

Experiment No. 4

Aim: To understand Continuous Integration, install and configure jenkins with maven/ant/gradle to setup a build job

Theory:

Continuous Integration (CI):

Continuous Integration is a software development practice where code changes are automatically built, tested, and integrated into a shared repository on a frequent basis. The primary goal of CI is to detect and address integration issues early in the development process, ensuring that the codebase remains stable and reliable. This approach allows teams to deliver high-quality software more efficiently and with greater confidence.

Key Concepts of Continuous Integration:

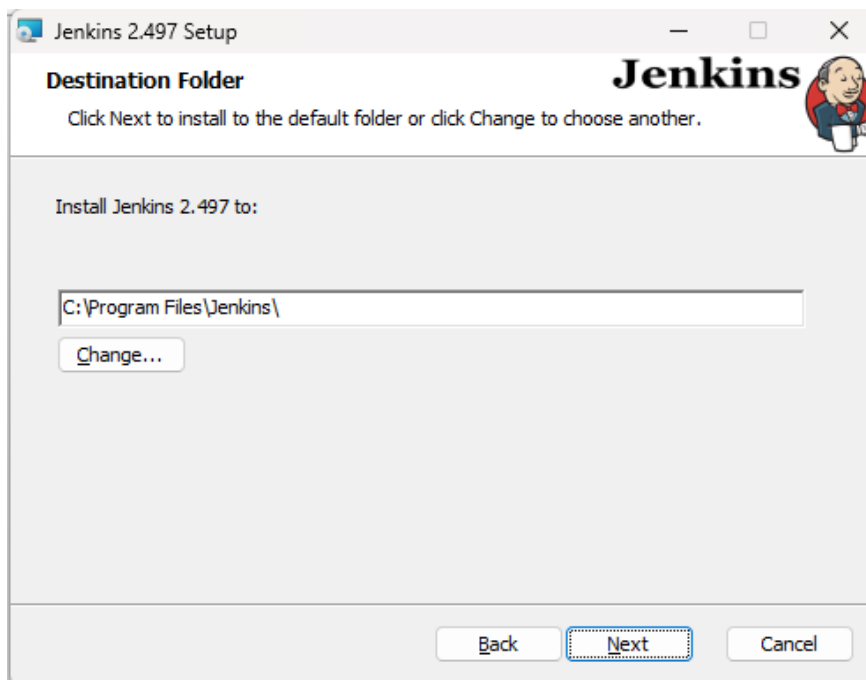
1. Version Control System (VCS): CI relies on a VCS (e.g., Git, SVN) to manage and track changes in the codebase. Developers commit their changes to the repository, triggering the CI process.
2. Automated Build: CI systems automate the process of compiling source code into executable artifacts. This ensures consistency and eliminates manual errors in the build process.
3. Automated Testing: CI includes automated testing to validate that code changes do not introduce new bugs or break existing functionality. Common types of tests include unit tests, integration tests, and acceptance tests.
4. Build Server: A CI server, like Jenkins, is responsible for orchestrating the CI process. It monitors the VCS for changes, triggers builds, runs tests, and provides feedback to developers.

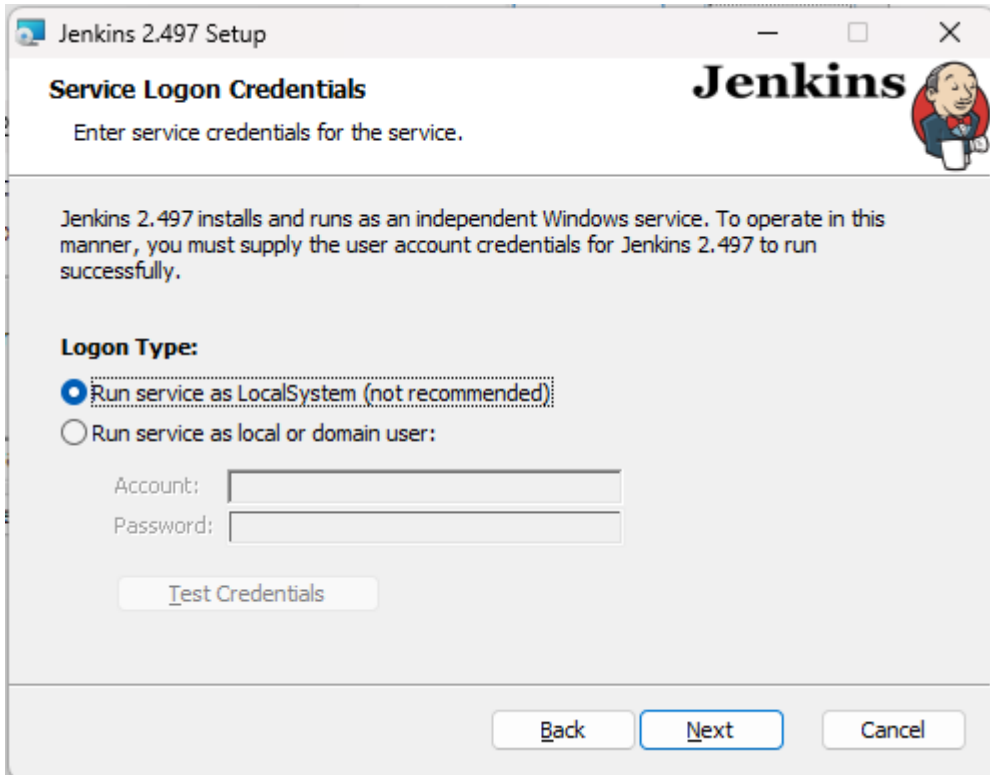
Installing and Configuring Jenkins with Maven/Ant/Gradle:

Here, we'll focus on setting up Jenkins with Maven, but the process is similar for other build tools.

1. Install Jenkins:

- Download and install Jenkins from the official website.
- Start the Jenkins service.





Jenkins 2.497 Setup

Service Logon Credentials

Enter service credentials for the service.

Jenkins 2.497 installs and runs as an independent Windows service. To operate in this manner, you must supply the user account credentials for Jenkins 2.497 to run successfully.

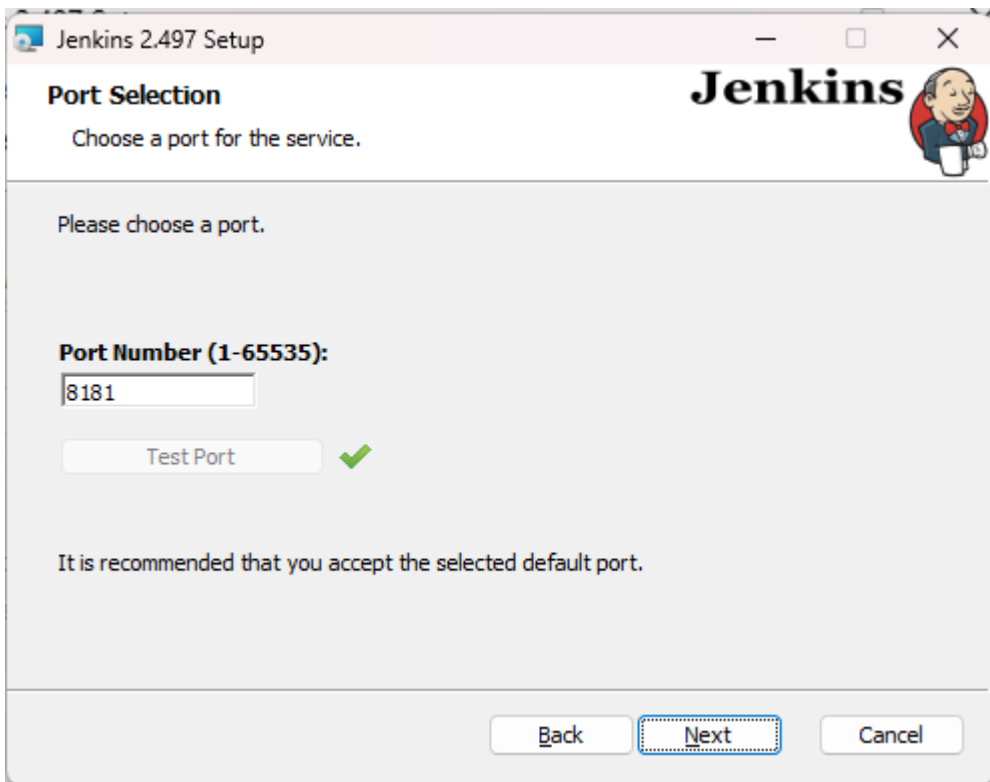
Logon Type:

☒ Run service as LocalSystem (not recommended)

☐ Run service as local or domain user:

Account:

Password:




Jenkins 2.497 Setup

Port Selection

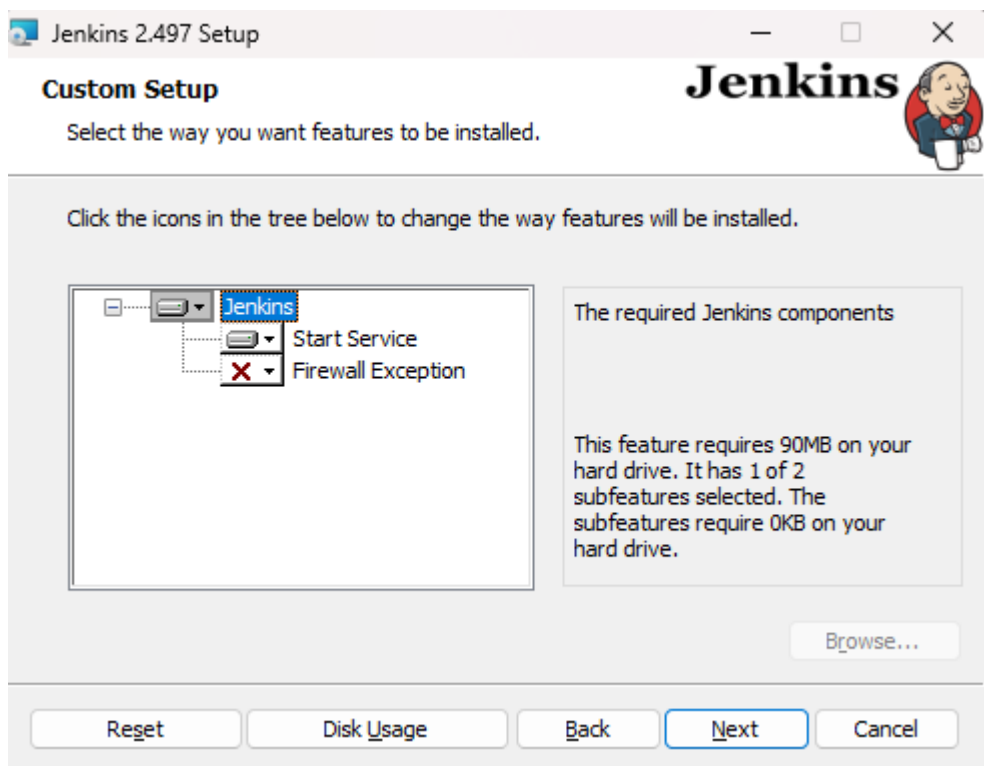
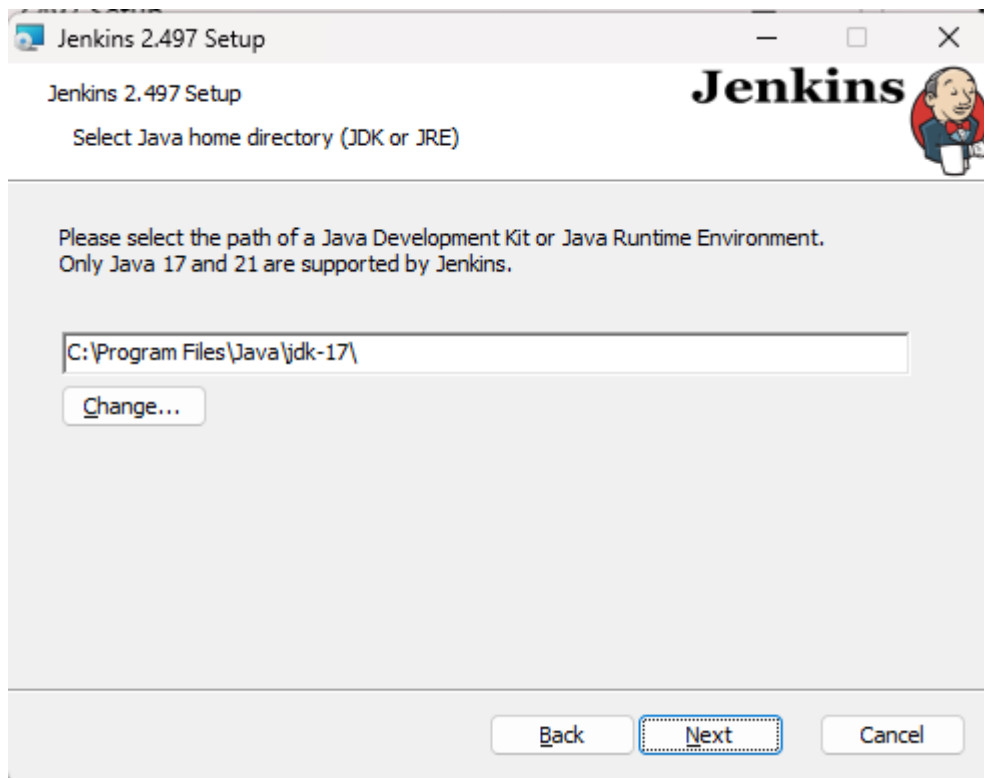
Choose a port for the service.

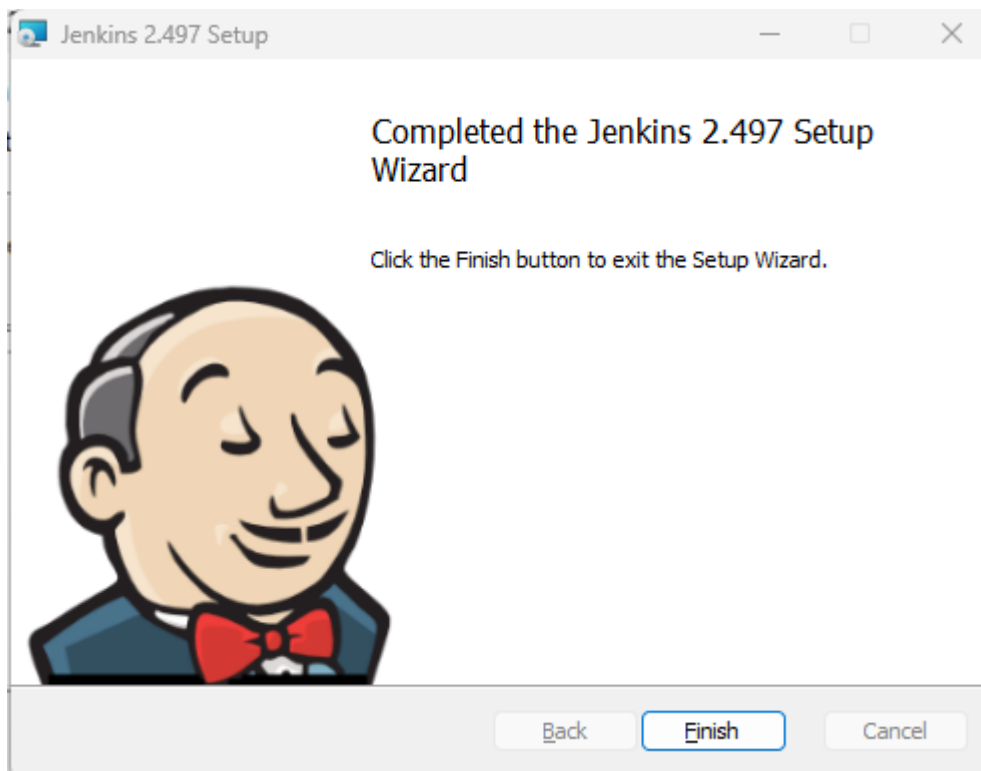
Please choose a port.







Port Number (1-65535):



It is recommended that you accept the selected default port.





| | | | | | |
|---|---------------------------------|-----------------|---------|-----------------|----------------|
|  | IP Helper | Provides tu... | Running | Automatic | Local Syste... |
|  | IP Translation Configuration... | Configures ... | | Manual (Trig... | Local Syste... |
|  | IPsec Policy Agent | Internet Pro... | Running | Manual (Trig... | Network S... |
|  | Jenkins | Jenkins Aut... | Running | Automatic | Local Syste... |
|  | KtmRm for Distributed Tran... | Coordinates... | | Manual (Trig... | Network S... |
|  | Language Experience Service | Provides inf | | Manual | Local Syste... |

2. Configure Jenkins:

- Open Jenkins in a web browser and follow the initial setup wizard. - Install necessary plugins, including the ones for Maven, Ant, or

Gradle integration.

3. Create a Jenkins Job:

- Click on "New Item" to create a new job.
- Choose the type of project (e.g., Freestyle project or Pipeline).
- Configure the job settings, such as source code repository, build triggers, and post-build actions.

4. Configure Build Tool:

- If using Maven, specify the path to the Maven executable and configure Maven goals (e.g., clean install).
- For Ant or Gradle, configure the respective build tool settings.

5. Save and Build:

- Save the job configuration and manually trigger the build to test the setup. - Observe the build console output for any errors or issues.

Theory:

Continuous Integration addresses several challenges in software development:

1. Early Detection of Bugs: CI ensures that code changes are continuously tested, allowing for the early detection and resolution of bugs.
2. Consistent Builds: Automated builds eliminate variations caused by manual processes, ensuring that each build is consistent and reproducible.
3. Integration Testing: CI helps in integrating code changes frequently, reducing the likelihood of integration issues that arise when merging changes from multiple developers.
4. Rapid Feedback: Developers receive rapid feedback on the quality of their code through automated tests and build reports, enabling them to address issues promptly.
5. Improved Collaboration: CI promotes collaboration by providing a central platform where developers can share and integrate their work regularly.

By installing and configuring Jenkins with Maven, Ant, or Gradle, teams can establish a robust CI pipeline that enhances software quality, accelerates development cycles, and facilitates collaboration among team members.