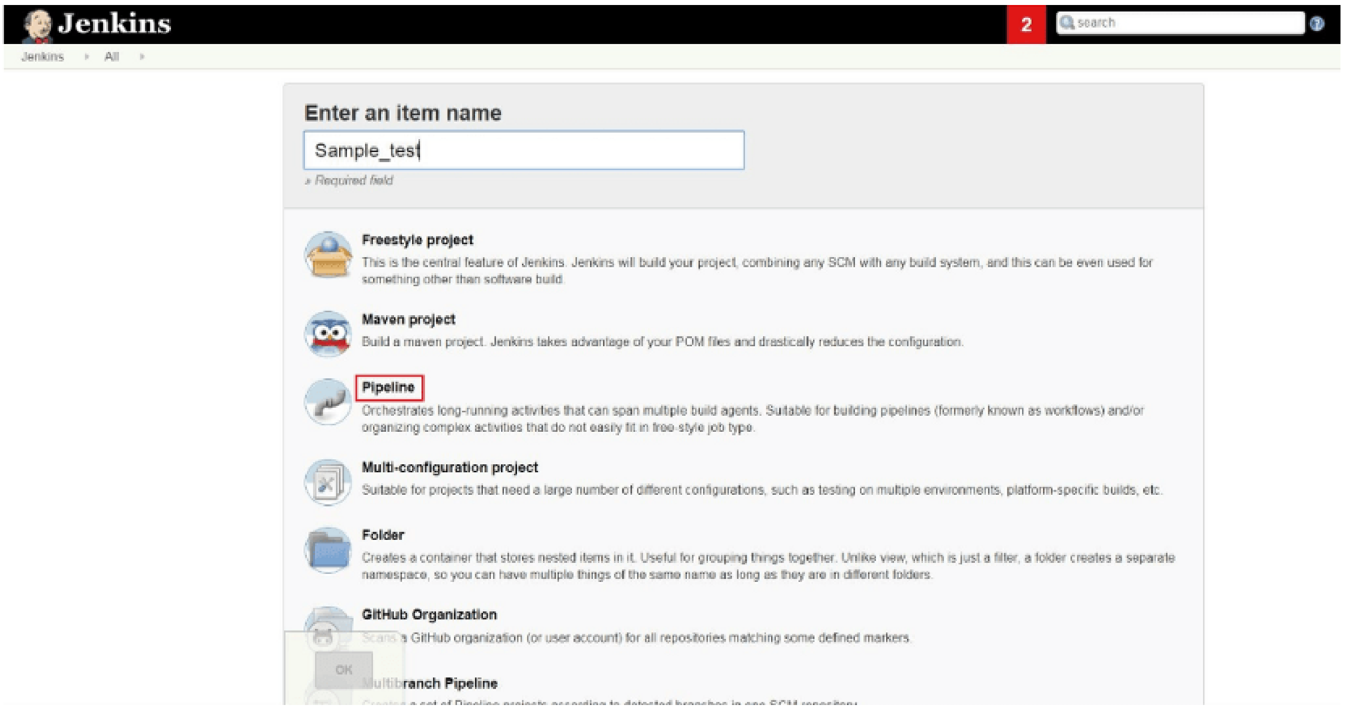
In the last section of this Jenkins pipeline tutorial, we will create a Jenkins CI/CD pipeline of our own and then run our first test. Below is the sample Jenkins File for the Pipeline, which has the required configuration details.

```
#!/usr/bin/env groovy

node {
    withEnv(["LT_USERNAME=Your LambdaTest UserName",
    "LT_ACCESS_KEY=Your LambdaTest Access Key",
    "LT_TUNNEL=true"]){

    echo env.LT_USERNAME
    echo env.LT_ACCESS_KEY

   stage('setup') {

      // Get some code from a GitHub repository
    try{
      git 'https://github.com/LambdaTest/nightwatch-selenium-
sample.git'

      //Download Tunnel Binary
      sh "wget https://s3.amazonaws.com/lambda-tunnel/LT_Linux.zip"

      //Required if unzip is not installed
      sh 'sudo apt-get install --no-act unzip'
      sh 'unzip -o LT_Linux.zip'

      //Starting Tunnel Process
      sh "./LT -user ${env.LT_USERNAME} -key ${env.LT_ACCESS_KEY} &"
      sh  "rm -rf LT_Linux.zip"
    }
    catch (err){
      echo err
   }

   }
   stage('build') {
      // Installing Dependencies
      sh 'npm install'
    }

   stage('test') {
   }
   stage('end') {
     echo "Success"
     }
 }
}
```

Step 1: We create a **New Project** for the Pipeline by navigating to Jenkins and then click on New Item.



Step 2: Next, we select **Pipeline** from the given list of options.



Enter an item name

Sample_test

» Required field

Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

Maven project
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

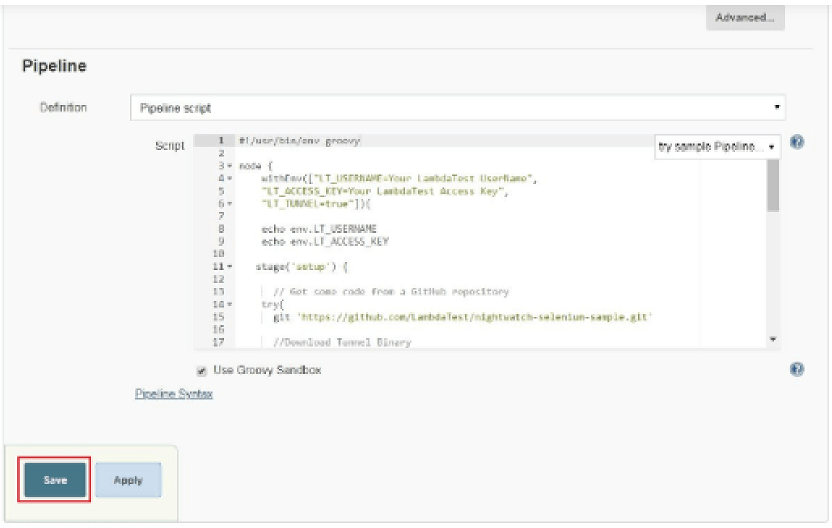
Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

GitHub Organization
Scans a GitHub organization (or user account) for all repositories matching some defined markers.

Multibranch Pipeline

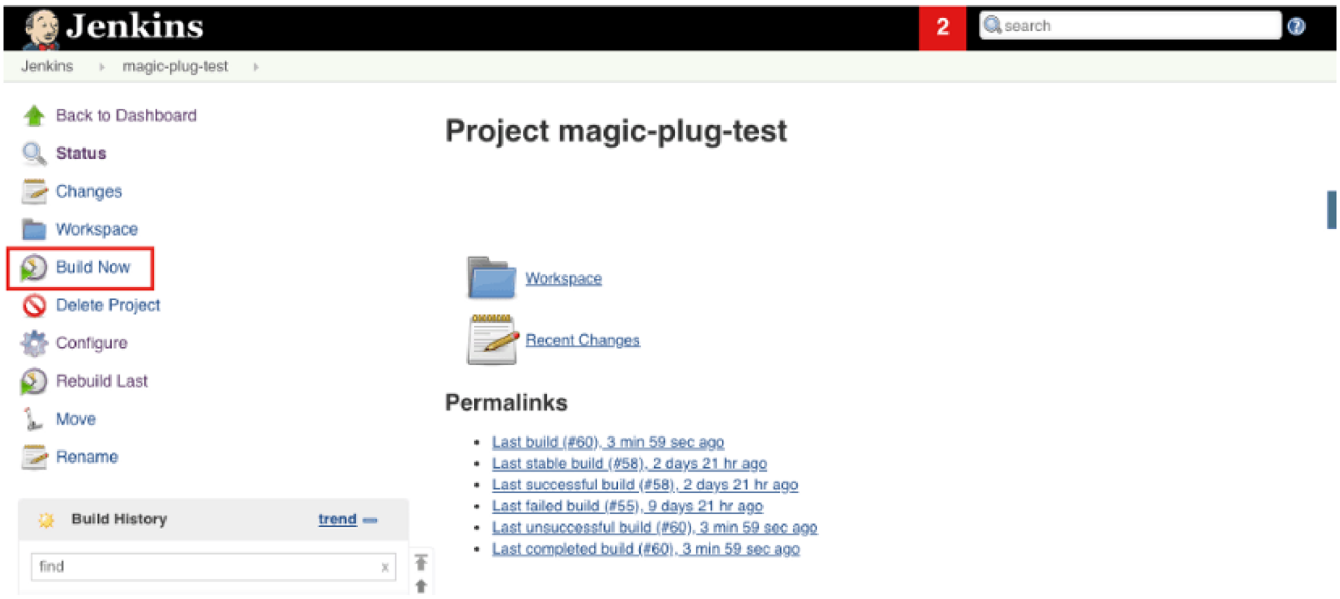
Step 3: Then, we will scroll down to **Advanced Project Options** and paste the pipeline script code that we saw above into the code pane and hit the **Save** button.



```
#!/usr/bin/env groovy

node {
    withEnv(["LT_USERNAME=Your LambdaTest UserName",
    "LT_ACCESS_KEY=Your LambdaTest Access Key",
    "LT_TUNNEL=true"]){

    echo env.LT_USERNAME
    echo env.LT_ACCESS_KEY

   stage('setup') {

     // Get some code from a GitHub repository
    try{
     git 'https://github.com/LambdaTest/nightwatch-selenium-sample.git'

     //Download Tunnel Binary
```

You’re now ready to run your first test using LambdaTest Jenkins plugin.

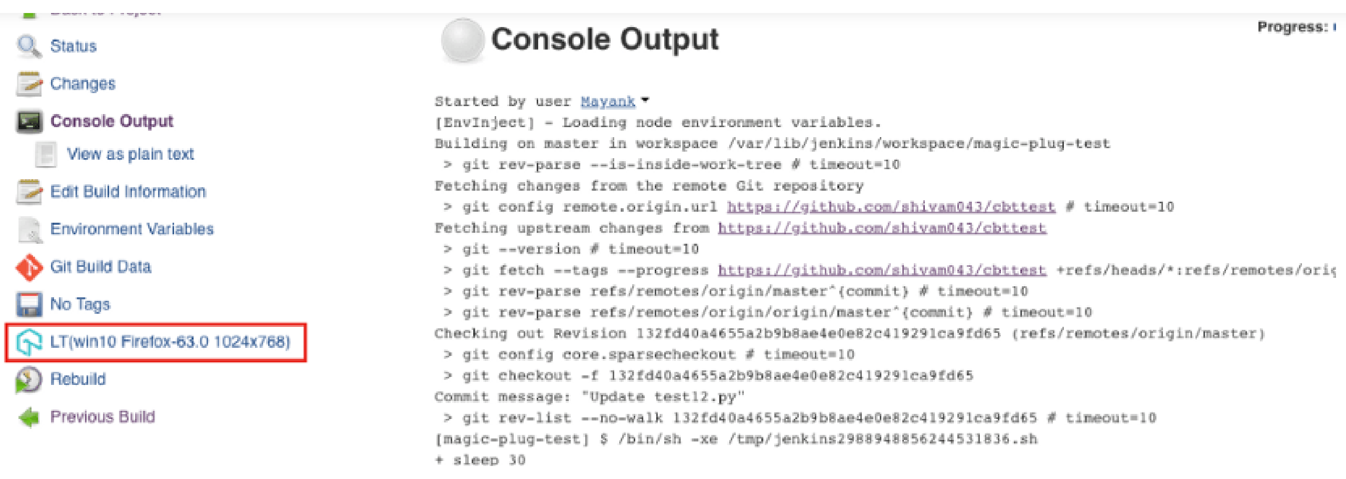
Step 4: We would need to set up a test build, which can be quickly done via the Jenkins UI by clicking **Build Now**.



- Last build (#60), 3 min 59 sec ago
- Last stable build (#58), 2 days 21 hr ago
- Last successful build (#58), 2 days 21 hr ago
- Last failed build (#55), 9 days 21 hr ago
- Last unsuccessful build (#60), 3 min 59 sec ago
- Last completed build (#60), 3 min 59 sec ago

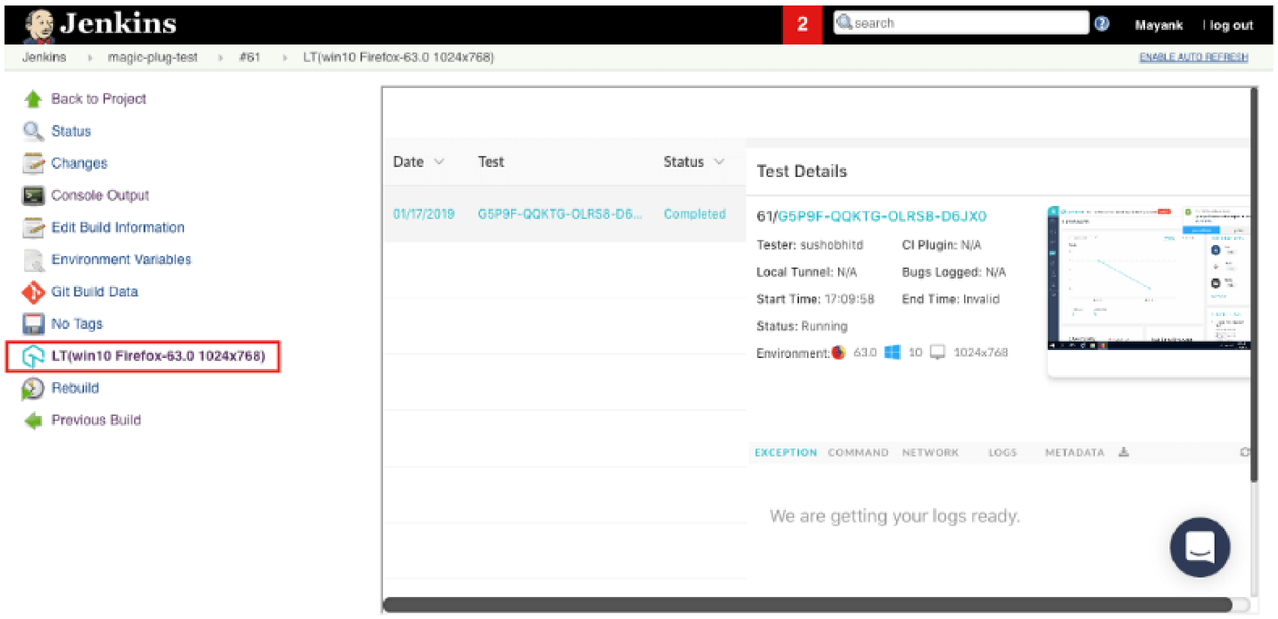
Step 5: We can see a new build generating under the build history as we click on **Build Now**. The console logs output can be viewed by clicking on the Build.

Clicking on **LambdaTest Icon** on the left will provide more details about the Selenium tests. This will also provide more insight into Jenkins test automation with Selenium, like the test sessions, video logs, etc. on the page loaded.



Console Output

```
Started by user Mayank
[EnvInject] - Loading node environment variables.
Building on master in workspace /var/lib/jenkins/workspace/magic-plug-test
 > git rev-parse --is-inside-work-tree # timeout=10
Fetching changes from the remote Git repository
 > git config remote.origin.url https://github.com/shivam043/cbttest # timeout=10
Fetching upstream changes from https://github.com/shivam043/cbttest
 > git --version # timeout=10
 > git fetch --tags --progress https://github.com/shivam043/cbttest +refs/heads/*:refs/remotes/ori
 > git rev-parse refs/remotes/origin/master^{commit} # timeout=10
 > git rev-parse refs/remotes/origin/origin/master^{commit} # timeout=10
Checking out Revision 132fd40a4655a2b9b8ae4e0e82c419291ca9fd65 (refs/remotes/origin/master)
 > git config core.sparsecheckout # timeout=10
 > git checkout -f 132fd40a4655a2b9b8ae4e0e82c419291ca9fd65
Commit message: "Update test12.py"
 > git rev-list --no-walk 132fd40a4655a2b9b8ae4e0e82c419291ca9fd65 # timeout=10
[magic-plug-test] $ /bin/sh -xe /tmp/jenkins2988948856244531836.sh
+ sleep 30
```



Date	Test	Status
01/17/2019	G5P9F-QQKTG-OLRS8-D6...	Completed

Test Details

61/G5P9F-QQKTG-OLRS8-D6JX0

Tester: sushobhitd CI Plugin: N/A
Local Tunnel: N/A Bugs Logged: N/A
Start Time: 17:09:58 End Time: Invalid
Status: Running
Environment: 63.0 10 1024x768

We are getting your logs ready.

Web Testing Platform: Check your websites and web applications in all major browsers https://www.lambdatest.com/web-testing

Conclusion

A well-defined Jenkins pipeline can help shorten production times and improve the quality of applications. It provides a definitive structure to your existing building process, committing, automation testing, and deployment.

I hope this Jenkins pipeline tutorial was helpful for you and you were able to create your first Jenkins Pipeline successfully and incorporate automation testing using an online Selenium grid like LambdaTest. **Happy Testing!**