

Ragul R R

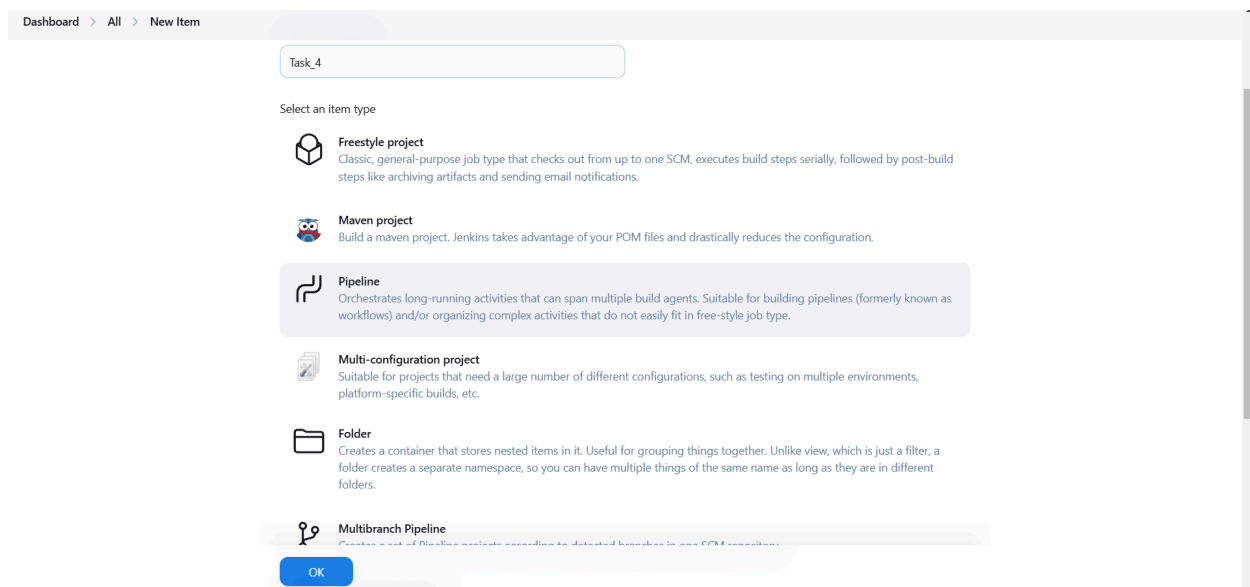
05.02.2025

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.

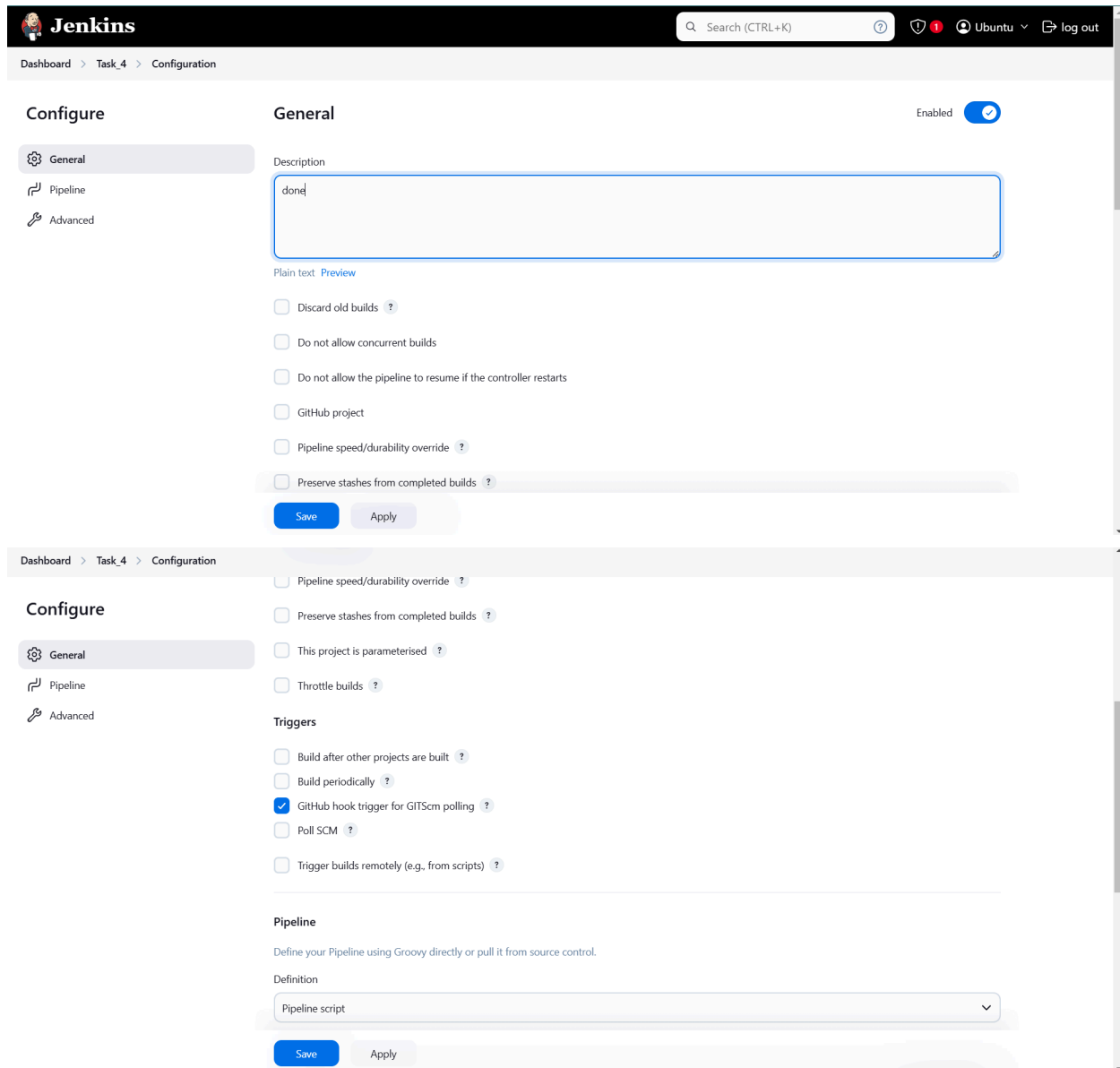


2. Jenkins Configuration Settings

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.

Dashboard > Task_4 > Configuration

Configure

General

Pipeline

Advanced

Pipeline

Define your Pipeline using Groovy directly or pull it from source control.

Definition

Pipeline script

Pipeline script

Pipeline script from SCM

try sample Pipeline...

☒ Use Groovy Sandbox ?

Pipeline Syntax

Save Apply

Dashboard > Task_4 > Configuration

Configure

General

Pipeline

Advanced

Pipeline

Define your Pipeline using Groovy directly or pull it from source control.

Definition

Pipeline script from SCM

SCM ?

None

None

Git

Jenkinsfile

☒ Lightweight checkout ?

Pipeline Syntax

Advanced

Advanced

Save Apply

Dashboard > Task_4 > Configuration

Configure

- General
- Pipeline**
- Advanced

Repository URL ? ✕

https://github.com/ragul2222/capstone.git

Credentials ?

- none - ▼

+ Add

Advanced ▼

Add Repository

Branches to build ?

Branch Specifier (blank for 'any') ? ✕

*/main

Add Branch

Save Apply

4. Docker Hub Repository Setup

The fourth image presents a Docker Hub repository (rragul/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.

NewMore Docker. Easy Access. New Streamlined Plans. Learn more. →

docker hub

Explore

Repositories

Organizations

Usage

Search Docker Hub

ctrl+k

R

Repositories / Create

Using 0 of 1 private repositories.

Create repository

Namespace

rragul

Repository Name *

task_4

Short description

A short description to identify your repository. If the repository is public, this description is used to index your content on Docker Hub and in search engines, and is visible to users in search results.

Visibility

Using 0 of 1 private repositories. Get more

Public

Appears in Docker Hub search results

Private

Only visible to you

Cancel

Create

Pushing images

You can push a new image to this repository using the CLI:

docker tag local-image:tagname new-repo:tagname

docker push new-repo:tagname

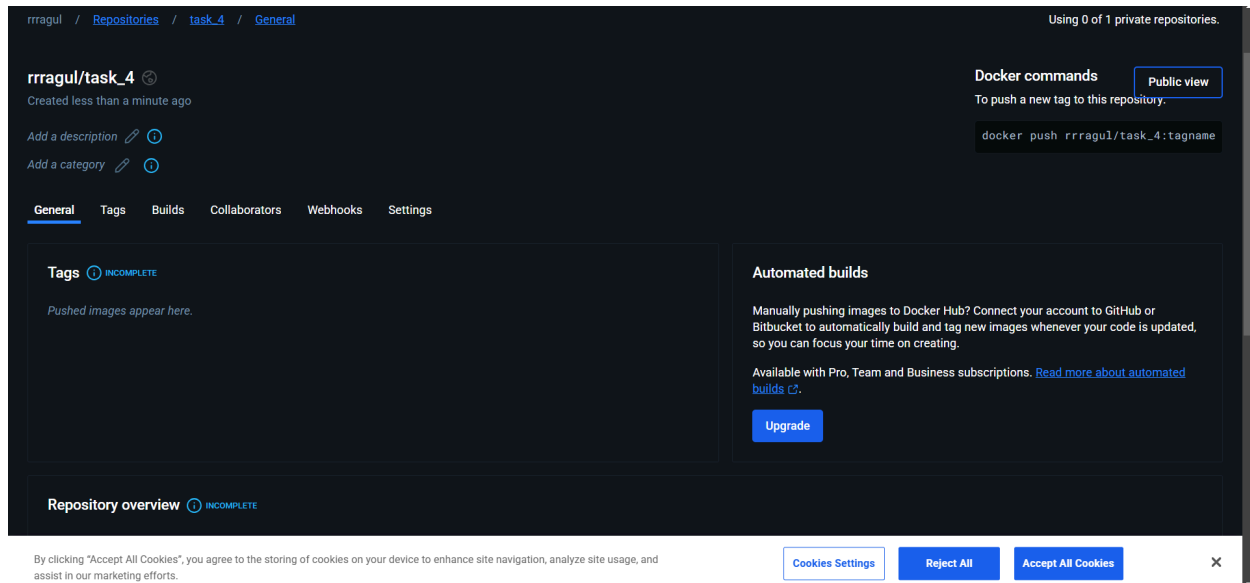
Make sure to replace tagname with your desired image repository tag.

By clicking "Accept All Cookies", you agree to the storing of cookies on your device to enhance site navigation, analyze site usage, and assist in our marketing efforts.

Cookies Settings

Reject All

Accept All Cookies



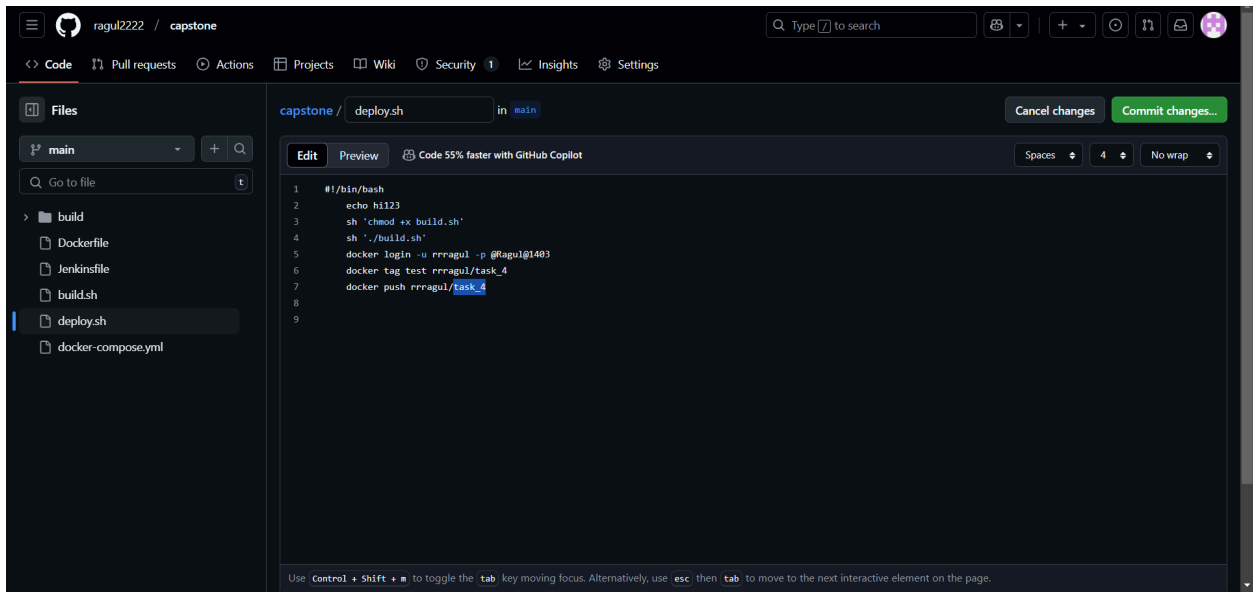
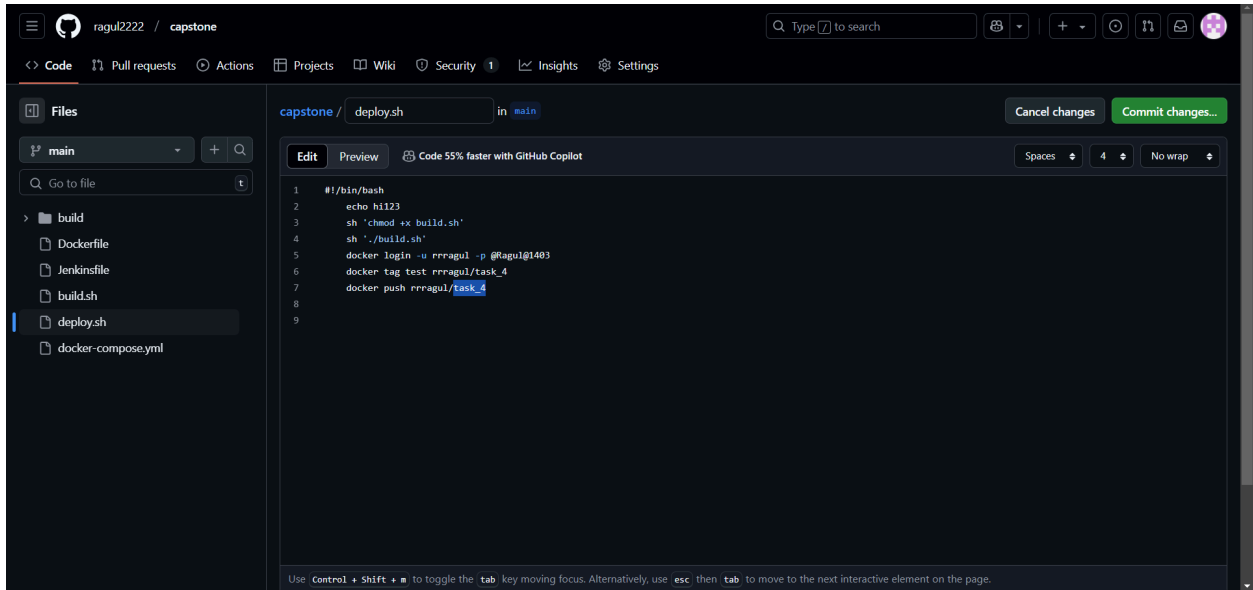
5. GitHub Repository & Deployment

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.

The image shows two screenshots. The top screenshot is the Jenkins build interface for a build named '#1' on '5 Feb 2025, 04:33:17'. The left sidebar contains a 'Status' tab and a list of actions: Changes, Console Output, Edit Build Information, Delete build '#1', Timings, Git Build Data, Pipeline Overview, Pipeline Console, Restart from Stage, Replay, Pipeline Steps, Workspaces, and Next Build. The main area shows the build status as successful (green checkmark). It indicates the build was started by 'Ubuntu' and took 29 seconds. A 'git' section shows the revision '374df0a73c44a9b84a5951b7530715b70cc6dee8' and the repository 'https://github.com/rragul222/capstone.git'. The 'Changes' section shows 'No changes.' The bottom right corner of the Jenkins interface shows 'REST API' and 'Jenkins 2.479.3'.

The bottom screenshot is the Docker Hub repository page for 'rragul/task_4'. The page shows the repository size as 69.4 MB and the last push as 'less than a minute ago'. It includes tabs for 'General', 'Tags', 'Builds', 'Collaborators', 'Webhooks', and 'Settings'. The 'Tags' section shows a table with one tag, 'latest', which is an 'Image' type, pushed 'a few seconds ago'. The 'Automated builds' section provides information on connecting to GitHub or Bitbucket for automated builds. A 'Docker commands' section shows the command 'docker push rragul/task_4:tagname'. At the bottom, there is a cookie consent banner with 'Cookies Settings', 'Reject All', and 'Accept All Cookies' buttons.

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.