Introduction to DevOps
Assignment 1 - GitHub Assignment

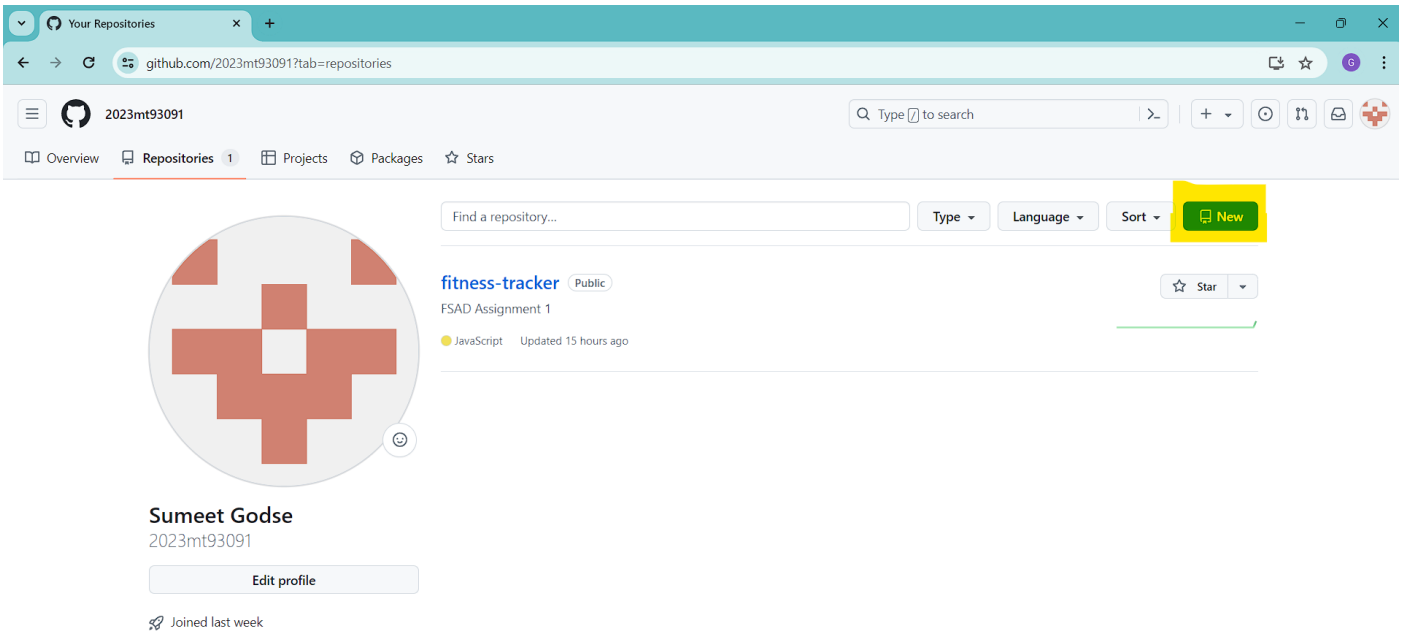
By - Godse Sumeet Deelip (2023mt93091)

GitHub - https://github.com/2023mt93091/devops-assignment-1

Problem Statement - ABC Organization would like to opt for the distributed version control system to upgrade their environment; where Git has been selected as the solution. You been assigned as a consultant to educate the migration process to move their Source Code from Centralized to Distributed systems. As a phase one, you would like to go ahead with a workshop to demonstrate below operation to make the ABC team comfortable.

Part 1 -

1. Open GitHub and create a repository



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere?
Import a repository.

Required fields are marked with an asterisk (*).

Owner * / Repository name *

2023mt93091 / devops-assignment-1

devops-assignment-1 is available.

Great repository names are short and memorable. Need inspiration? How about cautious-broccoli ?

Description (optional)

Introduction to DevOps - Assignment 1 - GitHub and Jenkins demo

- Public
  Anyone on the internet can see this repository. You choose who can commit.
- Private
  You choose who can see and commit to this repository.

Initialize this repository with:

- [x] Add a README file
  This is where you can write a long description for your project. Learn more about READMEs.

Add .gitignore

.gitignore template: None

Choose which files not to track from a list of templates. Learn more about ignoring files.

Choose a license

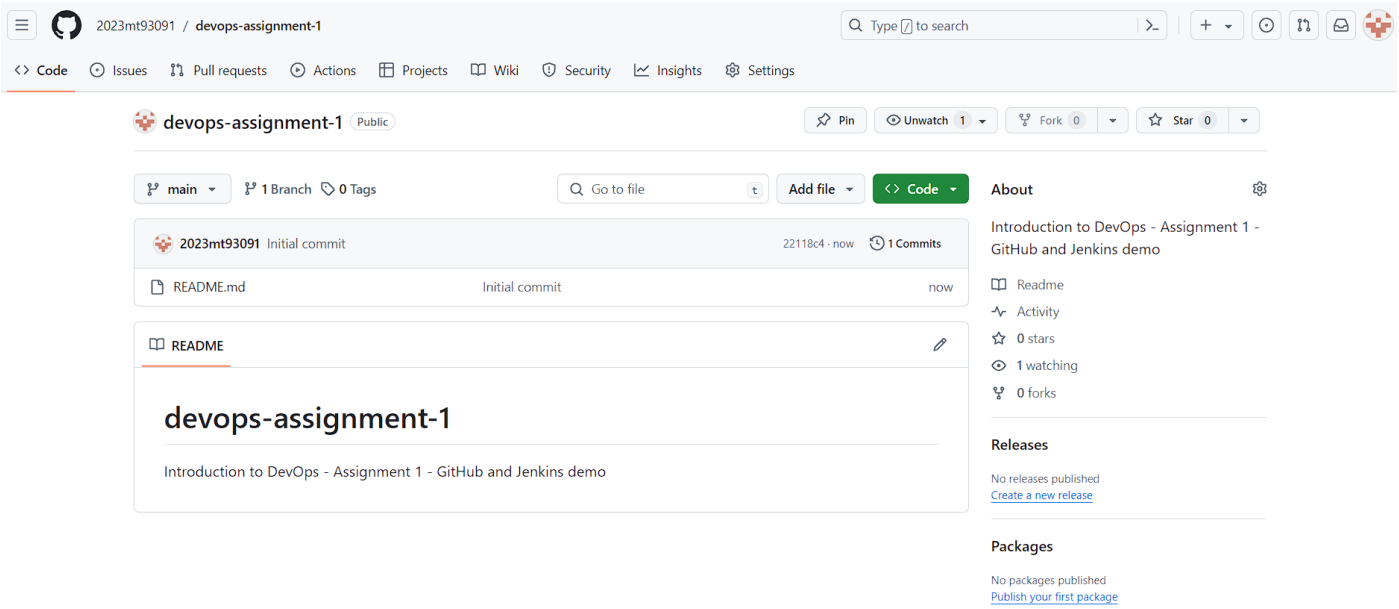
License: None

A license tells others what they can and can't do with your code. Learn more about licenses.

This will set `main` as the default branch. Change the default name in your settings.

You are creating a public repository in your personal account.

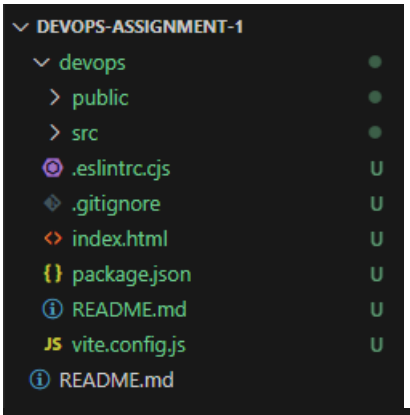
Create repository



2. Add two or more directories and some raw code files to the repository. I will be creating a react based application here and adding the same to the repository.

```
sumee@LAPTOP-I8BBG5OJ MINGW64 /d/DevProjects
$ git clone https://github.com/2023mt93091/devops-assignment-1.git
Cloning into 'devops-assignment-1'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.

sumee@LAPTOP-I8BBG5OJ MINGW64 /d/DevProjects
$ cd devops-assignment-1/

sumee@LAPTOP-I8BBG5OJ MINGW64 /d/DevProjects/devops-assignment-1 (main)
$ ls
README.md
```

```
sumee@LAPTOP-I8BBG5OJ MINGW64 /d/DevProjects/devops-assignment-1 (main)
$ yarn create vite
yarn create v1.22.10
[1/4] Resolving packages...
[2/4] Fetching packages...
[3/4] Linking dependencies...
[4/4] Building fresh packages...

success Installed "create-vite@5.2.3" with binaries:
      - create-vite
      - cva
√ Project name: ... devops
√ Select a framework: » React
√ Select a variant: » JavaScript

Scaffolding project in D:\DevProjects\devops-assignment-1\devops...

Done. Now run:

  cd devops
  yarn
  yarn dev

Done in 14.54s.
```




```
sumee@LAPTOP-I8BBG5OJ MINGW64 /d/DevProjects/devops-assignment-1/devops (main)
$ git add *
warning: LF will be replaced by CRLF in devops/README.md.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in devops/index.html.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in devops/package.json.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in devops/src/App.css.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in devops/src/App.jsx.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in devops/src/index.css.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in devops/src/main.jsx.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in devops/vite.config.js.
The file will have its original line endings in your working directory

sumee@LAPTOP-I8BBG5OJ MINGW64 /d/DevProjects/devops-assignment-1/devops (main)
$ git commit -m "add a sample react project with two directories"
[main 2d2e2d5] add a sample react project with two directories
 10 files changed, 211 insertions(+)
 create mode 100644 devops/README.md
 create mode 100644 devops/index.html
 create mode 100644 devops/package.json
 create mode 100644 devops/public/vite.svg
 create mode 100644 devops/src/App.css
 create mode 100644 devops/src/App.jsx
 create mode 100644 devops/src/assets/react.svg
 create mode 100644 devops/src/index.css
 create mode 100644 devops/src/main.jsx
 create mode 100644 devops/vite.config.js

sumee@LAPTOP-I8BBG5OJ MINGW64 /d/DevProjects/devops-assignment-1/devops (main)
$ git push
info: please complete authentication in your browser...
Enumerating objects: 17, done.
Counting objects: 100% (17/17), done.
Delta compression using up to 12 threads
Compressing objects: 100% (14/14), done.
Writing objects: 100% (16/16), 5.91 KiB | 2.95 MiB/s, done.
Total 16 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/2023mt93091/devops-assignment-1.git
   22118c4..2d2e2d5  main -> main
```

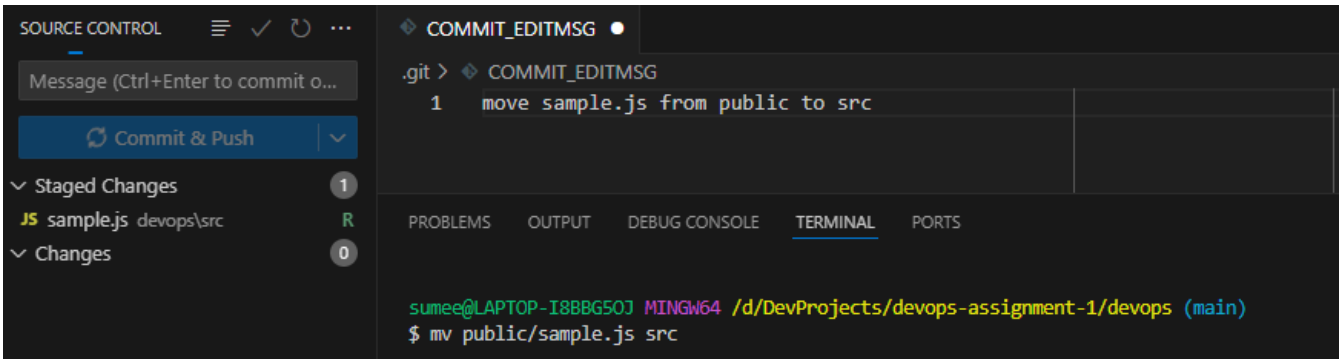
devops-assignment-1 / devops /

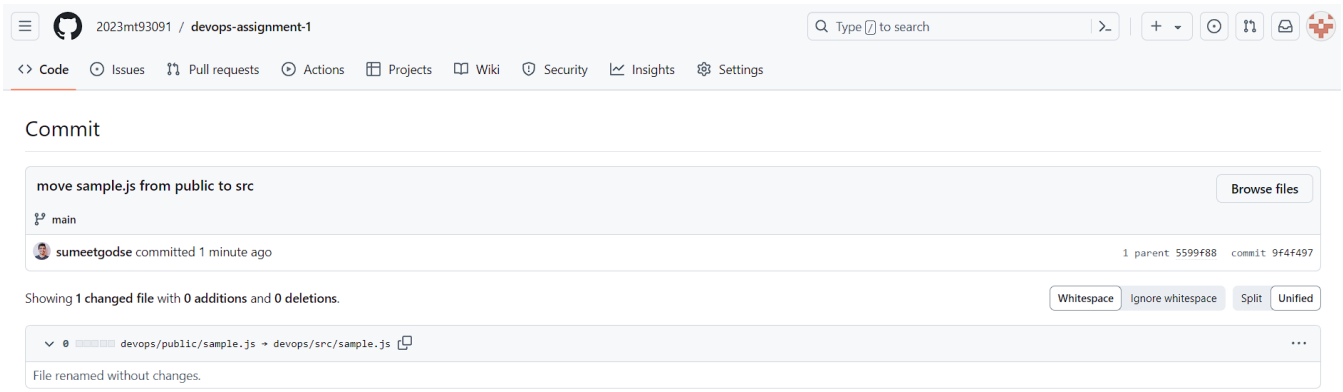
Add file

sumeetgodse add a sample react project with two directories 2d2e2d5 · 1 minute ago History

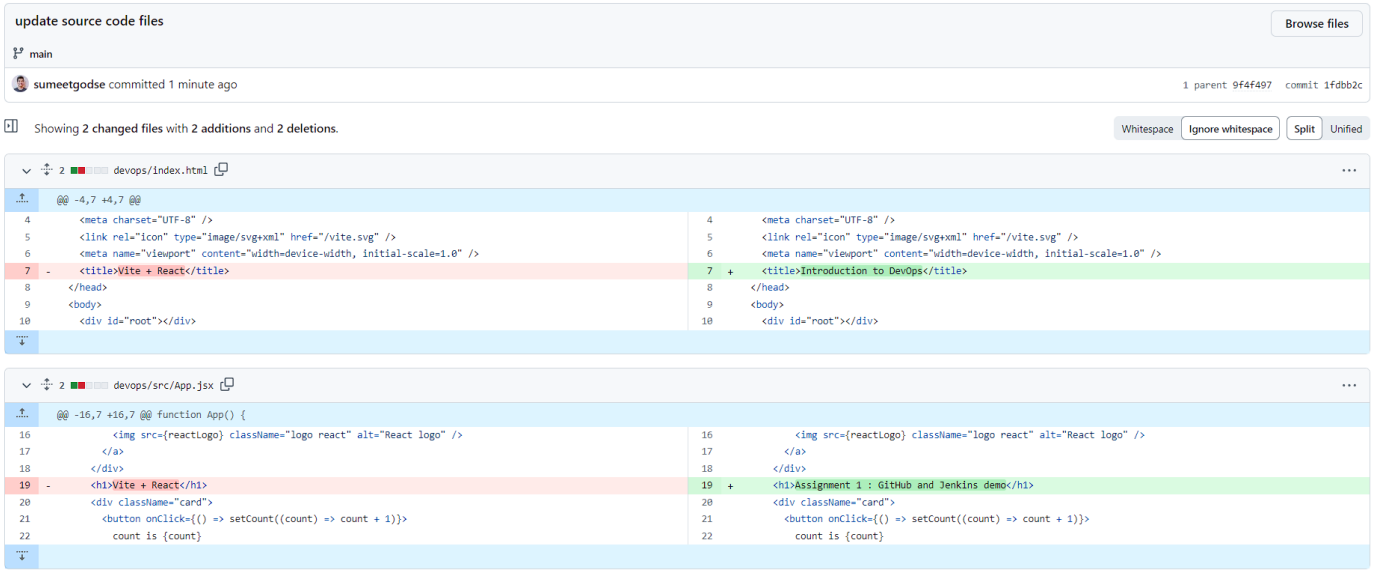
Name	Last commit message	Last commit date
..		
public	add a sample react project with two directories	1 minute ago
src	add a sample react project with two directories	1 minute ago
README.md	add a sample react project with two directories	1 minute ago
index.html	add a sample react project with two directories	1 minute ago
package.json	add a sample react project with two directories	1 minute ago
vite.config.js	add a sample react project with two directories	1 minute ago

3. Move code from one directory to another. I will be moving sample.js from public to src.



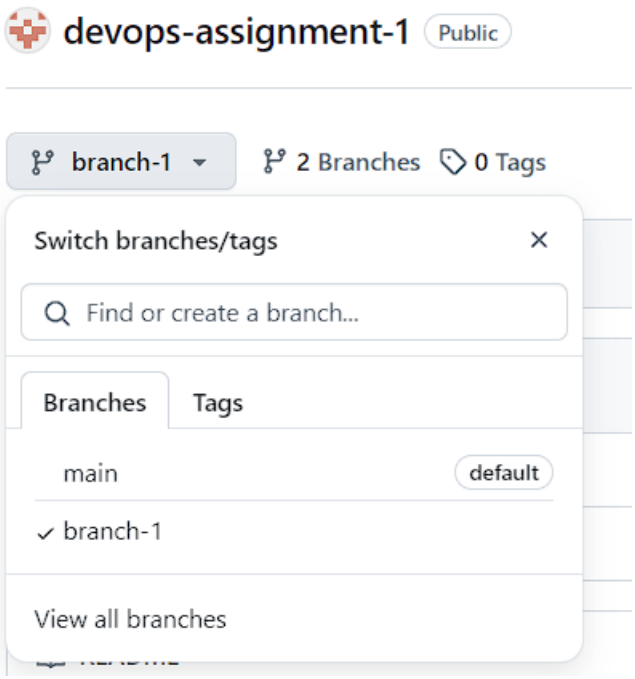


4. Update source code files and display the difference.

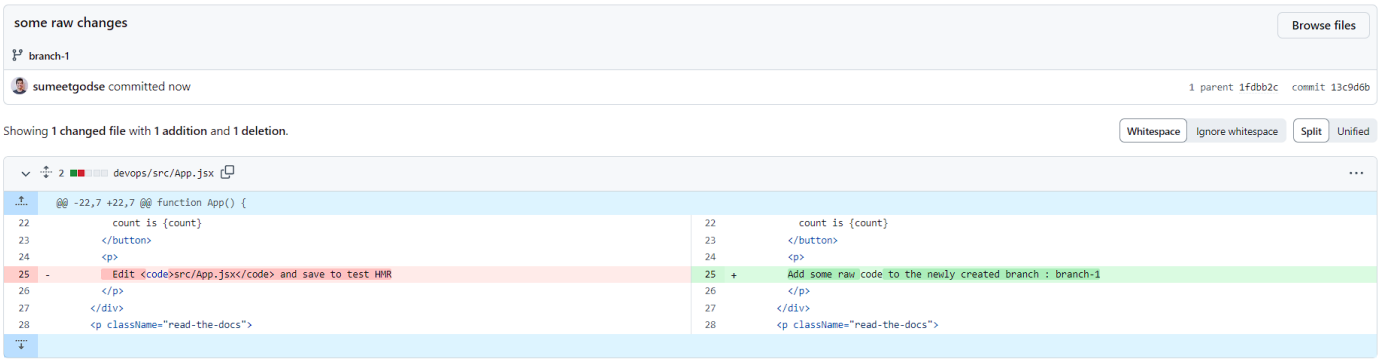


5. Create a branch

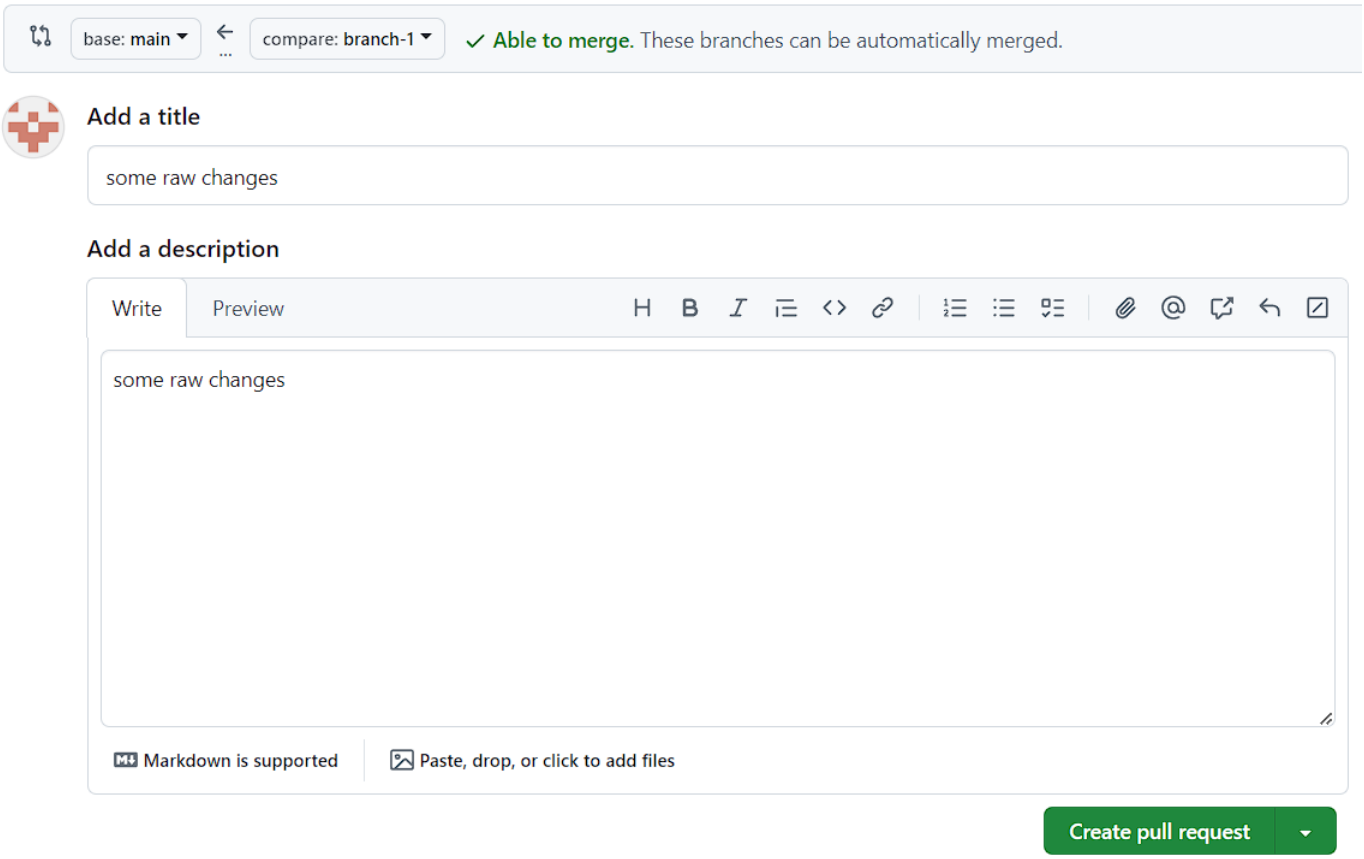
```
sumee@LAPTOP-I8BBG5OJ MINGW64 /d/DevProjects/devops-assignment-1/devops (main)
$ git checkout -b branch-1
Switched to a new branch 'branch-1'

sumee@LAPTOP-I8BBG5OJ MINGW64 /d/DevProjects/devops-assignment-1/devops (branch-1)
$ git push -u origin branch-1
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'branch-1' on GitHub by visiting:
remote:      https://github.com/2023mt93091/devops-assignment-1/pull/new/branch-1
remote:
To https://github.com/2023mt93091/devops-assignment-1.git
 * [new branch]      branch-1 -> branch-1
Branch 'branch-1' set up to track remote branch 'branch-1' from 'origin'.
```

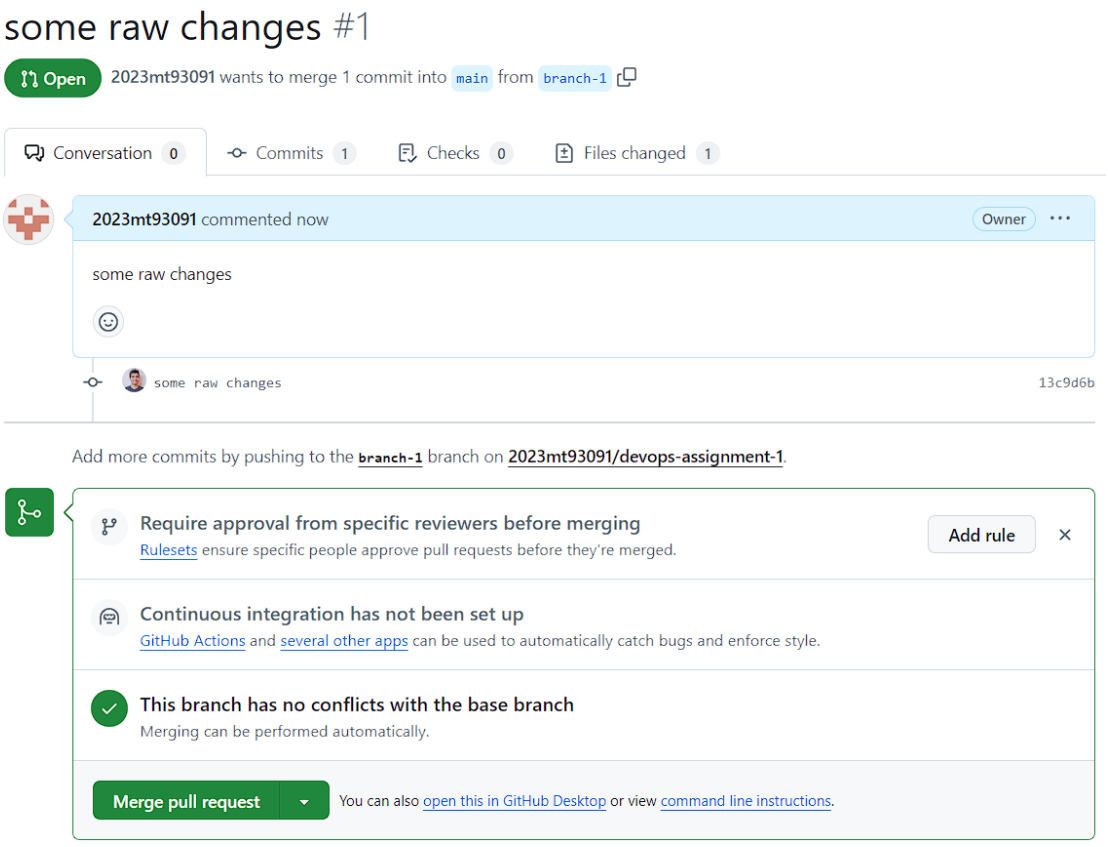


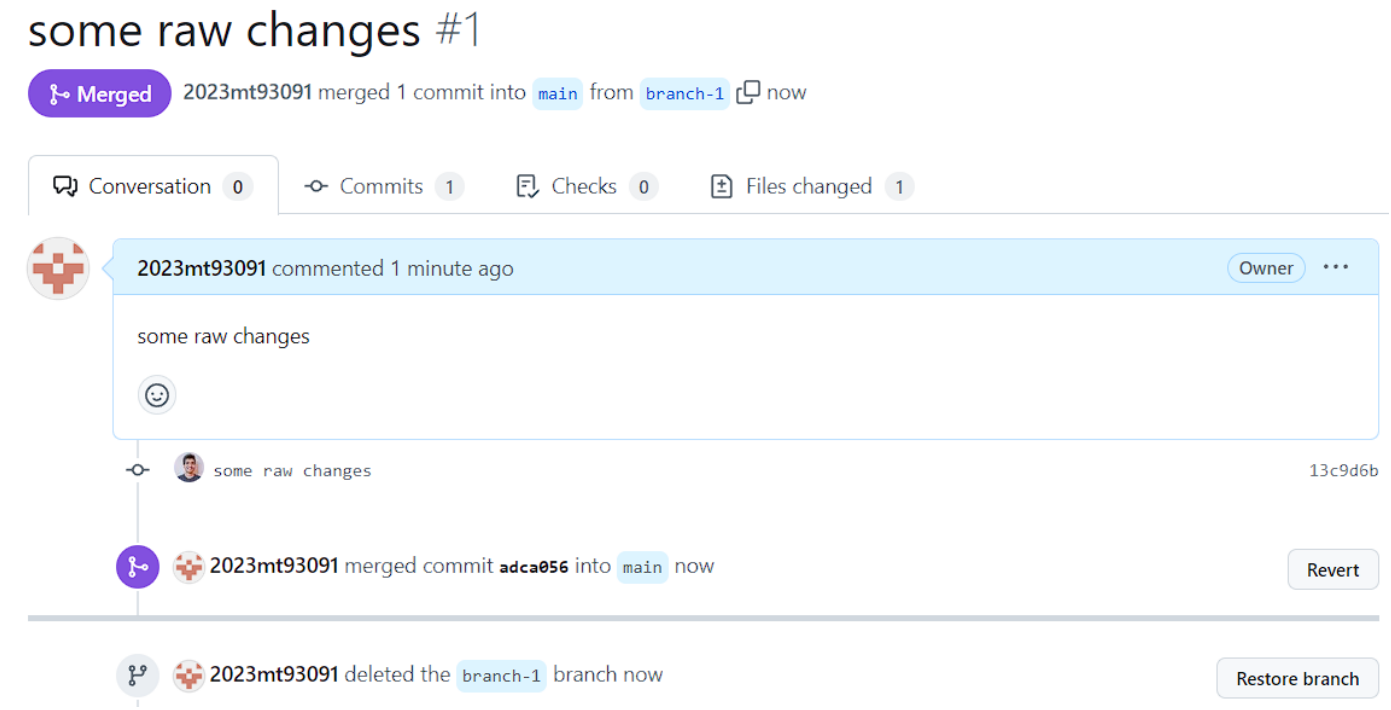
6. Add some raw code to the branch

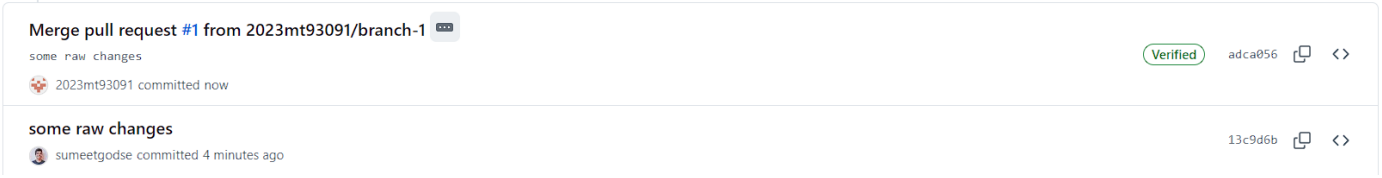


7. Merge the branch with main line. We can create a pull request to do this.









Advantages of moving from Centralized Source Code to Distributed Version Control.

Migrating from a centralized source code management system (SCM) to a distributed version control system (DVCS) like Git offers several advantages:

1. **Offline Work**:
   With DVCS, developers can work offline since they have a local copy of the entire repository. They can commit changes, create branches, and perform other operations without needing a constant connection to a central server.

2. **Redundancy and Backup**:
   Every developer has a complete copy of the repository, providing redundancy. This means if a central server goes down, developers can continue working and push changes once the server is back up. Additionally, each developer's local repository serves as a backup.

3. **Flexible Workflows**:
   DVCS systems like Git support various workflow models, such as Gitflow, GitHub flow, or GitLab flow. Teams can choose the workflow that best fits their development process.

4. **Easy Branching and Merging**:
   DVCS systems excel at branching and merging. Developers can create branches for features, bug

fixes, or experiments without affecting the main codebase. Merging is also easier and less error-prone compared to centralized systems.

5. **Parallel Development**:
   Since each developer works on their own local copy, multiple developers can work in parallel without stepping on each other's toes. This fosters faster development cycles and better collaboration.

6. **Scalability**:
   DVCS scales better as the project and team size grow. Since most operations are performed locally, the performance doesn't degrade as the repository size increases or when there are many concurrent users.

7. **Forking and Pull Requests**:
   DVCS encourages contribution from external developers through forking and pull requests. Forking allows developers to create their own copy of the repository to work on a feature or fix independently. Pull requests enable developers to propose changes and request them to be merged into the main repository.

8. **Security**:
   With DVCS, developers can sign commits, ensuring the authenticity of code changes. Additionally, access control mechanisms can be enforced both locally and on remote repositories, enhancing security.

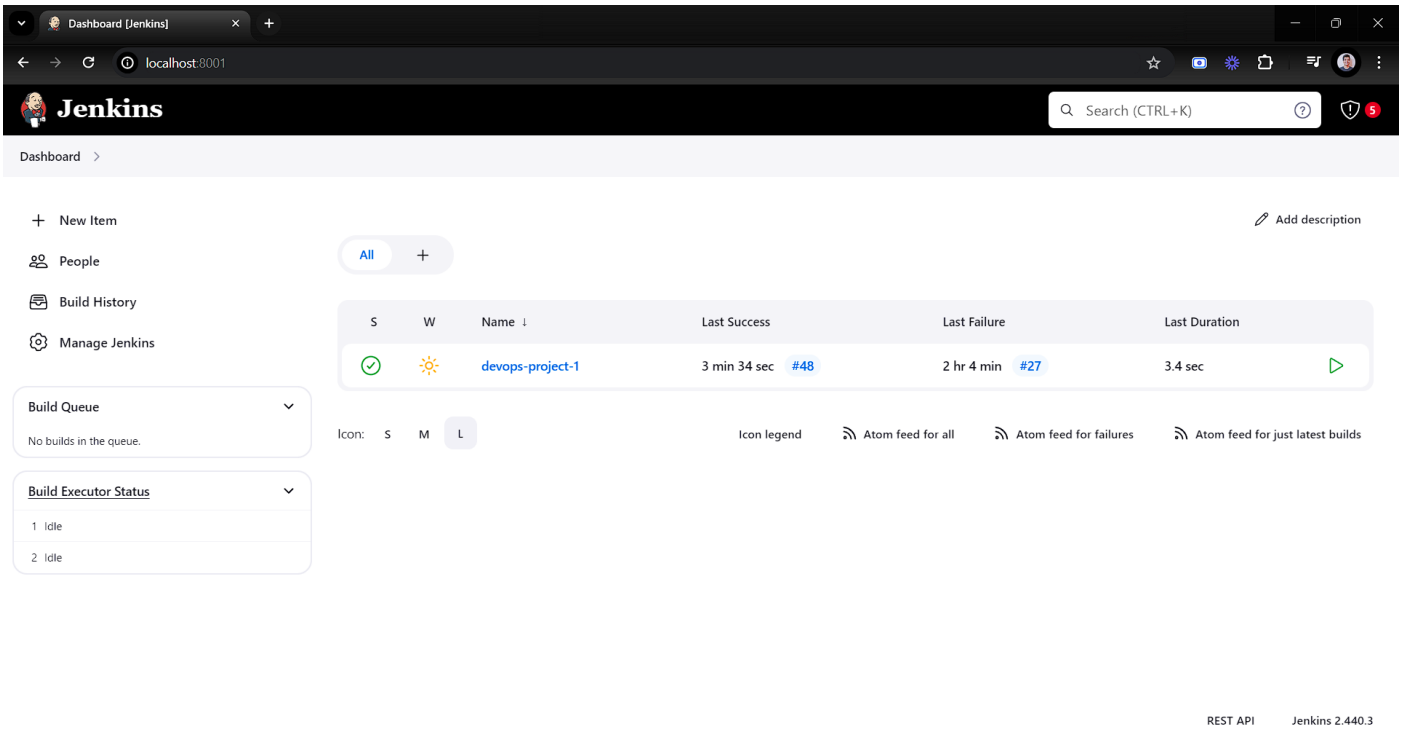
9. **Decentralization**:
   DVCS reduces the reliance on a single central server, distributing the control and responsibility among all developers. This decentralization can lead to increased agility and resilience in large, distributed teams.

Overall, migrating from a centralized source code system to a distributed version control system brings numerous benefits, improving collaboration, productivity, and flexibility in software development workflows.

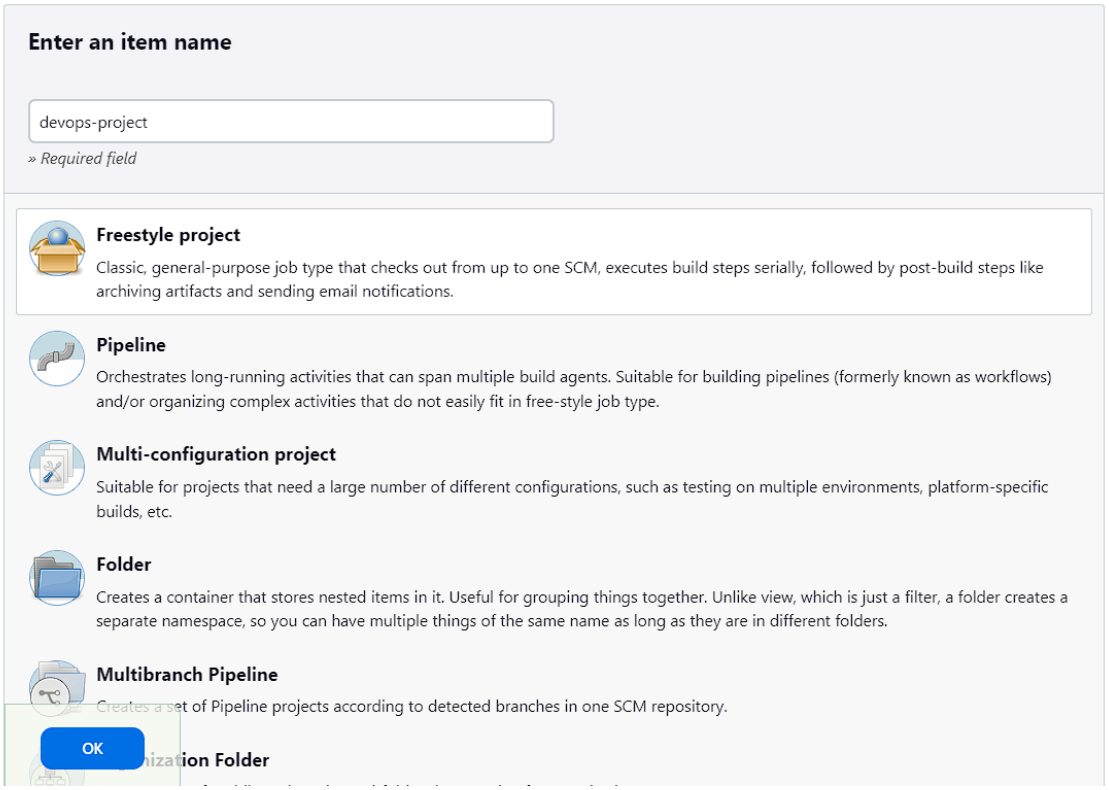
Part 2 -

Create the Jenkins pipeline, which is checking out the code and build and compiling it on every commit automatically.

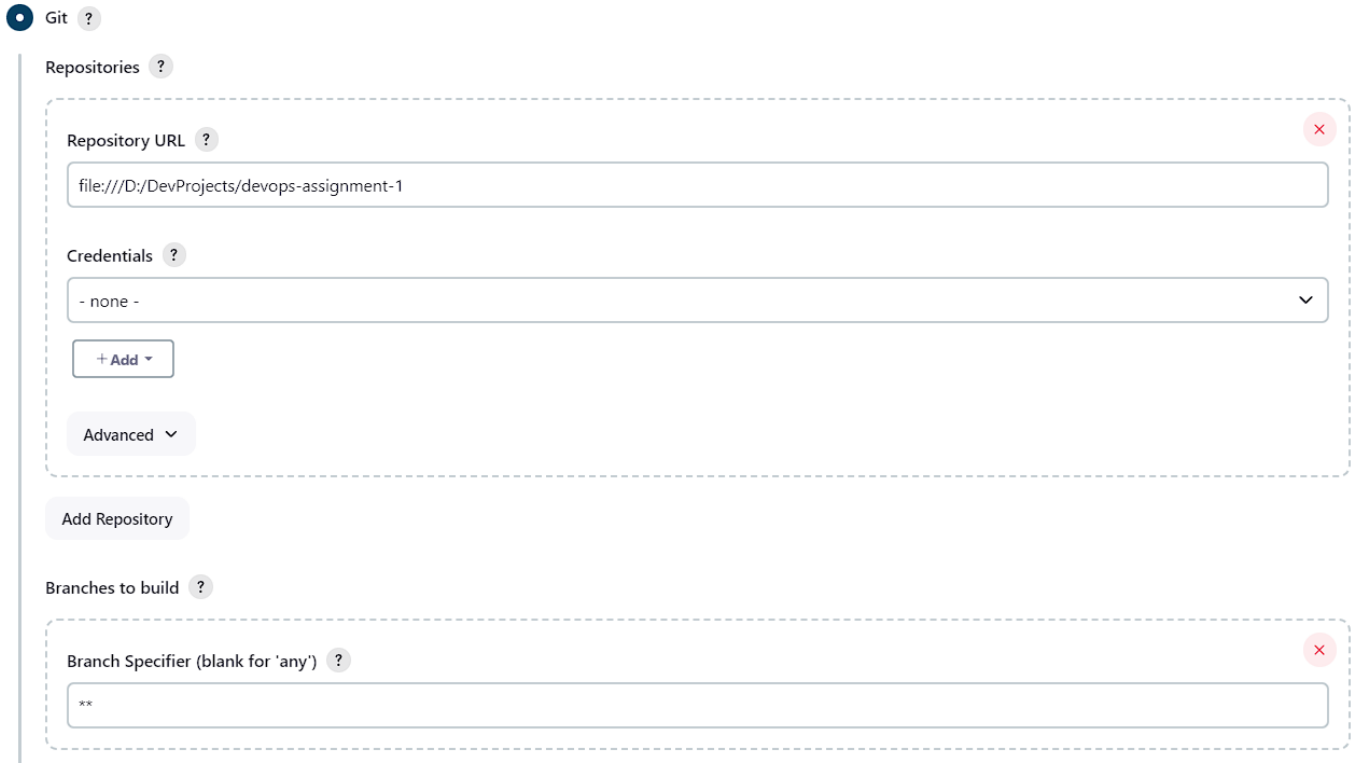
1. Install Jenkins on local machine, by referring the documentation. You must also have compatible java installed in your machine. Specify Jenkins port number etc. Once all setup is done Jenkins will run as a service which can be accessed using the services section of control panel. The GUI can be accessed on http://localhost:{PORT_THAT_YOU_SET} (I gave 8001 as port number)



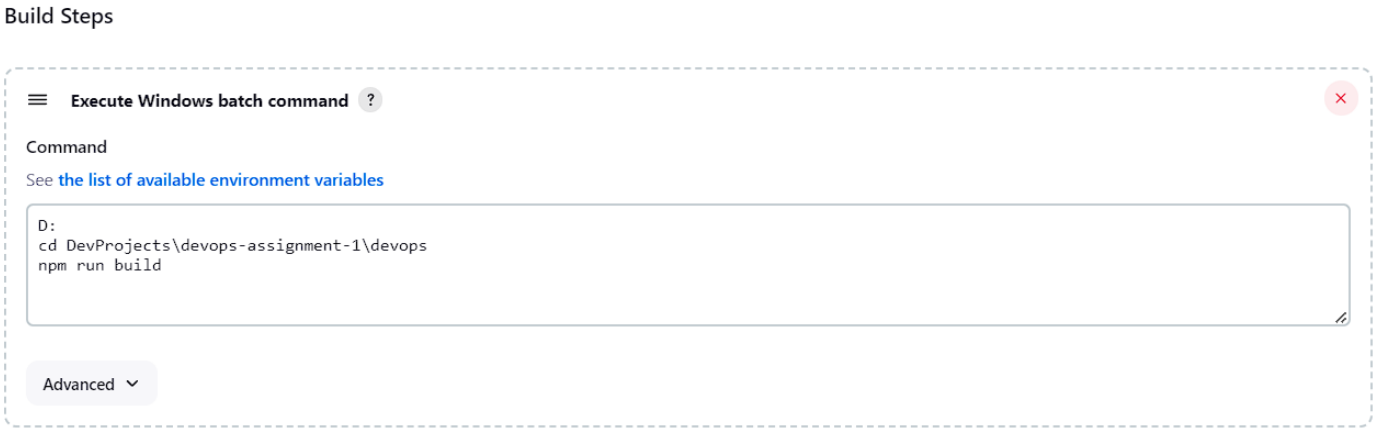
2. Create a new freestyle project. Do the configuration as below.



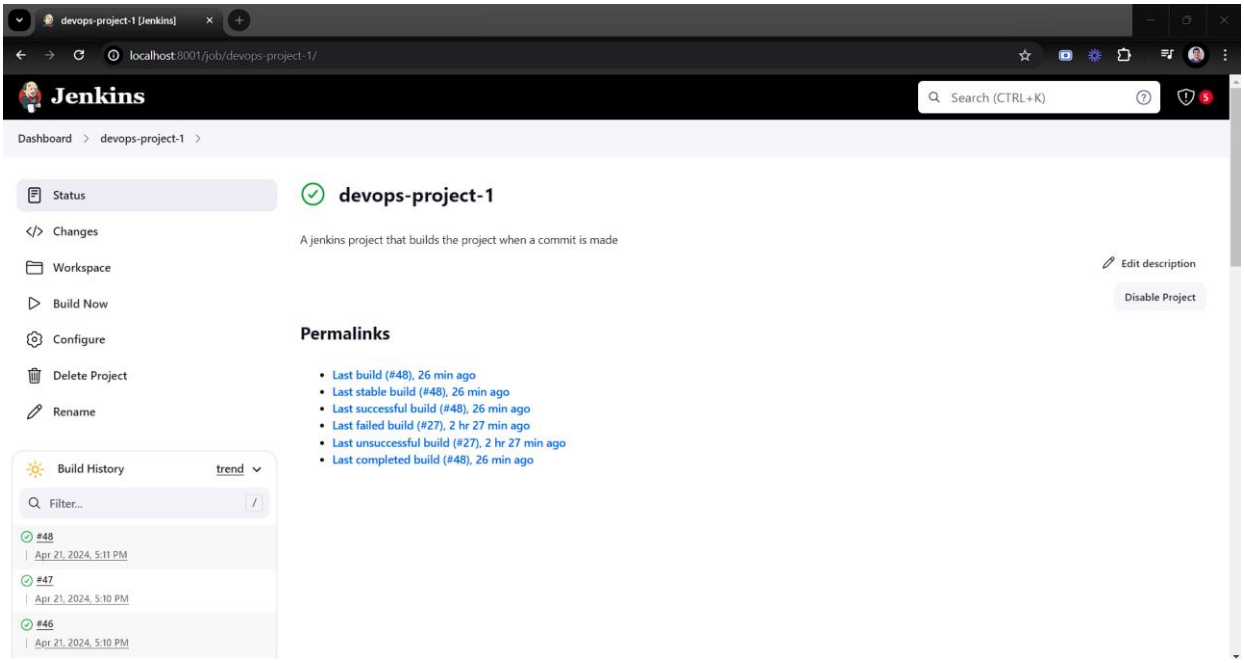
3. Update configuration - Source Code Management. Add the path to your local git repository.



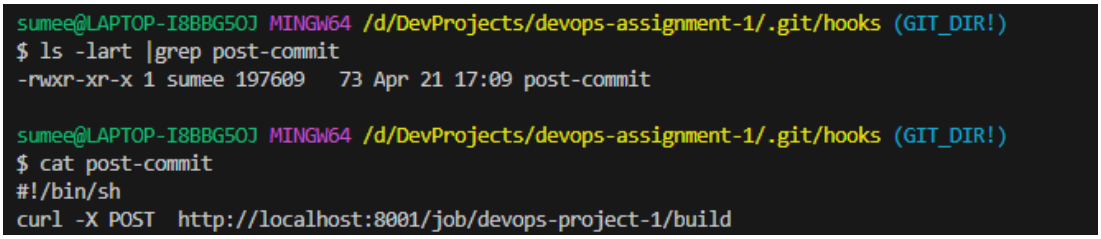
4. Update configuration - Build Steps. Add the steps for your project compilation and build



5. Keep all other settings as default and hit Save button. You will see a new project created.

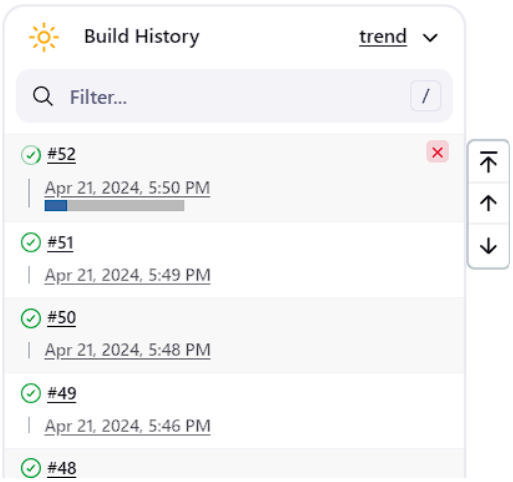


6. Now create a new executable file called ‘post-commit’ inside your projects `.git/hooks` directory. It should contain a curl command that makes a request to your Jenkins project to run a new build.
`curl -X POST` http://localhost:8001/job/{project-name}/build

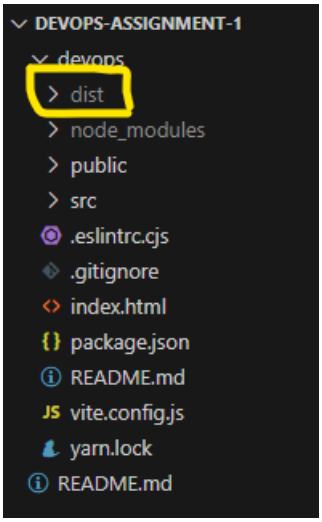


7. Now we are all set to go. Make a new commit in your GitHub repository. A new build will be then triggered in Jenkins.

```
sumee@LAPTOP-I8BBG5OJ MINGW64 /d/DevProjects/devops-assignment-1/devops (main)
$ git add src/sample.js

sumee@LAPTOP-I8BBG5OJ MINGW64 /d/DevProjects/devops-assignment-1/devops (main)
$ git commit -m "This commit will trigger a new build"
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
[main e14bf92] This commit will trigger a new build
 1 file changed, 1 insertion(+), 1 deletion(-)
```



8. This will compile and build our application code and create a new folder named ‘dist’ in our project which can be served to end users.



9. You can click on the build number check the console output.

Hence, in this way we can leverage Jenkins to checkout the code and do automated build and compilation on every commit.