# DEPARTMENT OF INFORMATION TECHNOLOGY

COURSE CODE: DJS22ITHN1L1 DATE: 31-01-2025

COURSE NAME: DevOps Laboratory CLASS: TY BTech

NAME: Anish Sharma ROLL: I011 DIV: IT1-1

# EXPERIMENT NO. 4

CO/LO: Apply DevOps principles to meet software development requirements.

AIM / OBJECTIVE: To implement the pipeline of jobs using Maven in Jenkins, create a pipeline script to Test and deploy an application.

THEORY:

Implementing a Continuous Integration/Continuous Deployment (CI/CD) pipeline in Jenkins for a Maven-based Java application involves several key steps:

1. Setting Up Jenkins and Required Tools:
   * Jenkins Installation: Ensure Jenkins is installed and running.
   * Maven Integration: Configure Maven in Jenkins by navigating to

"Manage Jenkins" > "Global Tool Configuration" and adding a Maven installation. o Version Control: Integrate your source code repository (e.g., GitHub) with Jenkins.

1. Creating the Jenkins Pipeline:
   * Pipeline Script (Jenkinsfile): Define your build, test, and deployment stages in a Jenkinsfile stored in your repository. o Declarative Pipeline Syntax: Utilize Jenkins' declarative pipeline syntax for clarity and maintainability.
2. Defining Pipeline Stages:
   * Build Stage: Compile the code and package it using Maven. o Test Stage: Execute unit tests to ensure code quality. o Deploy Stage: Deploy the application to the desired environment, such as a web server like Apache Tomcat.
3. Sample Jenkinsfile: Below is an example of a declarative Jenkins Pipeline script for a Maven project:

groovy CopyEdit pipeline { agent any tools {

maven 'Maven' // Assumes 'Maven' is configured in Global Tool Configuration

} stages { stage('Checkout') {

steps { git 'https://github.com/your-repo/your-project.git'

}

}

stage('Build') { steps { sh 'mvn clean package -DskipTests'

}

}

stage('Test') { steps { sh 'mvn test'

} post { always { junit 'target/surefire-reports/\*.xml'

}

}

}

stage('Deploy') { steps {

// Deployment steps, e.g., copying files to a server sh 'scp target/your-app.war user@server:/path/to/deploy/'

}

} } post { cleanup { cleanWs()

}

}

}

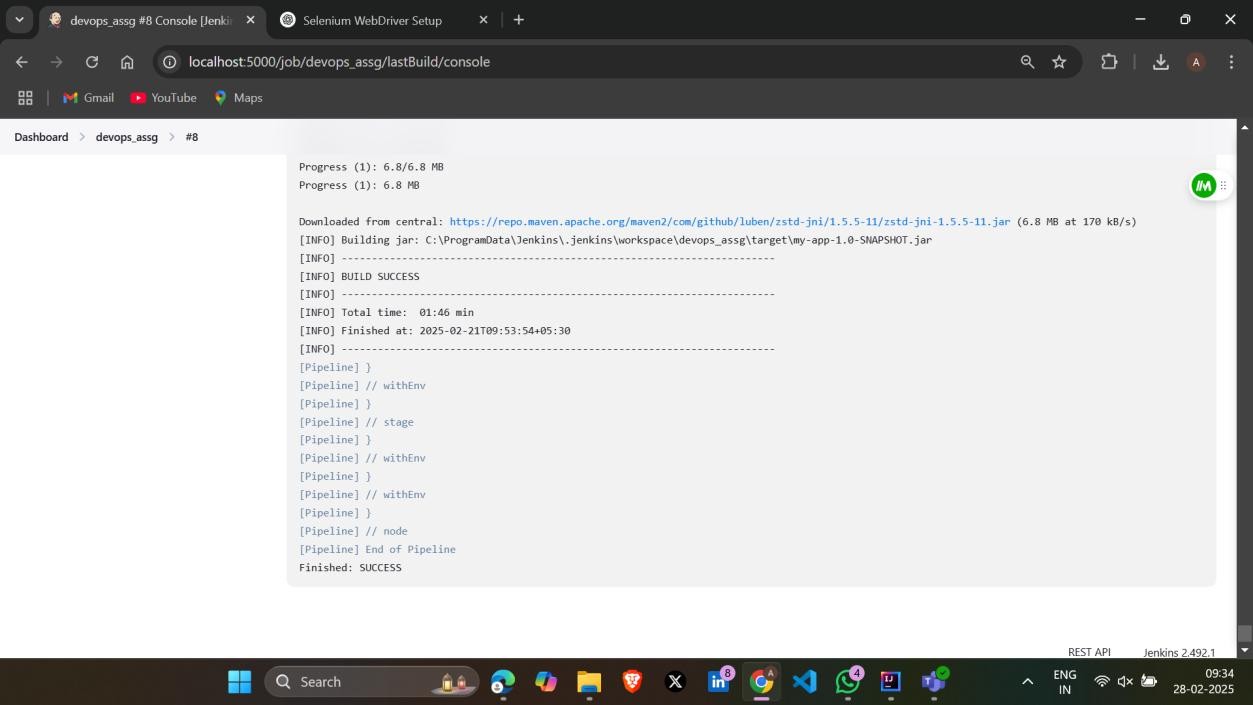
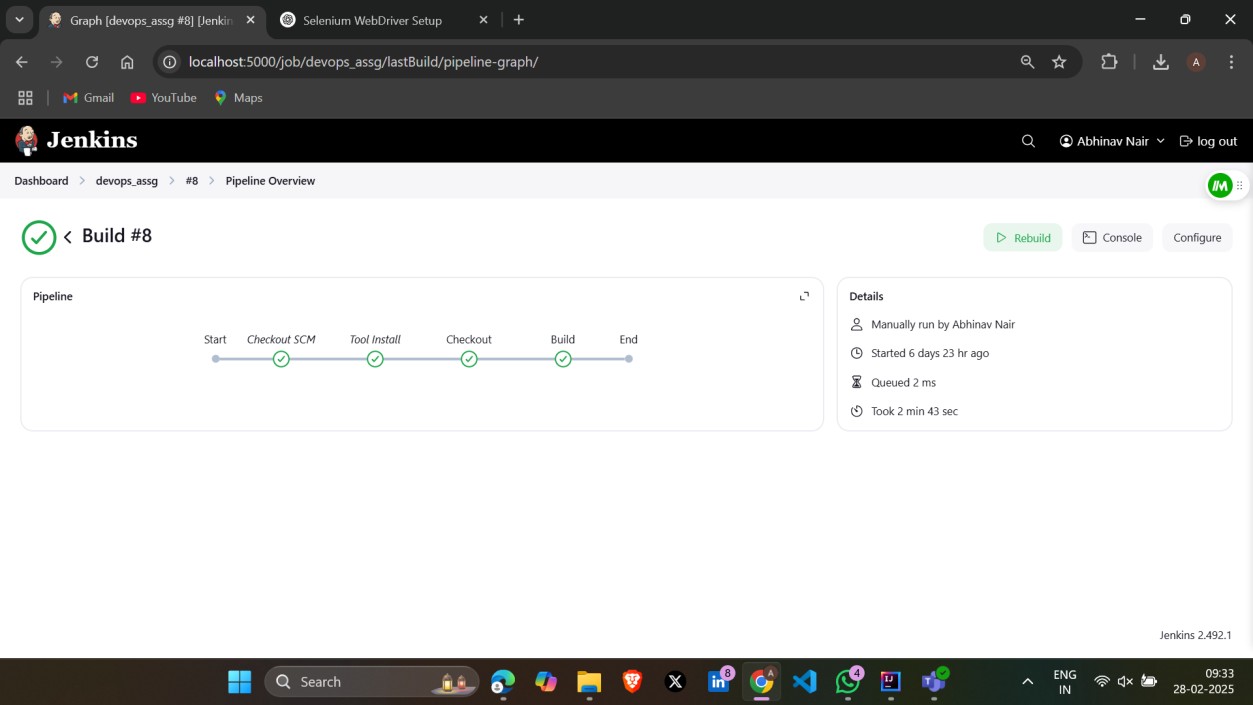
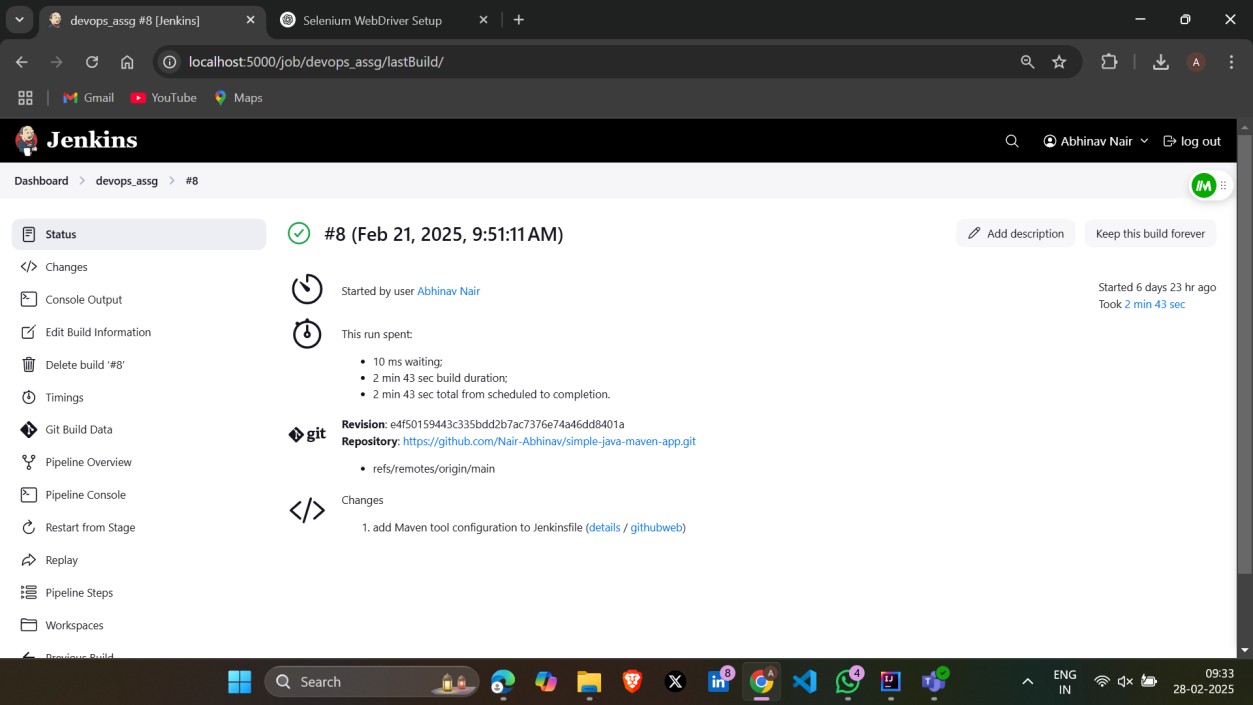
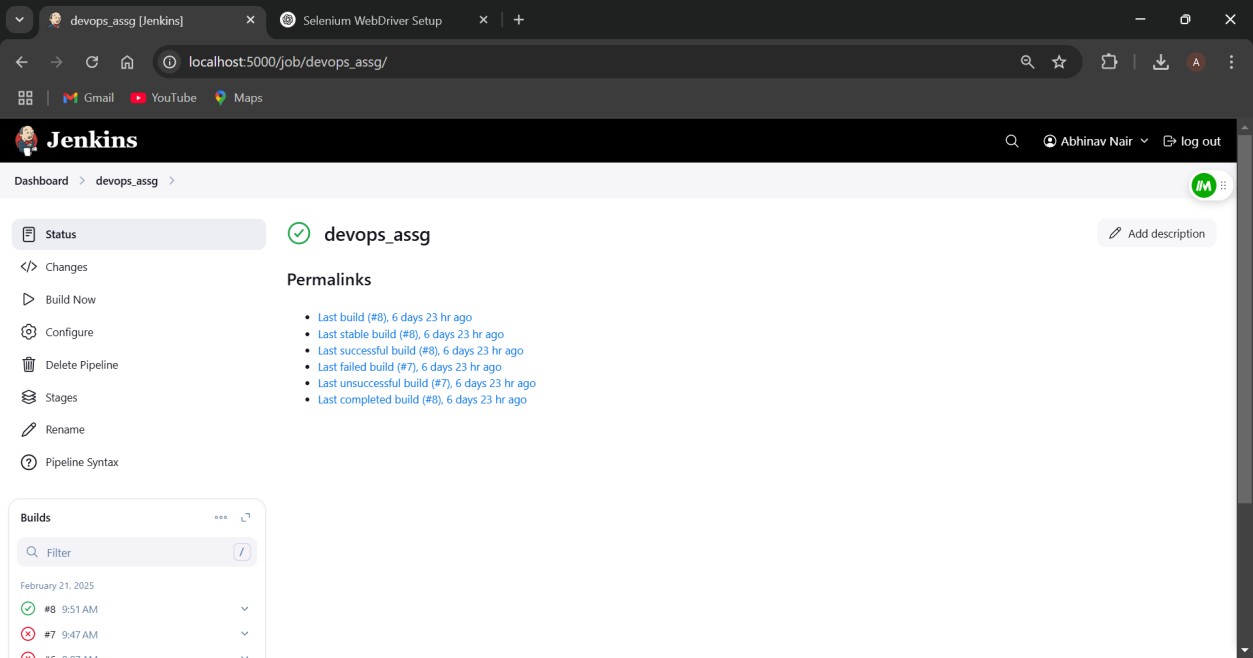
Explanation:

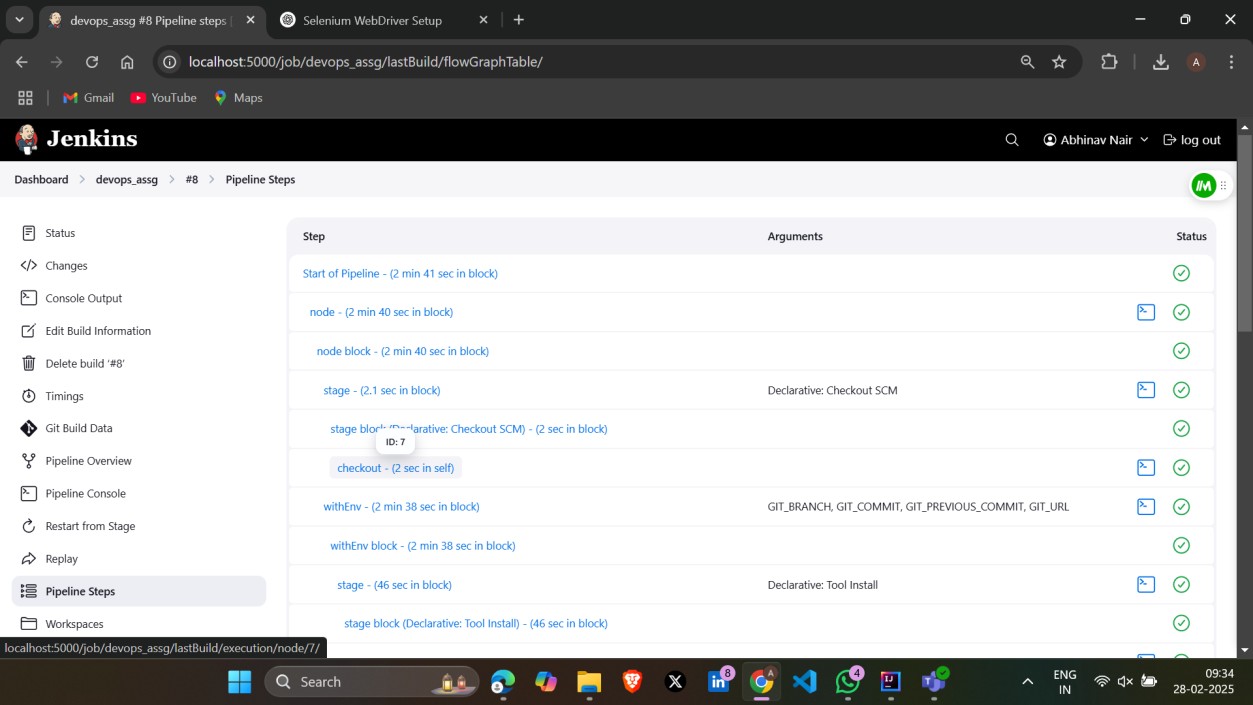
o tools: Specifies the Maven installation to use. o stages: Defines the sequence of stages: Checkout, Build, Test, and Deploy. o post: Contains actions to perform after each stage or the entire pipeline, such as archiving test results or cleaning up the workspace.

5. Enhancing the Pipeline:

o Parallel Testing: Run tests in parallel to reduce execution time. o Environment-Specific Deployments: Use parameters to deploy to different environments (development, staging, production). o Notifications: Integrate with communication tools (e.g., email, Slack) to send build and deployment notifications.

OUTPUT:





Conclusion:

In this experiment, we implemented the pipeline of jobs using Maven in Jenkins, create a pipeline script to Test and deploy an application.

References:

1. [How to Use Git and GitHub – Version Control Basics for Beginners (freecodecamp.org)](https://www.freecodecamp.org/news/git-and-github-the-basics/)
2. [Version Control Systems - GeeksforGeeks](https://www.geeksforgeeks.org/version-control-systems/)
3. [VCS Program Details - Verra](https://verra.org/programs/verified-carbon-standard/vcs-program-details/)