

Experiment 4: To understand Continuous Integration, install and configure Jenkins with Maven/Ant/Gradle to set up a build job.

Aim: To understand the concept of **Continuous Integration (CI)** and implement it by installing and configuring **Jenkins** with **Maven, Ant, or Gradle** to automate the build process. This study aims to explore how Jenkins helps in setting up a CI pipeline, executing automated builds, and improving software development efficiency.

Theory:

Theory of Continuous Integration Using Jenkins with Maven, Ant, or Gradle

Introduction to Continuous Integration (CI)

Continuous Integration (CI) is a **software development practice** where developers frequently integrate their code changes into a shared repository. Each integration is verified using **automated builds and tests**, ensuring that issues are detected early. CI helps streamline the development process, reduces manual errors, and improves software quality.

Key principles of CI:

1. **Frequent Code Integration** – Developers merge changes multiple times a day.
2. **Automated Build Process** – Code is compiled, built, and tested automatically.
3. **Immediate Feedback** – Issues are detected early and fixed promptly.
4. **Consistent Environment** – CI ensures that software builds are reproducible across different environments.

To implement Continuous Integration, organizations use **CI tools like Jenkins**, which automates the build, test, and deployment process.

Jenkins: A CI/CD Automation Tool

Jenkins is an open-source **automation server** that enables developers to **automate software builds, tests, and deployments**. It supports integration with version control systems (Git, SVN) and build tools like **Maven, Ant, and Gradle**.

Key Features of Jenkins

- **Automated Builds:** Supports scheduled or triggered builds based on repository

changes.

- **Build Pipelines:** Allows chaining multiple jobs for end-to-end automation.
- **Plugin Support:** Offers 1,500+ plugins for integration with tools like Docker, Kubernetes, and Slack.
- **Scalability:** Can distribute builds across multiple nodes for faster execution.

Build Tools: Maven, Ant, and Gradle

Build tools are essential in CI to **compile source code, resolve dependencies, and generate deployable artifacts.**

1. Apache Maven

- A widely used **Java-based build automation tool.**
- Uses **POM.xml (Project Object Model)** to define project dependencies, build lifecycle, and plugins.
- Supports phases like **clean, compile, test, package, install, and deploy.**
- Command to build a project:
- mvn clean install

2. Apache Ant

- **Older than Maven**, but still used for Java builds.
- Uses an **XML-based build script (build.xml)** to define tasks.
- More flexible but requires explicit configurations. •

Command to execute a build:

- ant build

3. Gradle

- **Newer build tool**, used for **Java, Kotlin, and Android development**.
- Uses a **Groovy or Kotlin-based build script** instead of XML.
- Faster than Maven due to its **incremental build mechanism**.
- Command to build a project:
- gradle build

Jenkins Integration with Maven, Ant, and Gradle

Jenkins can be configured to **automate builds** using these tools. The integration process involves:

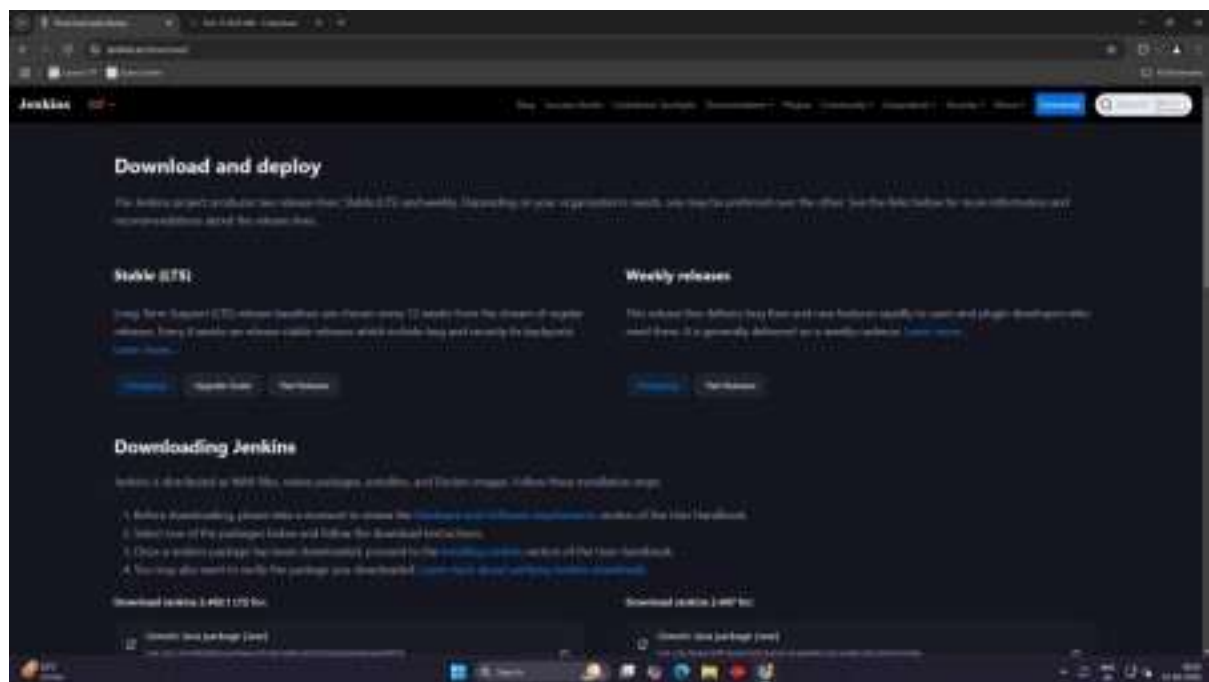
1. **Installing Jenkins** and setting up build tools.

2. **Creating a job in Jenkins** that fetches source code from Git.
3. **Configuring build steps** to invoke Maven, Ant, or Gradle commands.
4. **Executing automated builds** and monitoring results.

Advantages of Using Jenkins for CI

- **Faster Development Cycle** – Automated builds and testing reduce manual effort.
- **Early Bug Detection** – Continuous integration ensures quick issue identification.
- **Improved Collaboration** – Developers work on the latest stable codebase.
- **Efficient Deployment** – Jenkins supports integration with **Docker**, **Kubernetes**, and cloud platforms.

Implementation :



The screenshot shows the Jenkins website's 'Download and deploy' section. The page is dark-themed with white text. At the top, there's a navigation bar with links like 'Home', 'Features', 'Architecture', 'Getting started', 'Plugins', 'Contributing', 'Roadmap', 'Security', 'About', and a search bar. The main heading is 'Download and deploy'. Below it, a paragraph explains that Jenkins is available in two release types: Stable LTS and Weekly releases. The 'Stable (LTS)' section mentions that LTS releases are supported for 12 months and are the recommended version for production use. The 'Weekly releases' section states that these releases are published every Friday and are used for testing new features. Below these sections, there are buttons for 'Download' and 'Get Release' for both LTS and Weekly releases. The 'Downloading Jenkins' section provides instructions on how to download Jenkins, including downloading the Jenkins WAR file, the Jenkins Docker image, or the Jenkins MSI installer. It also includes a list of links for downloading Jenkins, such as 'Download Jenkins 2.401.1 LTS Bin', 'Download Jenkins 2.401.1 LTS Docker', and 'Download Jenkins 2.401.1 LTS MSI'. At the bottom, there are social media links for Jenkins on GitHub, Twitter, and YouTube, and a footer with links to 'About', 'Project', 'Community', and 'Blog'.

Download and deploy

The Jenkins project publishes two release types: Stable (LTS) and weekly. Depending on your organization's needs, you may prefer one over the other. See the table below for more information and recommended links for the release types.

Release Type	Stable (LTS)	Weekly releases
Description	Long-term support (LTS) releases (available for 12 months) from the stream of regular releases. Every 6 months, we release a stable release, which includes bug and security fixes for the last 12 months.	This release type follows the Jenkins 2.x release cycle and includes new features and plugins. It is generally recommended for use in development environments.
Download	Download	Download

Downloading Jenkins

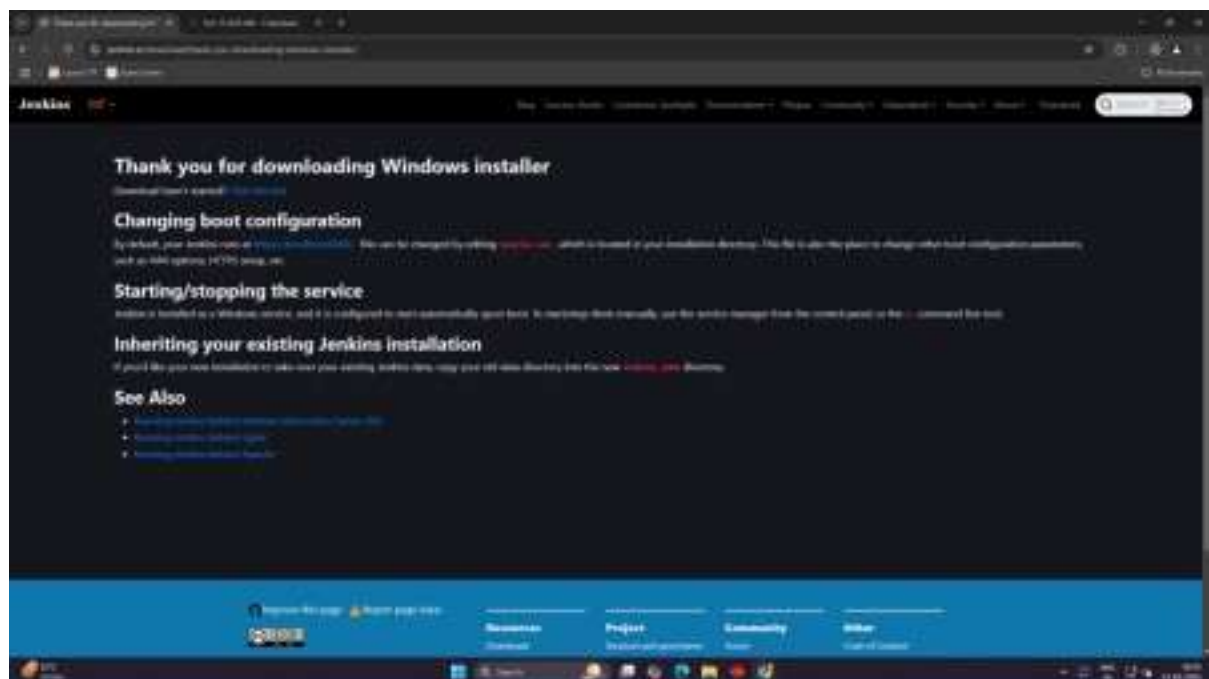
Jenkins is distributed as WAR files, native packages, containers, and Docker images. Follow these installation steps:

- Before downloading, please visit the Jenkins website to ensure the [latest version](#) is available.
- Select one of the packages below and follow the download instructions.
- Once a native package has been downloaded, proceed to the [installation](#) section of the user handbook.
- For more information on how to use the package you downloaded, [visit the documentation](#).

Download Jenkins 2.401.1 LTS Bin

Download Jenkins 2.401.1 LTS Docker

Download Jenkins 2.401.1 LTS MSI



The screenshot shows the Jenkins website's 'Thank you for downloading Windows installer' page. The page is dark-themed with white text. At the top, there's a navigation bar with links like 'Home', 'Features', 'Architecture', 'Getting started', 'Plugins', 'Contributing', 'Roadmap', 'Security', 'About', and a search bar. The main heading is 'Thank you for downloading Windows installer'. Below it, a paragraph explains that the user has downloaded the Jenkins Windows installer. The 'Changing boot configuration' section provides instructions on how to change the boot configuration of the Jenkins Windows installer. The 'Starting/stopping the service' section provides instructions on how to start and stop the Jenkins service. The 'Inheriting your existing Jenkins installation' section provides instructions on how to inherit an existing Jenkins installation. At the bottom, there are social media links for Jenkins on GitHub, Twitter, and YouTube, and a footer with links to 'About', 'Project', 'Community', and 'Blog'.

Thank you for downloading Windows installer

Downloaded successfully! [View instructions](#)

Changing boot configuration

By default, your Jenkins runs at [https://localhost:8080/](#). You can for emergency by using [jenkins.exe](#), which is located in your installation directory. This file is also the place to change other boot configuration settings, such as JVM options (JVM_OPTS).

Starting/stopping the service

Jenkins is installed as a Windows service, and it is configured to start automatically upon boot. To manage this manually, use the Jenkins Manager from the context panel in the [Jenkins Manager](#) for boot.

Inheriting your existing Jenkins installation

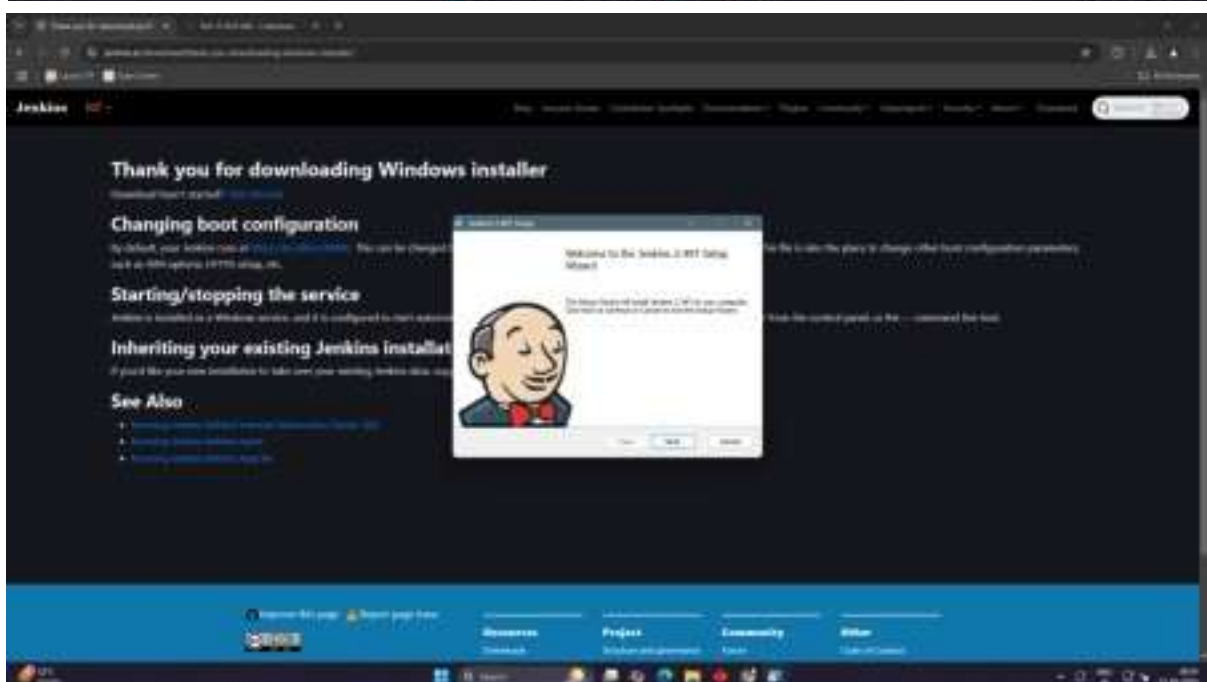
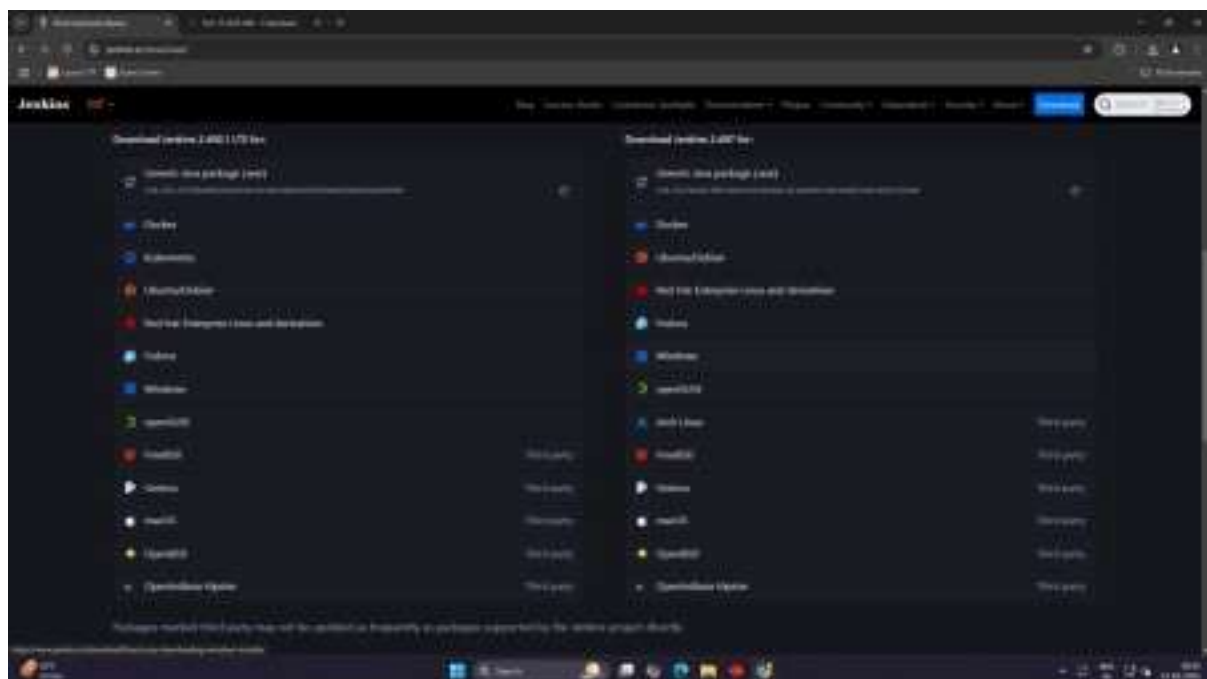
If you'd like your new installation to take over your existing Jenkins data, copy your old data directory into the new [Jenkins](#) data directory.

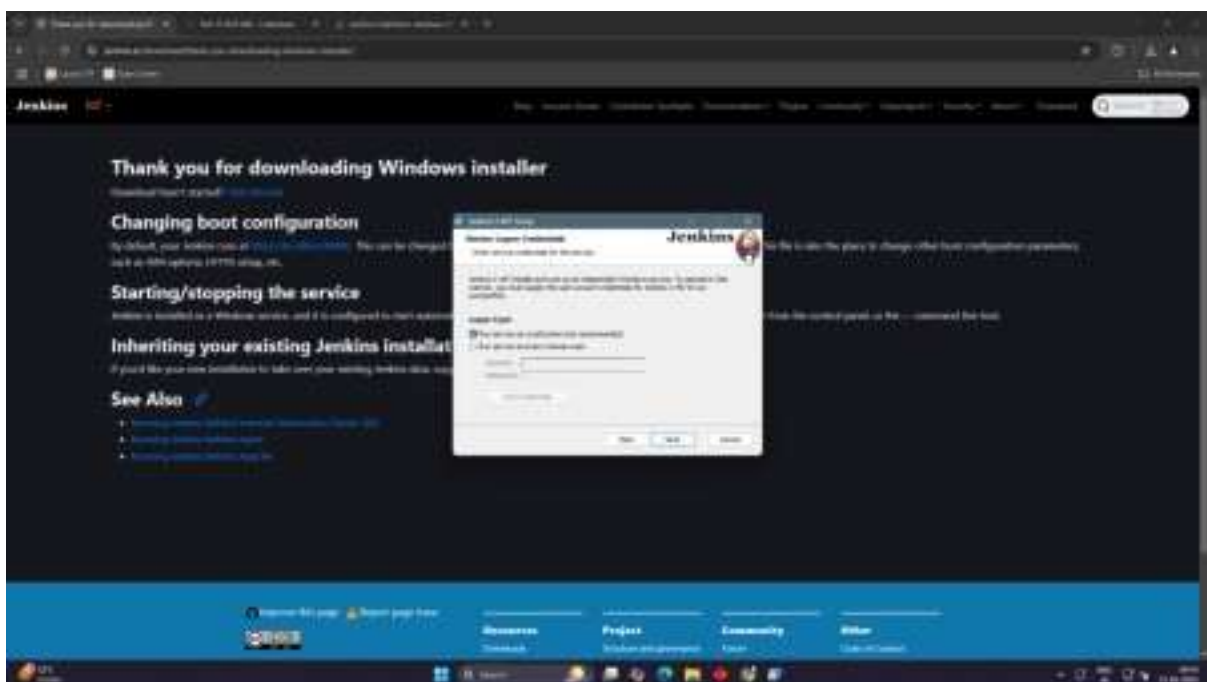
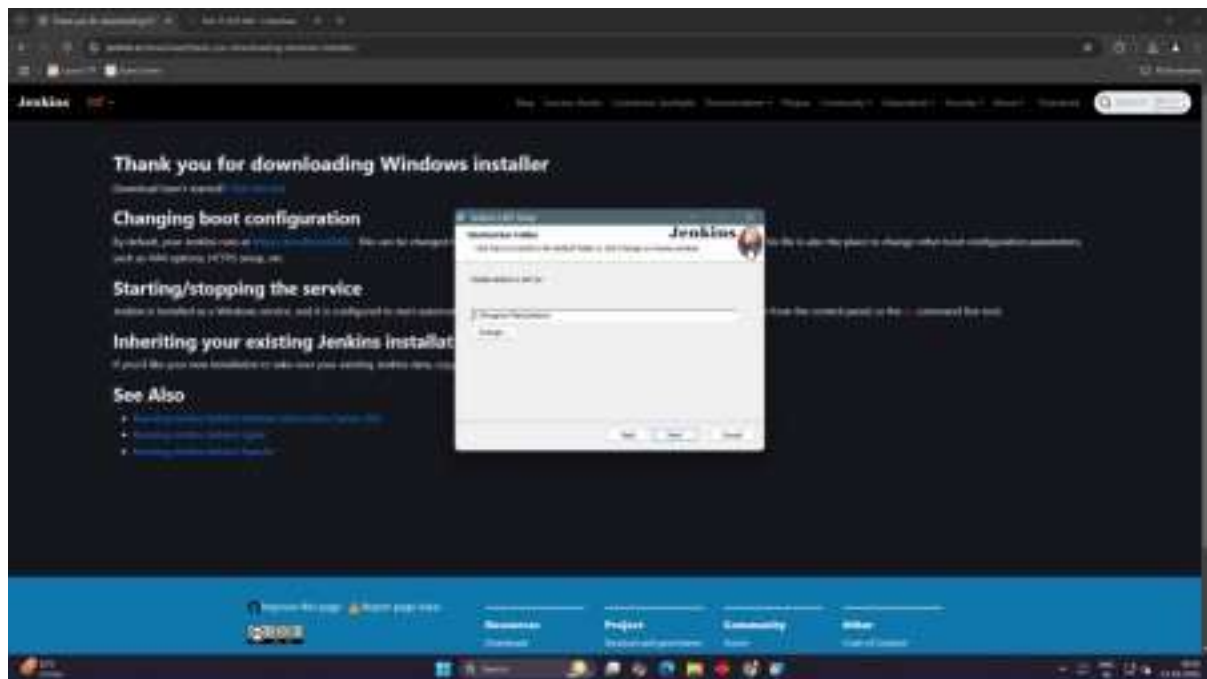
See Also

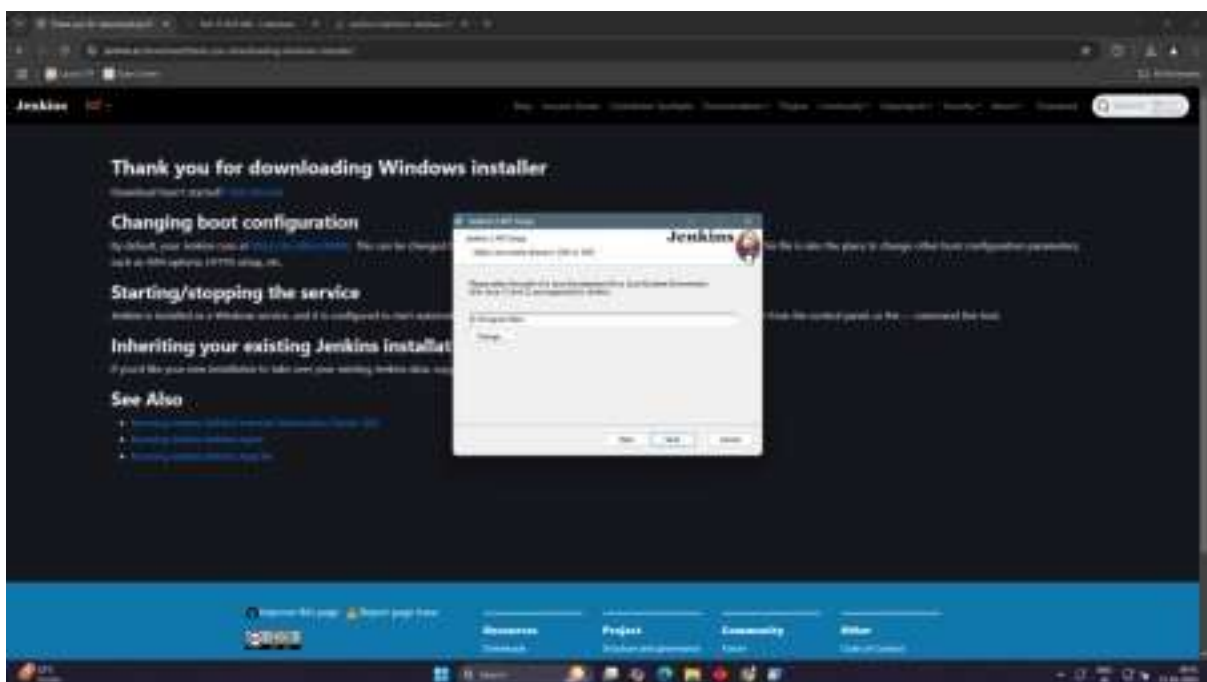
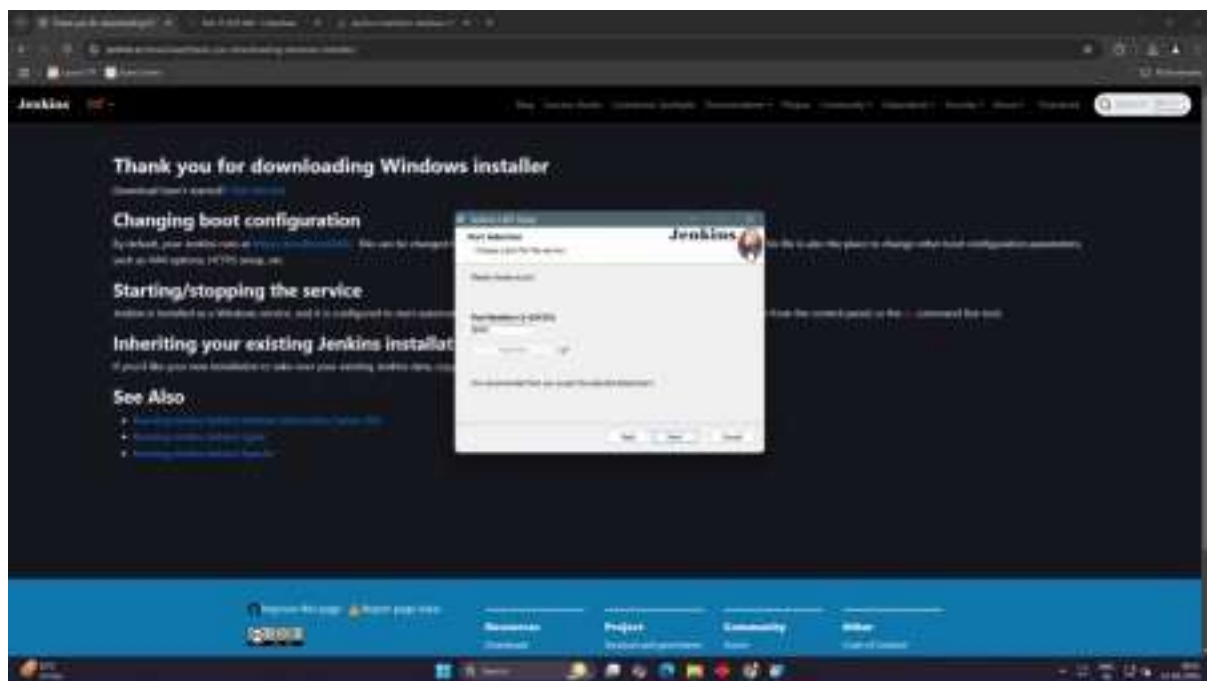
- [Installing Jenkins on Windows](#)
- [Installing Jenkins on Linux](#)
- [Installing Jenkins on macOS](#)

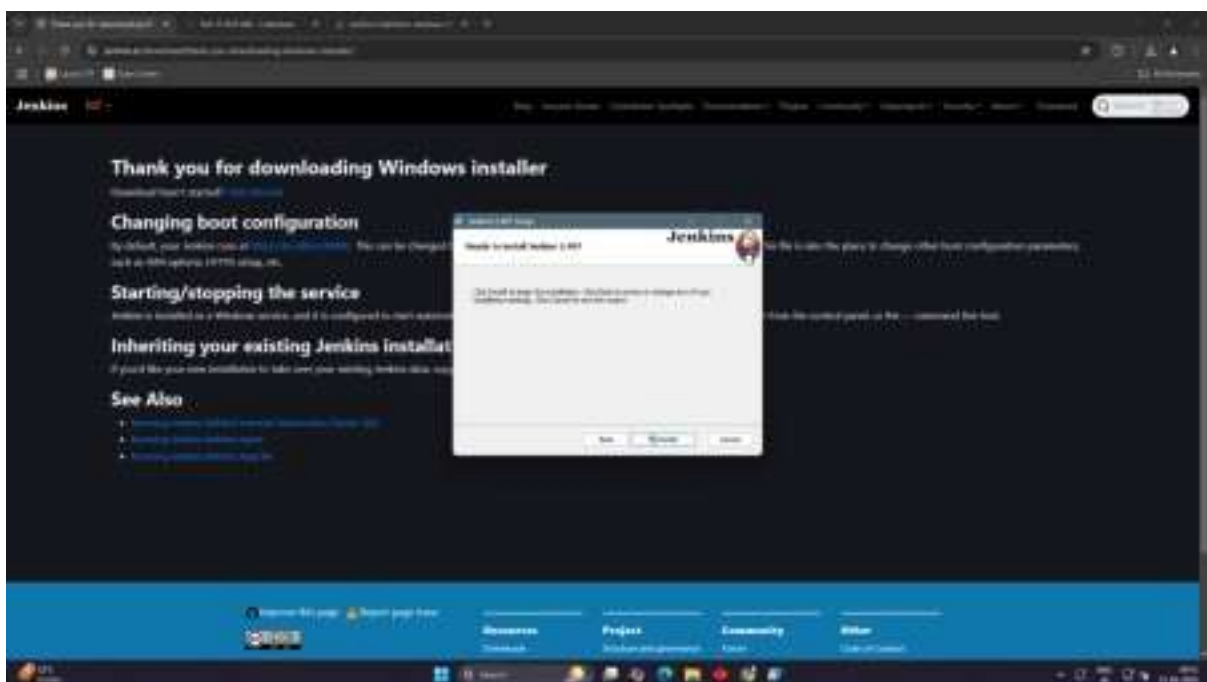
Improve this page | Report page issue

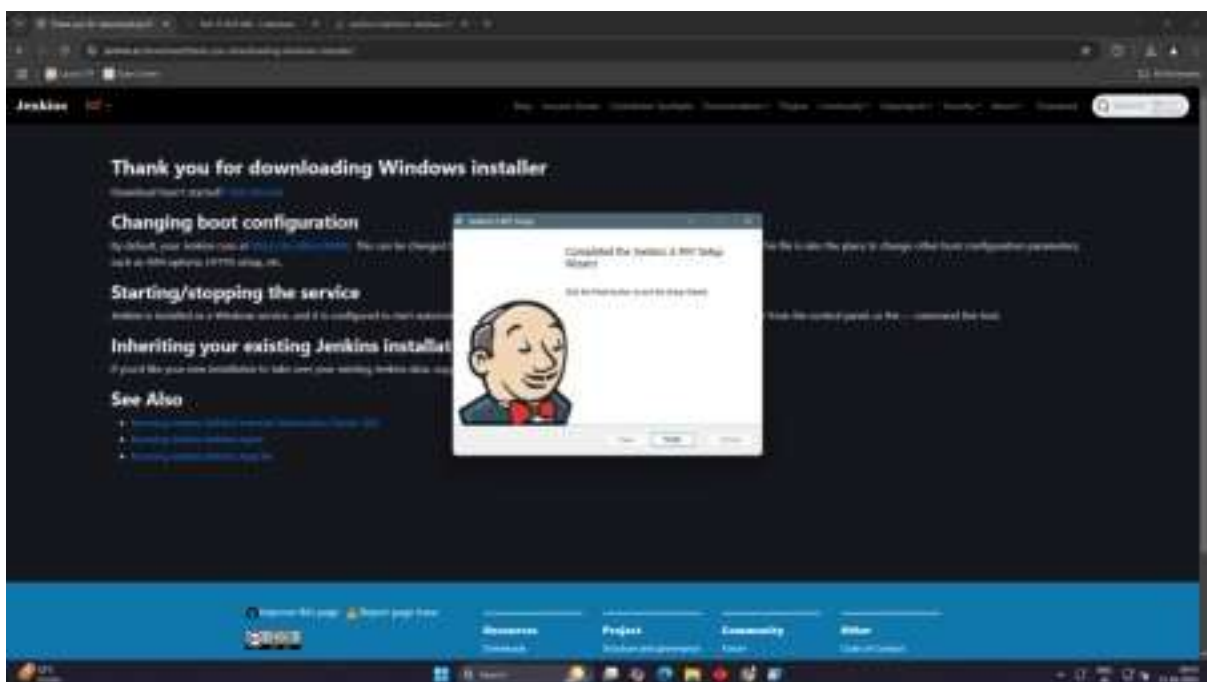
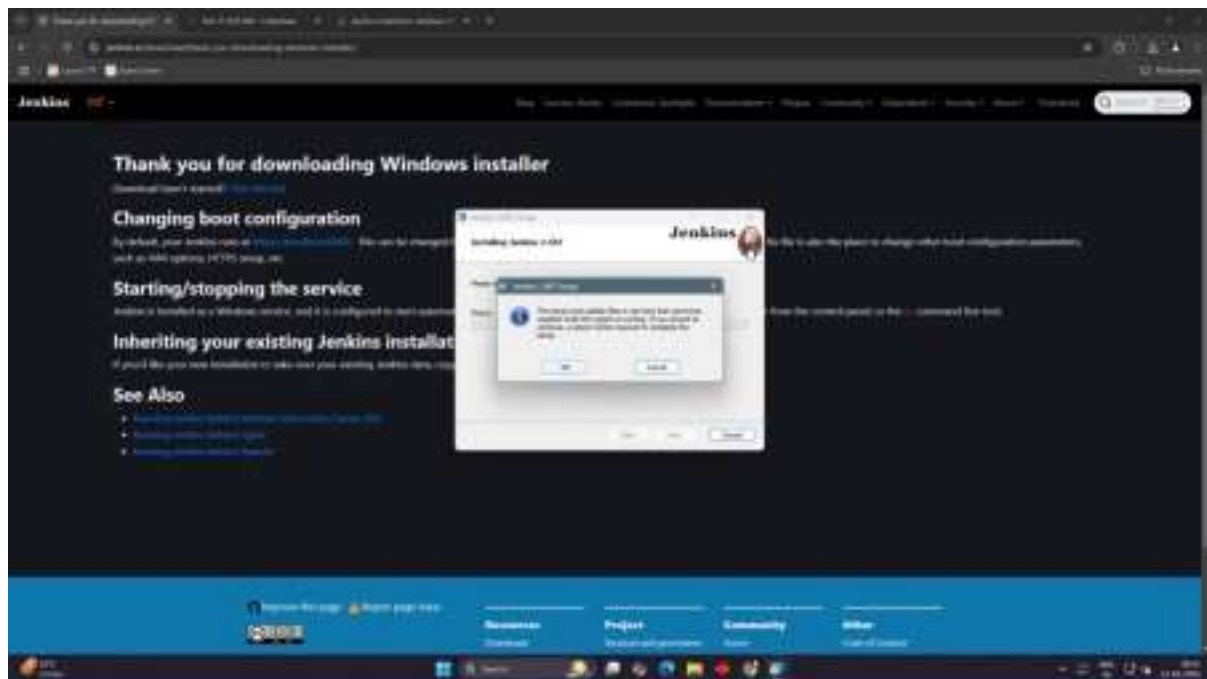
[Home](#) | [Project](#) | [Community](#) | [Blog](#)

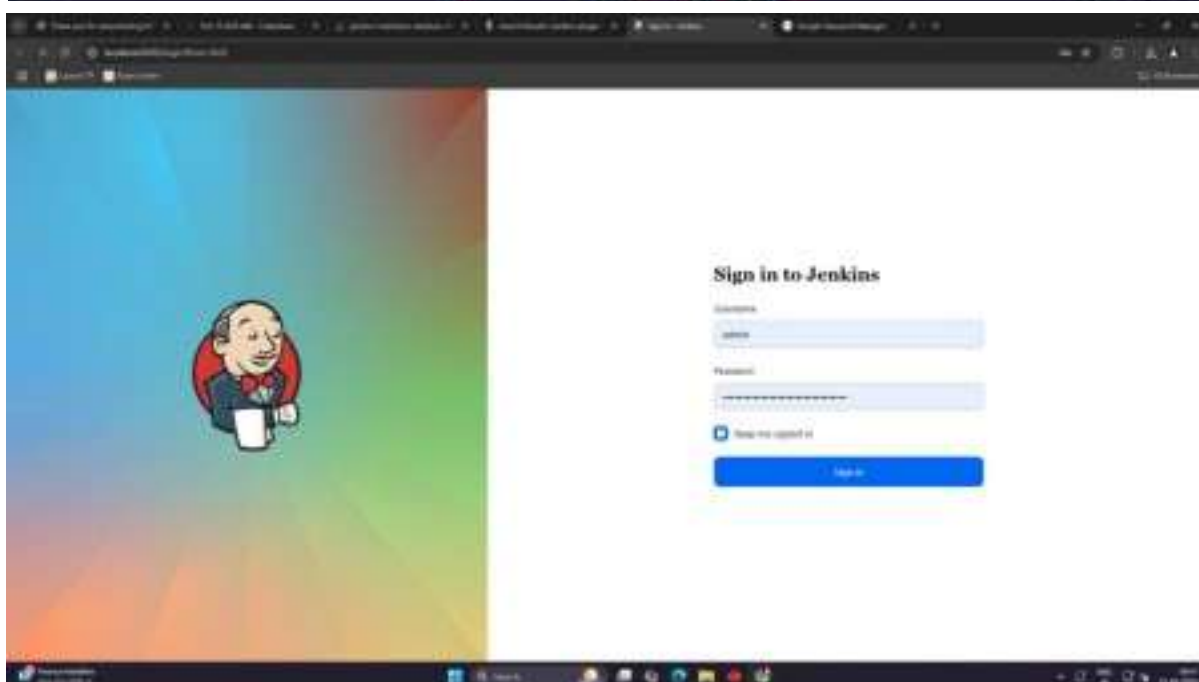
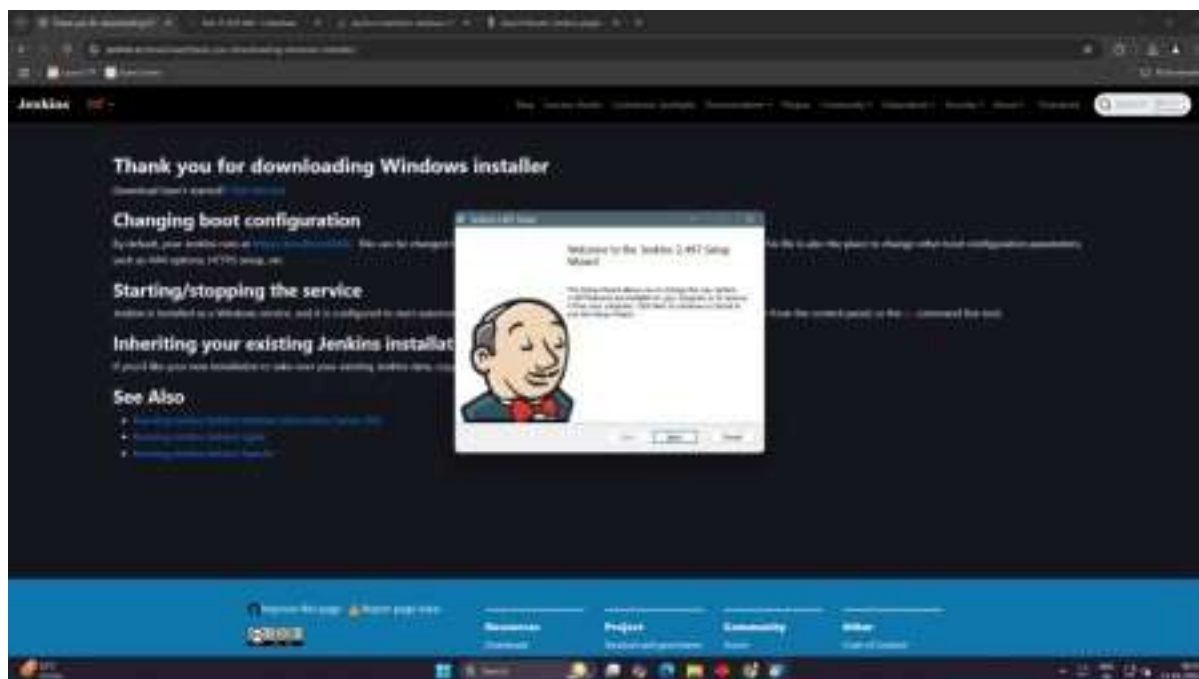














Conclusion : Thus we have successfully installed and configured Jenkins.