ROUNAK KATIYAR
Batch: T - 13
Roll no. 47

EXPERIMENT 4

AIM: To understand continuous integration, install and configure Jenkins with maven / ant / gradle to set up a build job.

THEORY:

Continuous Integration (CI):

Continuous Integration (CI) is a software development practice where code changes are automatically integrated, tested, and deployed frequently, often multiple times a day. The goal of CI is to improve code quality, reduce integration issues, and speed up the development lifecycle.

Key Concepts of CI:

- **Automated Testing:** Code is automatically tested every time it is committed to the repository to catch bugs early.
- **Frequent Integration:** Developers integrate their code changes into a shared repository regularly to avoid conflicts.
- **Continuous Feedback:** Developers receive quick feedback about the health of their codebase, enabling them to fix issues promptly.
- **Build Automation:** The process of compiling code, running tests, and deploying applications is automated.

Benefits of Continuous Integration:

- **Early Detection of Errors:** Automated tests catch bugs soon after they are introduced, reducing the cost of fixing them.
- **Improved Code Quality:** Continuous testing ensures that only high-quality code is merged into the main codebase.
- **Faster Development Cycles:** Developers can work in parallel without waiting for integration, enabling faster feature releases.
- **Reduced Manual Effort:** Automating builds and tests reduces manual errors and saves time.

Key CI Practices:

- **Version Control System (VCS) Integration:** CI tools integrate with Git, SVN, etc., to monitor code changes.
- **Automated Builds:** Code is automatically compiled and built into deployable artefacts.
- **Automated Testing:** Unit tests, integration tests, and UI tests run automatically after every code change.
- **Deployment Automation:** Code can be automatically deployed to staging or production environments.

Jenkins: A Leading CI/CD Tool:

Jenkins is an open-source automation server commonly used to implement CI/CD pipelines. It automates parts of software development related to building, testing, and deploying code.

Key Features of Jenkins:

- **Extensibility:** Jenkins has a rich ecosystem of plugins to support building, deploying, and automating projects for multiple languages and technologies.
- **Distributed Builds:** Jenkins can distribute tasks across multiple machines for faster processing.
- **Pipeline as Code:** Using Jenkins Pipeline DSL (Domain-Specific Language), you can define CI/CD workflows as code in a Jenkinsfile.
- **Integration with VCS:** Jenkins integrates seamlessly with Git, SVN, and other version control systems.

Jenkins Architecture

Jenkins operates on a master-slave (or controller-agent) architecture:

- **Master (Controller):** The central server that manages the CI/CD environment, schedules jobs, and monitors their execution.
- **Agents (Slaves):** Machines that perform the actual build and test tasks. Jenkins can distribute workloads to multiple agents for scalability.

Jenkins Components:

- Jobs: Individual tasks that Jenkins can execute, such as building code, running tests, or deploying applications.
- Builds: The process of compiling and packaging the code. Each build can be triggered manually or automatically.
- Pipelines: A series of automated steps defined in a Jenkinsfile that describe the CI/CD process.
- Plugins: Extend Jenkins' functionality, allowing integration with various tools like Git, Docker, Maven, etc.

Setting Up Jenkins:

Step1: Install Jenkins

-> On Ubuntu:

```
sudo apt update

sudo apt install openjdk-11-jdk

wget -q -O - https://pkg.jenkins.io/debian/jenkins.io.key |
sudo apt-key add -

sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable
binary/

> /etc/apt/sources.list.d/jenkins.list'
sudo apt update sudo
apt install jenkins
sudo systemctl start
jenkins sudo
systemctl status
jenkins
```

-> On Windows:

Download the Jenkins installer from the Jenkins official website and follow the installation wizard.

Step2: Access Jenkins Web Interface

Open your browser and go to: http://localhost:8080

Retrieve the initial admin password:

```
sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```

Unlock Jenkins and install the suggested plugins.

Configuring Jenkins for CI:

Step 1: Create a New Job

1. Click “New Item” in the Jenkins dashboard.
2. Enter a name for the job.
3. Choose the job type:
    - Freestyle Project: Basic job configuration.
    - Pipeline: Advanced, code-defined CI/CD workflow.

Step 2: Configure Source Code Repository

For GitHub Integration:

- In the job configuration, go to “Source Code Management”.
- Select Git and enter the repository URL.
- Add credentials if necessary.

Step 3: Add Build Triggers

Configure triggers to automate builds:

- **Poll SCM:** Periodically check for changes in the repository.
- **GitHub Hook Trigger:** Trigger builds automatically when changes are pushed to GitHub.

Step 4: Add Build Steps

- **Execute Shell:** Run shell commands or scripts.
- **Invoke Ant/Maven/Gradle:** Run build tools for Java projects.
- **Execute Windows Batch Command:** Run batch files on Windows systems.

Step 5: Add Post-Build Actions

- **Archive Artefacts:** Save build outputs for later use.
- **Send Notifications:** Email alerts or Slack messages after builds.
- **Deploy Applications:** Automate deployment to servers.

Jenkins Pipeline (Declarative & Scripted):

Declarative Pipeline (Recommended for Most Cases):

A declarative pipeline is defined in a Jenkinsfile using a simplified syntax.

```
pipeline {
    agent any
```

```
    stages {
        stage('Build') {
            steps {
                echo 'Building the project...'
                sh 'mvn clean install'
            }
        }
        stage('Test') {
            steps {
                echo 'Running tests...'
                sh 'mvn test'
            }
        }
        stage('Deploy') {
            steps {
                echo 'Deploying application...'
                sh 'scp target/app.jar
                user@server:/path/to/deploy'
            }
        }
    }
}
```

Scripted Pipeline (More Flexible):
Scripted pipelines offer more control but require a deeper understanding of Groovy.

```
node {
    stage('Build') {
        sh 'mvn clean install'
    }
    stage('Test') {
        sh 'mvn test'
    }
    stage('Deploy') {
        sh 'scp target/app.jar user@server:/path/to/deploy'
    }
}
```

Common Jenkins Plugins:

- Git Plugin: Integrates Jenkins with Git repositories.
- Pipeline Plugin: Enables the use of Jenkins Pipelines.
- Docker Pipeline: Manages Docker containers in CI/CD workflows.
- Blue Ocean: A modern UI for visualizing Jenkins pipelines.
- Slack Notification Plugin: Sends notifications to Slack channels.

Advanced Jenkins Concepts:

1. **Jenkinsfile as Code:**
   Defining your pipeline in a Jenkinsfile allows you to version control your CI/CD process alongside your application code.
2. **Parallel Execution:**
   You can run multiple stages in parallel to speed up your pipeline:

```
pipeline {
    agent any
    stages {
        stage('Parallel Stages') {
            parallel {
                stage('Unit Test') {
                    steps {
                        sh 'mvn test'
                    }
                }
                stage('Integration Test') {
                    steps {
                        sh 'mvn verify'
                    }
                }
            }
        }
    }
}
```

3. **Parameterized Builds:**
   Allow users to pass parameters to Jenkins jobs:

```
pipeline {
    agent any
    parameters {
```

```
            string(name: 'ENV', defaultValue: 'staging',
                description: 'Deployment Environment')
        }
        stages {
            stage('Deploy') {
                steps {
                    sh "deploy.sh ${params.ENV}"
                }
            }
        }
}
```

Jenkins Security Best Practices:

- Use Role-Based Access Control: Limit permissions based on user roles.
- Secure Credentials: Store sensitive data (like API keys) in Jenkins Credentials Manager.
- Keep Jenkins Updated: Regularly update Jenkins and plugins to fix security vulnerabilities.

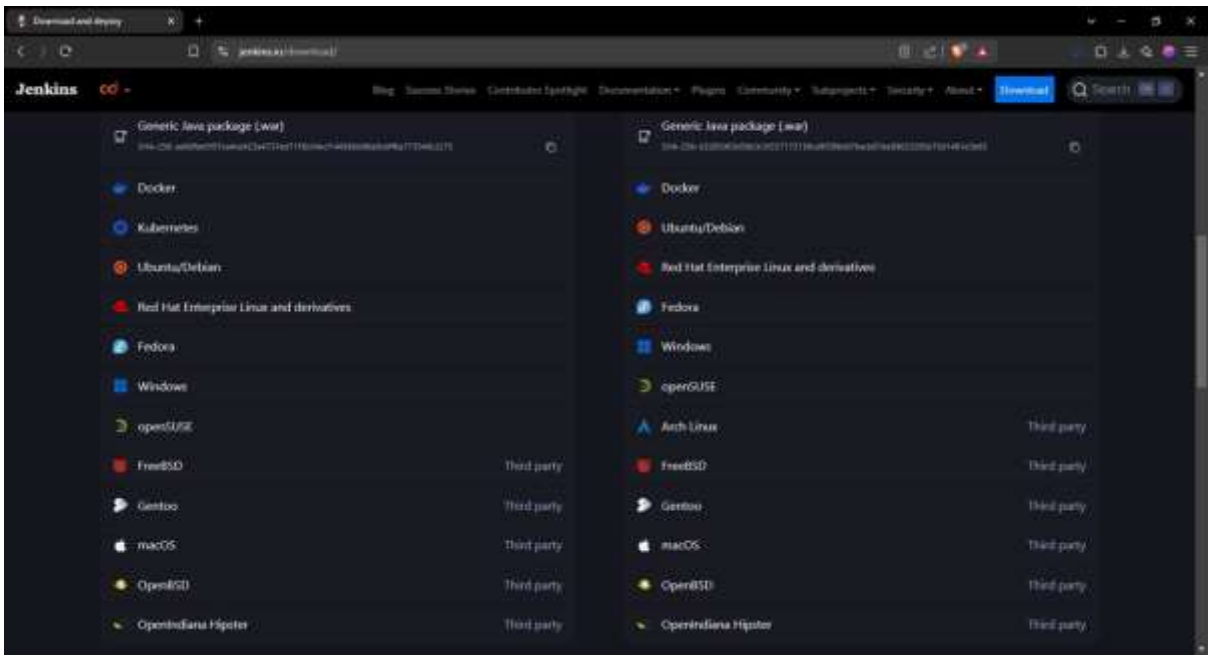
Troubleshooting Jenkins Issues:

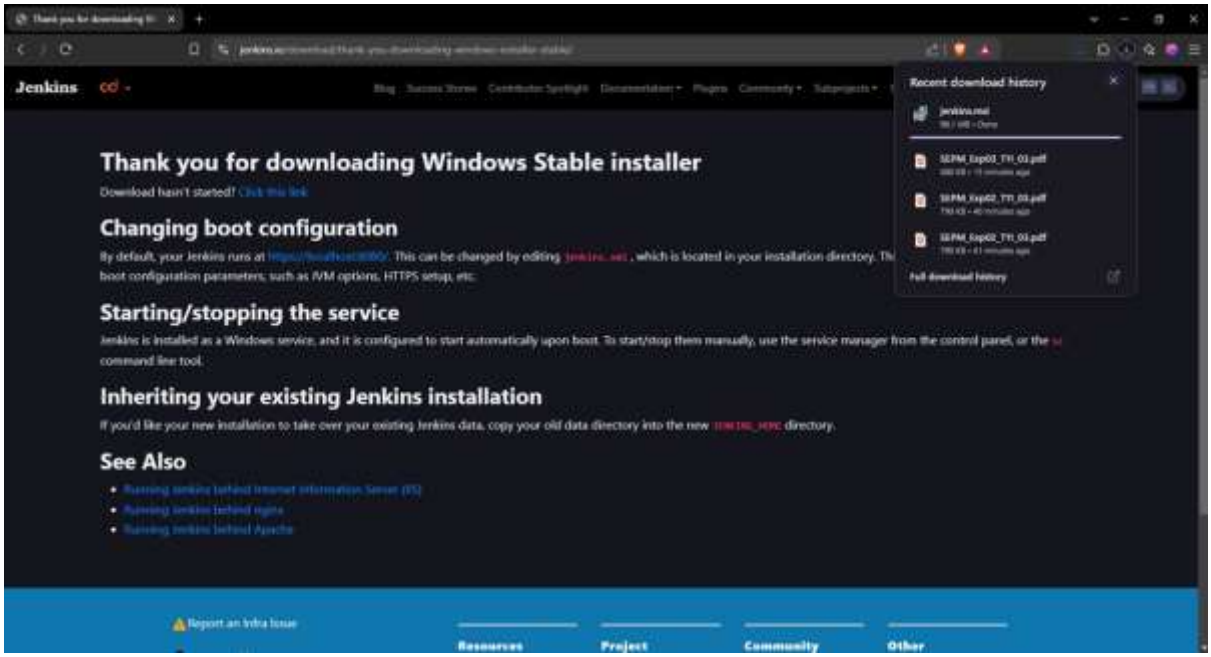
- Build Fails: Check the console output for error logs.
- Permissions Issues: Verify user permissions in Jenkins settings.
- Plugin Conflicts: Update or reinstall problematic plugins.

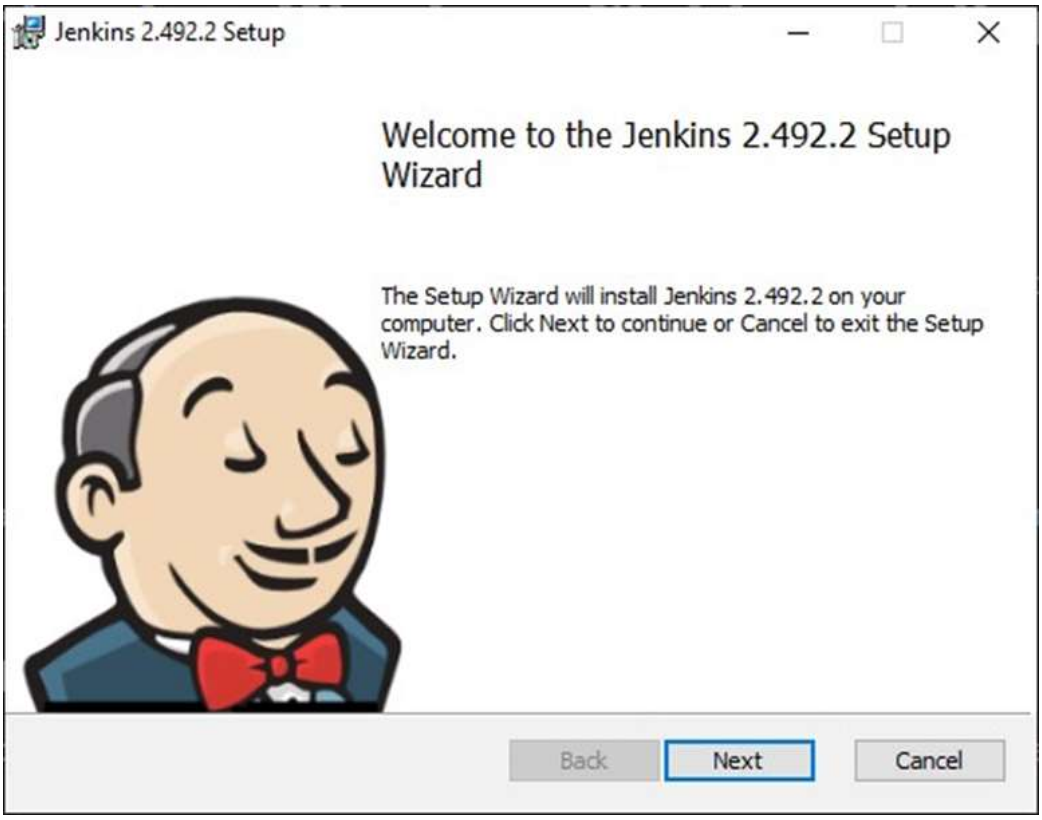
Summary of Key Jenkins Commands:

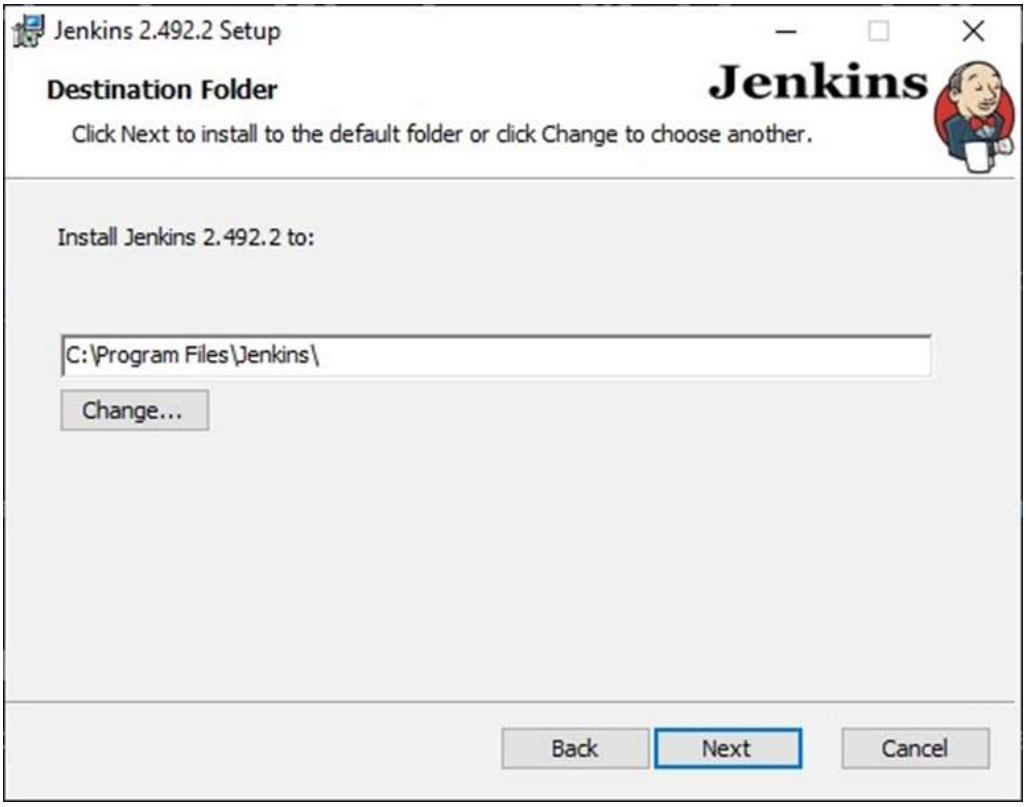
Start Jenkins	-	sudo systemctl start jenkins
Stop Jenkins	-	sudo systemctl stop jenkins
Restart Jenkins	-	sudo systemctl restart jenkins
Check Jenkins Status	-	sudo systemctl status jenkins
Access Jenkins Dashboard	-	http://localhost:8080
Install Plugins	-	Manage Jenkins → Manage Plugins
Run Pipeline	-	Click “Build Now” Manually
View Build Logs	-	Click on a build → Console Output

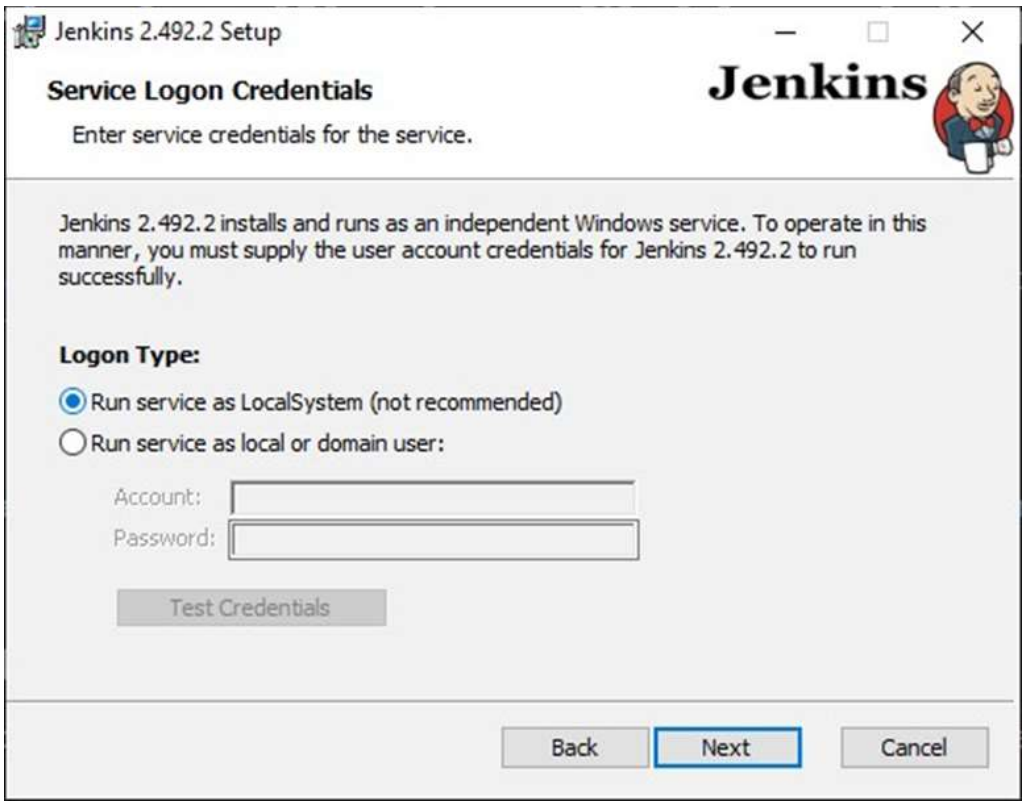
DEMONSTRATION:

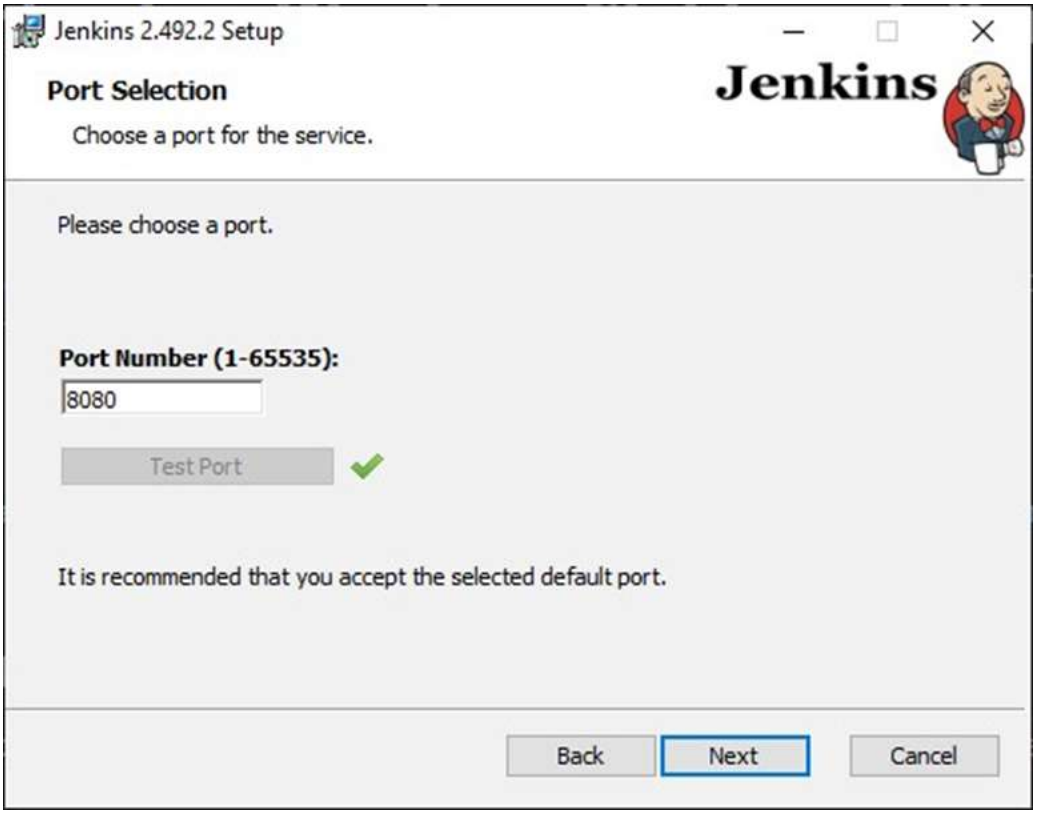


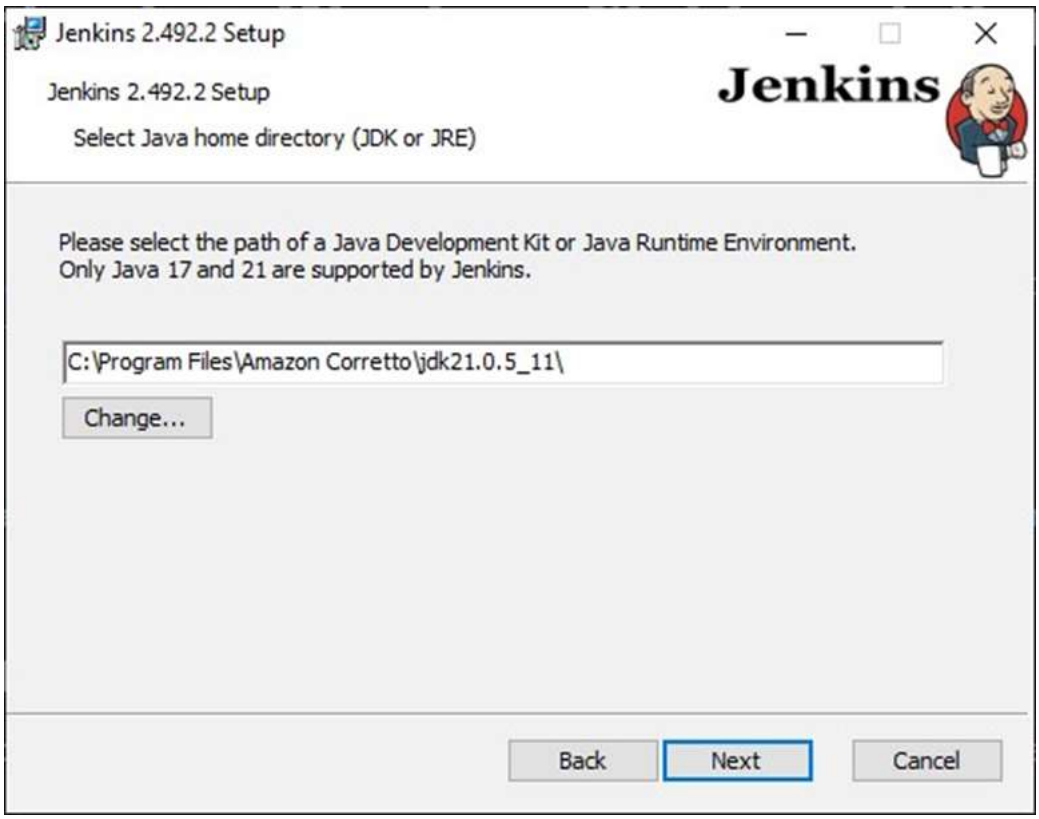

Jenkins
Recent download history
Thank you for downloading Windows Stable installer
Changing boot configuration
Starting/stopping the service
Inheriting your existing Jenkins installation
See Also
Report an Infra issue
Resources
Project
Community
Other

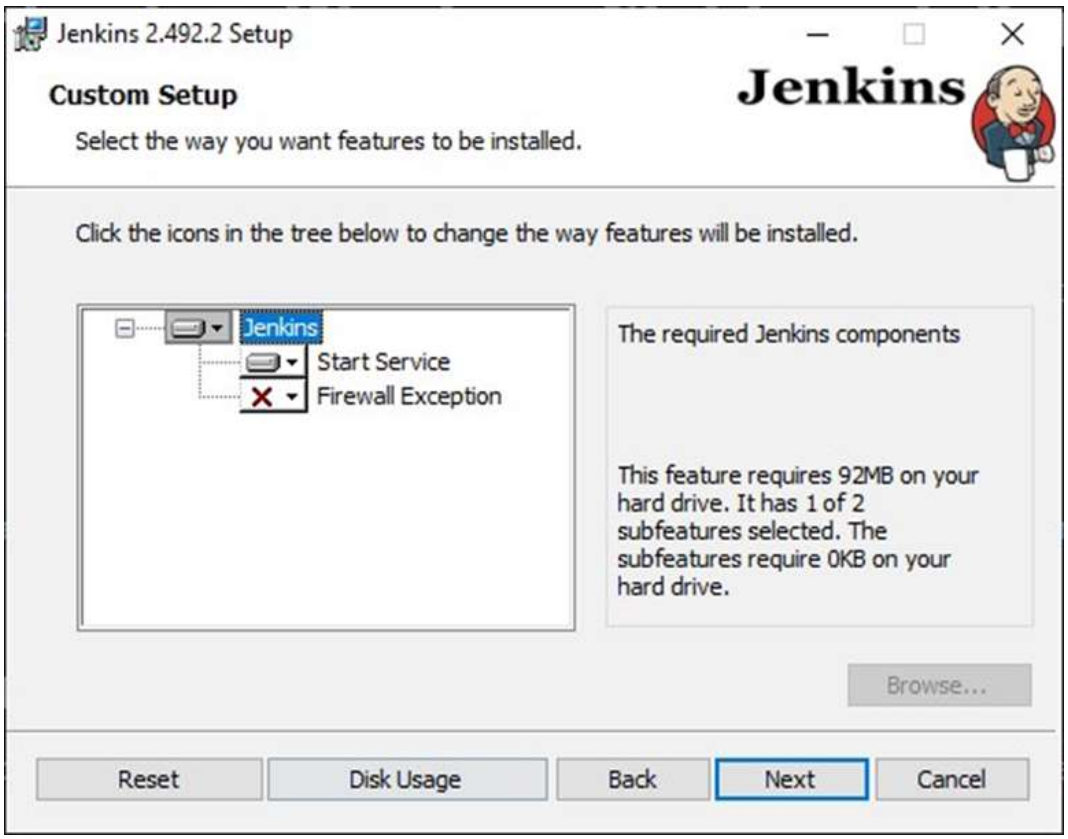

Jenkins 2.492.2 Setup
Welcome to the Jenkins 2.492.2 Setup Wizard
The Setup Wizard will install Jenkins 2.492.2 on your computer. Click Next to continue or Cancel to exit the Setup Wizard.
Back
Next
Cancel

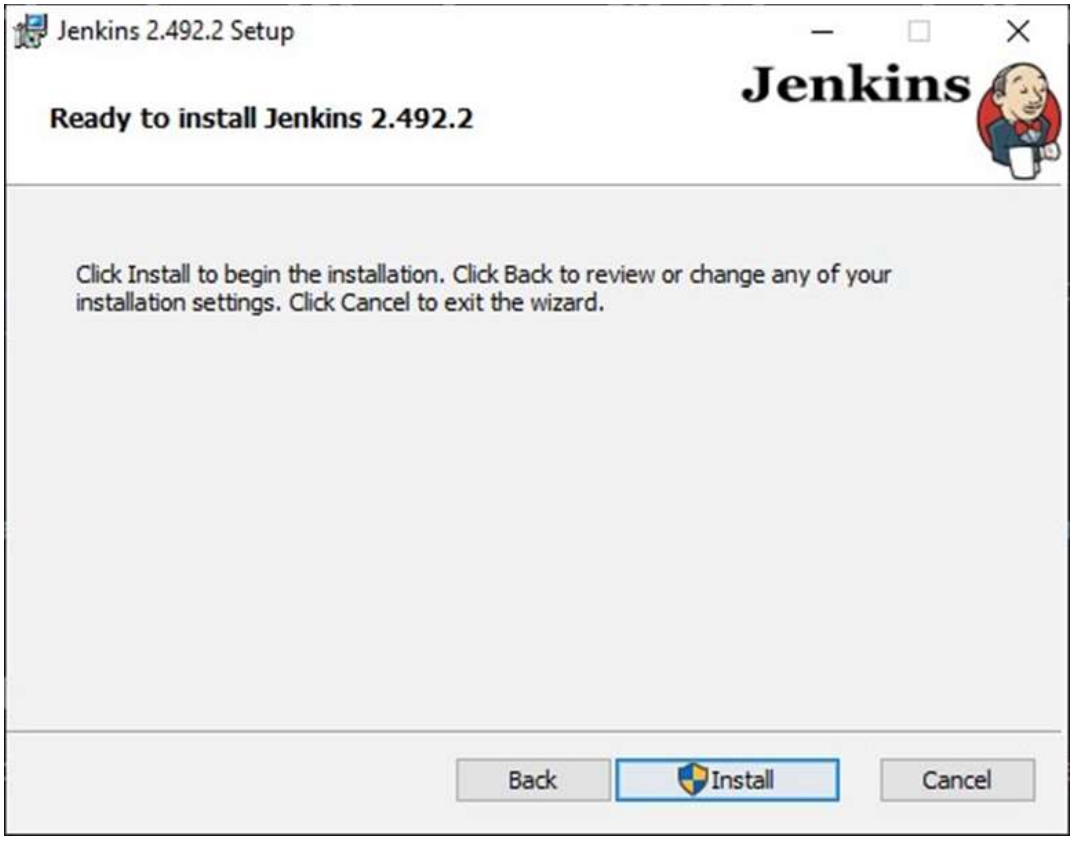

Jenkins 2.492.2 Setup
Destination Folder
Click Next to install to the default folder or click Change to choose another.
Jenkins
Install Jenkins 2.492.2 to:
C:\Program Files\Jenkins\
Change...
Back
Next
Cancel

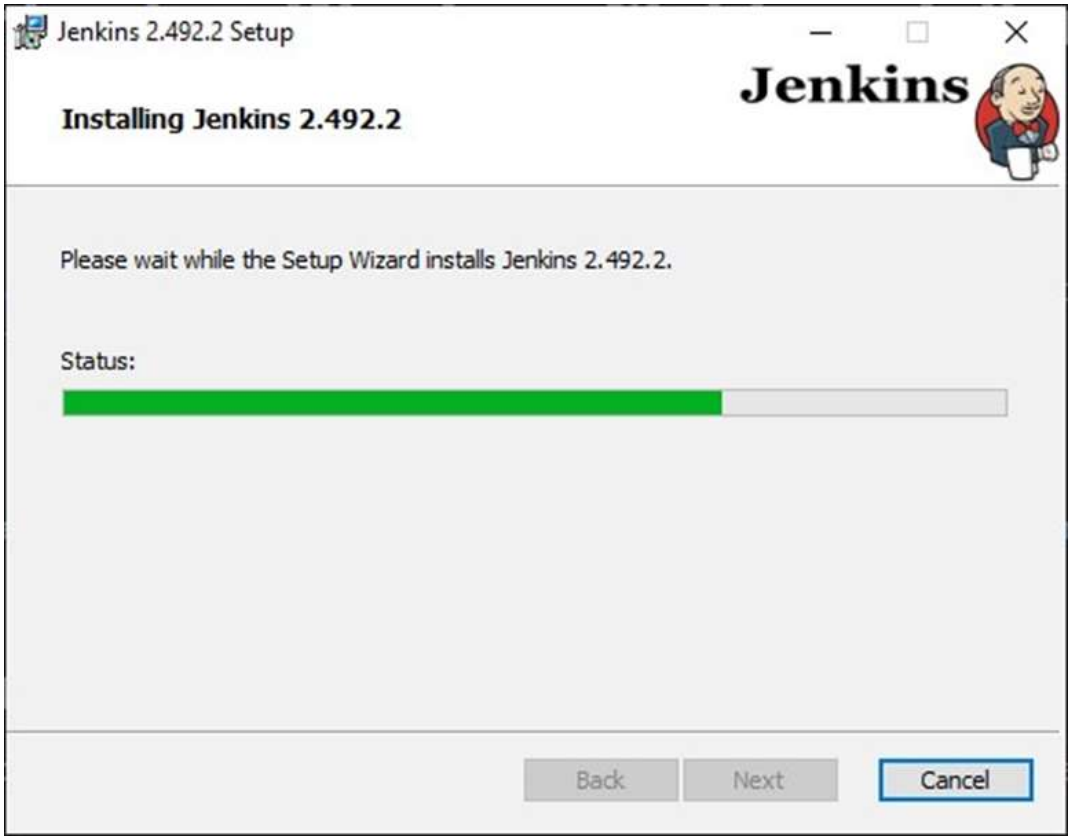

Jenkins 2.492.2 Setup
Service Logon Credentials
Enter service credentials for the service.
Jenkins
Jenkins 2.492.2 installs and runs as an independent Windows service. To operate in this manner, you must supply the user account credentials for Jenkins 2.492.2 to run successfully.
Logon Type:
Run service as LocalSystem (not recommended)
Run service as local or domain user:
Account:
Password:
Test Credentials
Back
Next
Cancel

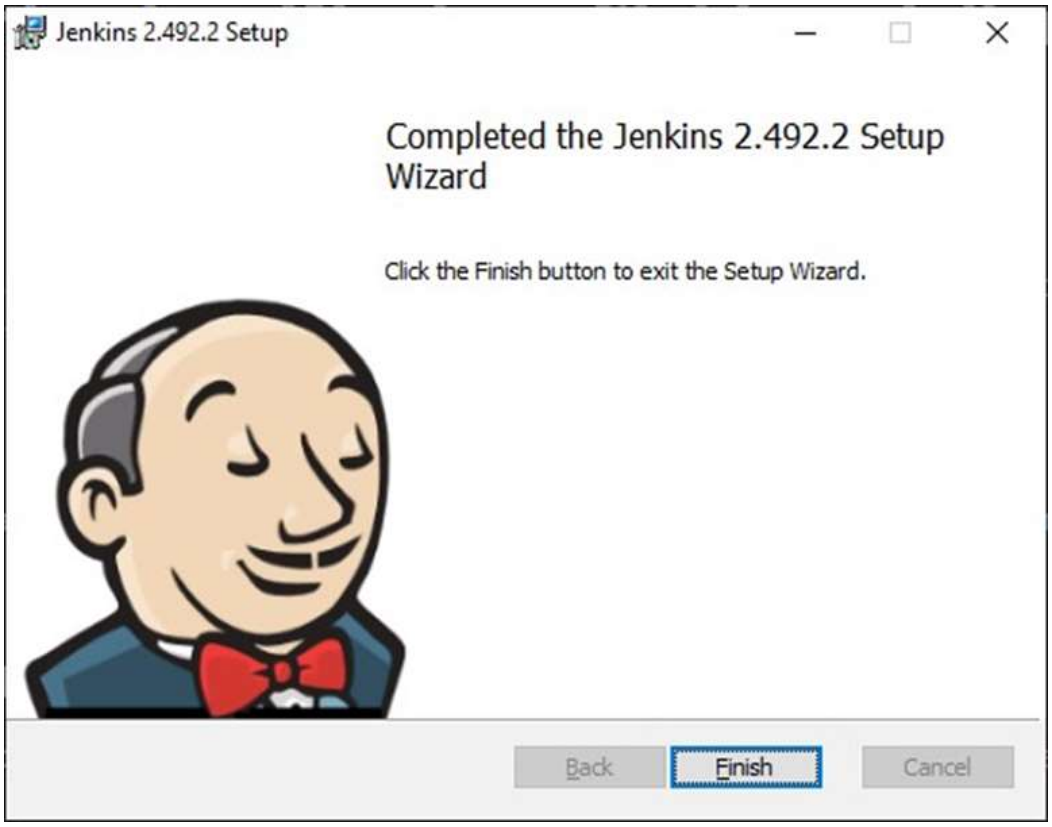

Jenkins 2.492.2 Setup
Port Selection
Choose a port for the service.
Jenkins
Please choose a port.
Port Number (1-65535):
8080
Test Port
It is recommended that you accept the selected default port.
Back
Next
Cancel


Jenkins 2.492.2 Setup
Jenkins 2.492.2 Setup
Select Java home directory (JDK or JRE)
Jenkins
Please select the path of a Java Development Kit or Java Runtime Environment.
Only Java 17 and 21 are supported by Jenkins.
C:\Program Files\Amazon Corretto\jdk21.0.5_11\
Change...
Back
Next
Cancel

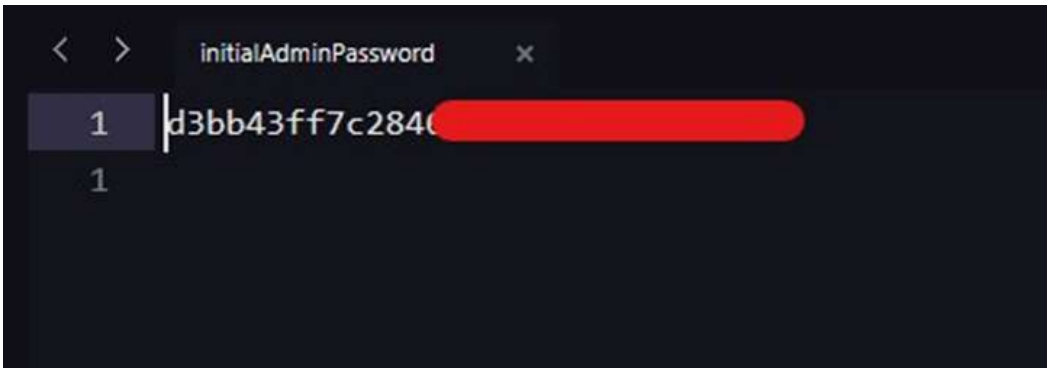

Jenkins 2.492.2 Setup
Custom Setup
Select the way you want features to be installed.
Jenkins
Click the icons in the tree below to change the way features will be installed.
Jenkins
Start Service
Firewall Exception
The required Jenkins components
This feature requires 92MB on your hard drive. It has 1 of 2 subfeatures selected. The subfeatures require 0KB on your hard drive.
Browse...
Reset
Disk Usage
Back
Next
Cancel

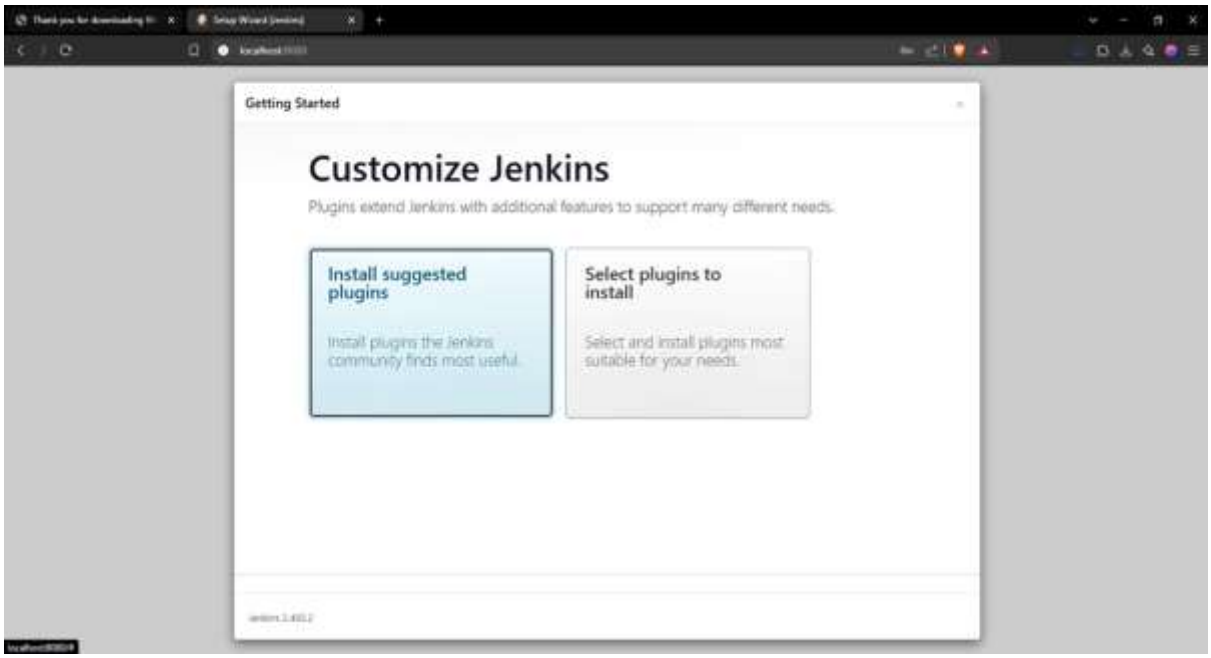

Jenkins 2.492.2 Setup
Ready to install Jenkins 2.492.2
Jenkins
Click Install to begin the installation. Click Back to review or change any of your installation settings. Click Cancel to exit the wizard.
Back
Install
Cancel

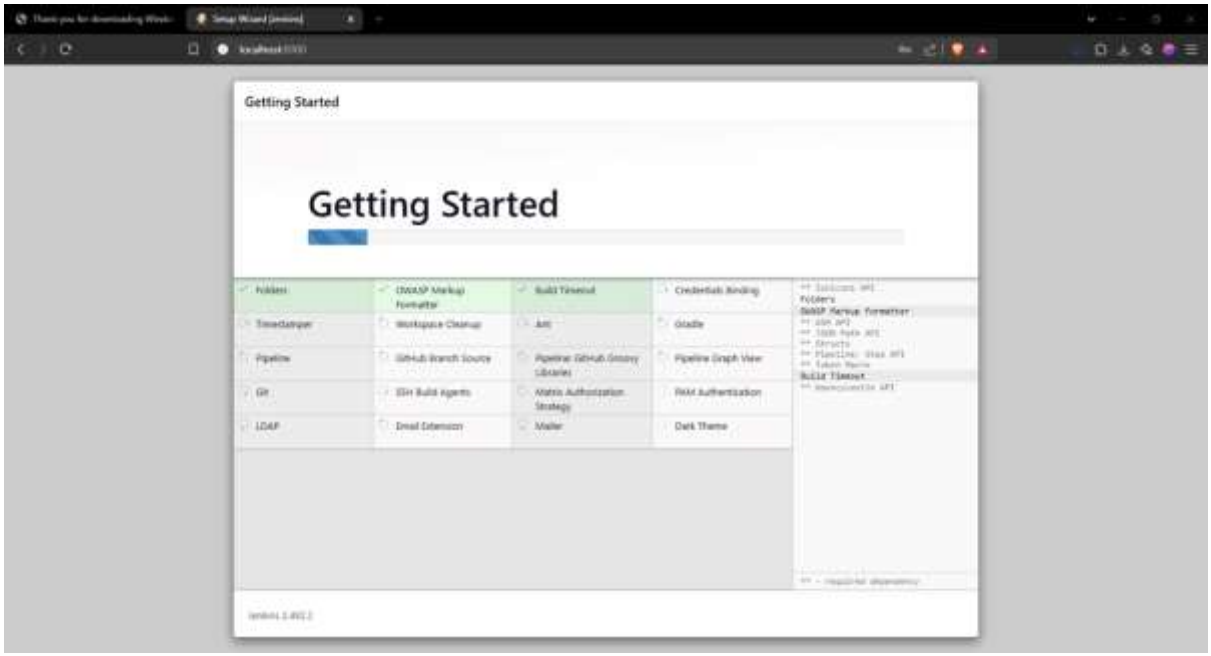

Jenkins 2.492.2 Setup
Installing Jenkins 2.492.2
Jenkins
Please wait while the Setup Wizard installs Jenkins 2.492.2.
Status:
Back
Next
Cancel

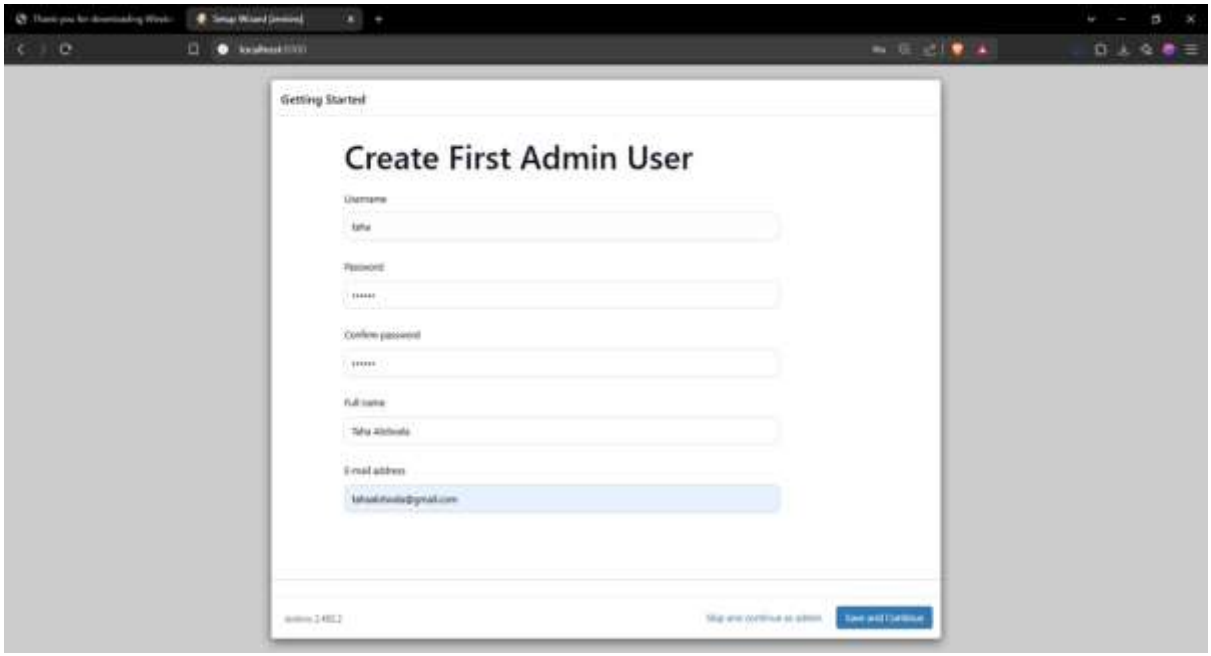

Jenkins 2.492.2 Setup
Completed the Jenkins 2.492.2 Setup Wizard
Click the Finish button to exit the Setup Wizard.
Back
Finish
Cancel

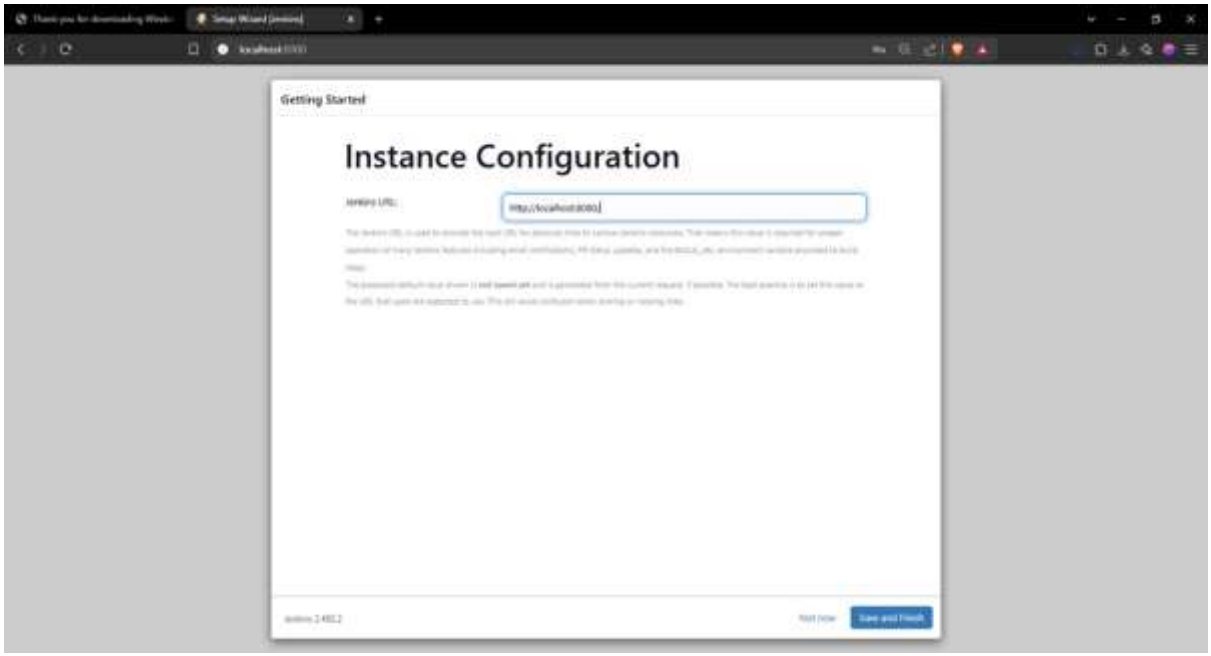

Getting Started
Unlock Jenkins
To ensure Jenkins is securely set up by the administrator, a password has been written to the log (not sure where to find it?) and this file on the server:
Please copy the password from either location and paste it below.
Administrator password
Continue

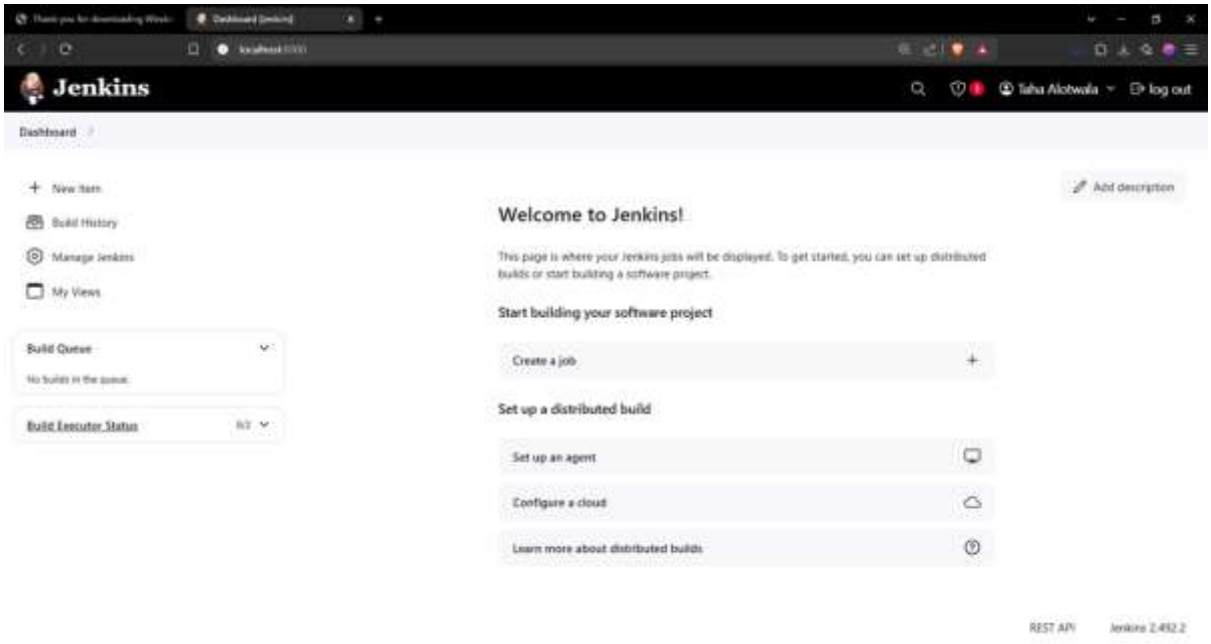

initialAdminPassword
1
d3bb43ff7c284
1

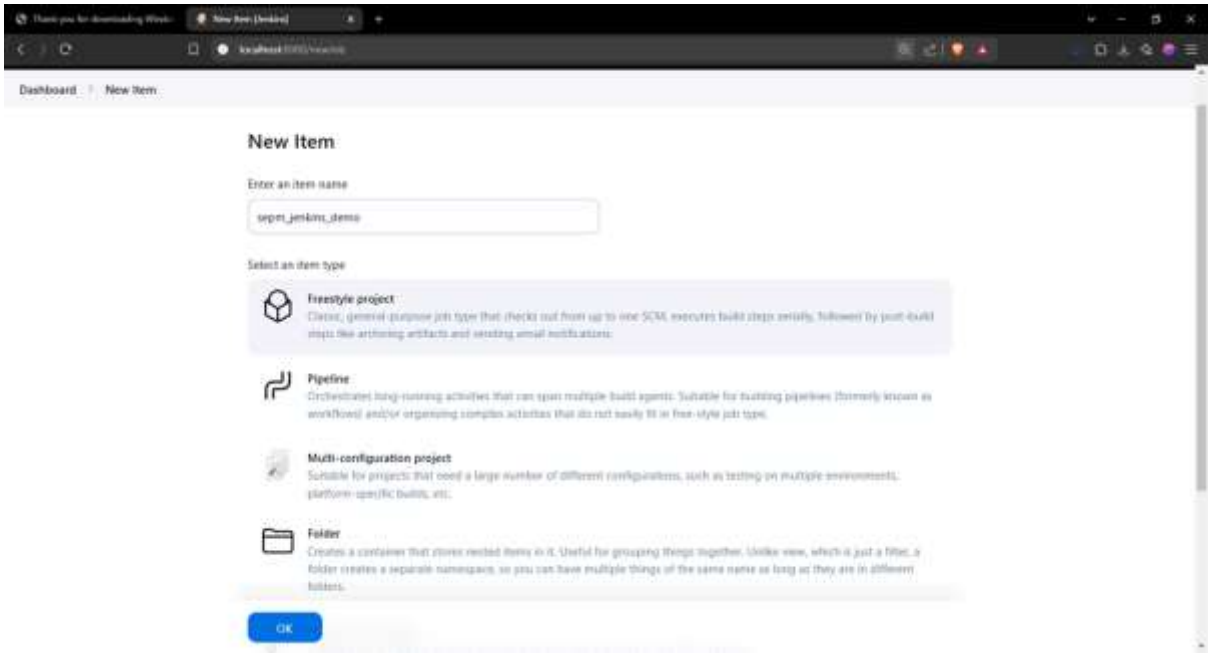

Getting Started
Customize Jenkins
Plugins extend Jenkins with additional features to support many different needs.
Install suggested plugins
Install plugins the Jenkins community finds most useful.
Select plugins to install
Select and install plugins most suitable for your needs.

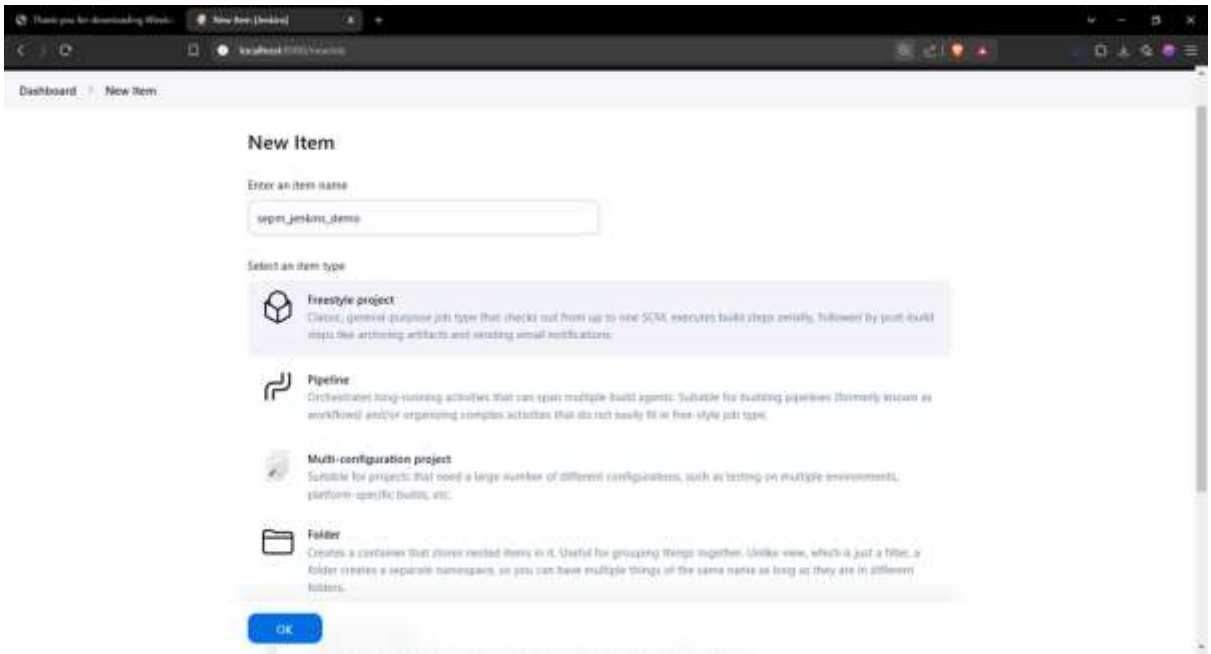

Getting Started
Getting Started

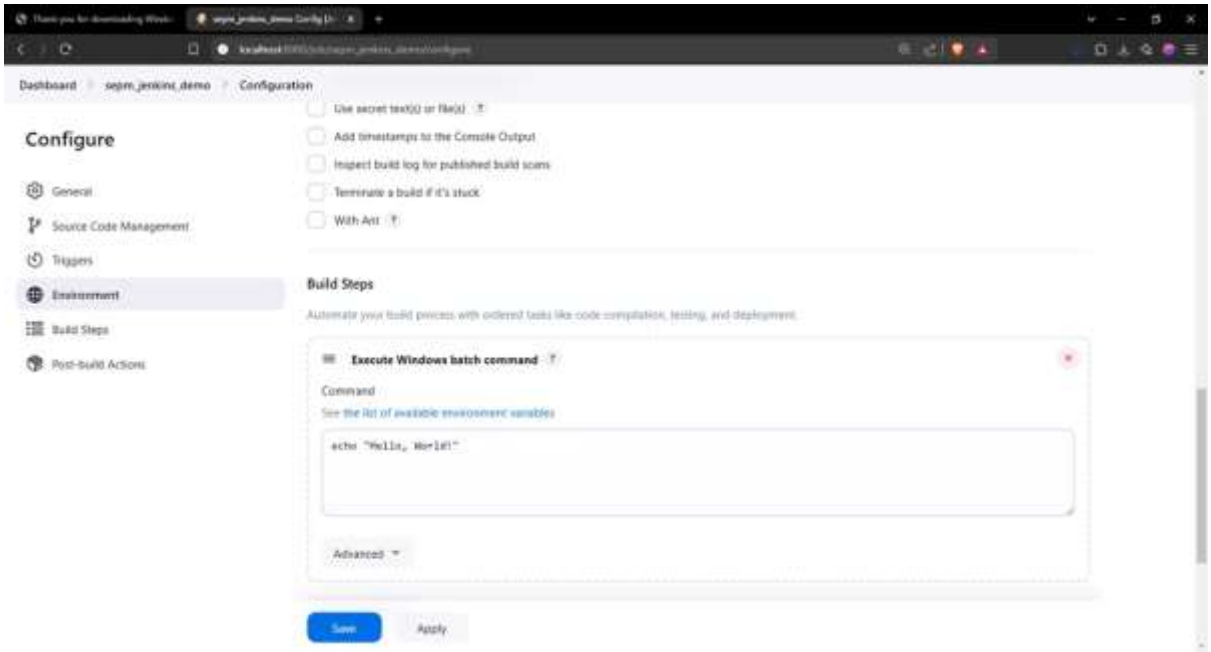

Getting Started
Create First Admin User

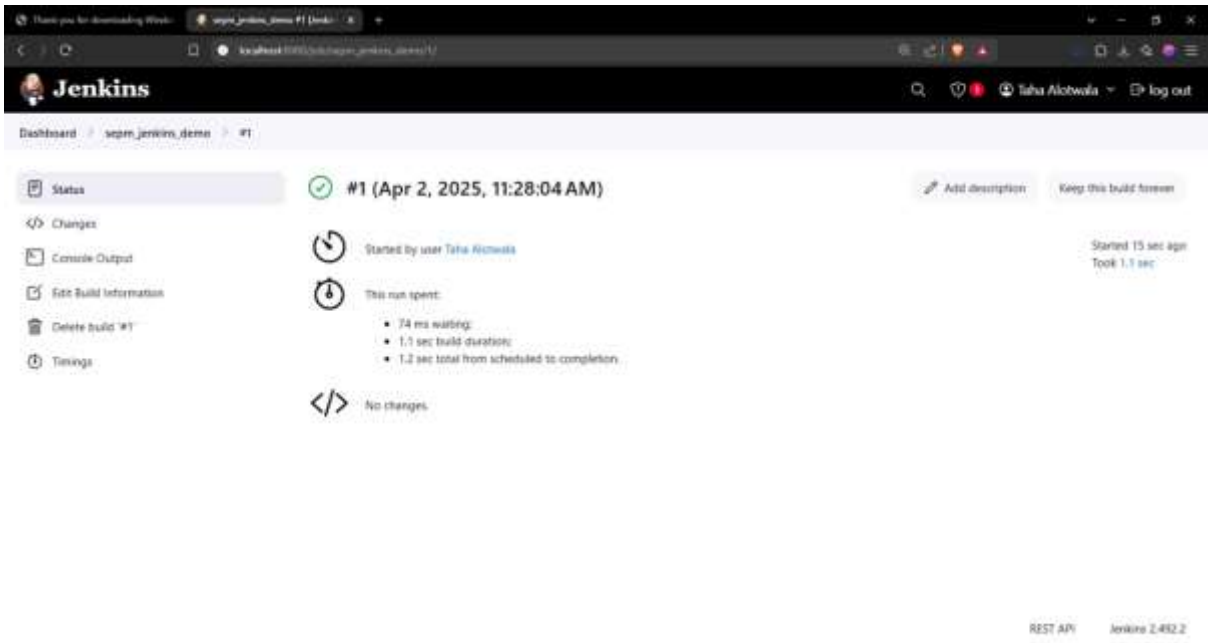

Getting Started
Instance Configuration

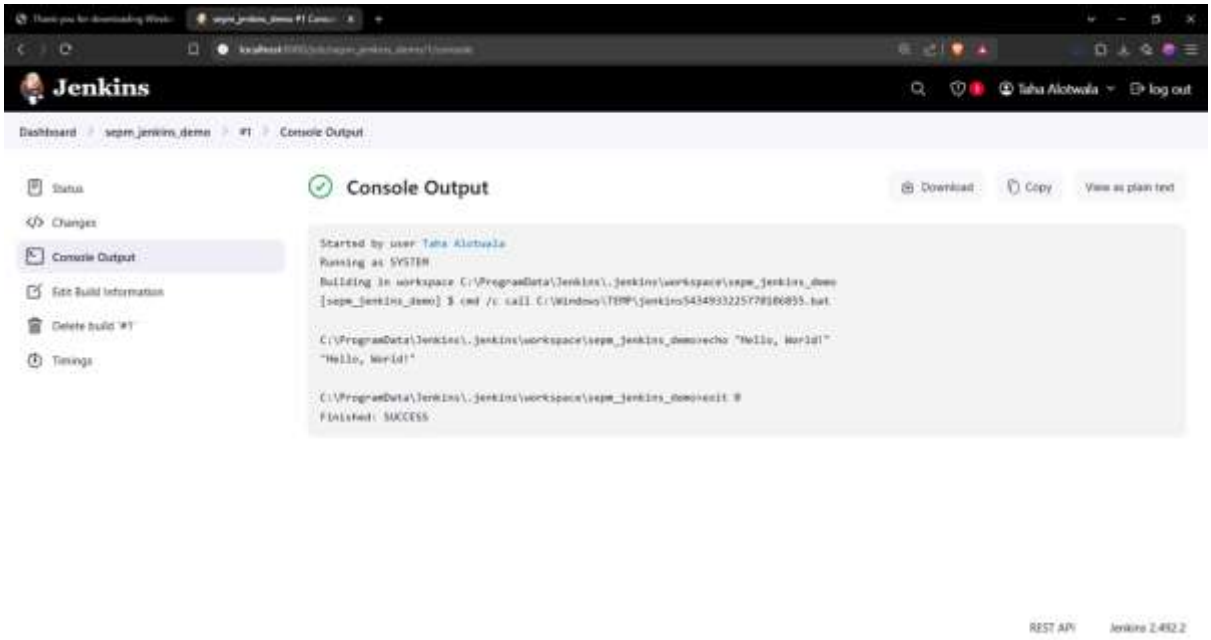

Jenkins
Taha Alotwala
log out
Dashboard
New Item
Build History
Manage Jenkins
My Views
Build Queue
No builds in the queue.
Build Executor Status
Add description
Welcome to Jenkins!
This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.
Start building your software project
Create a job
Set up a distributed build
Set up an agent
Configure a cloud
Learn more about distributed builds
REST API
Jenkins 2.452.2


Dashboard
New Item
New Item
Enter an item name
sepm_jenkins_demo
Select an item type
Freestyle project
Pipeline
Multi-configuration project
Folder
OK


Dashboard
New Item
New Item
Enter an item name
sepm_jenkins_demo
Select an item type
Freestyle project
Pipeline
Multi-configuration project
Folder
OK


Dashboard
sepm_jenkins_demo
Configuration
Configure
General
Source Code Management
Triggers
Environment
Build Steps
Post-build Actions
Add timestamps to the Console Output
Inspect build log for published build scans
Terminate a build if it's stuck
With Ant
Build Steps
Execute Windows batch command
Command
echo "Hello, World!"
Advanced
Save
Apply


Jenkins
Taha Alotwala
log out
Dashboard
sepm_jenkins_demo
#1
Status
Changes
Console Output
Edit Build Information
Delete build '#1'
Timings
#1 (Apr 2, 2025, 11:28:04 AM)
Add description
Keep this build forever
Started by user Taha Alotwala
Started 15 sec ago
Took 1.1 sec
This run spent:
74 ms waiting;
1.1 sec build duration;
1.2 sec total from scheduled to completion.
No changes.
REST API
Jenkins 2.492.2



CONCLUSION:

We have successfully installed and configured Jenkins with Maven/Ant/Gradle to setup a build Job and learnt about the implementation of Jenkins in open source continuous integration.