UNIT III CONTINUOUS INTEGRATION USING JENKINS 6

Install & Configure Jenkins, Jenkins Architecture Overview, Creating a Jenkins Job, Configuring a Jenkins job, Introduction to Plugins, Adding Plugins to Jenkins, Commonly used plugins (Git Plugin, Parameter Plugin, HTML Publisher, Copy Artifact and Extended choice parameters). Configuring Jenkins to work with java, Git and Maven, Creating a Jenkins Build and Jenkins workspace.

INTRODUCTION

Jenkins is an open source automation tool which allows continuous integration (CI). It is written in Java. Jenkins builds and tests our software projects continuously which makes it easy for developers to integrate the changes in the project and reduces the cost of manual testing and increases quality. Jenkins integrates all types of development life cycle processes including build, document, test, package, phase, deploy and so on. Jenkins supports a large number of plugins which makes it easy to configure and customize any project.

Features of Jenkins:

Following are the features of Jenkins:

Open-Source and Free:

Jenkins is freely available and supported by a large, active community, ensuring continuous development and extensive resources.

Easy Installation and Configuration:

It is a self-contained Java-based program with straightforward installation processes across various operating systems like Windows, Linux, and macOS. Configuration is managed through a user-friendly web interface.

Extensive Plugin Ecosystem:

Jenkins boasts a vast library of plugins (over 1,500) that extend its functionality and enable integration with a wide array of tools and technologies in the software development lifecycle, including version control systems (Git, SVN), build tools (Maven, Gradle), testing frameworks, and deployment platforms (Docker, Kubernetes, AWS).

Continuous Integration and Delivery (CI/CD) Capabilities:

Jenkins automates the entire CI/CD pipeline, including building, testing, and deploying software projects whenever code changes are committed.

Distributed Builds:

Jenkins can distribute build jobs across multiple agent machines (nodes), optimizing resource utilization and accelerating build and test times.

Monitoring and Reporting:

Jenkins provides built-in capabilities for monitoring build status, viewing logs, tracking performance trends, and generating reports to provide insights into the development process.

Notifications:

It can send notifications about build and deployment results through various channels like email or chat platforms.

ADVANTAGES AND DISADVANTAGES OF USING JENKINS

Advantages

- Highly extensible with a huge variety of existing plugins. Plugins contribute to Jenkins' flexibility and rich scripting and declarative language which supports advanced, custom pipelines.
- Robust and reliable at almost any scale.
- Mature and battle-tested.
- Supports hybrid and multi-cloud environments.
- Offers an extensive knowledge base, documentation, and community resources.
- Based on Java, an enterprise development language with a broad ecosystem, making it suitable for legacy enterprise environments.

Disadvantages:

- Single server architecture—uses a single server architecture, which limits resources to resources on a single computer, virtual machine, or container. Jenkins doesn't allow server-to-server federation, which can cause performance issues in large-scale environments.
- Jenkins sprawl—this is a common problem which also stems from lack of federation. Multiple teams using Jenkins can create a large number of standalone Jenkins servers that are difficult to manage.
- Relies on dated Java architectures and technologies—specifically Servlet and Maven. In general, Jenkins uses a monolithic architecture and is not designed for newer Java technologies such as Spring Boot or GraalVM.
- Not container native—Jenkins was designed in an era before containers and Kubernetes gained popularity, and while it supports container technology, it does not have nuanced support for container and orchestration mechanisms.
- Difficult to implement in production environments—developing continuous delivery pipelines with Jenkinsfiles requires coding in a declarative or scripting language, and complex pipelines can be difficult to code, debug, and maintain.
- Offers no functionality for real production deployments—"deploying with Jenkins" means running a fully customized set of scripts to handle the deployment.

- Jenkins itself requires deployment—this can be difficult to automate. Organizations that need to combine Jenkins with a continuous delivery solution have traditionally used configuration management to do this, but this adds another layer of complexity and is error-prone.
- Complicated plugin management—Jenkins has nearly 2,000 plugins, which can be overwhelming to sort through until you find a useful plugin. Many plugins also have dependencies that increase the management burden, while some plugins may conflict with each other. There is no guarantee a plugin you use will continue to be maintained.
- Groovy expertise requirements—Jenkins has programmatic pipelines implemented in Groovy, a language that is currently not in wide use and can make scripts difficult to work with. Jenkins supports scripted and declarative Groovy modes.

INSTALL & CONFIGURE JENKINS

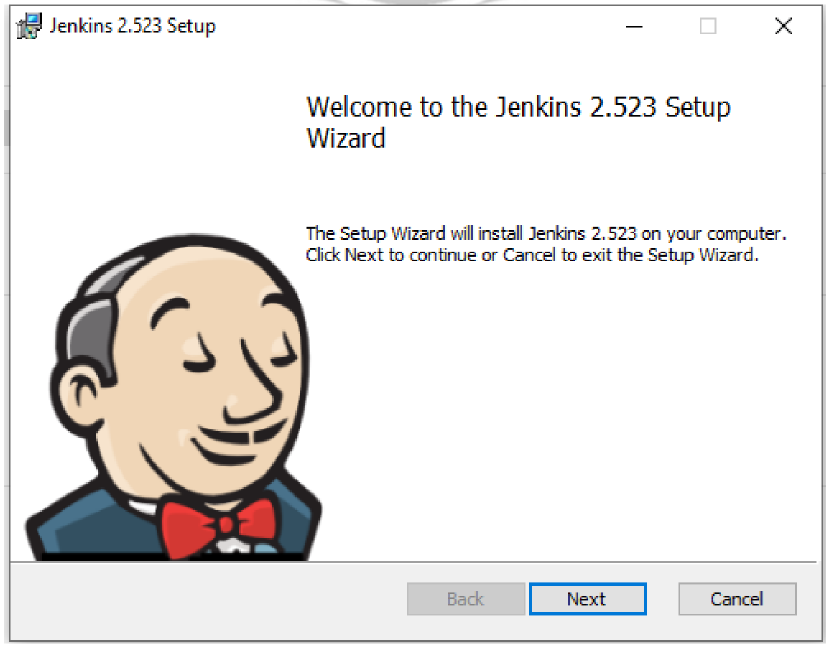
Since Jenkins runs on Java, the latest version of JDK or JRE must be present in the system.

Steps to be followed to install jenkins:

Step 1: Go to the website https://www.jenkins.io/download/ and select the platform on which Jenkins to be installed.

Step 2: Once the download is completed, run the jenkins.msi installation file.

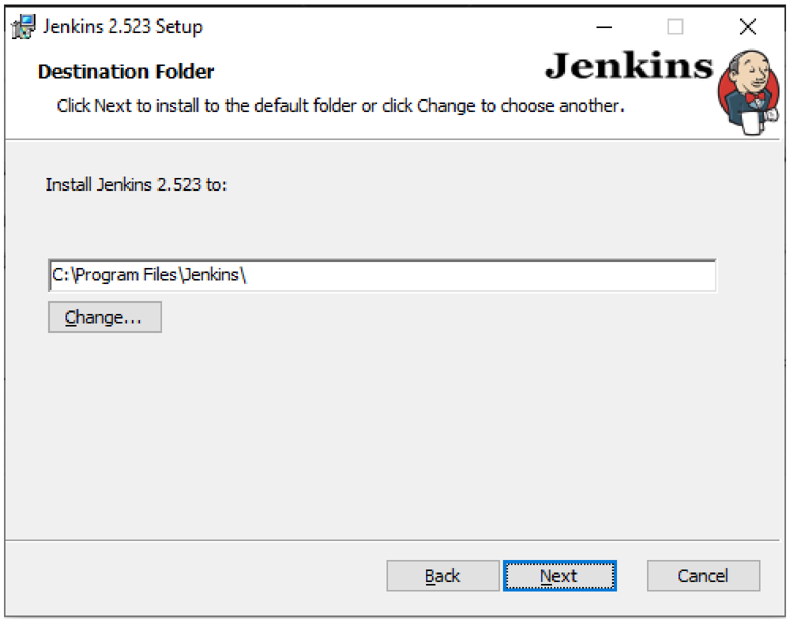
Step 3: The setup wizard starts as follows. Click the Next button to proceed.



Welcome to the Jenkins 2.523 Setup Wizard

The Setup Wizard will install Jenkins 2.523 on your computer. Click Next to continue or Cancel to exit the Setup Wizard.

Step 4: Now, select the destination folder for installation and click Next button.


Jenkins 2.523 Setup

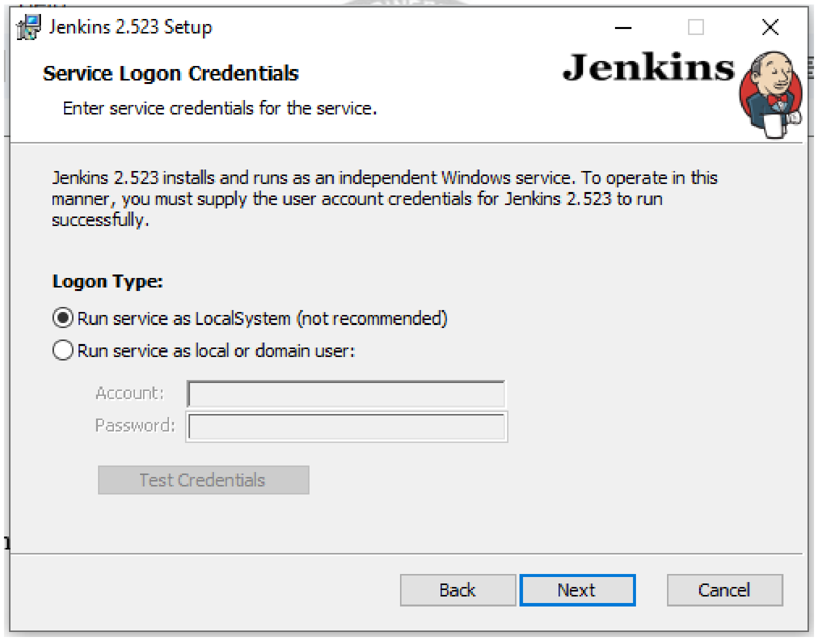
Destination Folder

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

Install Jenkins 2.523 to:

C:\Program Files\Jenkins\

Step 5: On the next screen, Select RunService as LocalSystem.


Jenkins 2.523 Setup

Service Logon Credentials

Enter service credentials for the service.

Jenkins 2.523 installs and runs as an independent Windows service. To operate in this manner, you must supply the user account credentials for Jenkins 2.523 to run successfully.

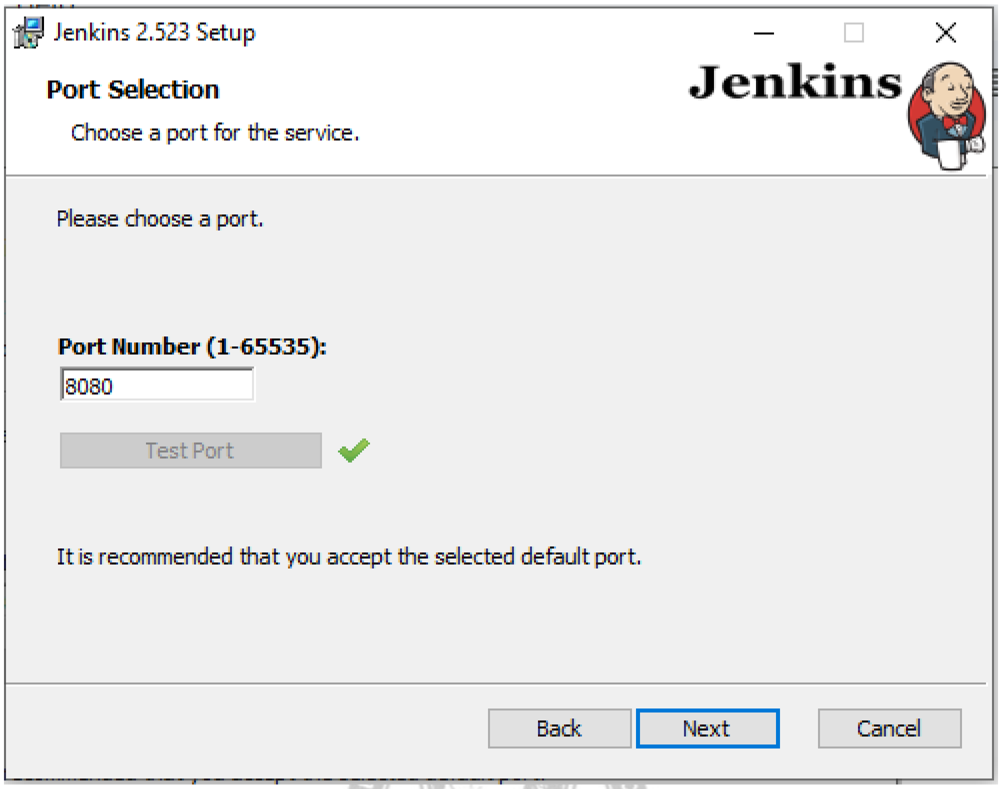
Logon Type:

● Run service as LocalSystem (not recommended)

○ Run service as local or domain user:

Account:

Password:


Jenkins 2.523 Setup

Port Selection

Choose a port for the service.

Please choose a port.

Port Number (1-65535):

8080

It is recommended that you accept the selected default port.

Click the next button. Select the default JDK folder by clicking on Next Button. Finally click on install to start installation.

Step 6: Once the installation is complete, Click finish to exit the installation wizard.

After completing the installation process, we need to unlock Jenkins.

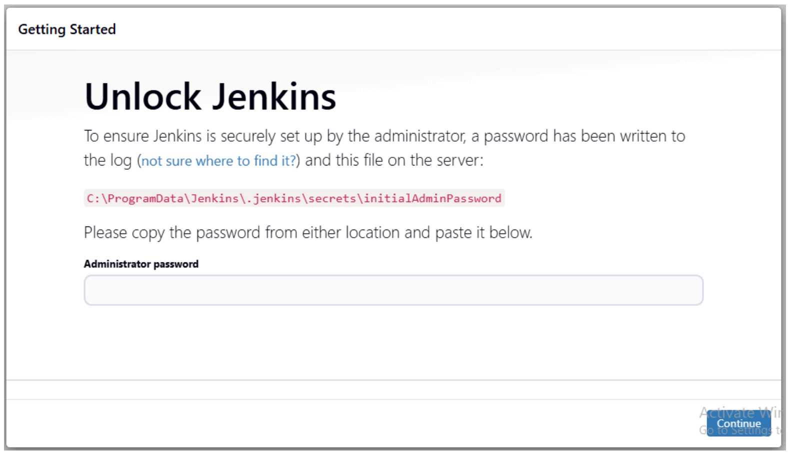
Unlocking Jenkins:

Step 1: Open web browser and navigate to the port number by giving the url https:localhost:8080/

Step 2: Navigate to the location of the system specified by the Unblock jenkins page. The location is :

```
C:\ProgramData\Jenkins\.jenkins\secrets\initialAdminPassword
```

Step 3: Open the initialAdminPassword in Notepad and copy it. Paste it to the Administrator Password on the Unblock Jenkins page and click continue to proceed.


Getting Started

Unlock Jenkins

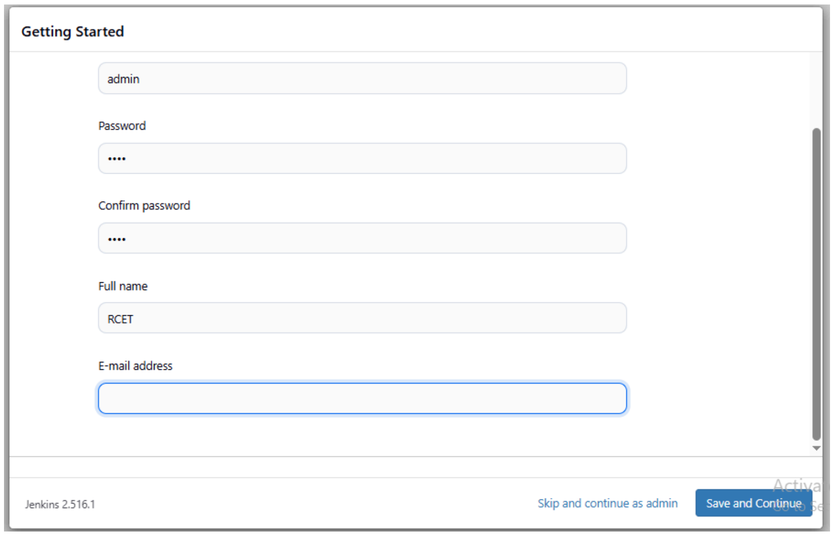
To ensure Jenkins is securely set up by the administrator, a password has been written to the log (not sure where to find it?) and this file on the server:

`C:\ProgramData\Jenkins\.jenkins\secrets\initialAdminPassword`

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

Administrator password

Step 4: Enter the required information on the Create First Admin User page. Click Save and Continue to proceed.


Getting Started

admin

Password

••••

Confirm password

••••

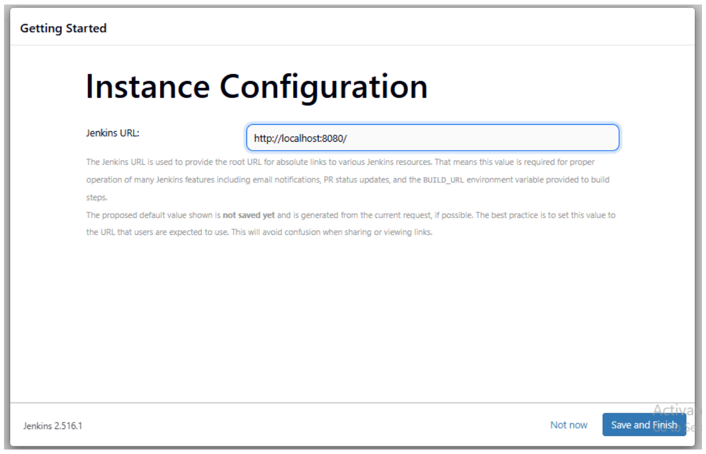
Full name

RCET

E-mail address

Jenkins 2.516.1

Step 5: On the Instance Configuration page confirm the port number 8080 and click save and finish to finish the initial customization.


Getting Started

Instance Configuration

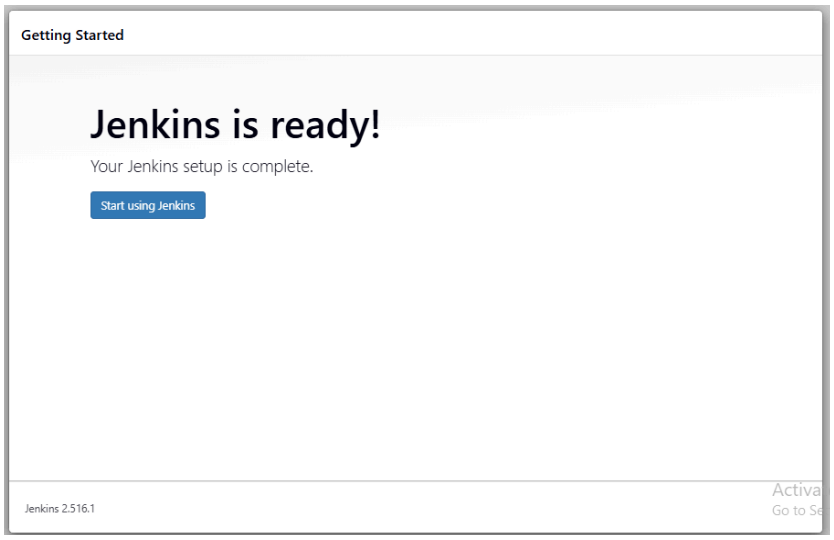
Jenkins URL: http://localhost:8080/

The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the `BUILD_URL` environment variable provided to build steps.

The proposed default value shown is **not saved yet** and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links.

Jenkins 2.516.1

Jenkins configuration is done.

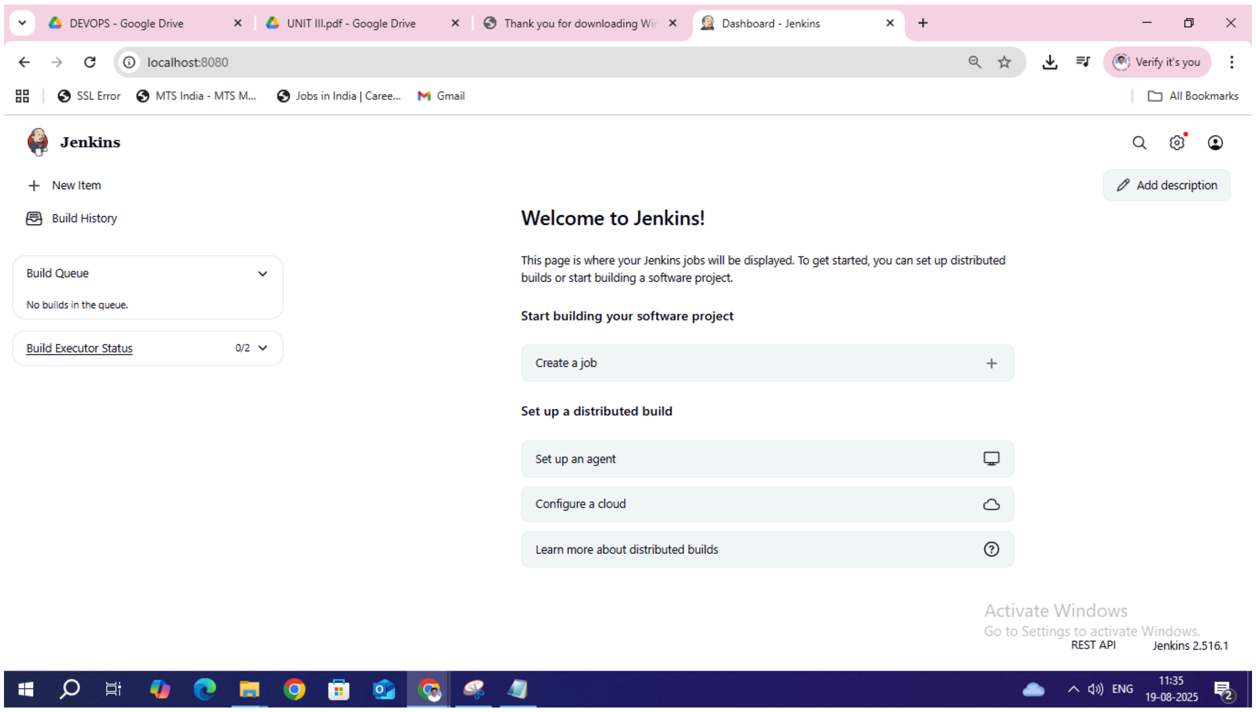


Getting Started

Jenkins is ready!

Your Jenkins setup is complete.

Jenkins dashboard appears as follows.



Welcome to Jenkins!

This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.

Start building your software project

Set up a distributed build

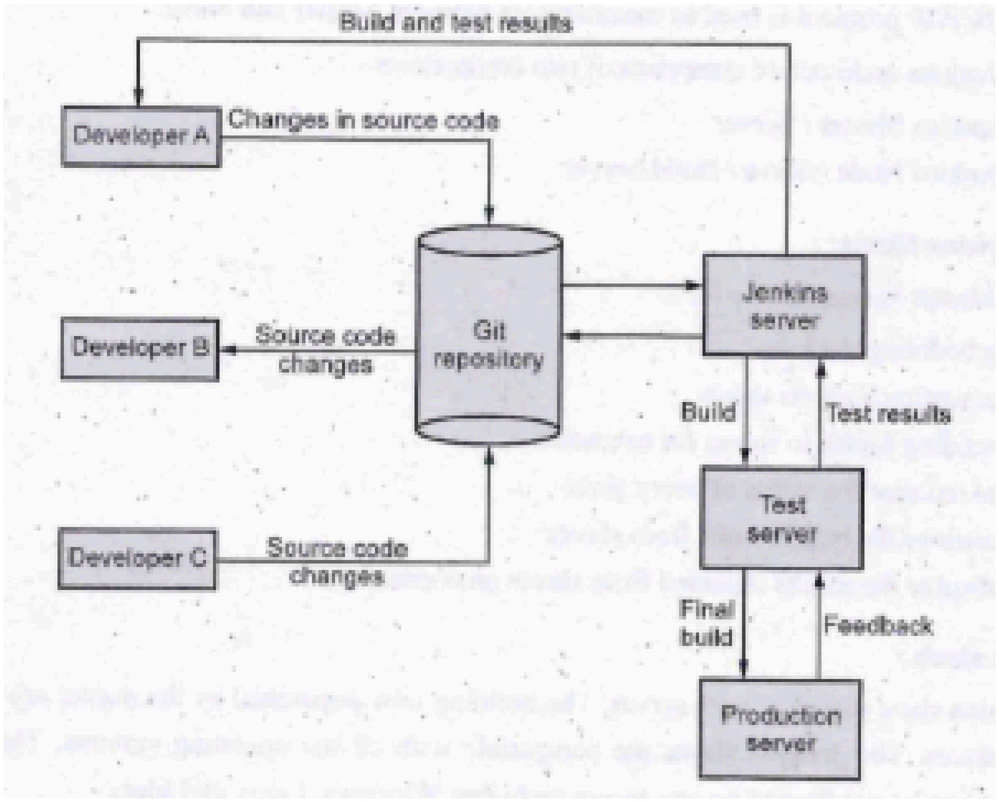
No builds in the queue.

Concept of Continuous Integration:

The continuous integration is a development practice in which developers commit the changes to the source code in a shared repository. Every commit made in the repository is then built. This allows the development team to detect any problem efficiently. Build is triggered whenever a commit occurs. The job of triggering the build for every commit can be automated by jenkins. Hence Jenkins is known as most mature continuous integration tool.

JENKINS ARCHITECTURE OVERVIEW:

Jenkins Workflow:



- The developers commit the code to a source code repository such as GitHub. Jenkins checks the repository at regular intervals for noticing the changes committed in the repository.
- When Jenkins server finds that the changes are committed in the repository, it starts preparing a new build.
- If the build fails, then the developers are notified about it.
- If the build is successful, then the Jenkins server deploys the build on the test server.
- The test server tests the build and generates the feedback. The Jenkins server gets this feedback and notifies about build and test results to the developers.
- If everything is perfect then the build is deployed on the production server
- In all the above activities, Jenkins server continuously verifies the code repository for changes made in the source code. If so, the above activities are repeated continuously.

Jenkins Architecture:

Jenkins works on Master-Slave Architecture also called Controller-Agent. It manages distributed builds using this Master-Slave architecture. The TCP/IP protocol is used to communicate between Master-Slave architecture. Jenkins architecture comprises of two components

1. Jenkins Master / Server
2. Jenkins Node / Slave / Build Server

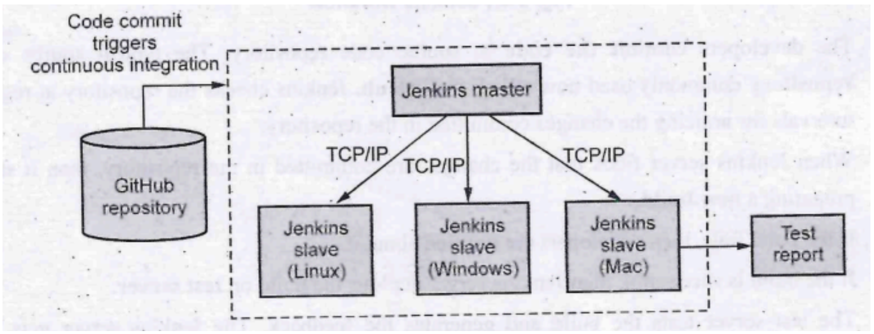
1. Jenkins Controller(Formerly Master)

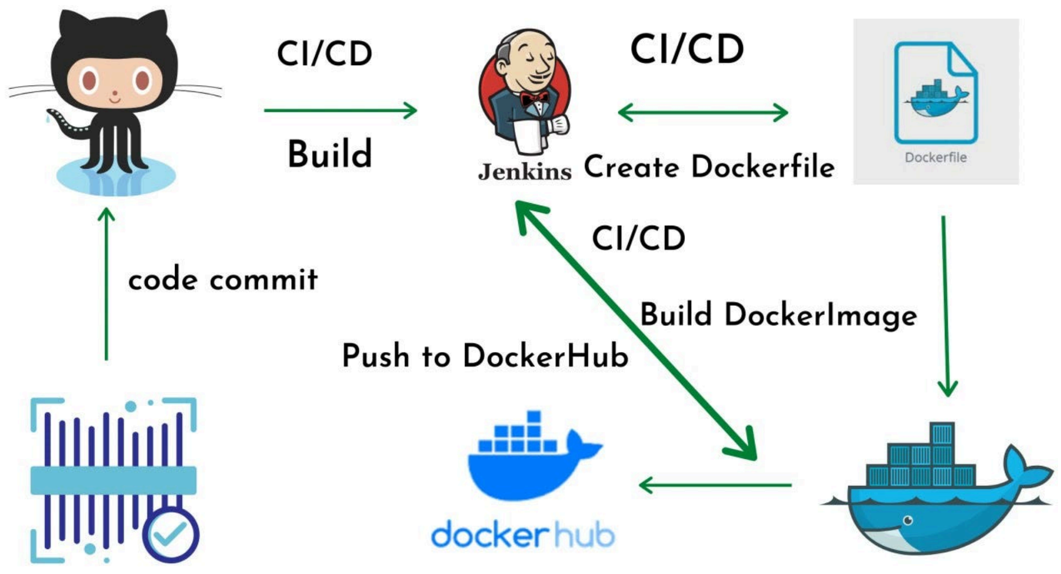
The Jenkins Controller serves as the central system for managing a Jenkins instance, often referred to as its "heart." It oversees agents and their connections, determining the tasks they should perform. Additionally, the Jenkins Controller loads plugins and ensures that jobs run in the correct sequence. The Jenkins Controller gives instructions to the agents, which follow the directions to complete their work efficiently.
Master is responsible for:

- Scheduling the jobs
- Assigning them to slaves
- Sending builds to slaves for execution
- Monitoring the status of every slave
- Retrieve the build results from the slaves
- Display the results received from the slave to the console.

2. Jenkins Agent(Formerly Slave)

Jenkins Agent is a machine that performs tasks like running scripts, executing tests, or building components, etc. These tasks are assigned by the Jenkins Controller. Each agent can have its setup, like different operating systems, software, or hardware. This helps Jenkins handle many types of tasks and work faster by spreading the load. The jenkins slave can be configured on any server including Windows, Linux and mac.
There are two main types of agents:
Permanent Agents: These are always ready and connected to Jenkins. They're like dedicated workers who are always on standby.
Ephemeral Agents: These are temporary. Jenkins starts them only when needed, usually in the cloud or using tools like Docker. When the job is done, they're shut down.




CI/CD
Build
Jenkins
CI/CD
Create Dockerfile
Dockerfile
code commit
CI/CD
Build DockerImage
Push to DockerHub
docker hub