Module 4 Assignment 2

Name: Jishnu Ganesh
Course: CSC 7435E – Secure DevOps Practices
Assignment: Module 4 Assignment 2 – Jenkins CI/CD Pipeline
GitHub Repository URL: https://github.com/jishnukrg/JishnuGanesh_M4Assgn1
Jenkins Job Name: BMIAppPipeline

1. Objective

The purpose of this assignment is to implement a CI/CD pipeline using Jenkins and GitHub, replicating the course walkthrough and customizing it for my BMI Calculator app. The goal is to automate code integration, building, testing, packaging, and conditional release processes, ensuring secure and efficient continuous delivery.

2. Jenkins Setup

Steps completed to configure Jenkins:
1. Installed required plugins: Git, Pipeline, and Blue Ocean.
2. Added GitHub Personal Access Token (PAT) under credentials named 'github-pat'.
3. Created a pipeline job named 'BMIAppPipeline' and linked it to my GitHub repository.
4. Configured the branch specifier to '*/main' and ensured the Jenkinsfile was at the repository root.

3. Pipeline Configuration

The Jenkinsfile automates the build, test, and packaging process for the Java BMI application. A conditional release stage was added to ensure releases occur only from the 'main' branch.

4. Jenkinsfile Overview

Stages in the Jenkinsfile include:
• Checkout – Pulls the code from GitHub.
• Build – Compiles Java source files.
• Test – Runs basic verification checks.
• Package – Packages compiled files into a '.tar' artifact.
• Release – Executes only when the branch is 'main'.

5. Modification Description (Required Customization)

The modification involved adding a conditional 'Release' stage that only executes when the build runs on the 'main' branch. This ensures controlled, production-only deployments. The condition uses Jenkins environment variables as shown below:

- when {
  expression { env.GIT_BRANCH?.contains('main') }
  }

This customization was verified by testing both 'main' and 'test-branch'. The main branch successfully triggered a release, while the test branch correctly skipped it.

6. Pipeline Execution Evidence

Pipeline runs were tested under two scenarios:
• Main Branch: Release stage executed successfully.
• Test Branch: Release stage skipped as expected.
The results confirmed that the pipeline logic and automation were implemented correctly.

7. Results and Observations

Branch	Release Stage Behavior	Result
main	✅ Executed successfully	Artifact released and pipeline succeeded
test-branch	⭕ Skipped as expected	No release, build successful

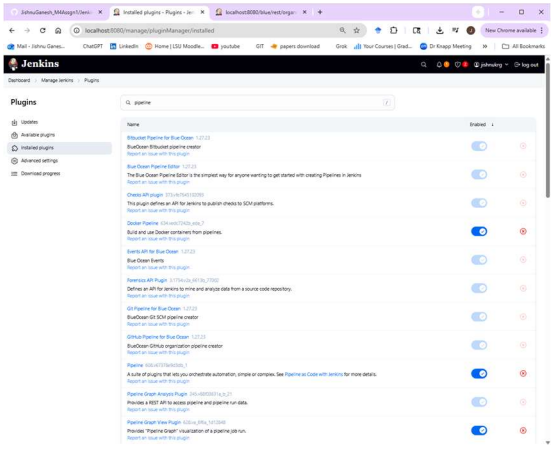
8. Conclusion

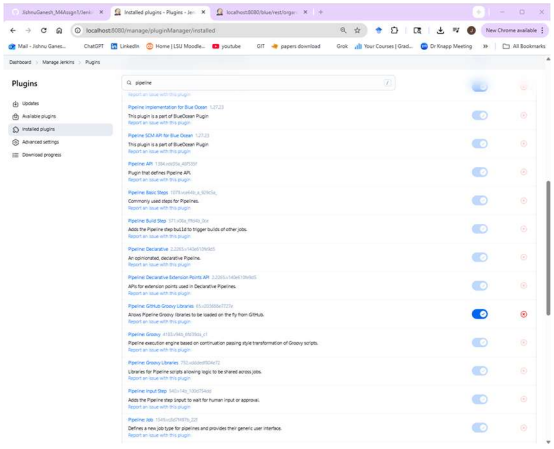
The CI/CD pipeline was implemented and customized successfully using Jenkins and GitHub integration. It automates building, testing, packaging, and conditional deployment for the BMI Calculator app. The modification ensures that only the 'main' branch can trigger releases, promoting stable and controlled deployments. Testing across multiple branches validated the functionality, meeting all assignment objectives and ensuring readiness for future CI/CD enhancements.

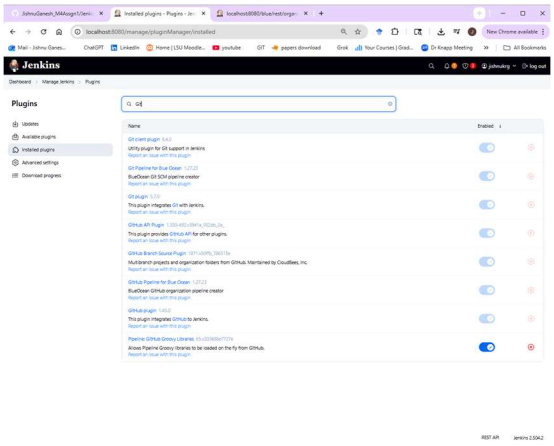
10. Appendix – Screenshots

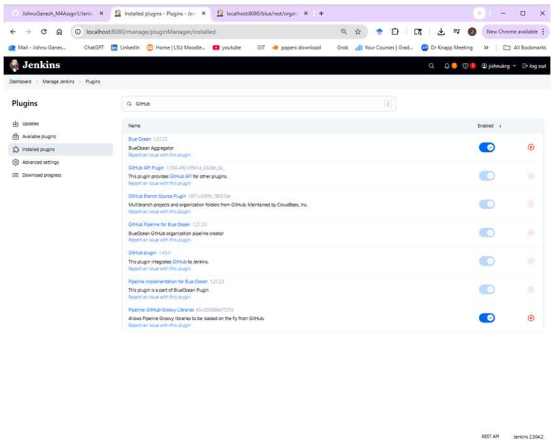
Below are all screenshots documenting the setup and execution of the Jenkins pipeline. Each figure includes a short caption explaining its purpose.

Figures: Jenkins Plugin Manager – Installed Git, Pipeline, and Blue Ocean.









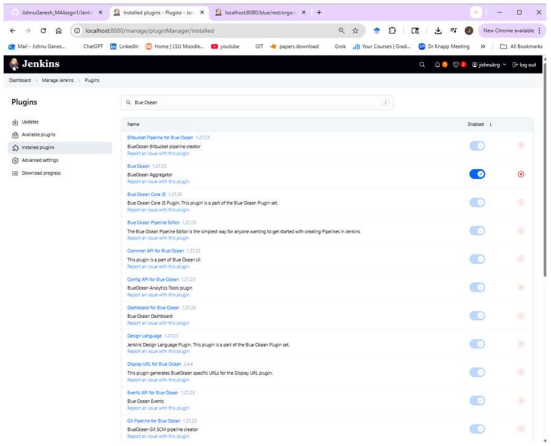


Figure: GitHub Credentials Setup – Personal Access Token added as 'github-pat'.

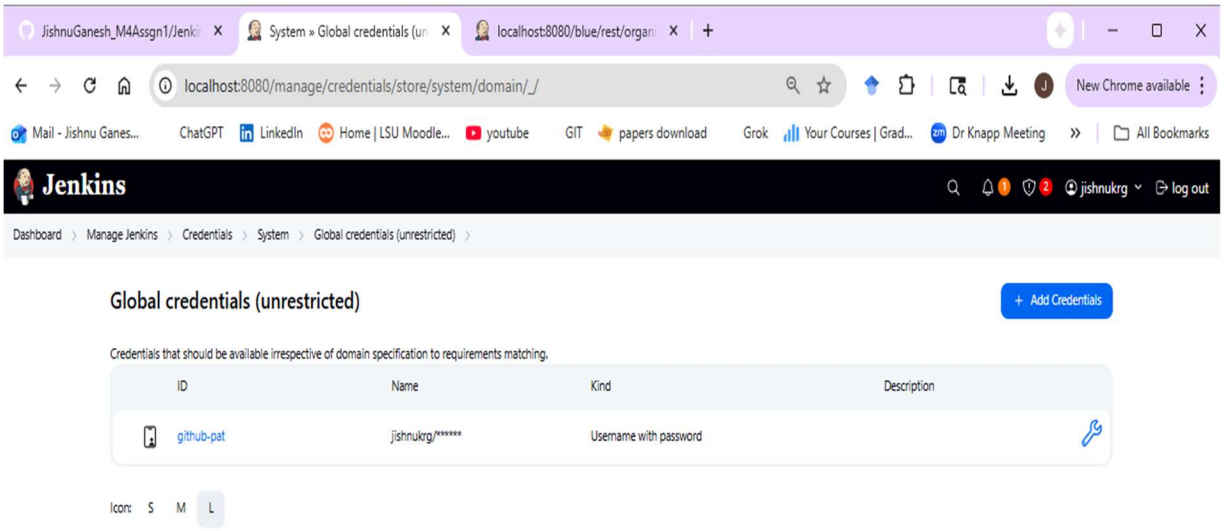


Figure: Jenkins Job Configuration – Repository URL, branch, and Jenkinsfile path set.

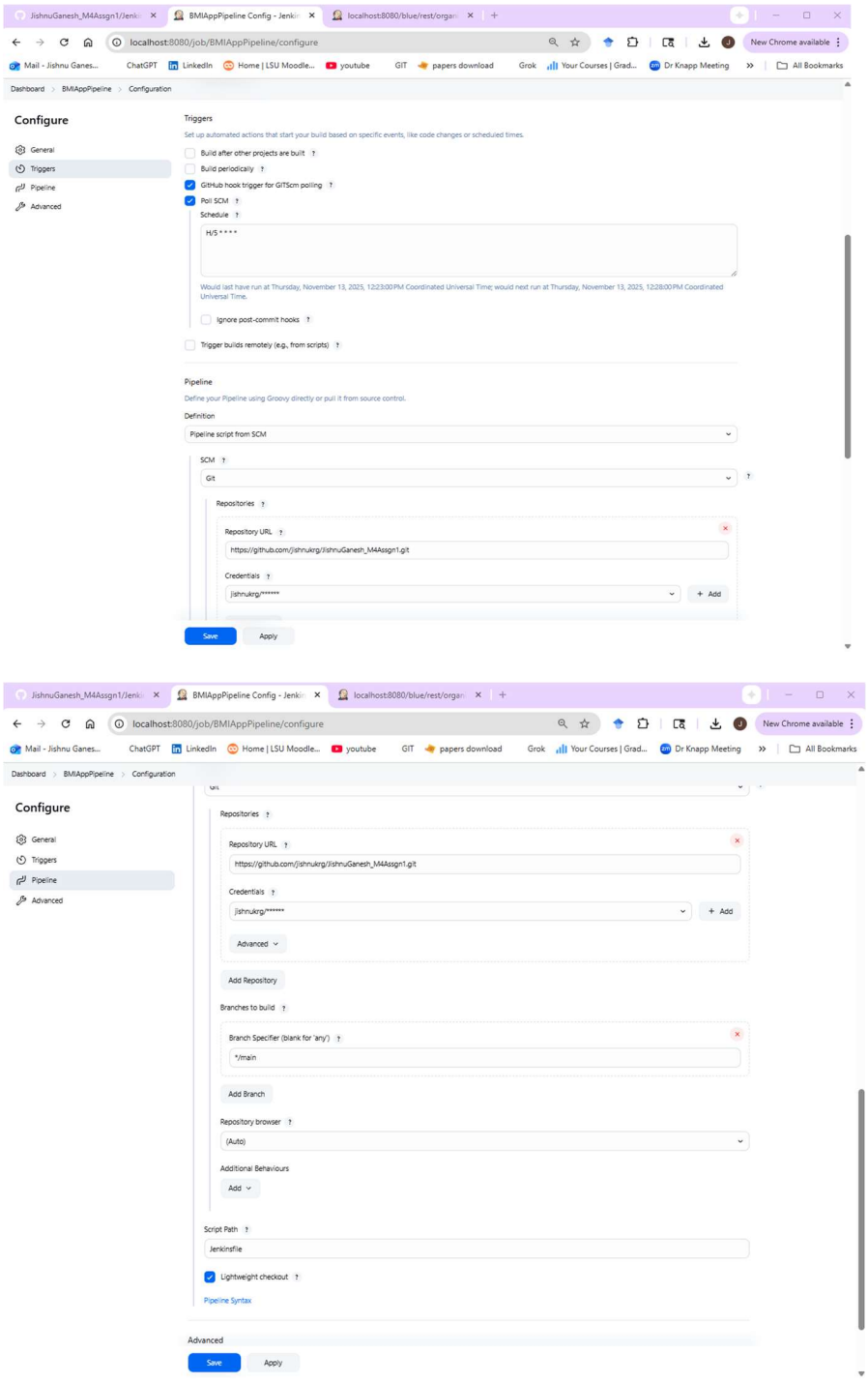


Figure: Blue Ocean – Test Branch showing Release skipped (gray).

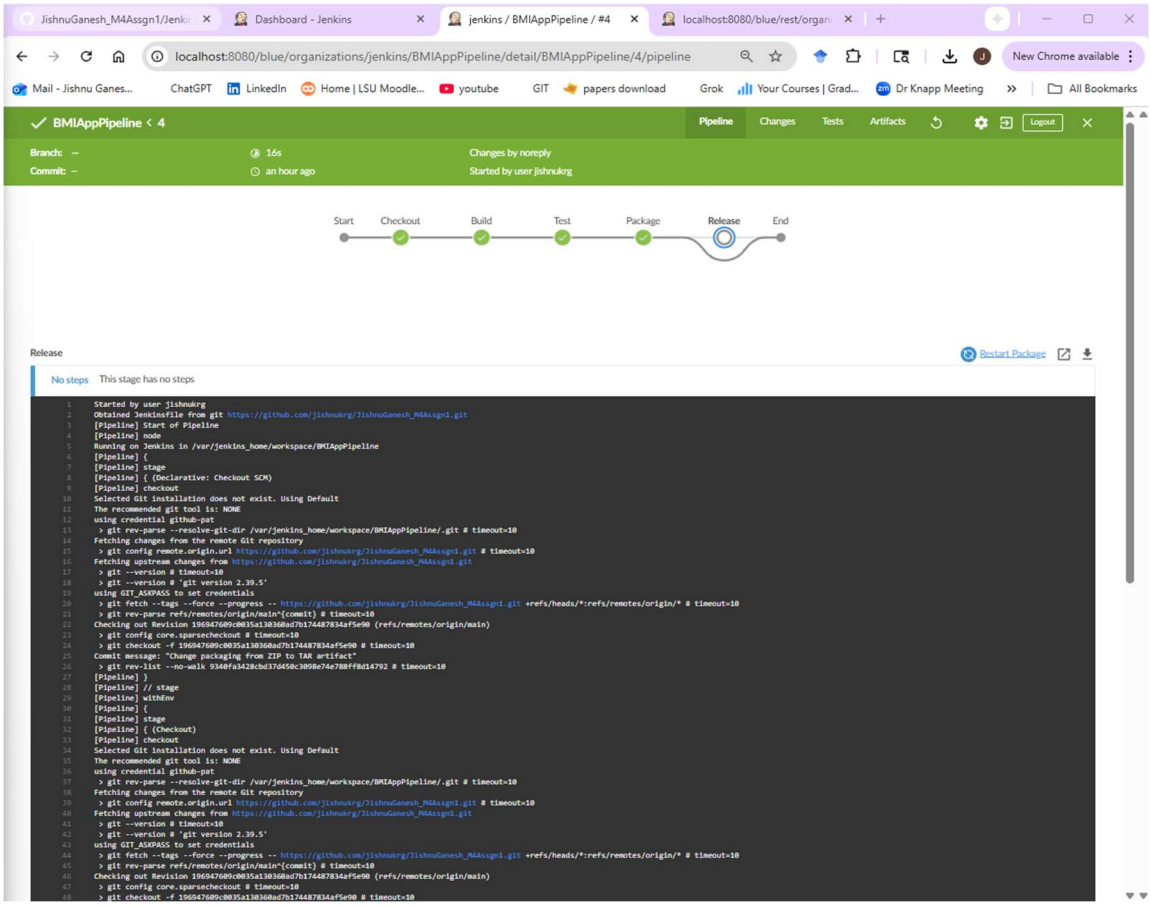


Figure: Blue Ocean – Main Branch showing all stages green including Release.

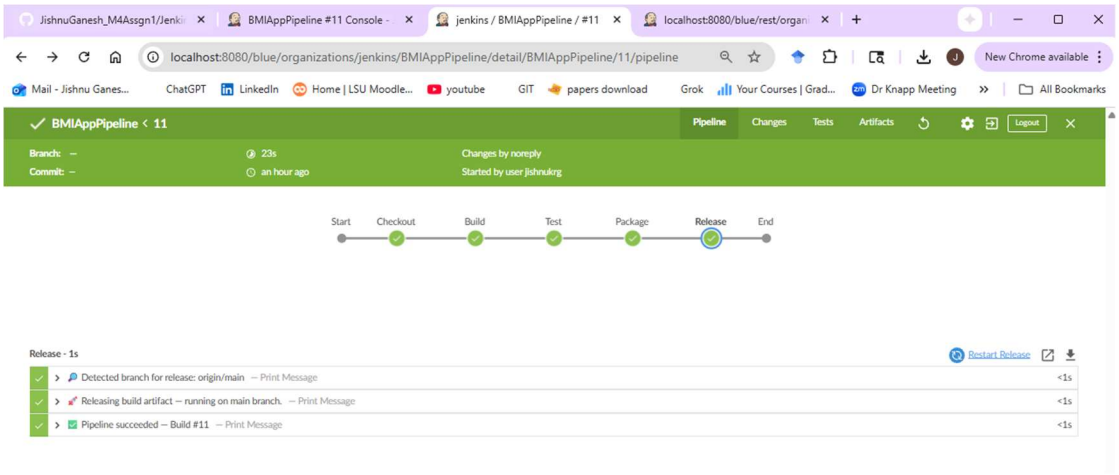


Figure: Console Output – Main Branch successful run with '🚀 Releasing build artifact'.

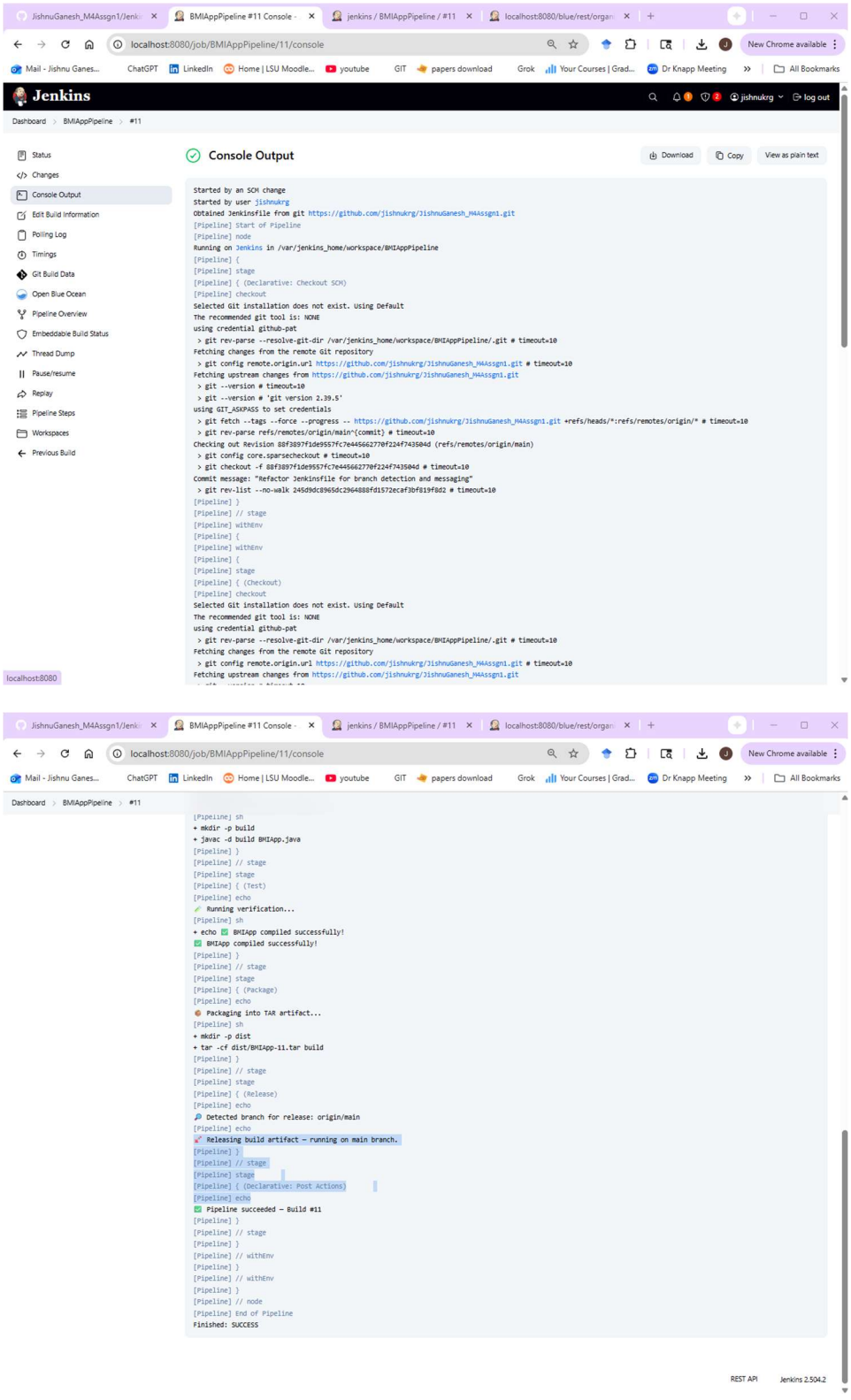
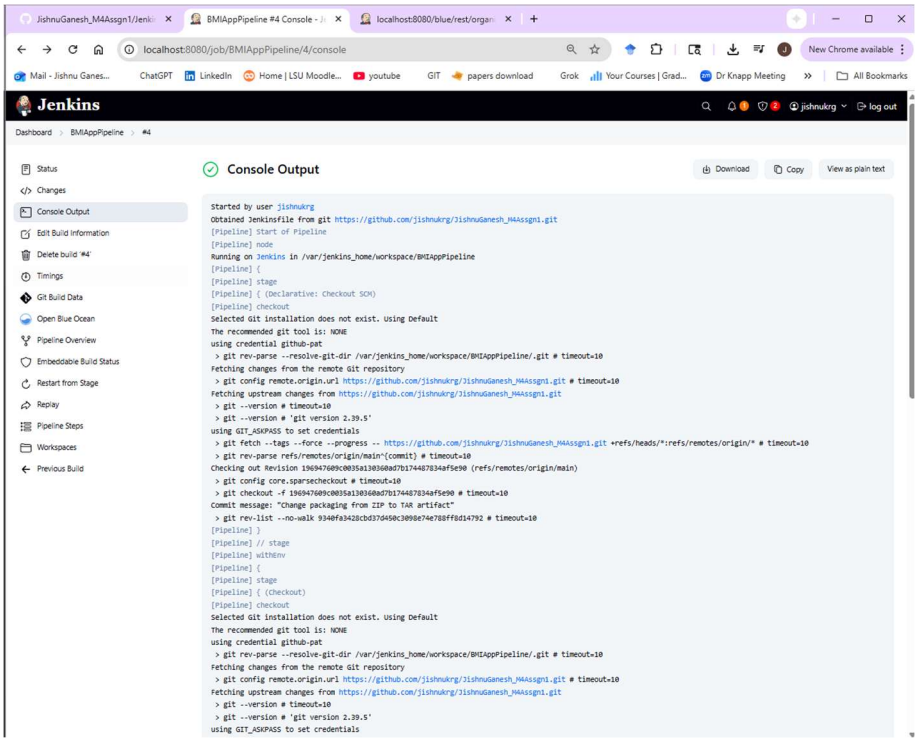


Figure: Console Output – Test Branch skipped Release stage confirmation.



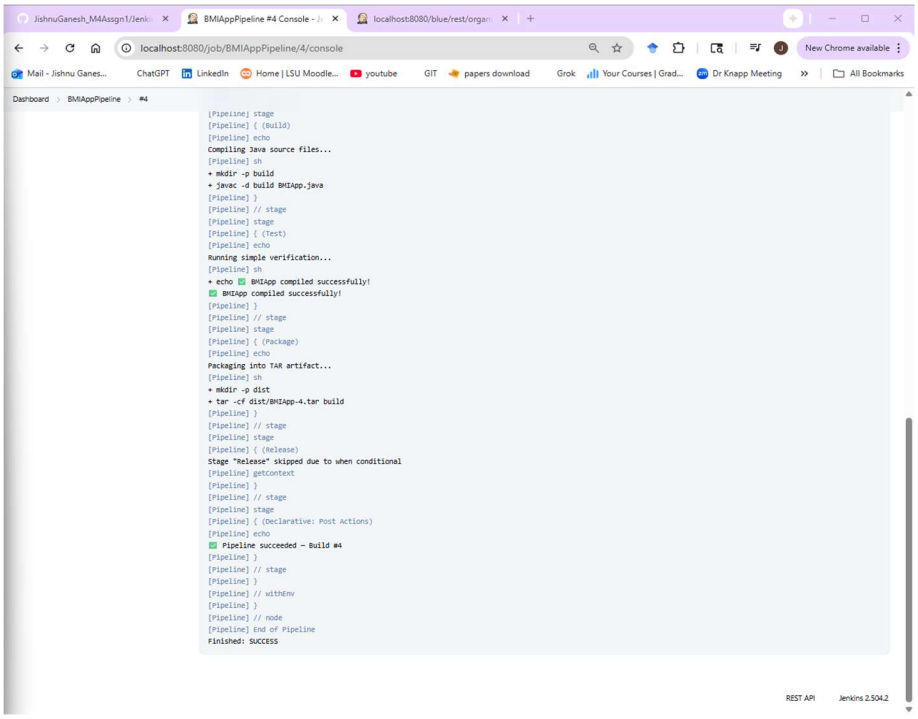


Figure: Dist Folder – Generated TAR artifact after successful pipeline run.