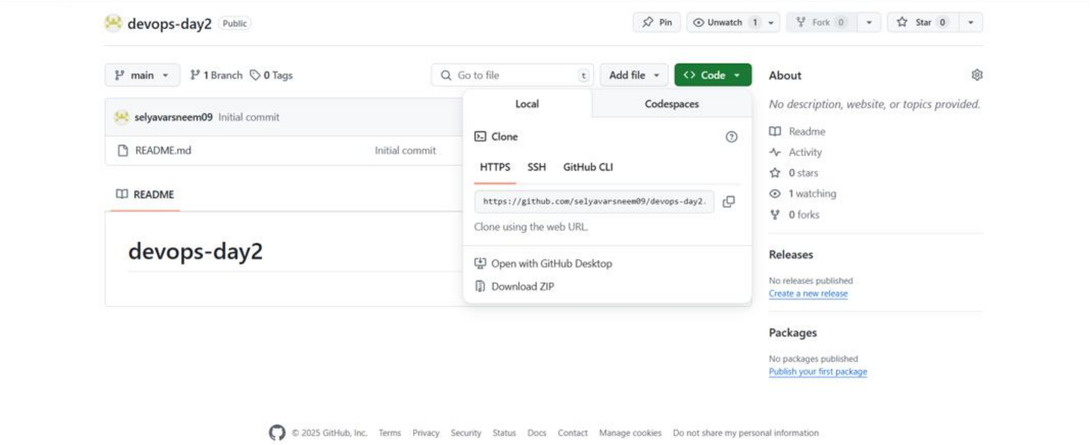
STEP 1:

Creating a new repository and copying the url.



selyavarsneem09 Initial commit

README.md Initial commit

devops-day2

https://github.com/selyavarsneem09/devops-day2.

Clone using the web URL.

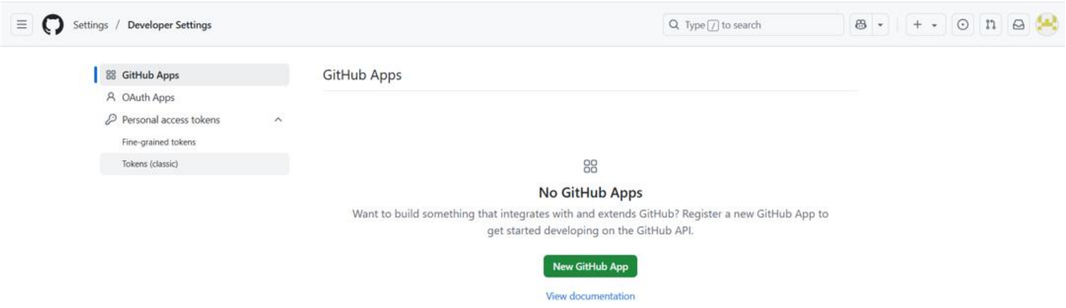
No description, website, or topics provided.

No releases published

No packages published

STEP 2:

Go to the Developer Settings



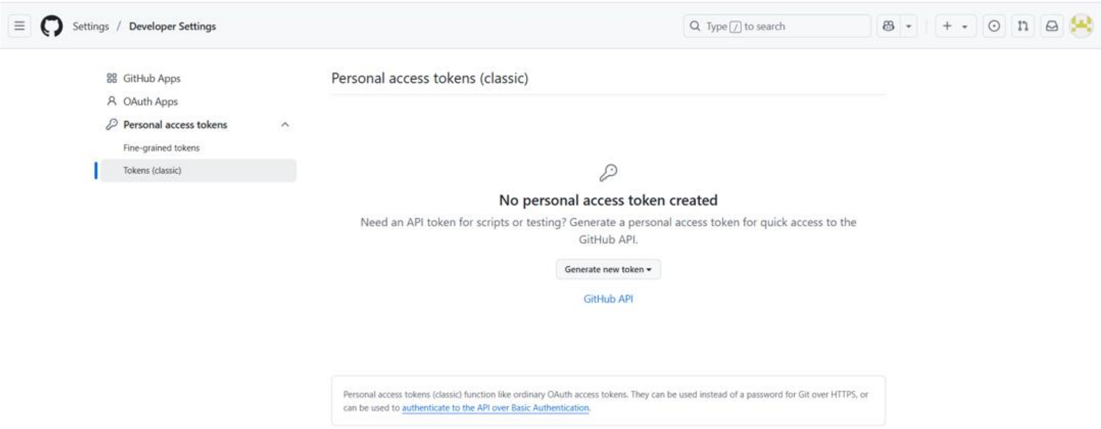
GitHub Apps

No GitHub Apps

Want to build something that integrates with and extends GitHub? Register a new GitHub App to get started developing on the GitHub API.

STEP 3:

Select Tokens(classic) Option



Personal access tokens (classic)

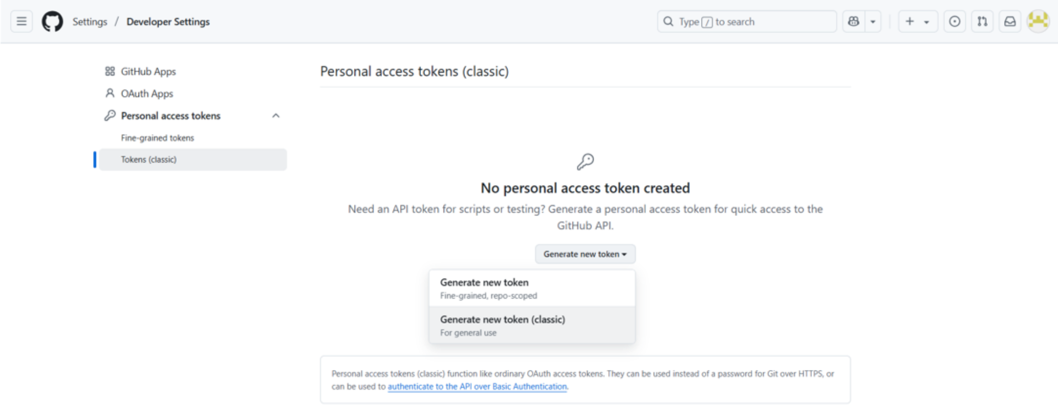
No personal access token created

Need an API token for scripts or testing? Generate a personal access token for quick access to the GitHub API.

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to authenticate to the API over Basic Authentication.

STEP 4:

Select the “Generate new token (classic) option



Personal access tokens (classic)

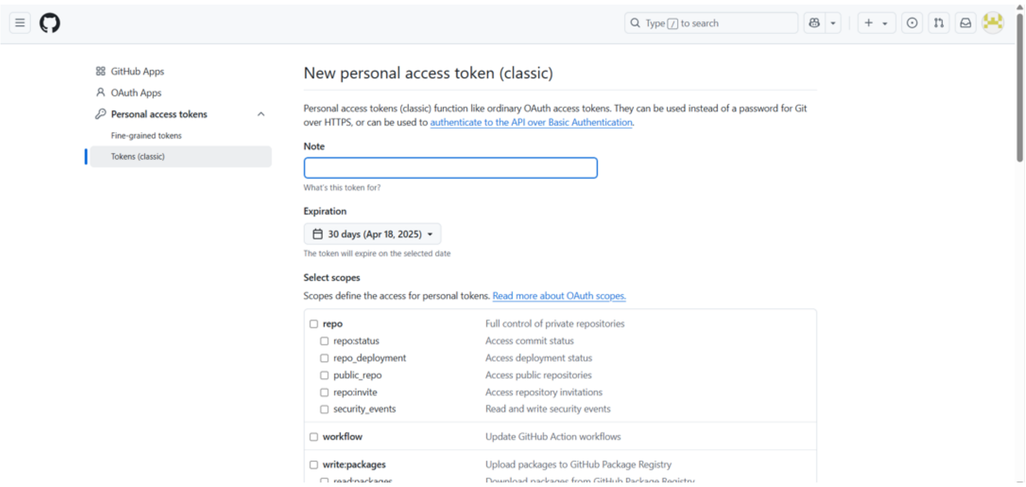
No personal access token created

Need an API token for scripts or testing? Generate a personal access token for quick access to the GitHub API.

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to authenticate to the API over Basic Authentication.

STEP 5:

Add note and generate token



New personal access token (classic)

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to authenticate to the API over Basic Authentication.

Note

What's this token for?

Expiration

30 days (Apr 18, 2025)

The token will expire on the selected date

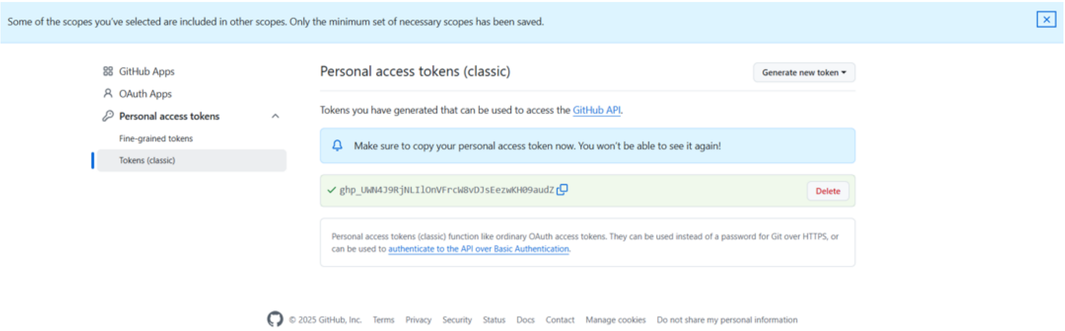
Select scopes

Scopes define the access for personal tokens. Read more about OAuth scopes.

□ repo	Full control of private repositories
□ repo:status	Access commit status
□ repo_deployment	Access deployment status
□ public_repo	Access public repositories
□ repo:invite	Access repository invitations
□ security_events	Read and write security events
□ workflow	Update GitHub Action workflows
□ write:packages	Upload packages to GitHub Package Registry

STEP 6:

Copy the token generated



Some of the scopes you've selected are included in other scopes. Only the minimum set of necessary scopes has been saved.

Personal access tokens (classic)

Tokens you have generated that can be used to access the GitHub API.

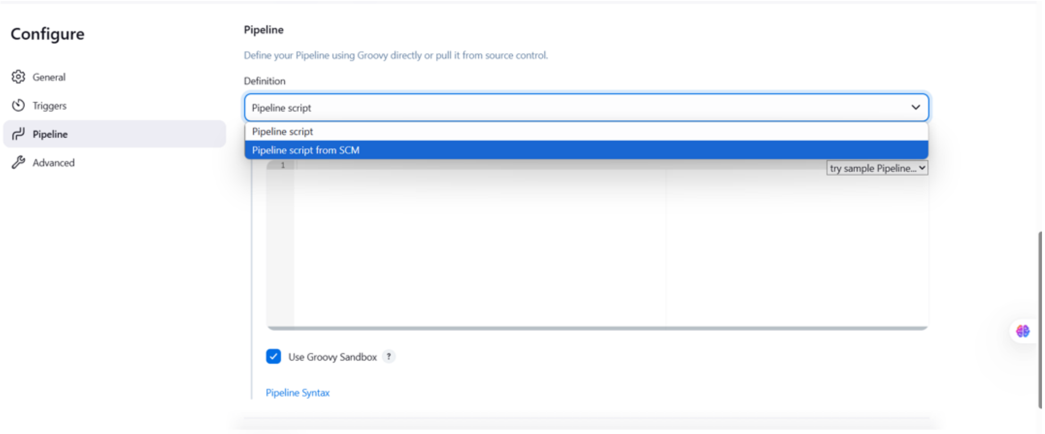
Make sure to copy your personal access token now. You won't be able to see it again!

✓ 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

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to authenticate to the API over Basic Authentication.

STEP 7:

Create a pipeline



Pipeline

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

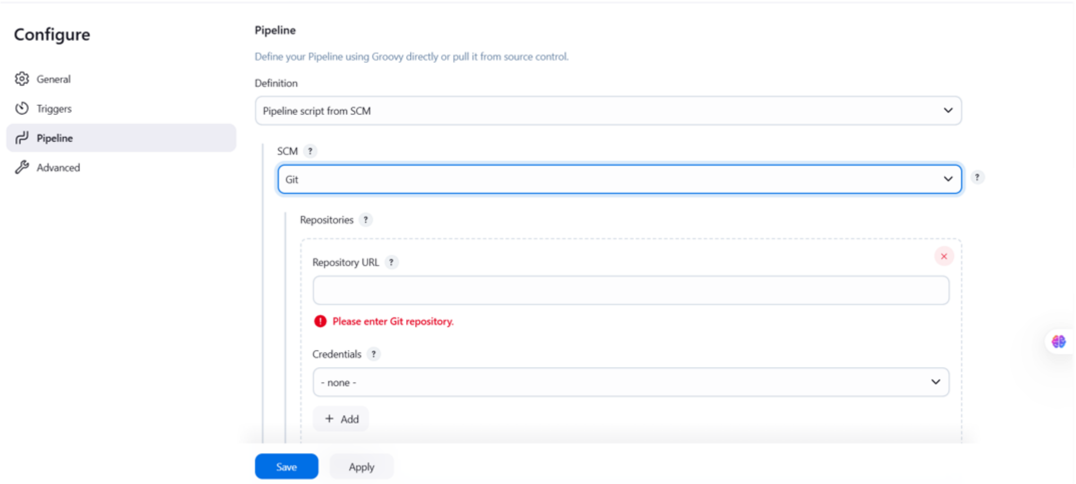
Definition: Pipeline script

- Pipeline script
- Pipeline script from SCM

☑ Use Groovy Sandbox

STEP 8:

Provide the credentials and save



Pipeline

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

Definition: Pipeline script from SCM

SCM: Git

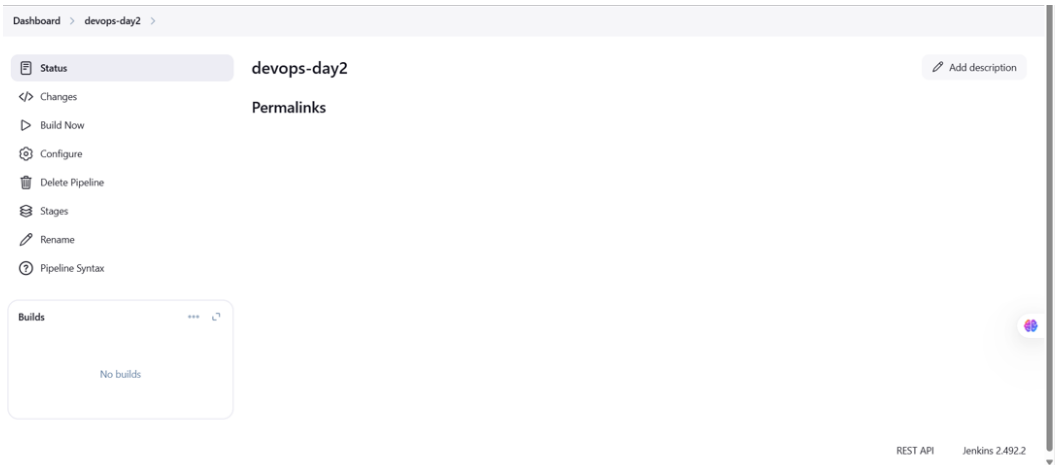
Repositories

Repository URL:

Please enter Git repository.

Credentials: - none -

STEP 9:



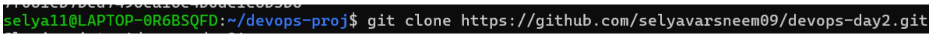
devops-day2

Permalinks

No builds

STEP 10:

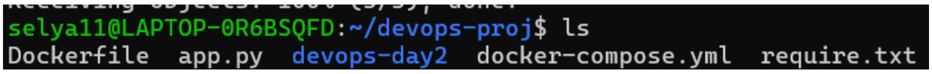
Clone the remote repository



```
selya11@LAPTOP-0R6BSQFD:~/devops-proj$ git clone https://github.com/selyavarsneem09/devops-day2.git
```

STEP 11:

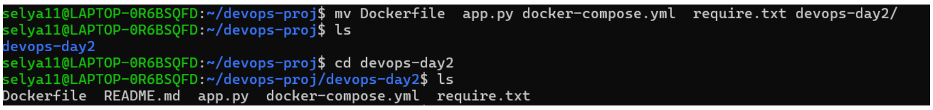
List the files in the directory “devops-proj”



```
selya11@LAPTOP-0R6BSQFD:~/devops-proj$ ls
Dockerfile  app.py  devops-day2  docker-compose.yml  require.txt
```

STEP 12:

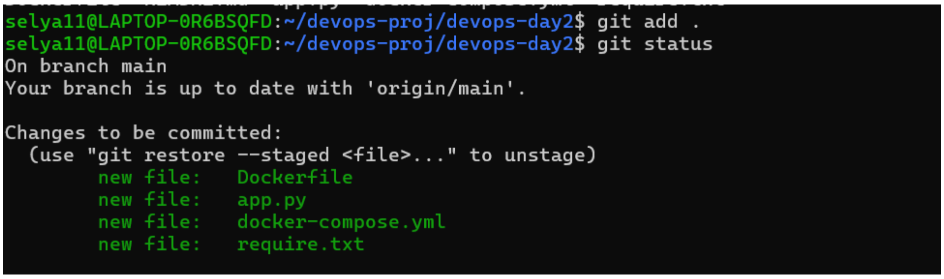
Move the files to the repo and change the directory and list it again



```
selya11@LAPTOP-0R6BSQFD:~/devops-proj$ mv Dockerfile  app.py docker-compose.yml  require.txt devops-day2/
selya11@LAPTOP-0R6BSQFD:~/devops-proj$ ls
devops-day2
selya11@LAPTOP-0R6BSQFD:~/devops-proj$ cd devops-day2
selya11@LAPTOP-0R6BSQFD:~/devops-proj/devops-day2$ ls
Dockerfile  README.md  app.py  docker-compose.yml  require.txt
```

STEP 13:

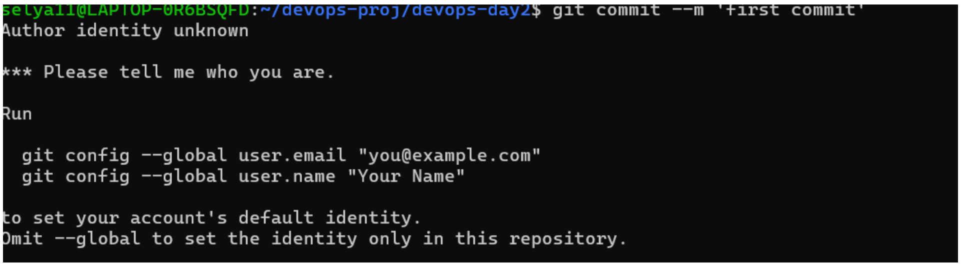
Now add the listed files to the repo



```
selya11@LAPTOP-0R6BSQFD:~/devops-proj/devops-day2$ git add .
selya11@LAPTOP-0R6BSQFD:~/devops-proj/devops-day2$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   Dockerfile
        new file:   app.py
        new file:   docker-compose.yml
        new file:   require.txt
```

STEP 14:

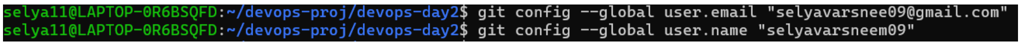
Commit your activity using git commit



```
selya11@LAPTOP-0R6BSQFD:~/devops-proj/devops-day2$ git commit --m 'first commit'
Author identity unknown

*** Please tell me who you are.

Run

  git config --global user.email "you@example.com"
  git config --global user.name "Your Name"

to set your account's default identity.
Omit --global to set the identity only in this repository.
```

STEP 15:

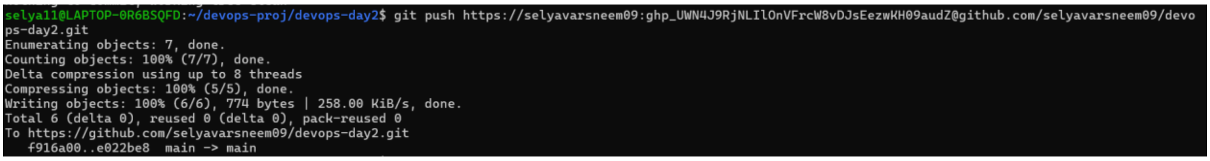
Now config using user name and email



```
selya11@LAPTOP-0R6BSQFD:~/devops-proj/devops-day2$ git config --global user.email "selyavarsnee09@gmail.com"
selya11@LAPTOP-0R6BSQFD:~/devops-proj/devops-day2$ git config --global user.name "selyavarsneem09"
```

STEP 16:

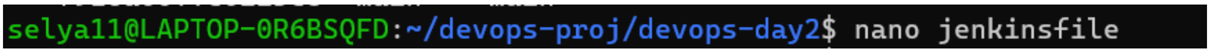
Use push command to push the changes



```
selya11@LAPTOP-0R6BSQFD:~/devops-proj/devops-day2$ git push https://selyavarsneem09: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@github.com/selyavarsneem09/devo
ps-day2.git
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 8 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (6/6), 774 bytes | 258.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/selyavarsneem09/devops-day2.git
   f916a00..e022be8  main -> main
```

STEP 17:

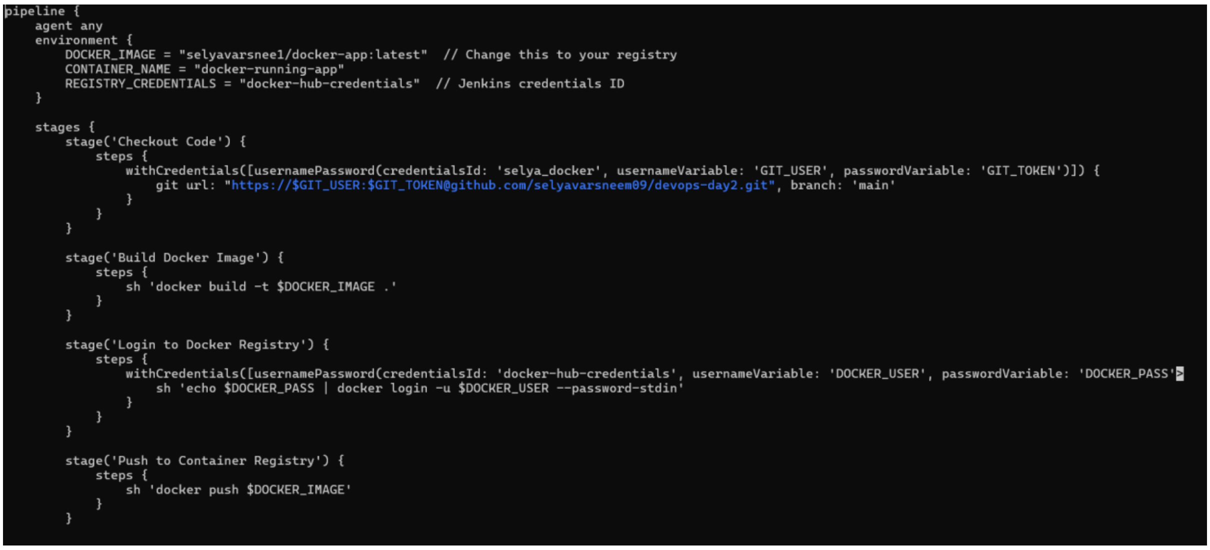
Now create a nano file called jenkinsfile



```
selya11@LAPTOP-0R6BSQFD:~/devops-proj/devops-day2$ nano jenkinsfile
```

STEP 18:

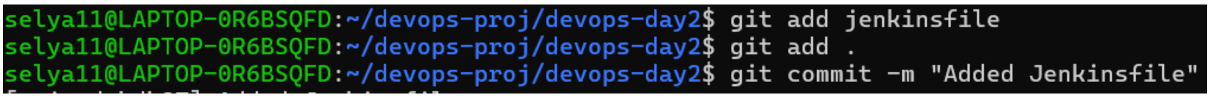
Edit the Jenkins file by providing docker credentials and git hub credentials



```
pipeline {
    agent any
    environment {
        DOCKER_IMAGE = "selyavarsnee1/docker-app:latest"  // Change this to your registry
        CONTAINER_NAME = "docker-running-app"
        REGISTRY_CREDENTIALS = "docker-hub-credentials"  // Jenkins credentials ID
    }

    stages {
        stage('Checkout Code') {
            steps {
                withCredentials([usernamePassword(credentialsId: 'selya_docker', usernameVariable: 'GIT_USER', passwordVariable: 'GIT_TOKEN')]) {
                    git url: "https://$GIT_USER:$GIT_TOKEN@github.com/selyavarsneem09/devops-day2.git", branch: 'main'
                }
            }
        }

        stage('Build Docker Image') {
            steps {
                sh 'docker build -t $DOCKER_IMAGE .'
            }
        }

        stage('Login to Docker Registry') {
            steps {
                withCredentials([usernamePassword(credentialsId: 'docker-hub-credentials', usernameVariable: 'DOCKER_USER', passwordVariable: 'DOCKER_PASS'>
                    sh 'echo $DOCKER_PASS | docker login -u $DOCKER_USER --password-stdin'
                }
            }
        }

        stage('Push to Container Registry') {
            steps {
                sh 'docker push $DOCKER_IMAGE'
            }
        }
```

STEP 19:

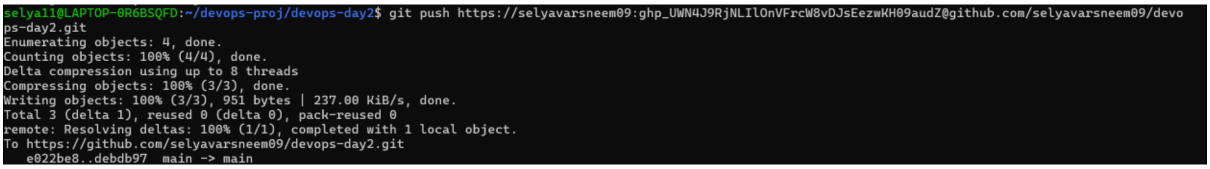
Now add the Jenkins file to the git repo, then commit it



```
selya11@LAPTOP-0R6BSQFD:~/devops-proj/devops-day2$ git add jenkinsfile
selya11@LAPTOP-0R6BSQFD:~/devops-proj/devops-day2$ git add .
selya11@LAPTOP-0R6BSQFD:~/devops-proj/devops-day2$ git commit -m "Added Jenkinsfile"
```

STEP 20:

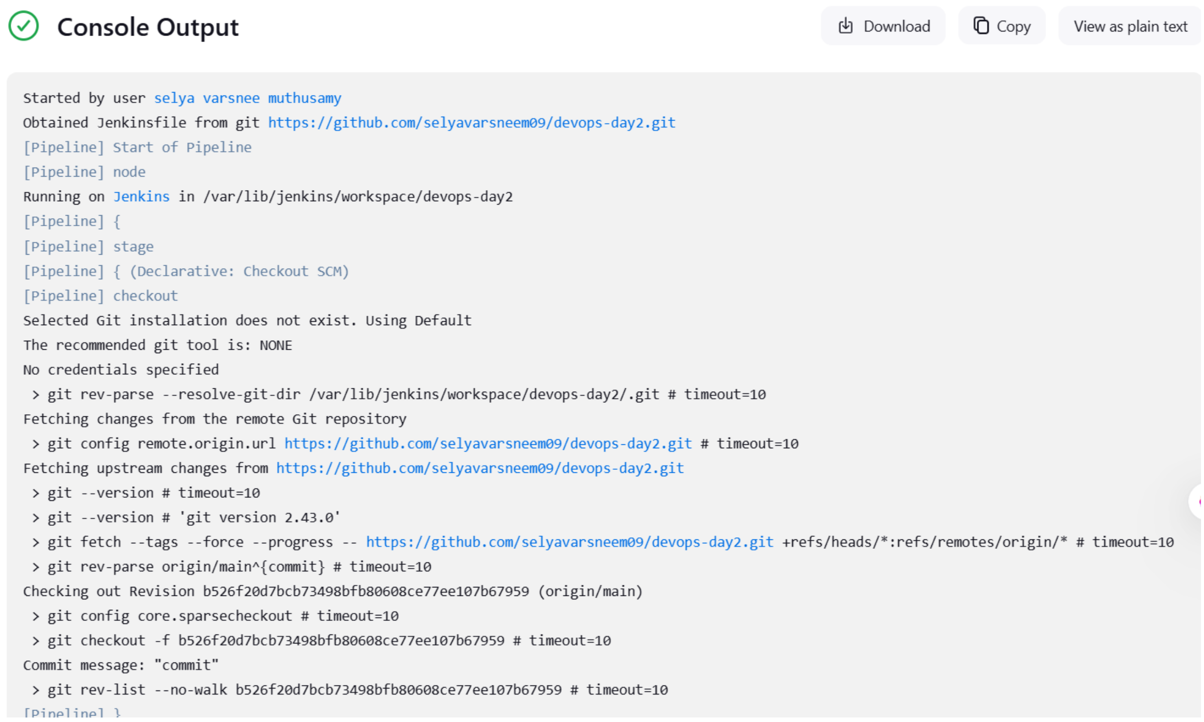
Then push it to the repository



```
selya11@LAPTOP-0R6BSQFD:~/devops-proj/devops-day2$ git push https://selyavarsneem09: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@github.com/selyavarsneem09/devo
ps-day2.git
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 951 bytes | 237.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/selyavarsneem09/devops-day2.git
   e022be8..debdb97  main -> main
```

STEP 21:

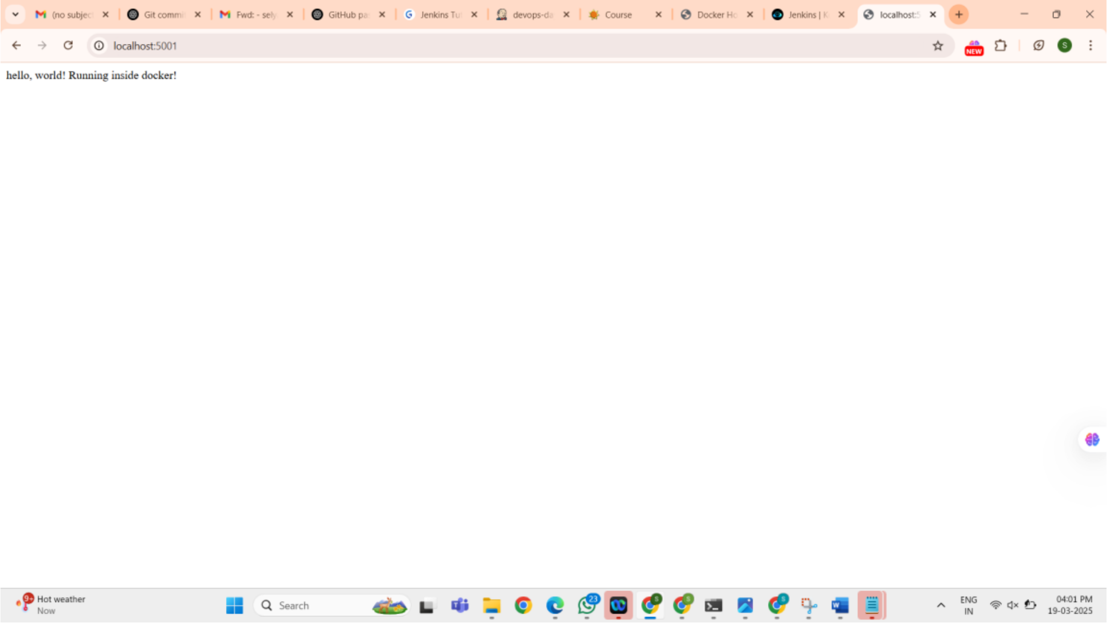
Go to Jenkins and give “build now” option



```
Started by user selya varsnee muthusamy
Obtained Jenkinsfile from git https://github.com/selyavarsneem09/devops-day2.git
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/devops-day2
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Declarative: Checkout SCM)
[Pipeline] checkout
Selected Git installation does not exist. Using Default
The recommended git tool is: NONE
No credentials specified
 > git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/devops-day2/.git # timeout=10
Fetching changes from the remote Git repository
 > git config remote.origin.url https://github.com/selyavarsneem09/devops-day2.git # timeout=10
Fetching upstream changes from https://github.com/selyavarsneem09/devops-day2.git
 > git --version # timeout=10
 > git --version # 'git version 2.43.0'
 > git fetch --tags --force --progress -- https://github.com/selyavarsneem09/devops-day2.git +refs/heads/*:refs/remotes/origin/* # timeout=10
 > git rev-parse origin/main^{commit} # timeout=10
Checking out Revision b526f20d7bcb73498bfb80608ce77ee107b67959 (origin/main)
 > git config core.sparsecheckout # timeout=10
 > git checkout -f b526f20d7bcb73498bfb80608ce77ee107b67959 # timeout=10
Commit message: "commit"
 > git rev-list --no-walk b526f20d7bcb73498bfb80608ce77ee107b67959 # timeout=10
[Pipeline] }
```

STEP 22:

Run the localhost on port number : 5001 and see the screen with message “hello world”



hello, world! Running inside docker!