



|  |  |  |
| --- | --- | --- |
| **Batch:B1** | **Roll No.:16010421119** | **Experiment No.:**04 |

**Aim:** Continuous Integration with Jenkins, Git and GitHub for each build.



**Resources needed:** Jenkins, GitHub Account, Git installation



**Theory:**

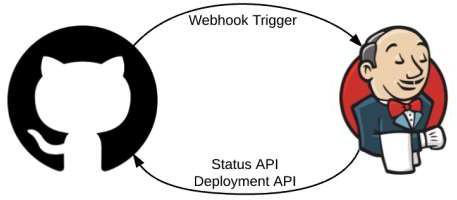
Implementing Continuous Integration (CI) can reduce errors in code, shorten development cycles, and help your team ship software faster.

**Why CI matters**

Companies need to release software faster than ever, and CI has become an important part of keeping up. With CI in place, teams can build, test, and update their code within minutes, increasing quality and reducing time to production.

With so much change in the tech field, teams can use many different kinds and combinations of tools. That’s why our focus is on making life easier for as many developers as possible, giving teams the freedom to integrate with the best tools out there. Today, GitHub integrates with hundreds of third party tools, including some of the most popular and well-documented integrations around—all available from Marketplace and the Works with GitHub directory.

**How does it work with GitHub?**



Jenkins scans your entire GitHub organization and creates Pipeline jobs for each repository containing a [Jenkinsfile](https://jenkins.io/doc/book/pipeline/jenkinsfile/)—a text file that defines the process of building, testing, and deploying your project using Jenkins. Immediately after code is checked in or a new pull request is created, Jenkins will execute the Pipeline job and return the status to GitHub indicating whether it failed or succeeded. This process allows you to run a build and subsequent automated tests at every check in so only the best code gets merged. Catching bugs early and automatically reduces the number of problems introduced into production, so your team can build better, more efficient software.

Deployments that occur in Jenkins can also be recorded back into GitHub to audit the entire lifecycle.

**Introduction to Git**



Git is a version control system for tracking changes in computer files and coordinating work on those files among multiple people. It is primarily used for source code management in software development, but it can be used to keep track of changes in any set of files.

**Git Commands:**

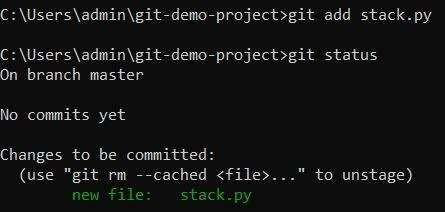
**Git Init**

The git init command creates a new Git repository. It can be used to convert an existing, unversioned project to a Git repository or initialize a new, empty repository.



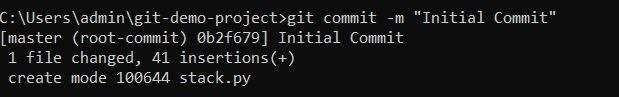
**Git Add**

When we create, modify or delete a file, these changes will happen in our local and won't be included in the next commit (unless we change the configurations). We need to use the git add command to include the changes of a file(s) into our next commit. The git add command doesn't change the repository and the changes are not saved until we use git commit. The git add command adds new or changed files in your working directory.



**Git Commit**

A commit, or "revision", is an individual change to a file (or set of files). It's like when you save a file, except with Git, every time you save it creates a unique ID (a.k.a. the "SHA" or "hash") that allows you to keep record of what changes were made when and by who. In version control systems, a commit is an operation which sends the latest changes to the source code to the repository, making these changes part of the head revision of the repository.



**Git- Push and Pull**

**Push operation:** The git push command is used to upload local repository content to aremote repository. Pushing is how you transfer commits from your local repository to a remote repository. The command for push is git push remote –all.

**Pull operation:** The git pull command is used to fetch and download content from a remote



repository and immediately update the local repository to match that content. The command is: git pull <remote>.

**Git Clone**

Git clone is a Git command line utility which is used to target an existing repository and create a clone, or copy of the target repository. In this page we'll discuss extended configuration options and common use cases of git clone. Clones a repository into a newly created directory, creates remote-tracking branches for each branch in the cloned repository (visible using git branch --remotes), and creates and checks out an initial branch that is forked from the cloned repository’s currently active branch.

**Branches:** Git branches are effectively a pointer to a snapshot of your changes. When you want to add a new feature or fix a bug, no matter how big or how small, you spawn a new branch to encapsulate your changes. This makes it harder for unstable code to get merged into the main code base, and it gives you the chance to clean up your future's history before merging it into the main branch.

A branch represents an independent line of development. Branches serve as an abstraction for the edit/stage/commit process. New commits are recorded in the history for the current branch, which results in a fork in the history of the project.

Files in master branch:



The files in the master branch are merged with newbranch.





**Procedure:**

1. Download Git tool from official Git website: <https://git-scm.com/downloads>2. Create GitHub Account <https://github.com/join>  
3. Execute Git Commands: version, global, init, add, commit, checkout, push, pull etc.

4. Write any program and push it on GitHub



5. Integrate Jenkins and GitHub

a. Install Git plugin on jenkins

b. Create a project on Jenkins and configure it for Git SCM

6. Run Git Push command to commit the modified code on GitHub

7. Check the changes on the GitHub and Jenkins for continuous integration for

each build.

**Results: (Document with screenshots)**

1. Program code

2. Pushed file on GitHub dashboard

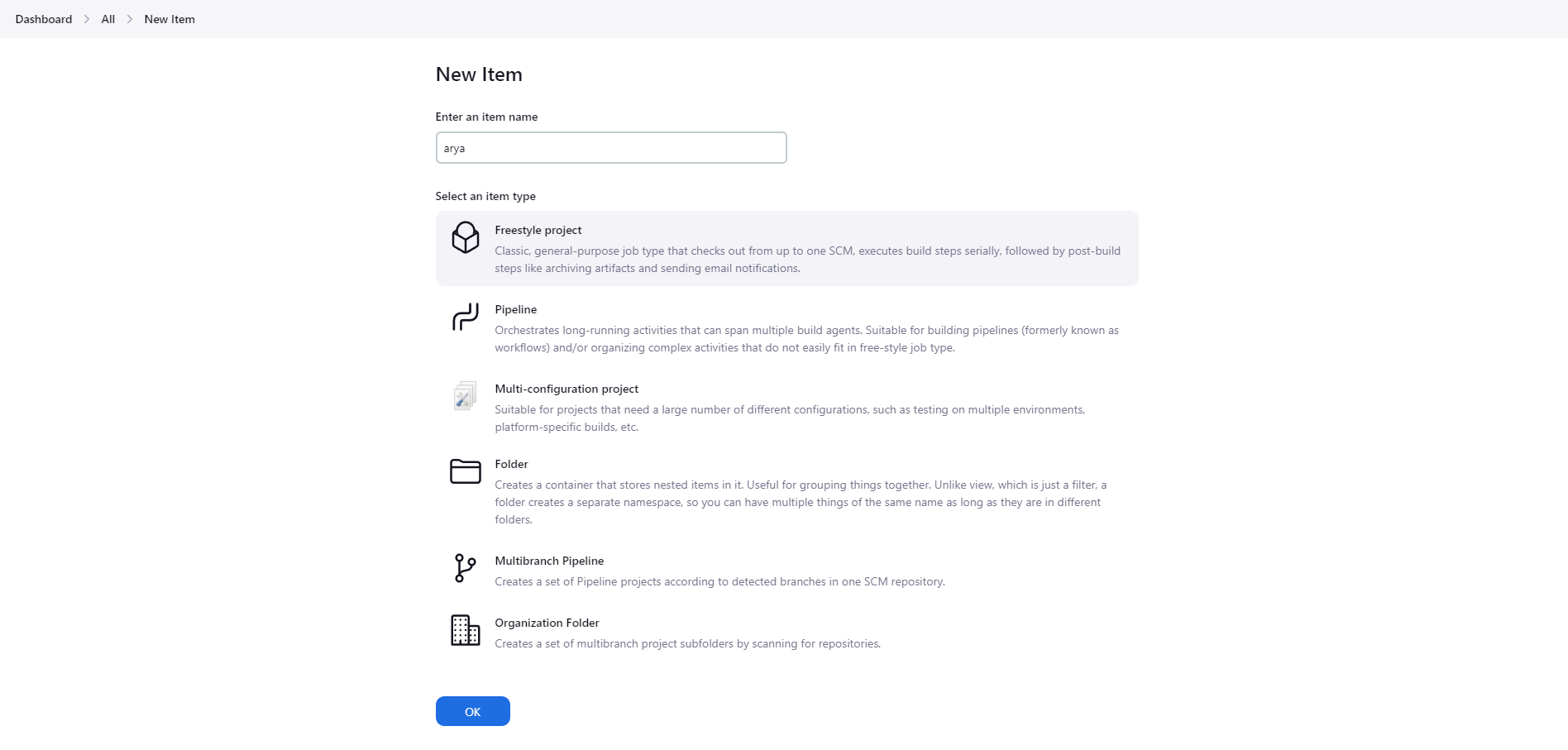
3. All steps to install and configure Git Plugin

4. Execution process of code before and after code modification

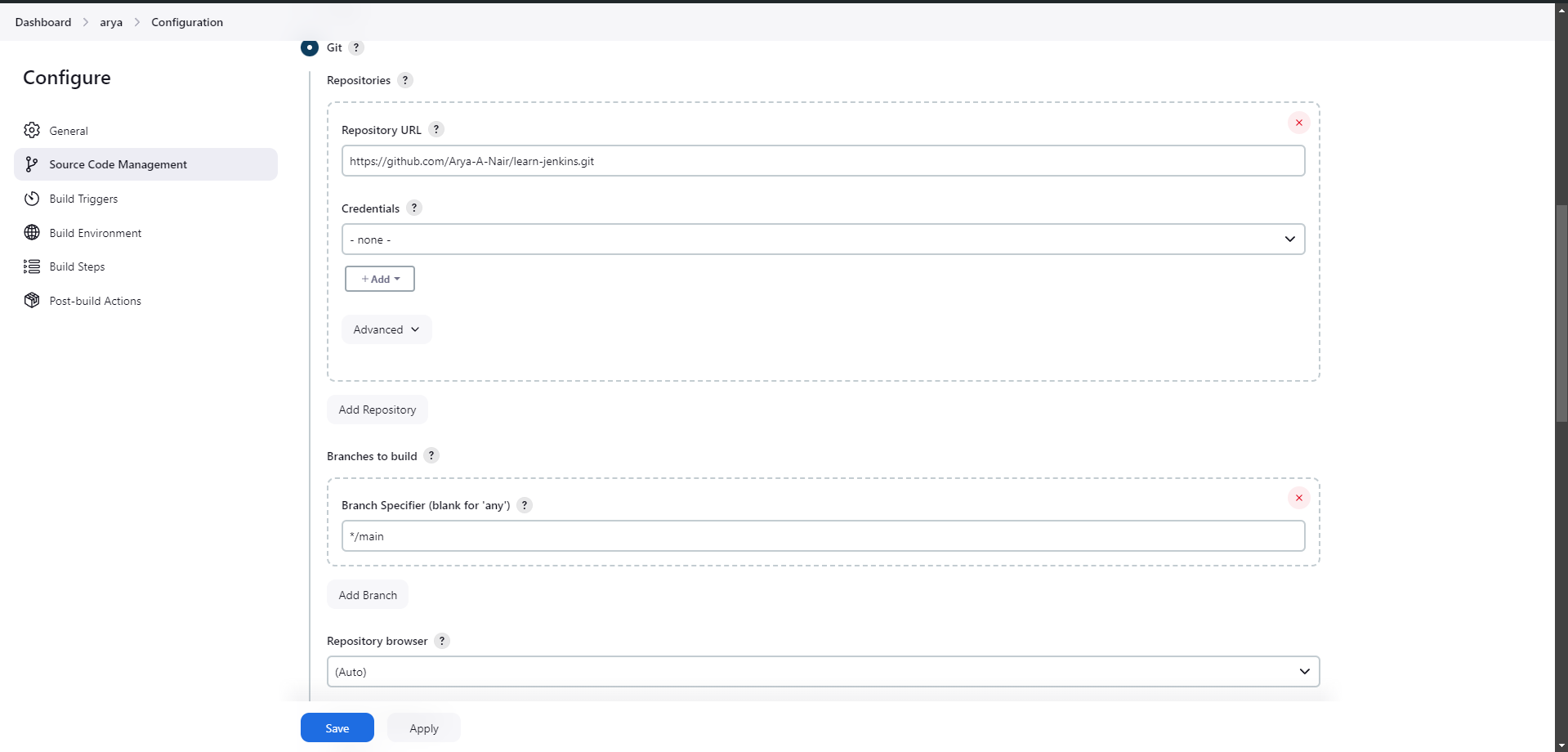
a. Git command

b. GitHub updates

c. Jenkins updates

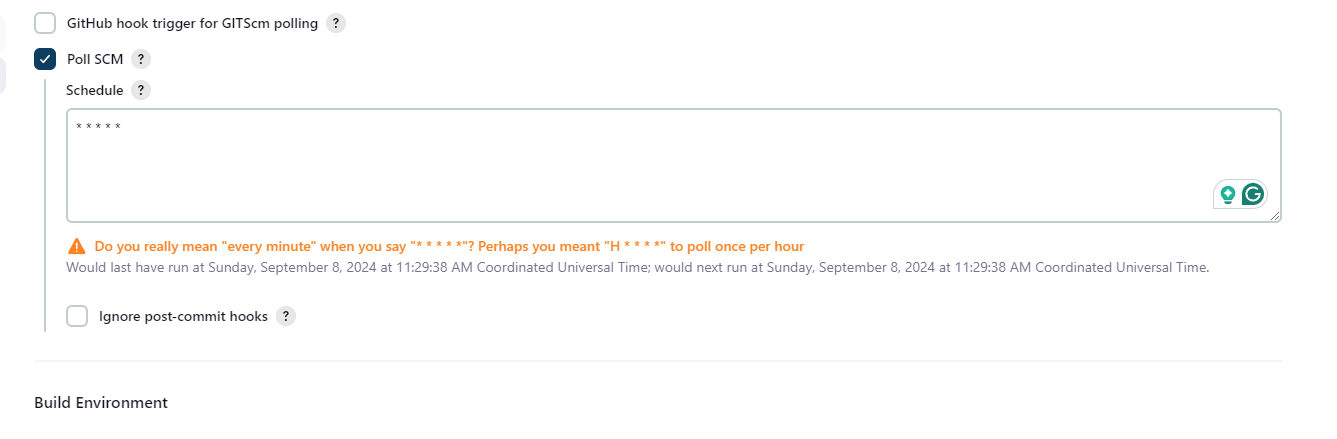


Create a new freestyle project

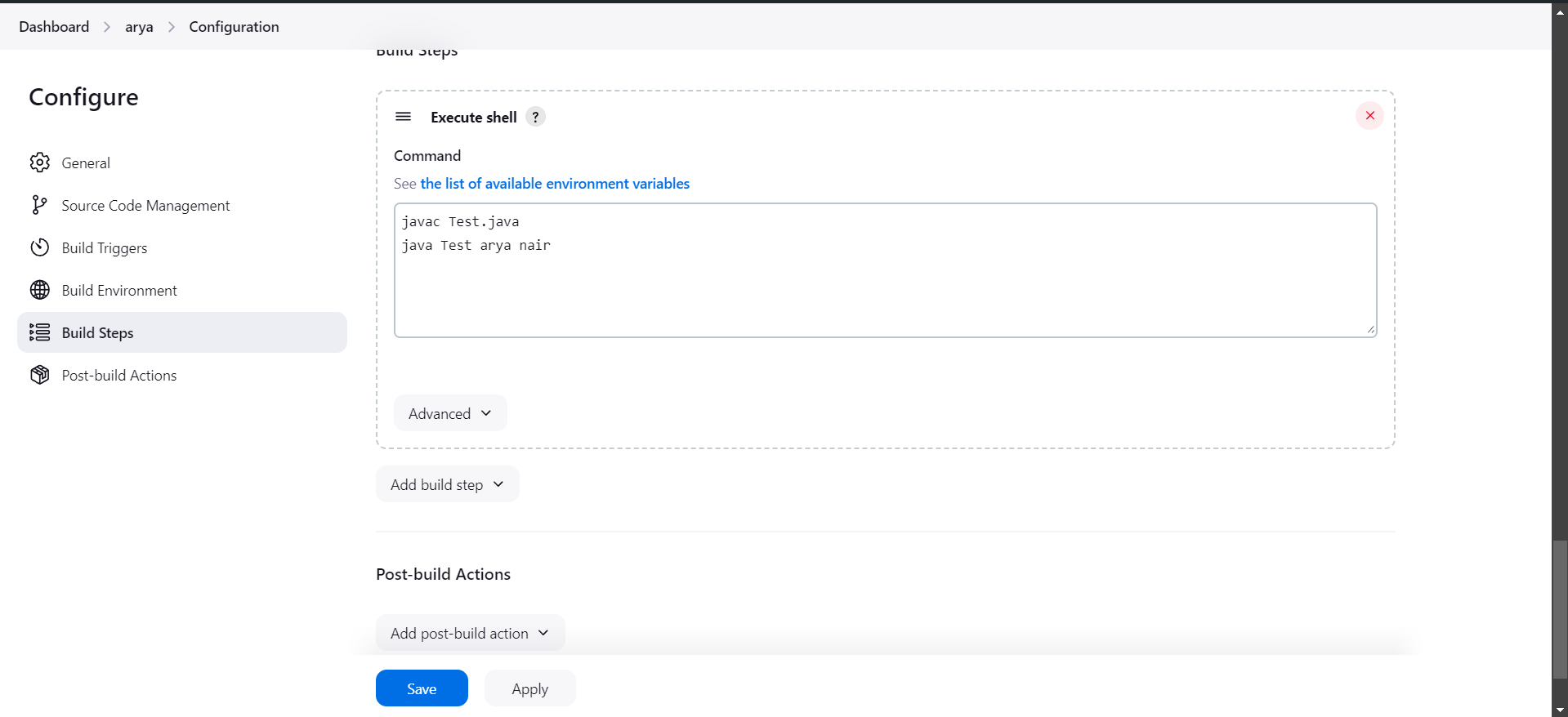


add repository URL to clone the repository during pipeline execution

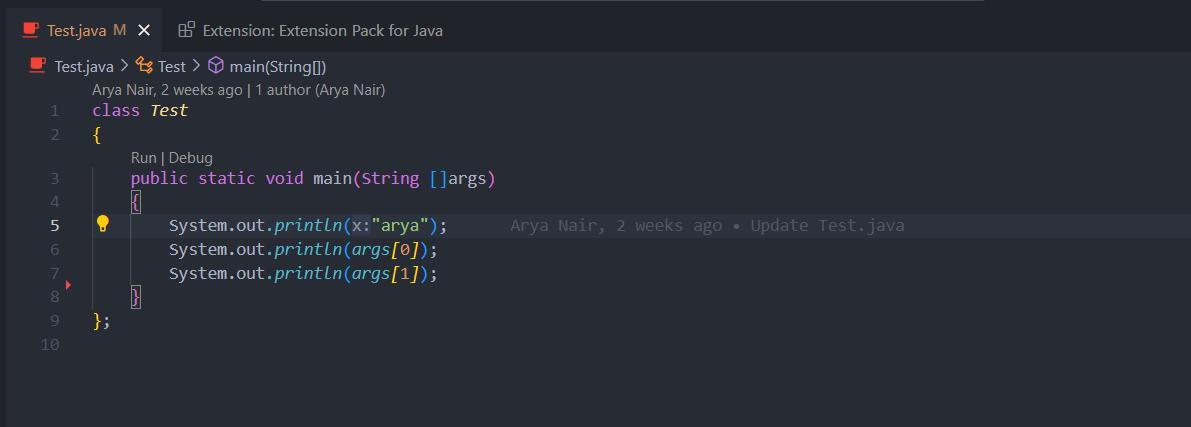




Add Poll SCM to check github for updates

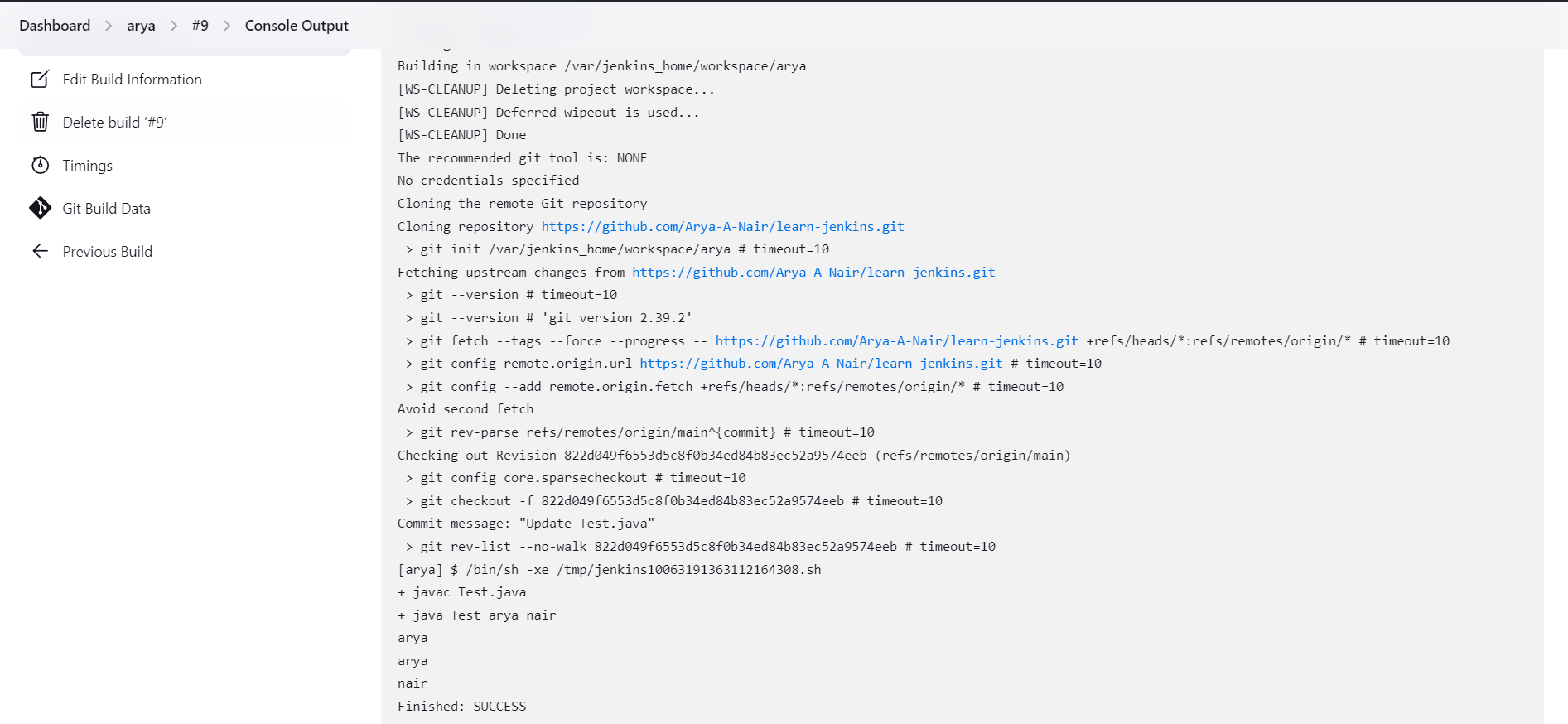


Add commands to be run



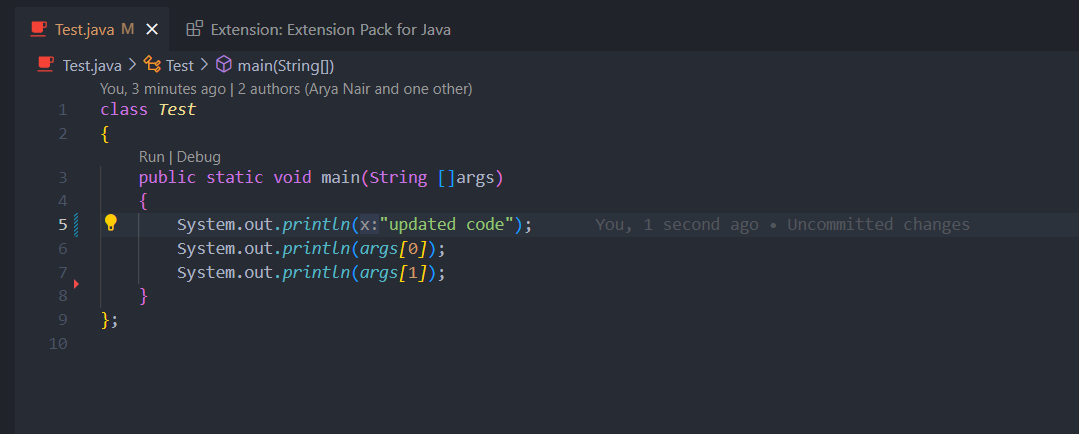
Code which is being run

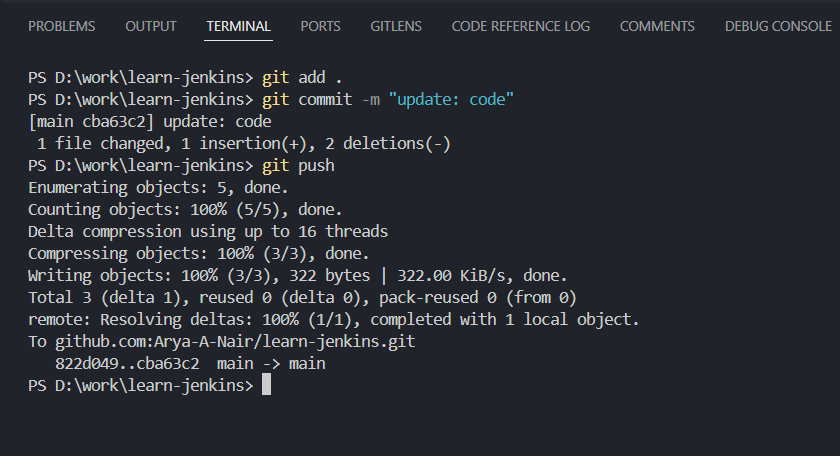




Code output

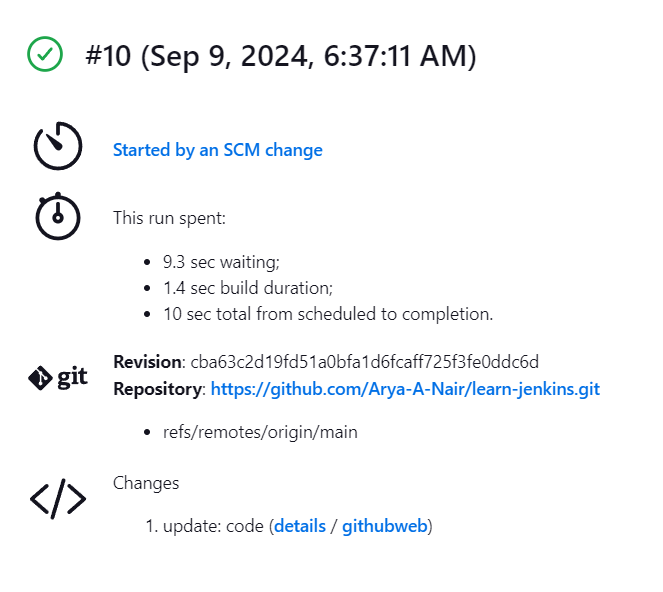
pushing updated code



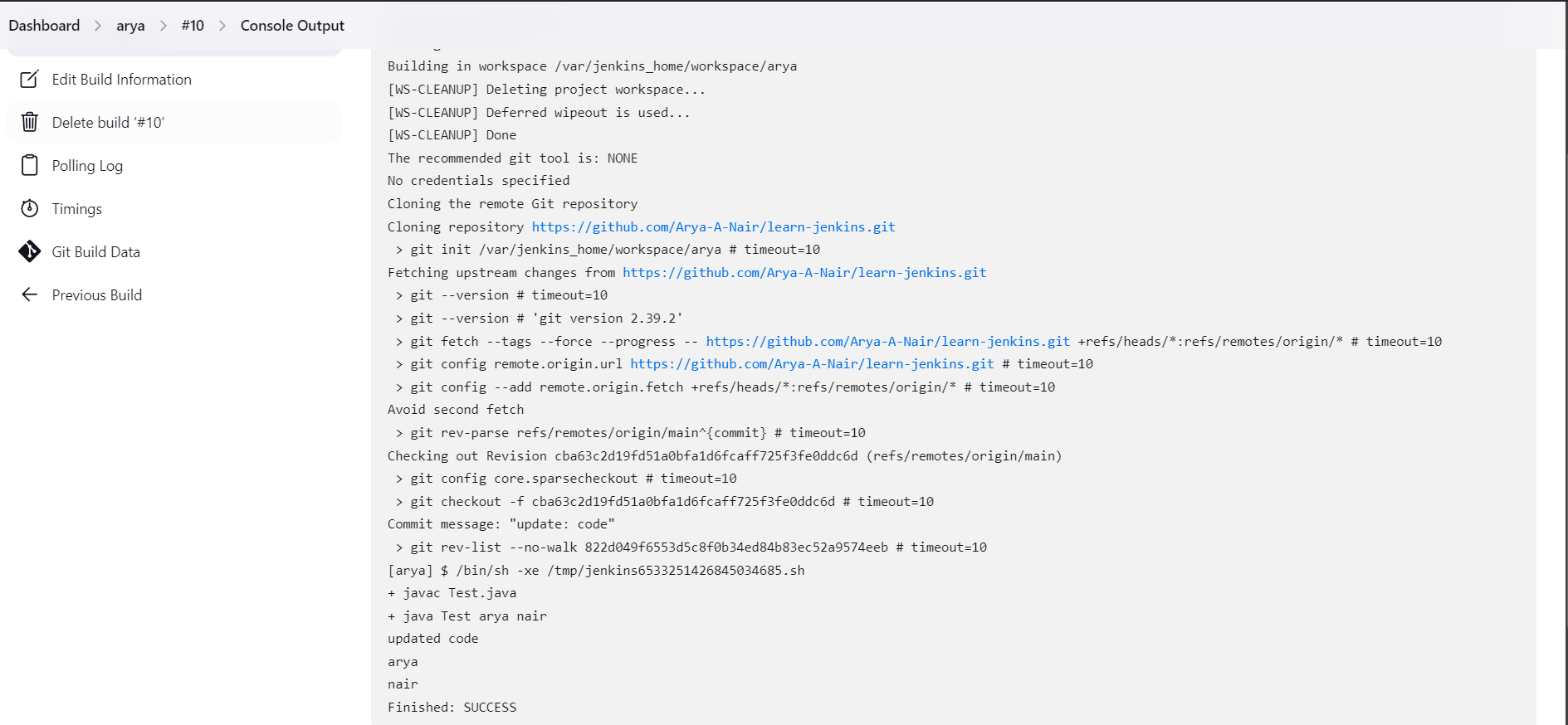












updated code output

**Questions:**

1. What is difference between Git and GitHub

Git is a distributed version control system used to track changes in source code locally, allowing multiple developers to work independently on a project. GitHub, on the other hand, is a web-based platform that hosts Git repositories online, offering tools for collaboration, such as pull requests and issue tracking, as well as social features for exploring and contributing to open-source projects. 2. What is Master and Branch in Github?

In Git and GitHub, "master" (or "main") is the default branch that typically holds the stable, production-ready code. A "branch" is a separate line of development within a repository, allowing you to work on new features or fixes independently from the main branch. Branches enable parallel development and experimentation without affecting the core codebase.

3. What is Jenkinsfile?

A Jenkinsfile is a script that defines the build and deployment pipeline for a project in Jenkins, a popular continuous integration/continuous deployment (CI/CD) tool. Written in either Declarative or Scripted Pipeline syntax, a Jenkinsfile outlines the stages and steps involved in building, testing, and deploying software, automating these processes to ensure consistent and repeatable workflows.

By storing the Jenkinsfile in the project's repository, you can version control your pipeline configuration alongside your code.



**Outcomes:CO3 Comprehend the effective code building and testing process**





**Conclusion: (Conclusion to be based on the Results and outcomes achieved)**



**run jenkins build using continuous integration setup using poll SCM**





**Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of faculty in-charge with date**



**References:**   
**Books/ Journals/ Websites:**

1. https://guides.github.com/introduction/git-handbook/   
2. Jenkins Beginner Tutorial 8 - Jenkins integration with GIT (SCM):   
 https:/[/www](http://www.youtube.com/watch?v=bGqS0f4Utn4&t=7s).[youtube.com/watch?v=bGqS0f4Utn4&t=7s](http://www.youtube.com/watch?v=bGqS0f4Utn4&t=7s)   
3. Integrate with GitHub: build after each commit (Get started with Jenkins, part 13): https:/[/www](http://www.youtube.com/watch?v=Z3S2gMBUkBo).[youtube.com/watch?v=Z3S2gMBUkBo](http://www.youtube.com/watch?v=Z3S2gMBUkBo)   
4. https://resources.github.com/whitepapers/practical-guide-to-CI-with-Jenkins-and - GitHub/   
5. https:/[/www.guru99.com/continuous](http://www.guru99.com/continuous-integration.html)-[integration.html](http://www.guru99.com/continuous-integration.html)



6. Effective DevOps: Building a Culture of Collaboration, Affinity, and Tooling at Scale, Jennifer Davis, Ryn Daniels, O'Reilly Media June 2016.

7. Practical DevOps: Implement DevOps in your organization by effectively building, deploying, testing, and monitoring code,Joakim Verona, Packt Publishing, 2nd Edition,2016