Ragul R R

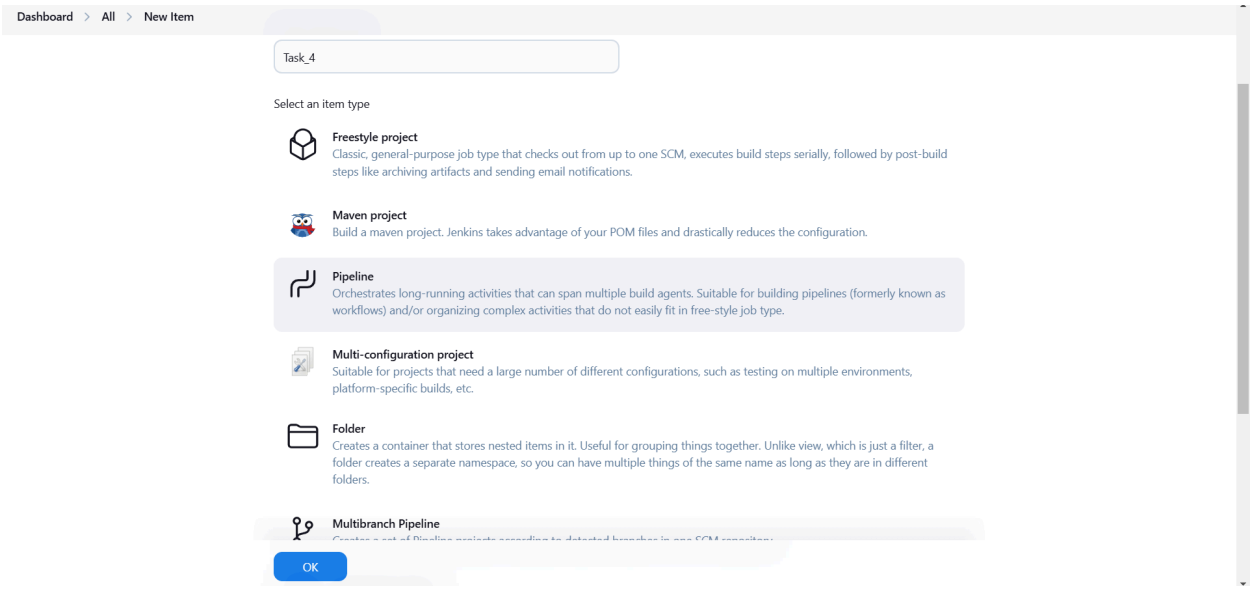
05.02.2025

Analysis of Jenkins Pipeline, Docker, and GitHub Repository Configuration

1. Jenkins Dashboard & New Item Creation

The first image displays the Jenkins dashboard, showcasing the process of creating a new job. The available options include:

- **Freestyle Project**: A flexible project type that allows various build and post-build steps.
- **Maven Project**: Uses POM files to handle builds efficiently.
- **Multi-Configuration Project**: Designed for projects requiring multiple configurations, such as matrix builds.
- **Pipeline Project**: Uses a scripted or declarative pipeline for complex CI/CD automation.

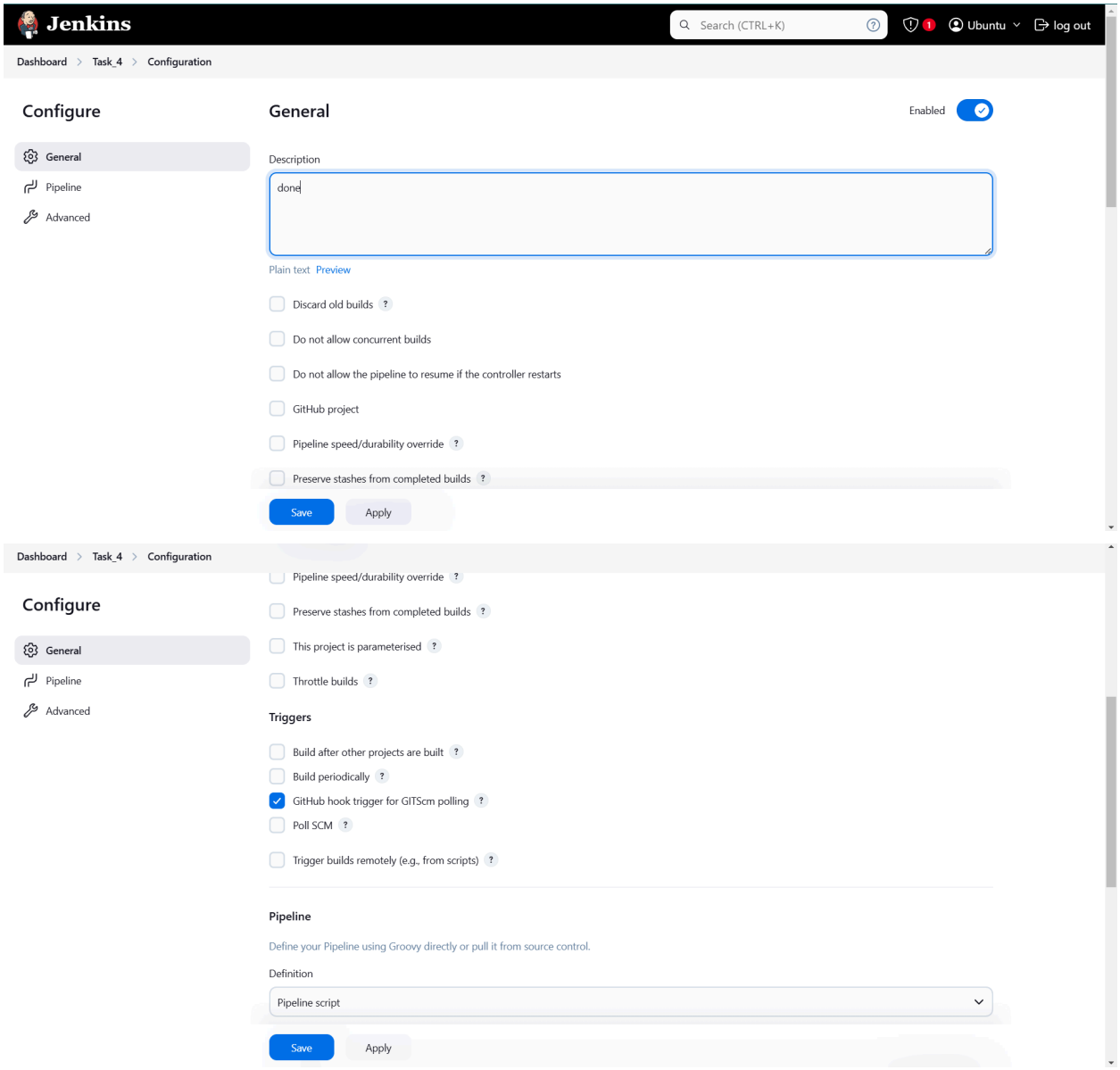


2. Jenkins Configuration Settings

The second image focuses on the configuration settings of a Jenkins pipeline job. Key options include:

- **Pipeline Speed/Durability Override**: Adjusts execution speed and resource allocation.

- **Preserve Stashes from Completed Builds**: Stores build stashes for debugging purposes.
- **GitHub Triggers**: Enables triggers like Webhooks and SCM polling.
- **Throttle Builds**: Limits concurrent builds to prevent system overload.

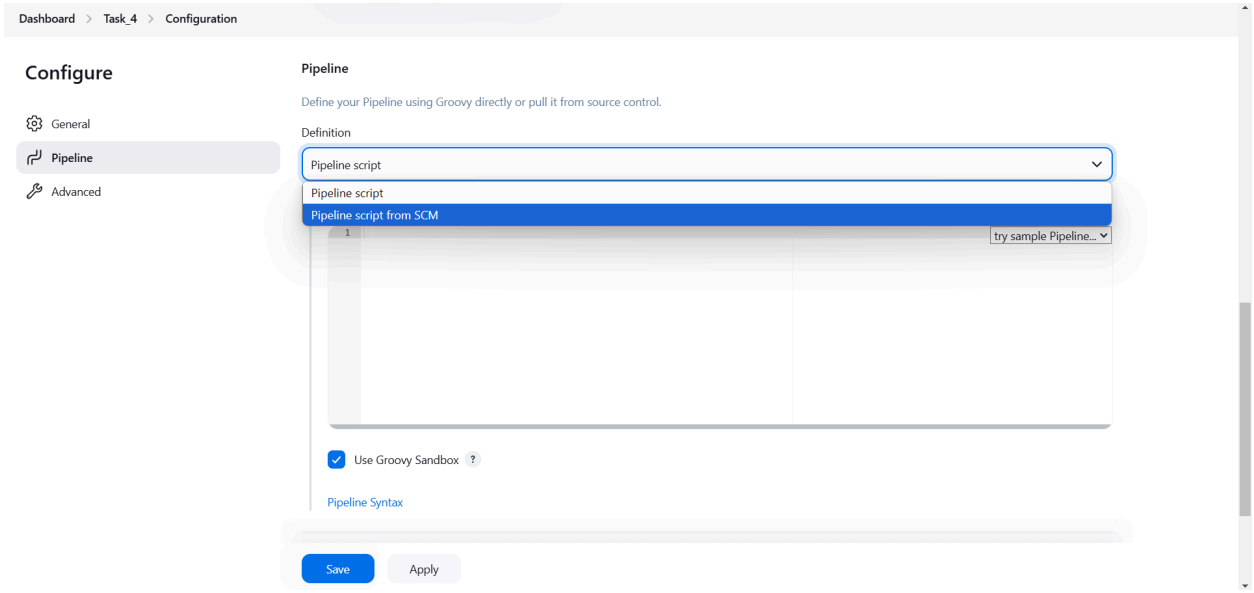


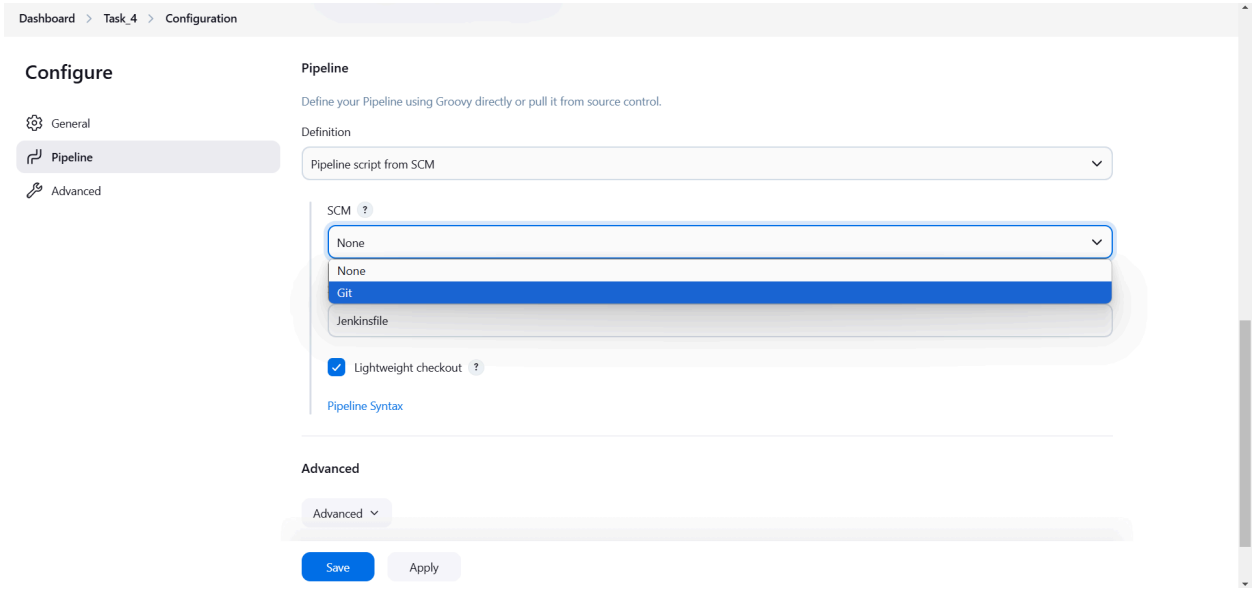
3. Pipeline Script & SCM Configuration

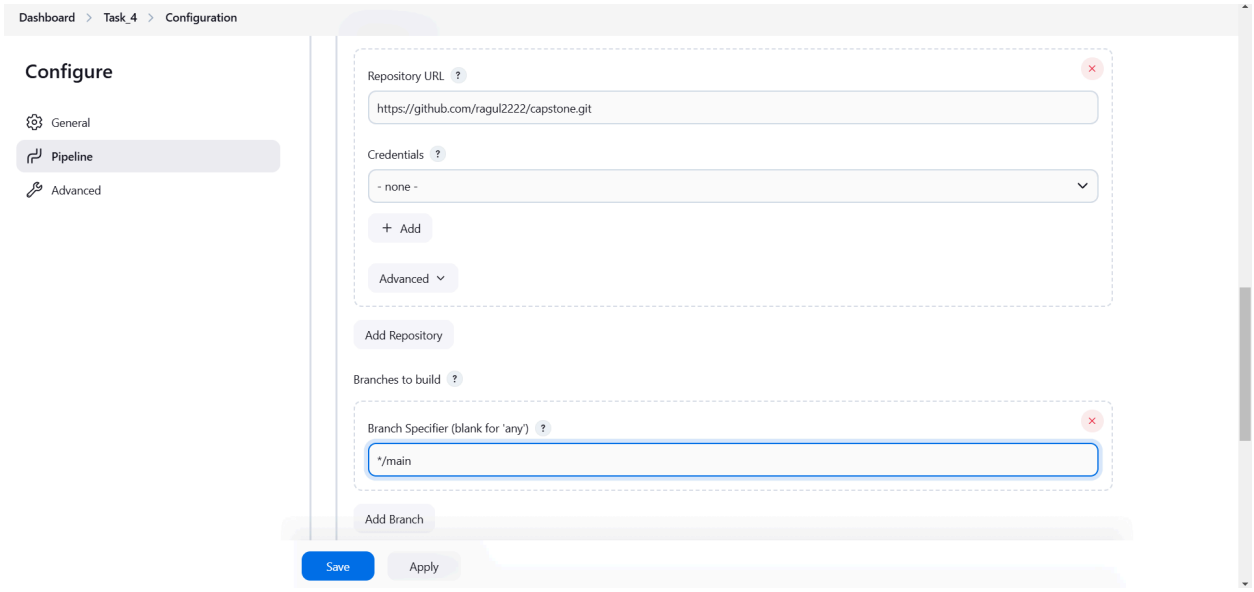
This image showcases:

- **Pipeline Script from SCM**: Fetches the Jenkins pipeline script from a Git repository.
- **Repository URL**: https://github.com/ragul2222/capstone.git.

- **Branch Specifier**: Set to main, meaning Jenkins builds only this branch.
- **Jenkinsfile Usage**: Defines pipeline stages and steps.







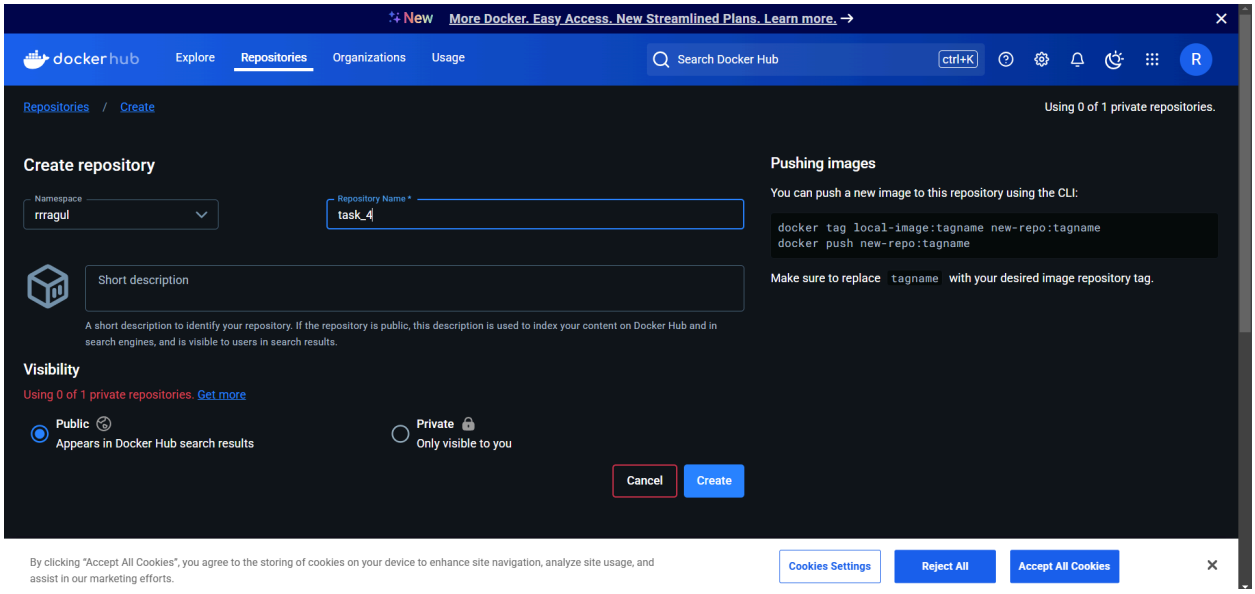
4. Docker Hub Repository Setup

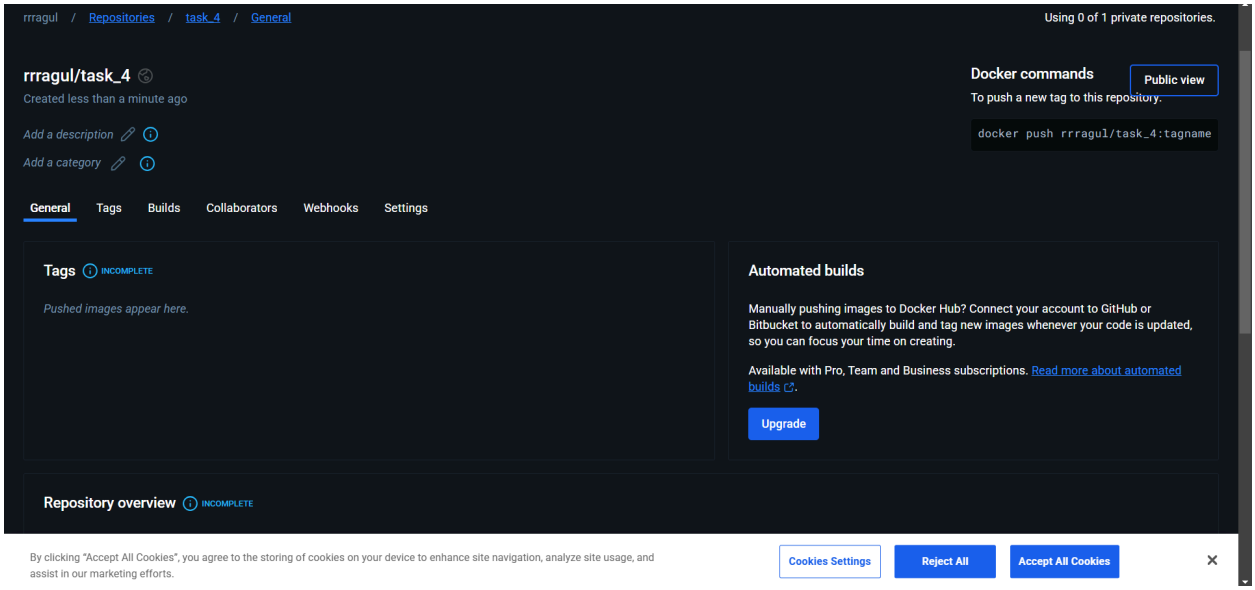
The fourth image presents a Docker Hub repository (rrragul/task_4). Important details include:

- **Public Visibility**: The repository is accessible to all users.

Pushing Docker Images: CLI commands for uploading images:
docker tag local-image:tagname new-repo:tagname
docker push new-repo:tagname

- **Automated Builds**: Enables linking with GitHub or Bitbucket for automatic image builds upon code updates.



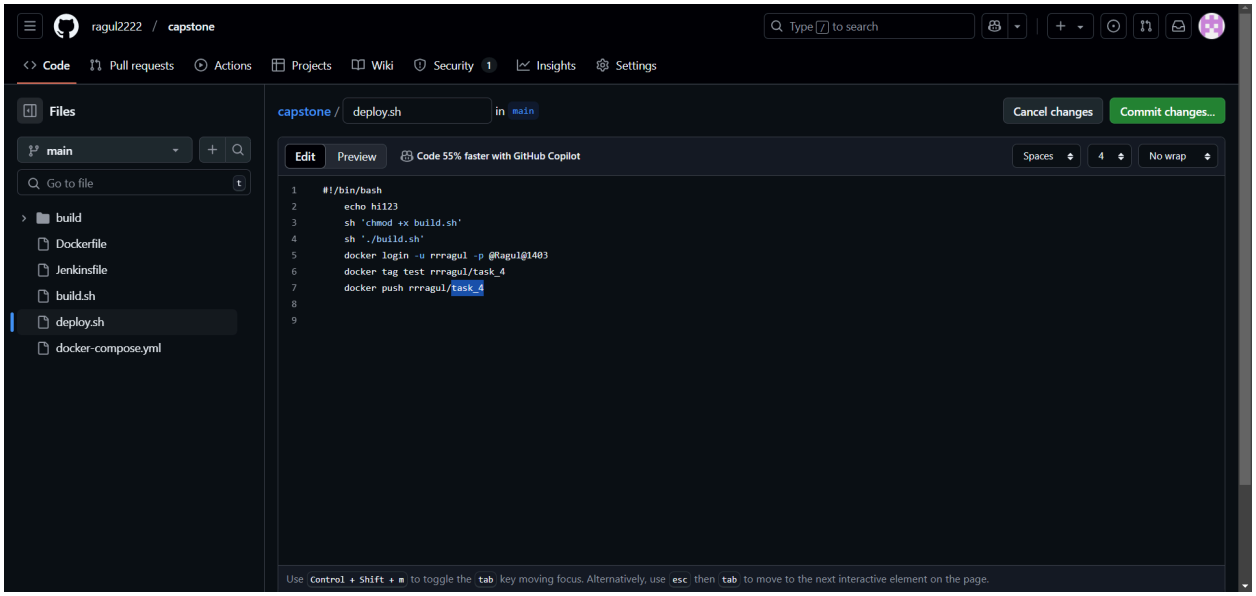


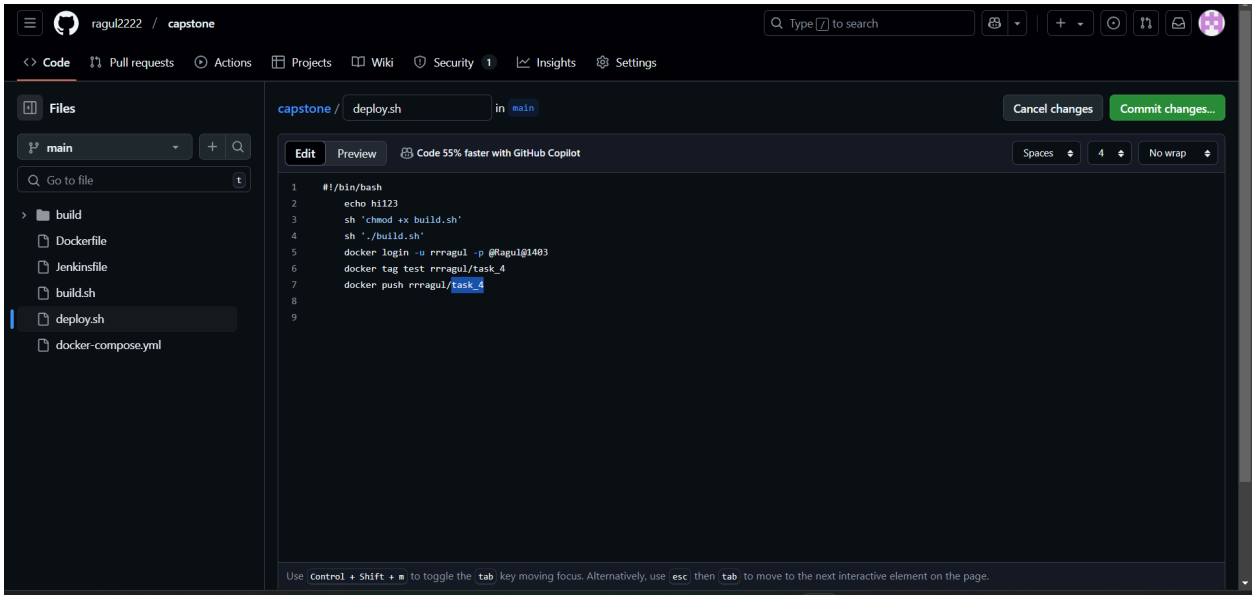
5. GitHub Repository & Deployment

This section highlights:

- **Files in the Repository**:
  - Dockerfile: Defines the container image structure.
  - Jenkinsfile: Contains pipeline definitions.
  - build.sh: Script for building the project.
  - deploy.sh: Script for deploying the application.
  - docker-compose.yml: Configuration for containerized multi-service applications.

Deployment Commands:
docker login -u username -p password
docker push rrragul/task_4:tagname

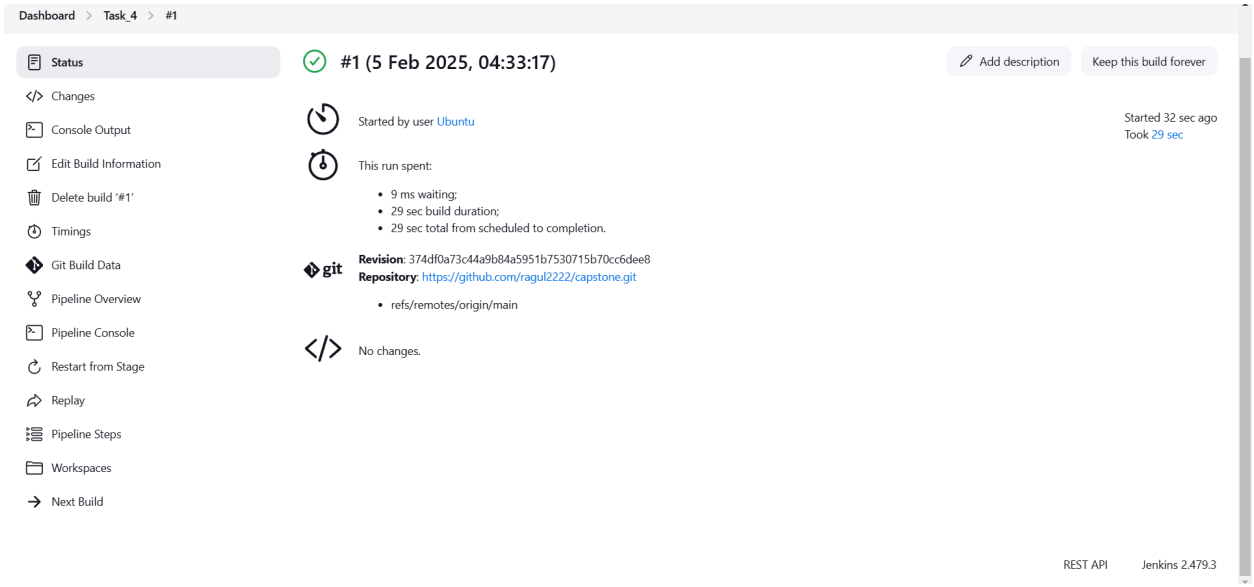


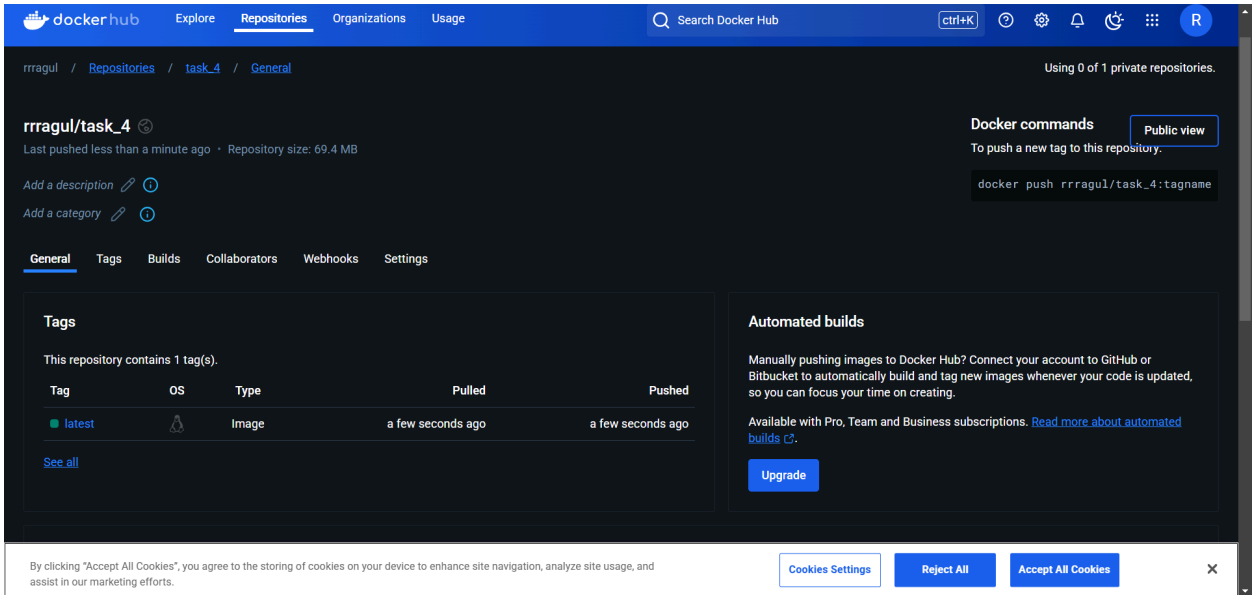


6. Jenkins Build Execution

The final image captures a successful pipeline execution:

- **Build Number**: #1 executed on **Feb 5, 2025**.
- **Repository Revision**: 3744f0a73044a984a5951b7530715b70ccbdees.
- **Execution Time**: 29 seconds.
- **Console Output**: Displays detailed logs of the build process.





Conclusion

The extracted images indicate a well-structured CI/CD pipeline using Jenkins, GitHub, and Docker. The workflow involves:

1. **Defining a Pipeline (Jenkinsfile)**.
2. **Storing Source Code in GitHub**.
3. **Using Jenkins for Automated Builds**.
4. **Pushing Docker Images to Docker Hub**.
5. **Executing CI/CD Pipelines Efficiently**.

This setup ensures seamless software development and deployment automation.