Introduction to Git and GitHub

Intro to Git
Git is a version control system. The full form of Git is
Global Information Tracker

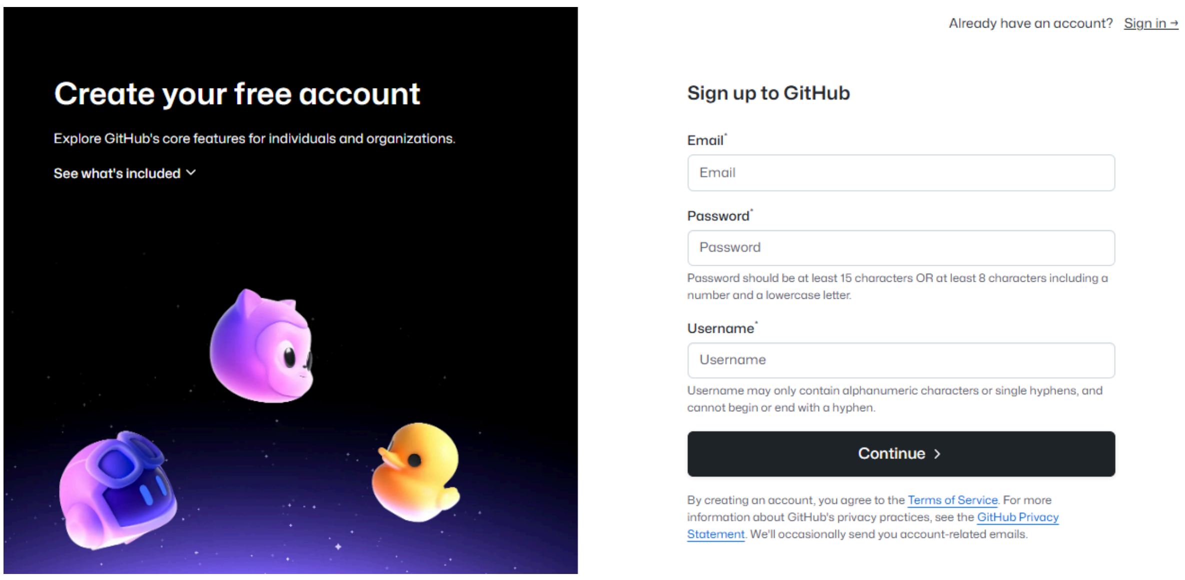
We primarily use Git to easily control and manage different versions of a project. Git is also very helpful for quickly switching from one state of the project to another. Even if a file is accidentally deleted, it can be restored using Git! Moreover, Git allows tracking all kinds of changes made to the project files.

Using Web Interface

1. Setting Up GitHub

1.1 Creating a GitHub Account

1. Visit GitHub and click **Sign Up**.
2. Enter your username, email, and password.
3. Complete the verification process and click **Create Account**.



Create your free account

Explore GitHub's core features for individuals and organizations.

See what's included

Already have an account? Sign in →

Sign up to GitHub

Email*
Email

Password*
Password

Password should be at least 15 characters OR at least 8 characters including a number and a lowercase letter.

Username*
Username

Username may only contain alphanumeric characters or single hyphens, and cannot begin or end with a hyphen.

Continue >

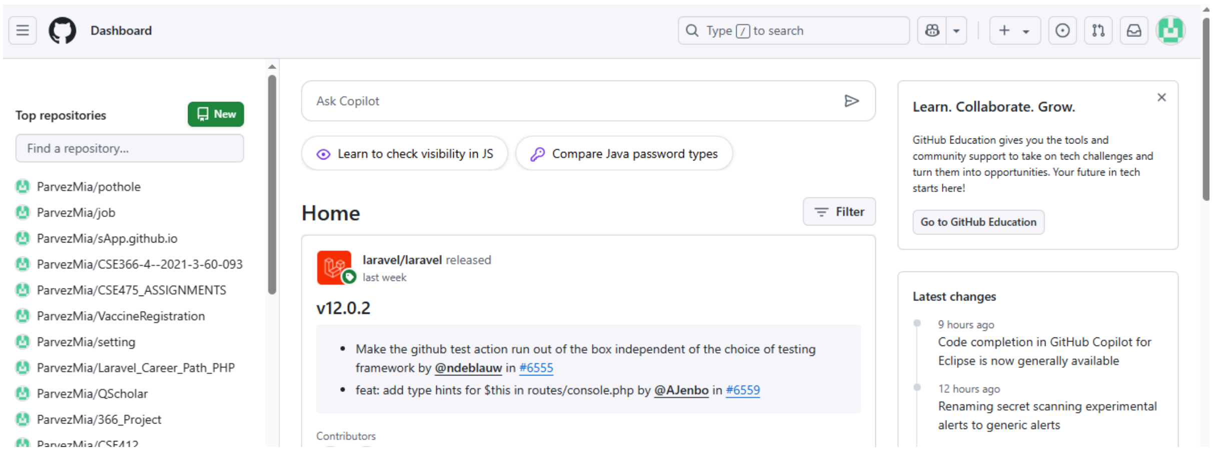
By creating an account, you agree to the Terms of Service. For more information about GitHub's privacy practices, see the GitHub Privacy Statement. We'll occasionally send you account-related emails.

Or Click the **Sign in** Page if you already have an account.

1.2 Exploring the Dashboard

After signing in, you'll see the **GitHub Dashboard**, where you can:

- View repositories
- See recent activities
- Access settings



ParvezMia/pothole
ParvezMia/job
ParvezMia/sApp.github.io
ParvezMia/CSE366-4--2021-3-60-093
ParvezMia/CSE475_ASSIGNMENTS
ParvezMia/VaccineRegistration
ParvezMia/setting
ParvezMia/Laravel_Career_Path_PHP
ParvezMia/QScholar
ParvezMia/366_Project

laravel/laravel released
last week

v12.0.2

- Make the github test action run out of the box independent of the choice of testing framework by @ndeblauw in #6555
- feat: add type hints for $this in routes/console.php by @AJenbo in #6559

Learn. Collaborate. Grow.

GitHub Education gives you the tools and community support to take on tech challenges and turn them into opportunities. Your future in tech starts here!

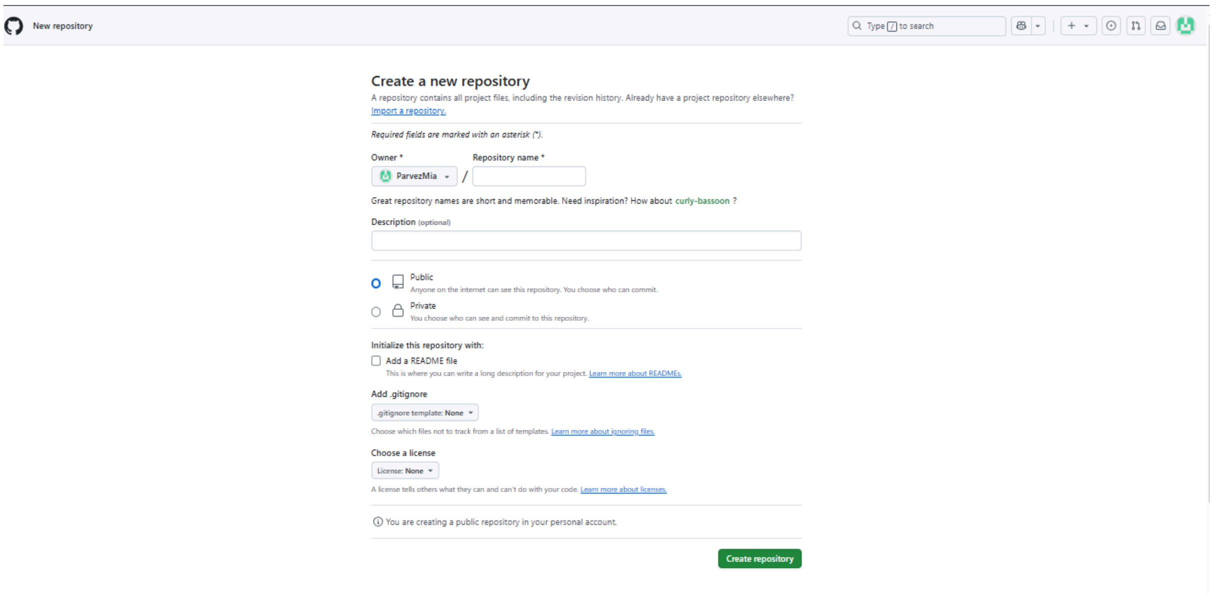
9 hours ago
Code completion in GitHub Copilot for Eclipse is now generally available

12 hours ago
Renaming secret scanning experimental alerts to generic alerts

2. Creating and Managing Repositories

2.1 Creating a New Repository

1. Click on the **+** icon at the top right and select **New 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 * ParvezMia / Repository name *

Great repository names are short and memorable. Need inspiration? How about curly-bassoon ?

Description (optional)

[✓ 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:

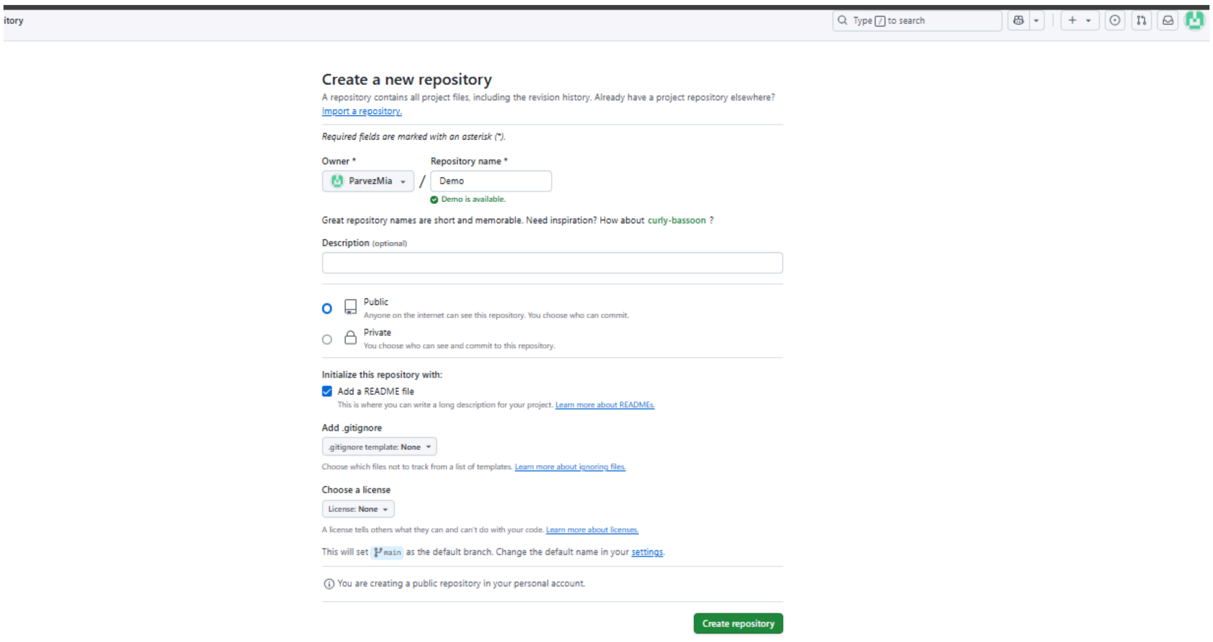
☐ Add a README file
This is where you can write a long description for your project. Learn more about READMEs.

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.

You are creating a public repository in your personal account.

1. Enter a repository name.
2. Choose **Public** (visible to everyone) or **Private** (only you and collaborators can see it).
3. (Optional) Check **Add a README file**.
4. Click **Create 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 * ParvezMia / Repository name * Demo

Demo is available.

Great repository names are short and memorable. Need inspiration? How about curly-bassoon ?

Description (optional)

[✓ 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:

[✓] 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.

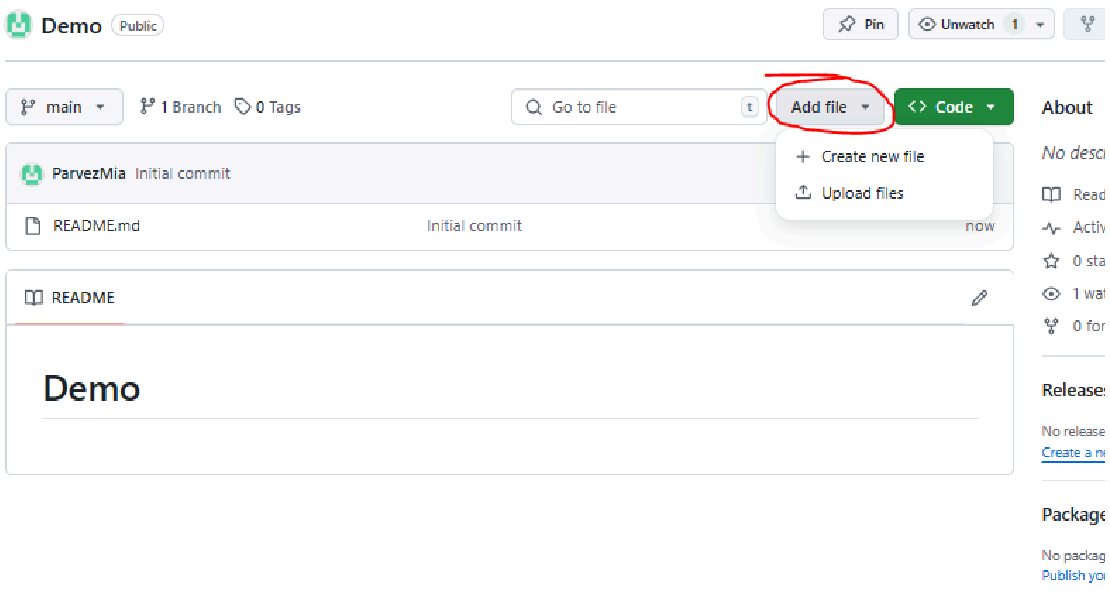
This will set `main` as the default branch. Change the default name in your settings.

You are creating a public repository in your personal account.

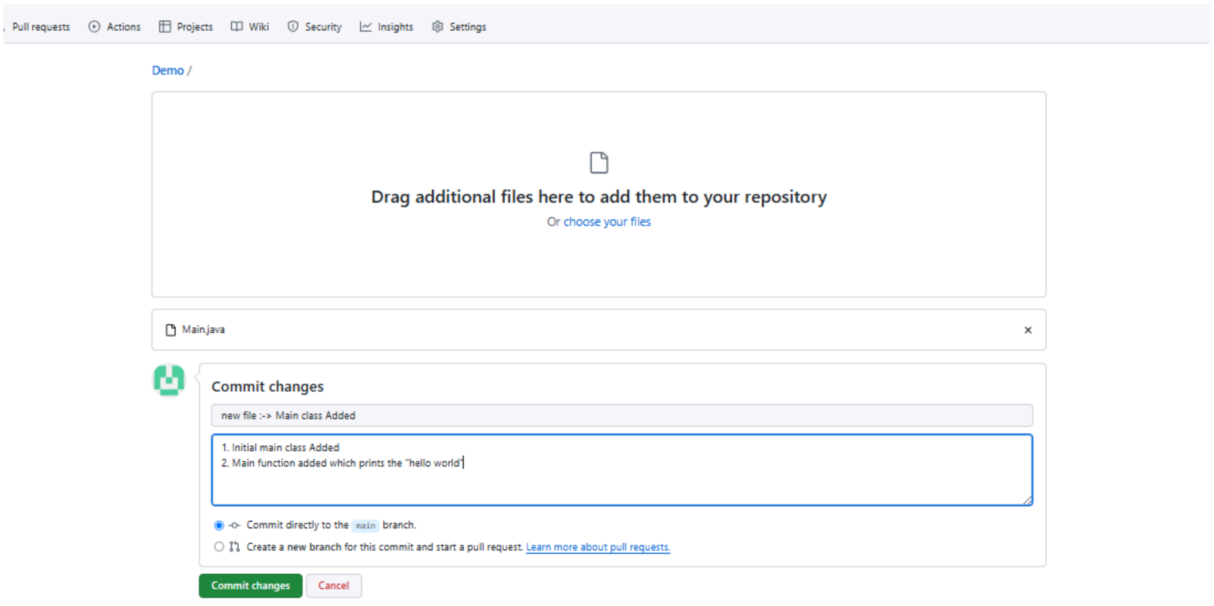
3. Adding and Managing Files

3.1 Uploading a File

1. Navigate to your repository.
2. Click **Add File** → **Upload Files**.



3. Drag and drop files or select them manually.
4. Click **Commit changes** to save them.

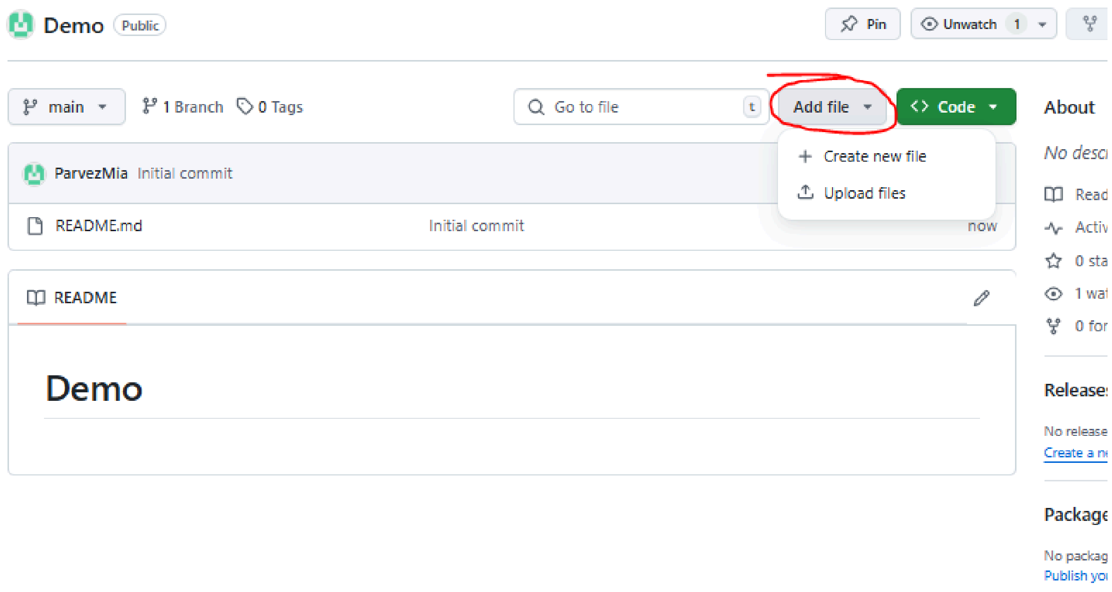

Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
Demo /
Drag additional files here to add them to your repository
Or choose your files
Main.java
Commit changes
Commit directly to the main branch.
Create a new branch for this commit and start a pull request. Learn more about pull requests.
Commit changes
Cancel

new file :-> Main class Added

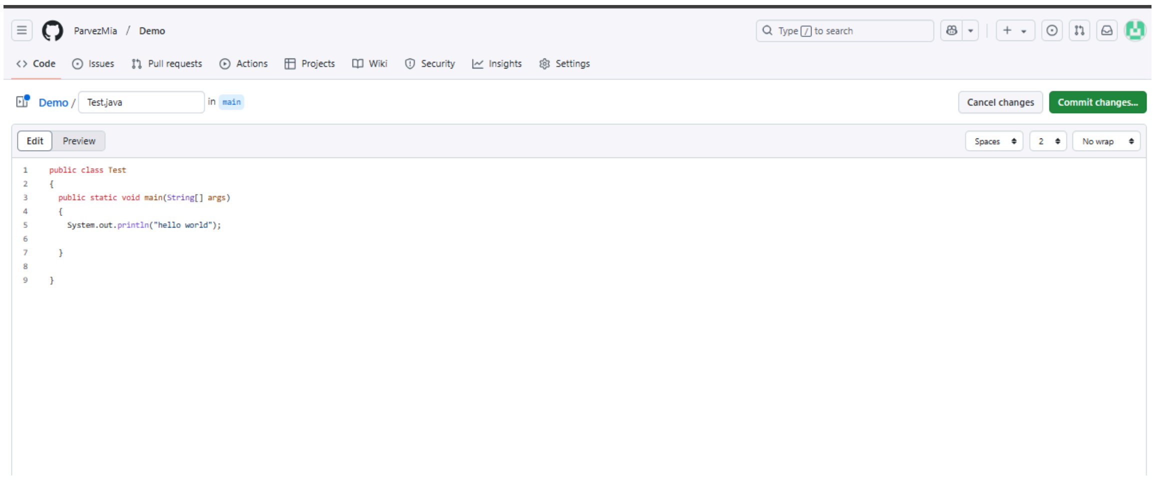
1. Initial main class Added
2. Main function added which prints the "hello world"

3.2 Creating a New File

1. Go to your repository and click **Add File** → **Create New File**.



2. Enter a filename and add content.

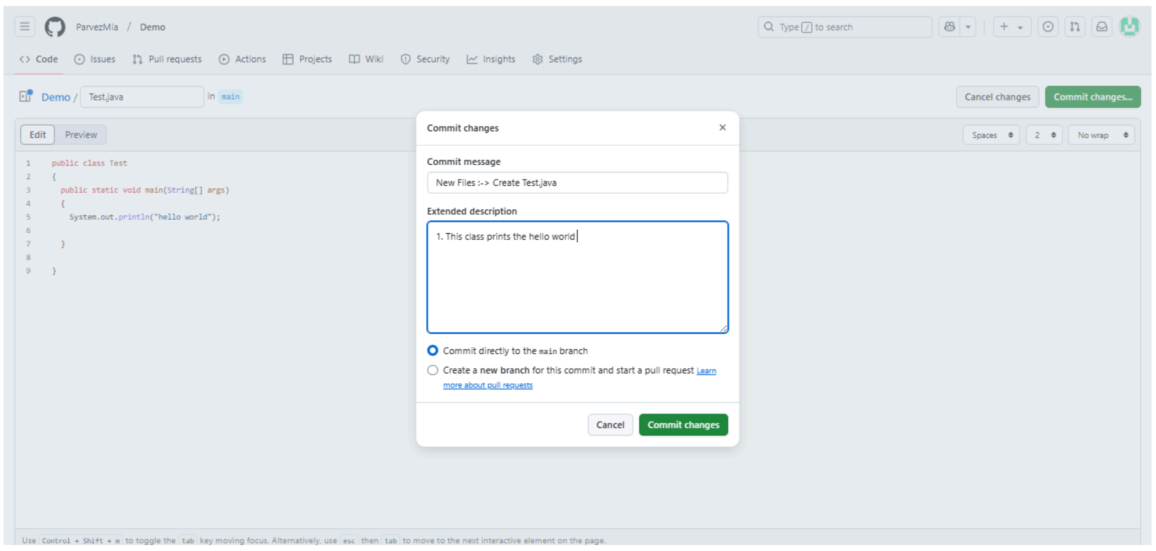


```
public class Test
{
  public static void main(String[] args)
  {
    System.out.println("hello world");

  }

}
```

3. Click **Commit changes**.



Commit changes

Commit message: New Files :-> Create Test.java

Extended description: 1. This class prints the hello world

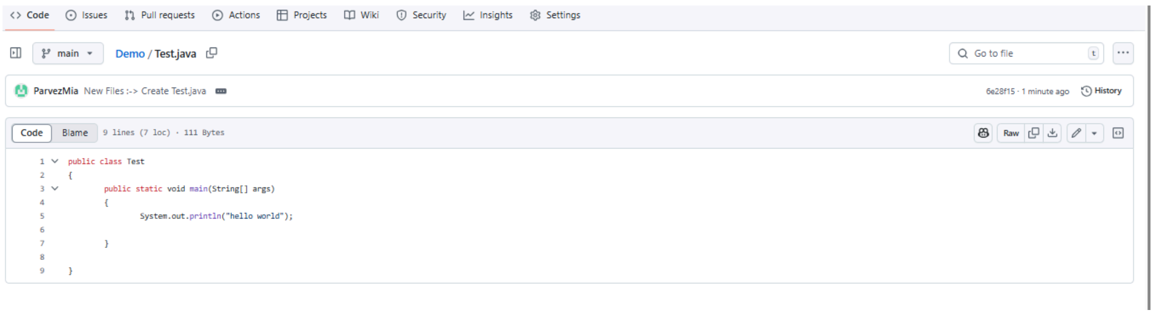
[✓ Commit directly to the main branch]

Create a new branch for this commit and start a pull request Learn more about pull requests

Use Control + Shift + m to toggle the tab key moving focus. Alternatively, use esc then tab to move to the next interactive element on the page.

3.3 Editing an Existing File

1. Open the file inside the repository.
2. Click the **Edit** (pencil) icon.



ParvezMia New Files :-> Create Test.java

6e28f15 · 1 minute ago

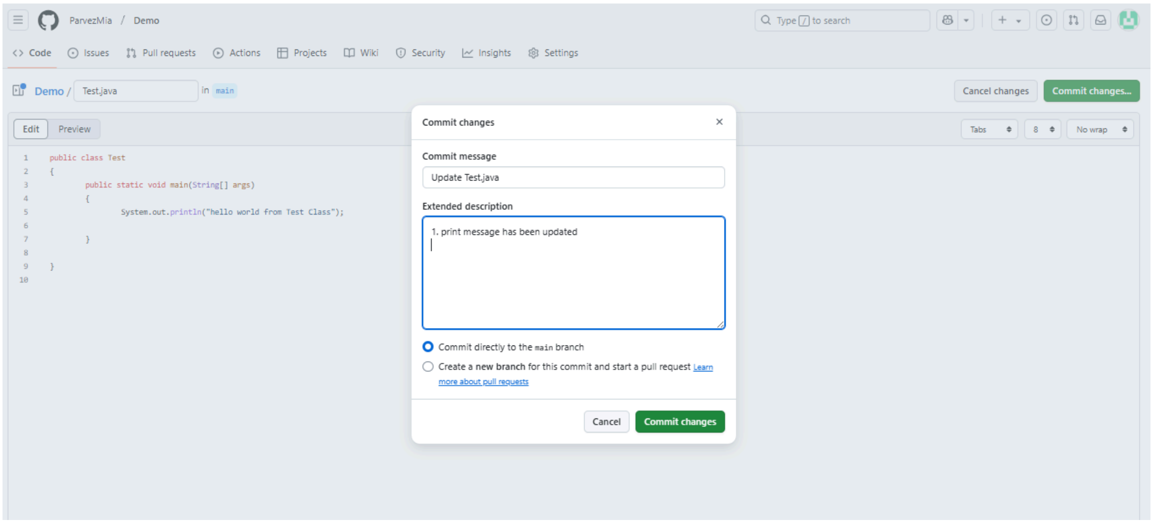
9 lines (7 loc) · 111 Bytes

```
public class Test
{
        public static void main(String[] args)
        {
                System.out.println("hello world");

        }

}
```

3. Make changes and click **Commit changes**.



```
public class Test
{
        public static void main(String[] args)
        {
                System.out.println("hello world from Test Class");

        }

}
```

Commit message: Update Test.java

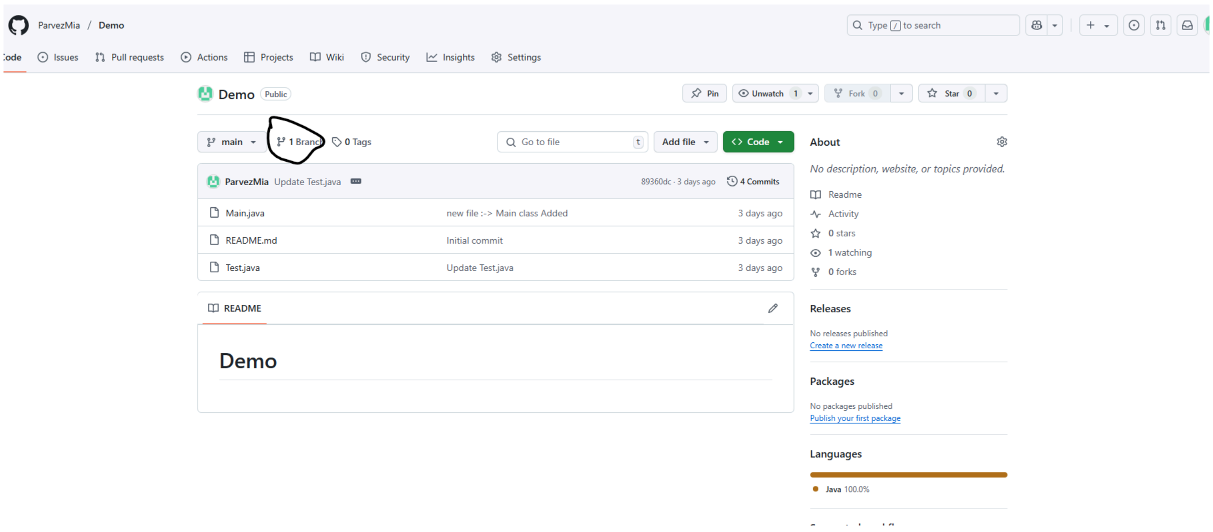
Extended description: 1. print message has been updated

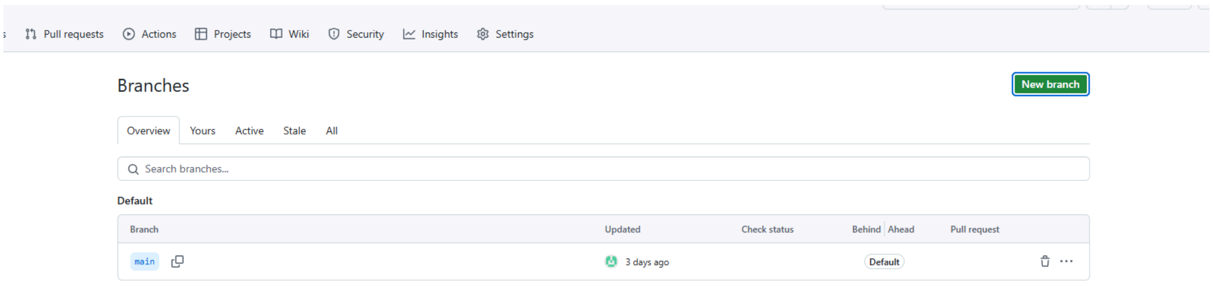
[✓ Commit directly to the main branch]
Create a new branch for this commit and start a pull request Learn more about pull requests

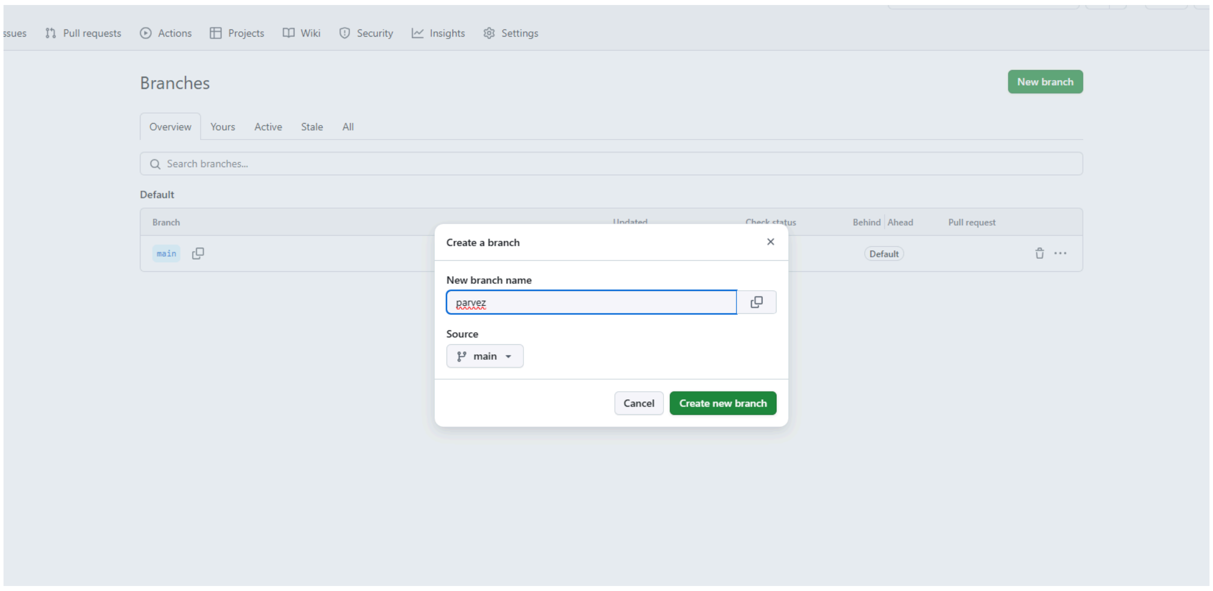
4. Branching in GitHub

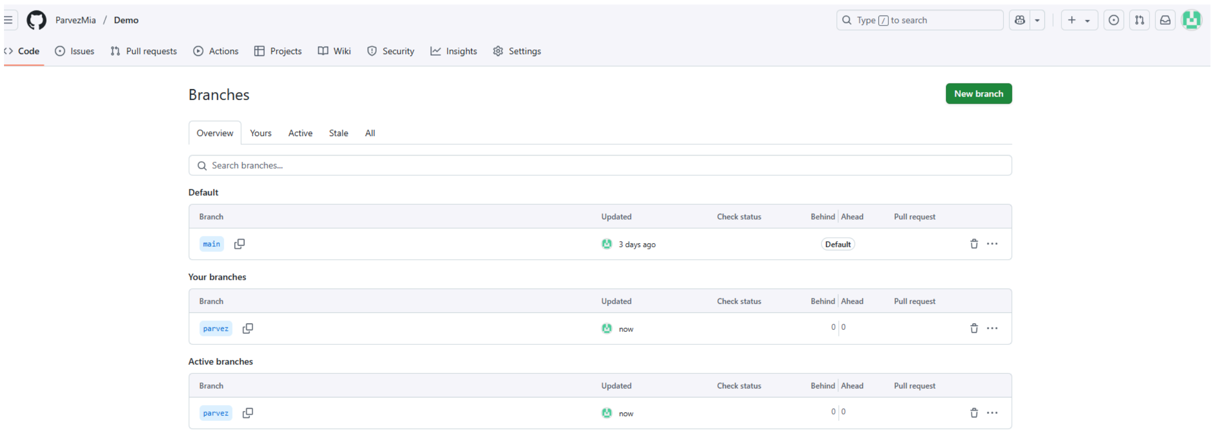
4.1 Creating a New Branch

1. Open your repository.
2. Click the **New Branch**
3. Type a new branch name and press **Enter**.


ParvezMia / Demo
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
Demo
Public
main
1 Branch
0 Tags
Go to file
Add file
Code
ParvezMia Update Test.java
4 Commits
Main.java
new file :-> Main class Added
3 days ago
README.md
Initial commit
Test.java
Update Test.java
README
Demo
About
No description, website, or topics provided.
Readme
Activity
0 stars
1 watching
0 forks
Releases
No releases published
Create a new release
Packages
No packages published
Publish your first package
Languages
Java 100.0%


Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
Branches
New branch
Overview
Yours
Active
Stale
All
Search branches...
Default
Branch
Updated
Check status
Behind
Ahead
Pull request
main
3 days ago
Default


Branches
New branch
Create a branch
New branch name
parvez
Source
main
Cancel
Create new branch


Branches

Default

Branch	Updated	Check status	Behind	Ahead	Pull request
main	3 days ago		Default		

Your branches

Branch	Updated	Check status	Behind	Ahead	Pull request
parvez	now		0	0	

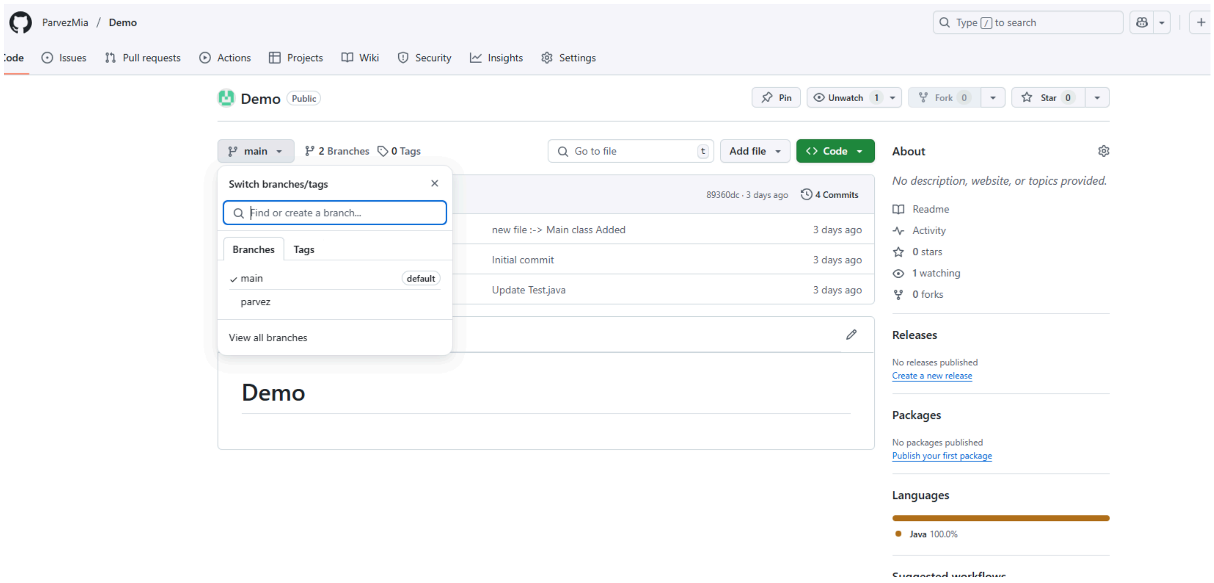
Active branches

Branch	Updated	Check status	Behind	Ahead	Pull request
parvez	now		0	0	

Note that all the team member should have a separate branch in the repo, where they will push their changes/contributions to the project

4.2 Switching Branches

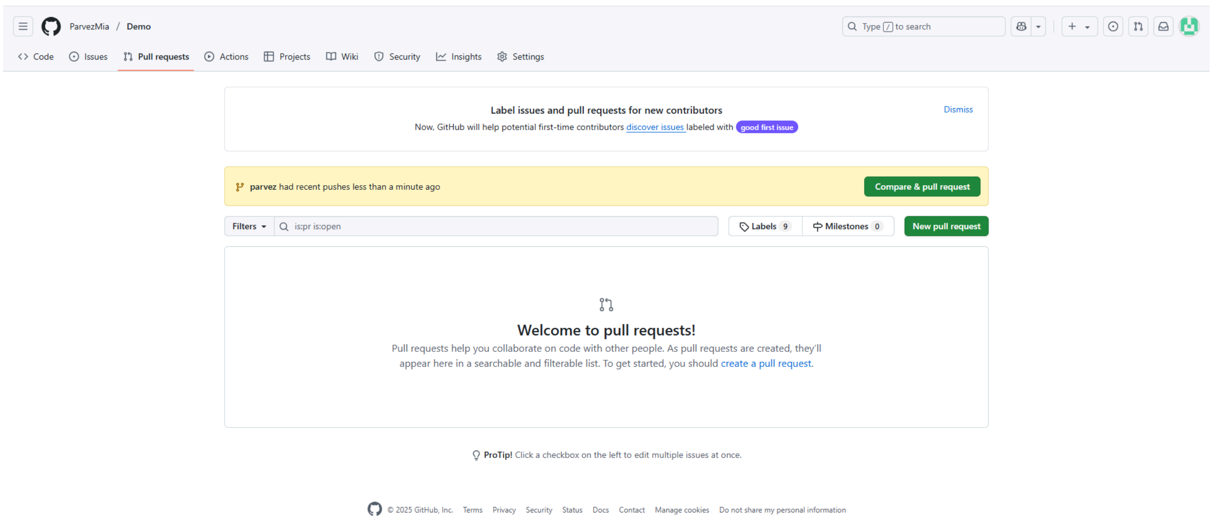
1. Click the **Branch dropdown**.
2. Select the branch you want to switch to.



89360dc · 3 days ago	4 Commits
new file :-> Main class Added	3 days ago
Initial commit	3 days ago
Update Test.java	3 days ago

4.3 Merging a Branch (Using Pull Requests)

1. Switch to the branch you want to merge.
2. Click **Pull Requests** → **New Pull Request**.



Label issues and pull requests for new contributors

Now, GitHub will help potential first-time contributors discover issues labeled with good first issue

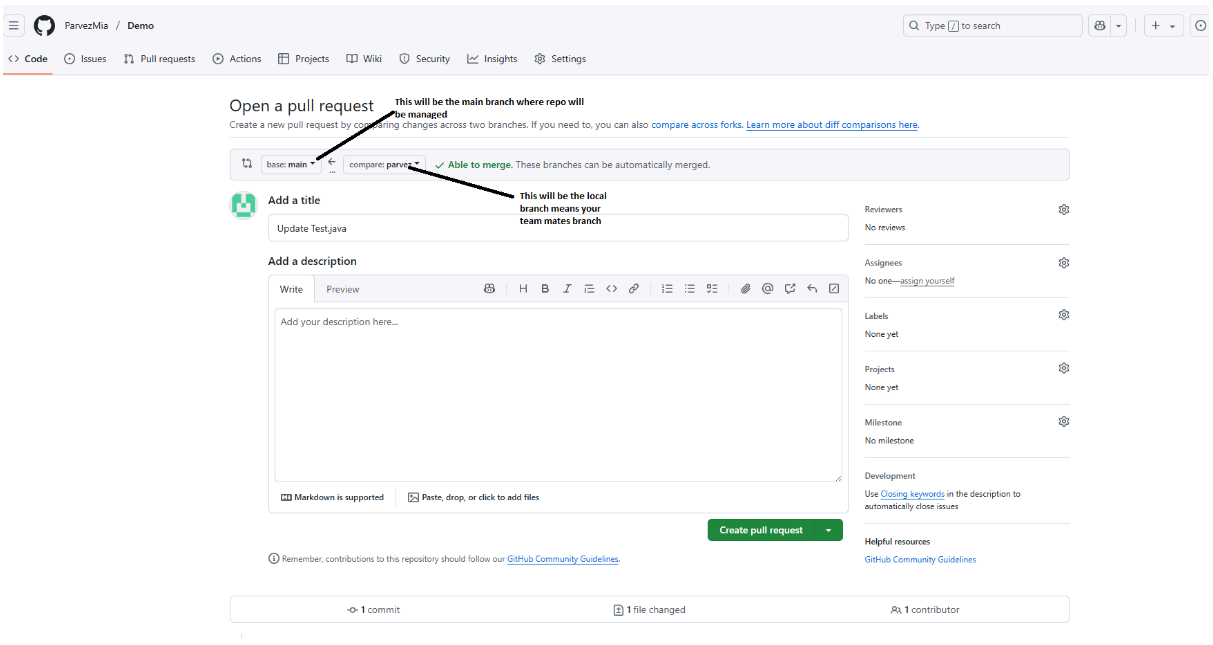
parvez had recent pushes less than a minute ago

Welcome to pull requests!

Pull requests help you collaborate on code with other people. As pull requests are created, they'll appear here in a searchable and filterable list. To get started, you should create a pull request.

ProTip! Click a checkbox on the left to edit multiple issues at once.

1. Select branches to compare.



Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also compare across forks. Learn more about diff comparisons here.

This will be the main branch where repo will be managed

Able to merge. These branches can be automatically merged.

This will be the local branch means your team mates branch

Add a title

Update Test.java

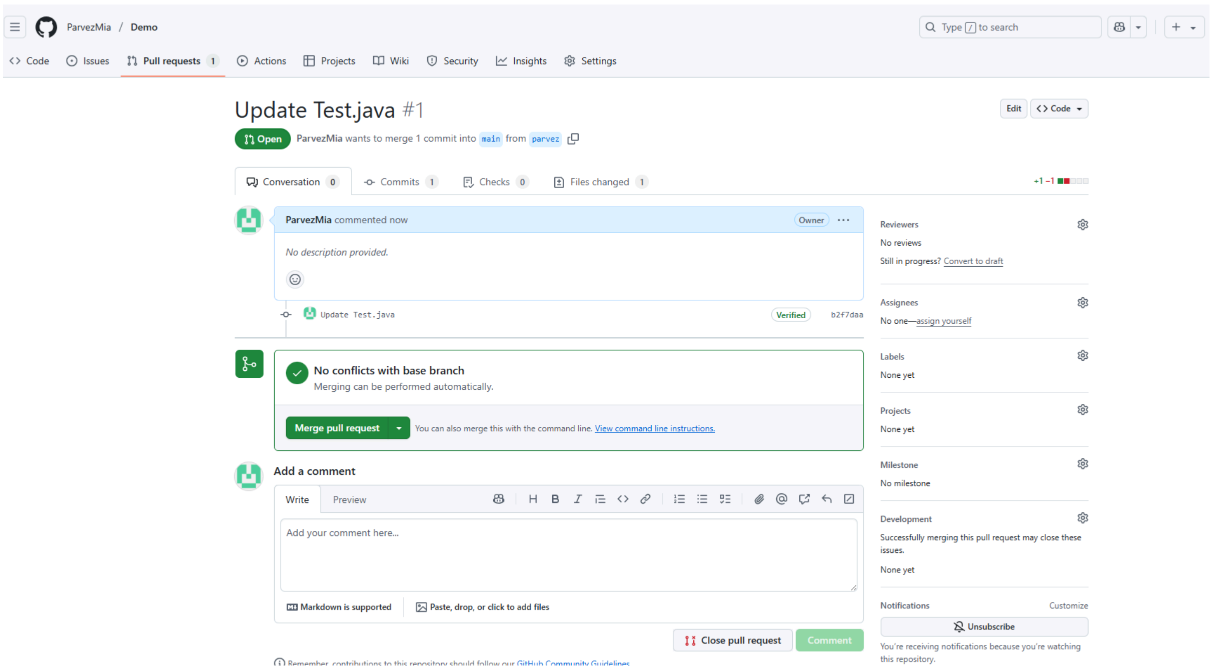
Add a description

Add your description here...

Use Closing keywords in the description to automatically close issues

Remember, contributions to this repository should follow our GitHub Community Guidelines.

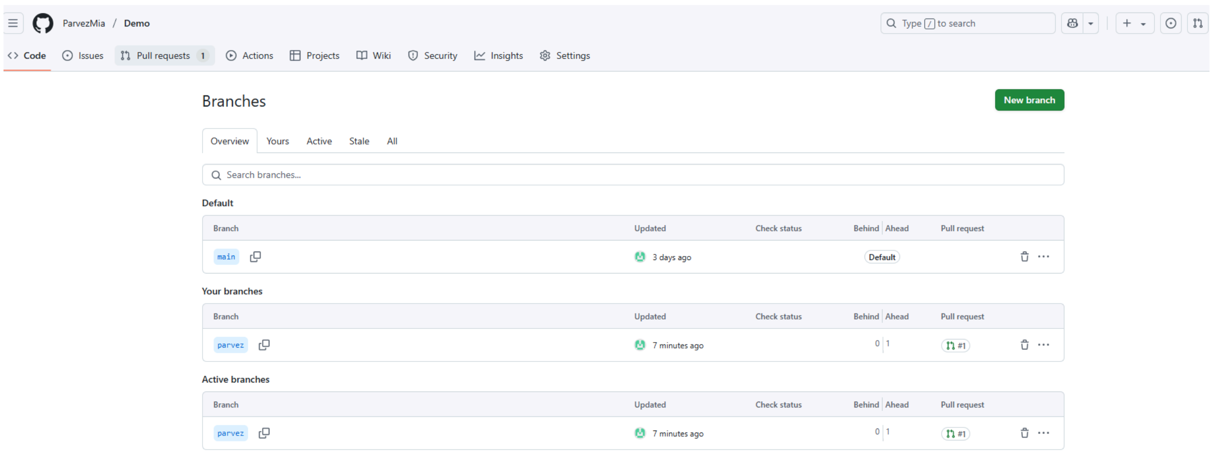
1. Click **Create Pull Request → Merge Pull Request**.



Congratulations first PR Request has been created now the team leader will review this PR and if there is no confect than he will marge this PR

4.4 Deleting a Branch

1. Go to **Branches**.
2. Click the **Delete** button next to the branch.

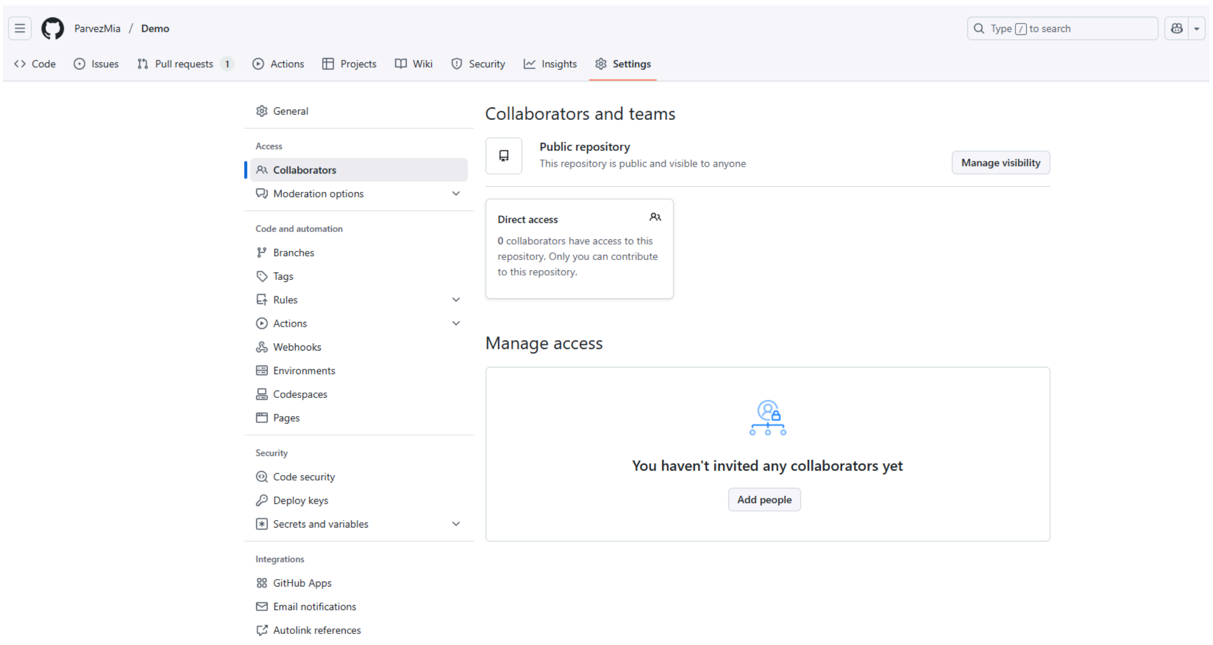


Branch	Updated	Check status	Behind | Ahead	Pull request
main	3 days ago		Default	
parvez	7 minutes ago		0 | 1	#1
parvez	7 minutes ago		0 | 1	#1

5. Working with GitHub Features

5.1 Adding Collaborators

1. Open your repository and click **Settings**.
2. Go to the **Manage Access** section.
3. Click **Invite a collaborator**.
4. Enter the GitHub username or email of the person you want to invite.
5. Click **Add Collaborator**.
6. The collaborator will receive an email invitation. Once accepted, they can contribute to your repository.



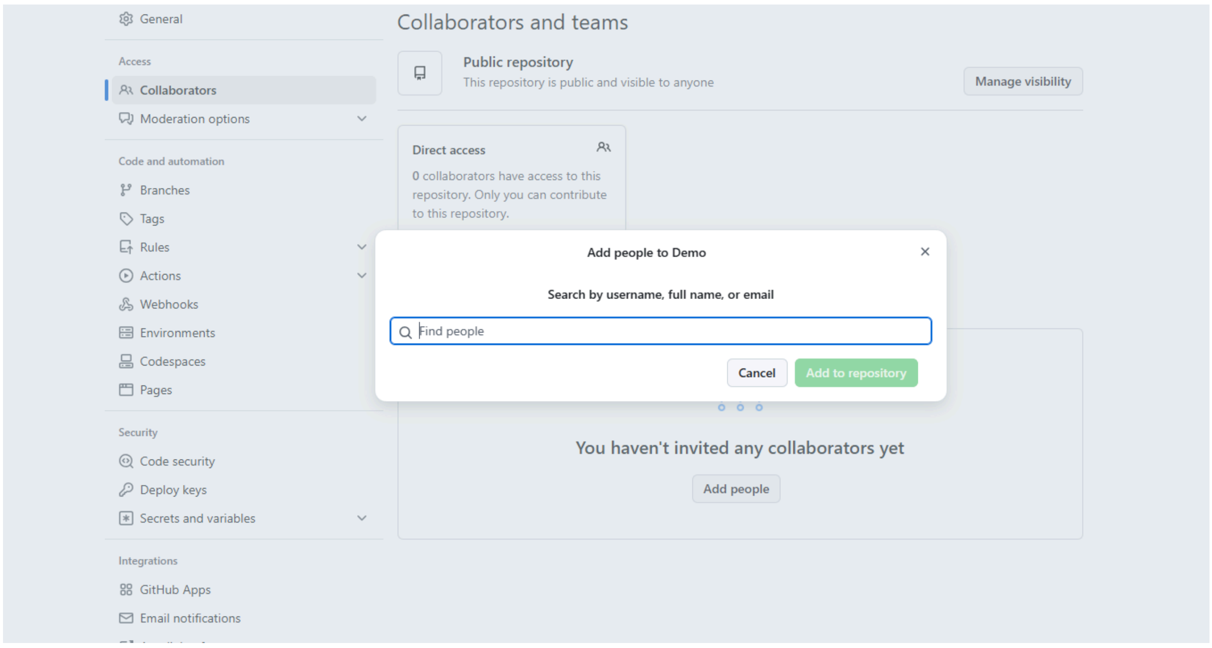
Collaborators and teams

Public repository
This repository is public and visible to anyone

Direct access
0 collaborators have access to this repository. Only you can contribute to this repository.

Manage access

You haven't invited any collaborators yet



Collaborators and teams

Public repository

This repository is public and visible to anyone

0 collaborators have access to this repository. Only you can contribute to this repository.

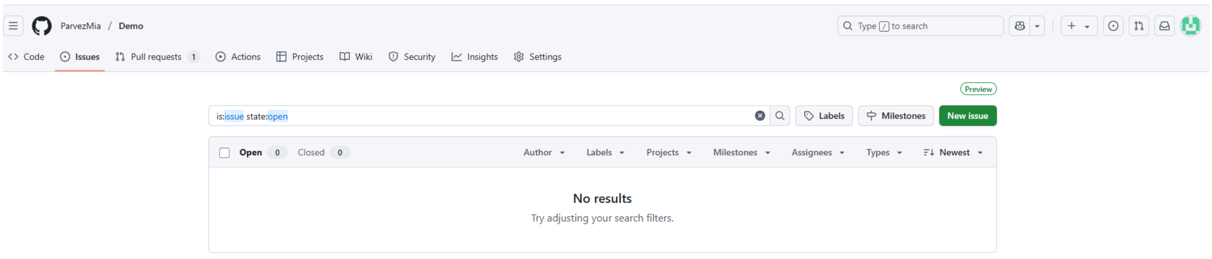
Add people to Demo

Search by username, full name, or email

You haven't invited any collaborators yet

5.2 Creating an Issue

1. Open your repository and go to the **Issues** tab.

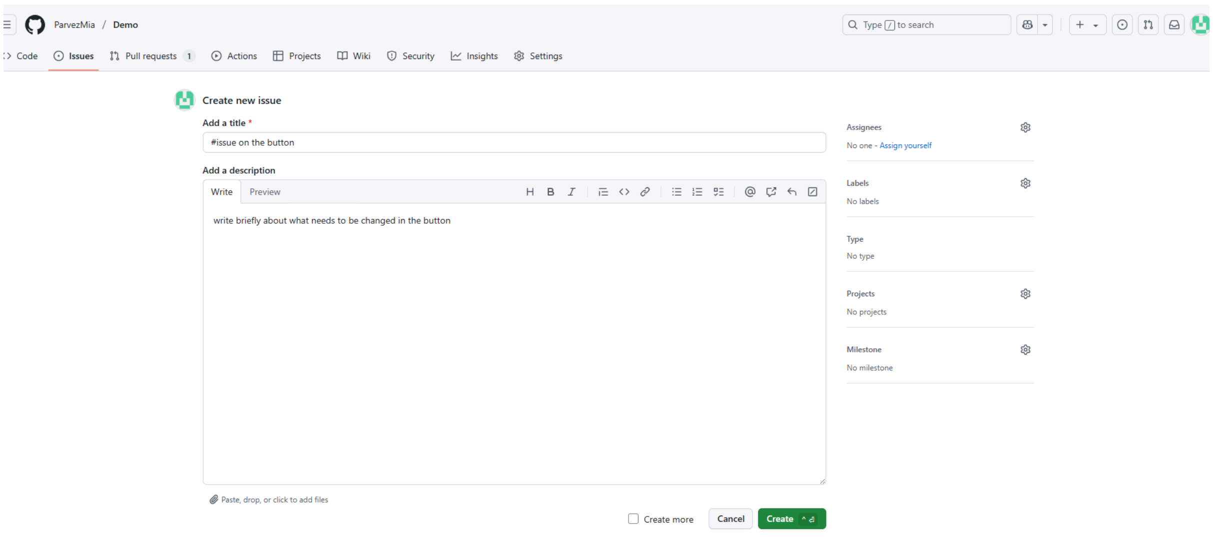


is:issue state:open

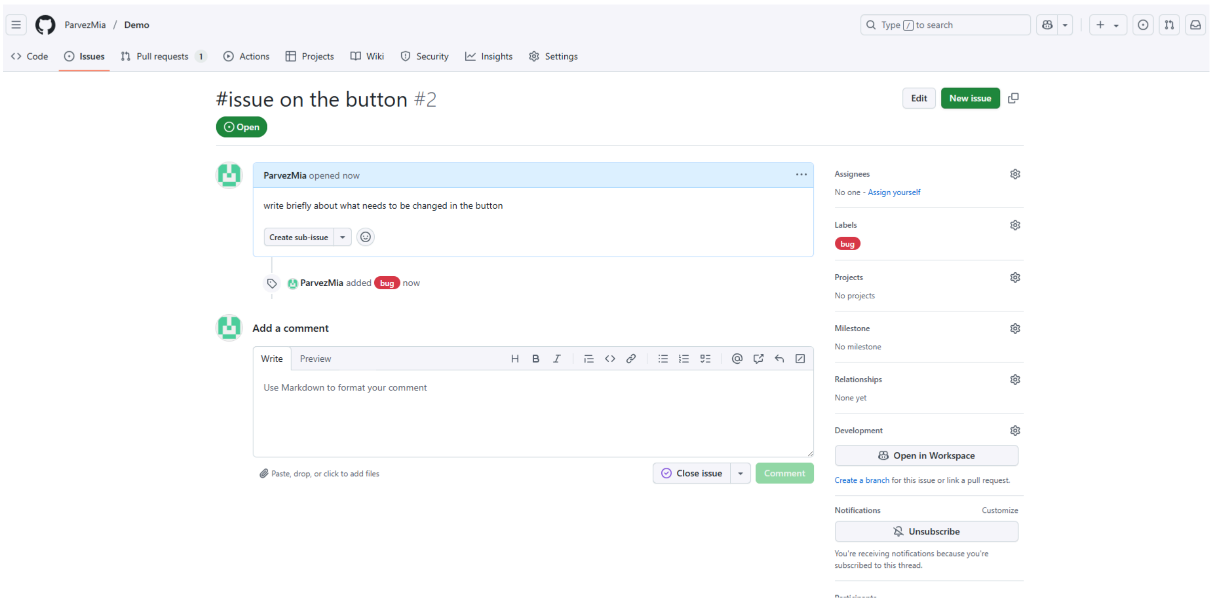
No results

Try adjusting your search filters.

1. Click **New Issue**.
2. Enter a title and detailed description of the issue.



1. (Optional) Assign the issue to a collaborator or add labels.
2. Click **Submit New Issue**.



Using CMD Interface

Installing Git

https://www.git-scm.com/downloads

You can easily install Git from the official website based on your operating system.

Once Git is successfully installed, you can verify it by running the command `git --version` in the terminal. If the version appears, congratulations! Git has been successfully installed. 😁

Using Git in a Project

Before initializing Git in a project, we need to set some configurations so that we don't have to do it repeatedly in the future. For that, we will run two commands in the terminal one by one. First:

```
git config --global user.name "Al Nahian"
```

after that,

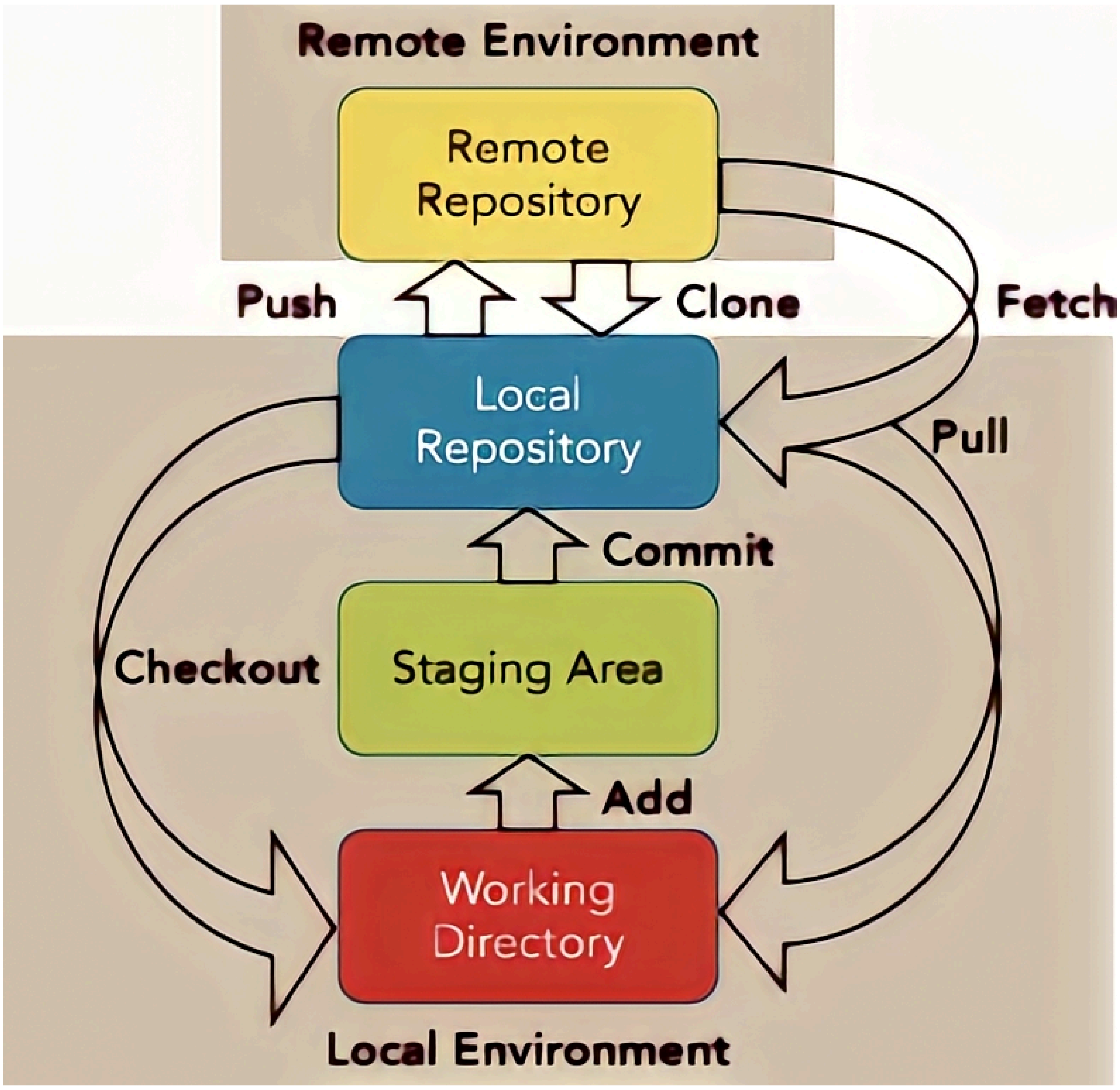
```
git config --global user.email "nahian@mail.com"
```

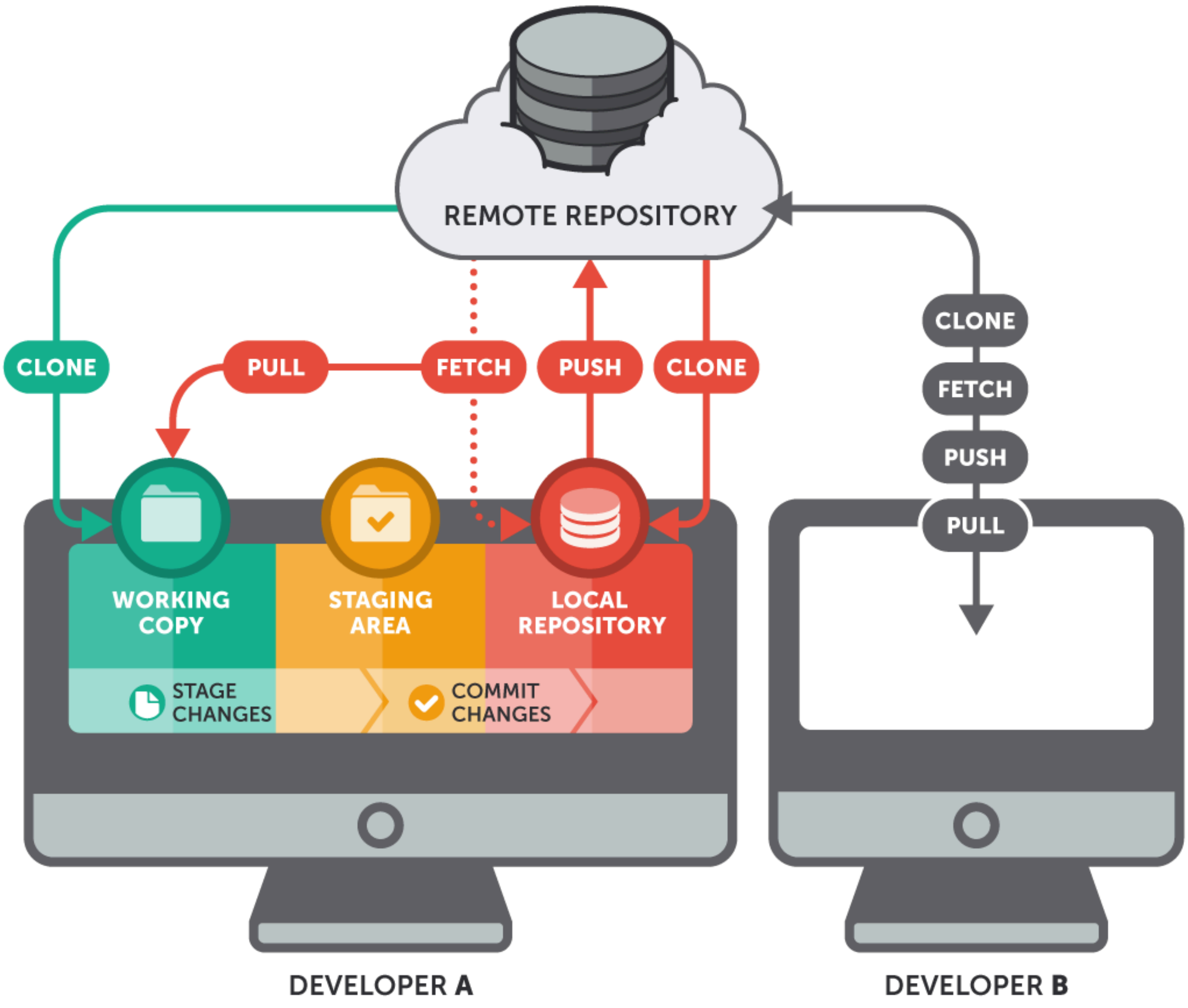
Additionally, to ensure that every repository is created with the `main` branch by default, we will set another configuration:

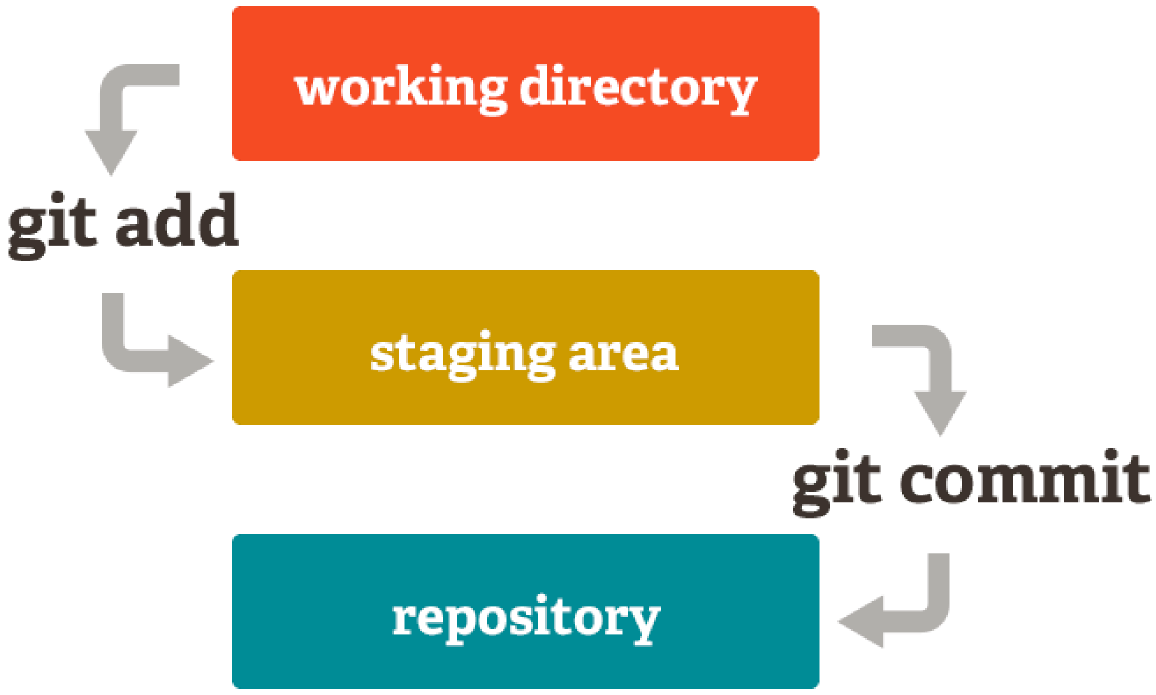
```
git config --global init.defaultBranch main
```

Next, you need to create an empty folder or use an existing project. Then, navigate inside that folder and run the `git init` command using **Git Bash Here** or through the terminal. This will initialize Git in that project.

Git Workflow


Remote Environment
Remote Repository
Push
Clone
Fetch
Local Repository
Pull
Commit
Checkout
Staging Area
Add
Working Directory
Local Environment


REMOTE REPOSITORY
CLONE
PULL
FETCH
PUSH
CLONE
CLONE
FETCH
PUSH
PULL
WORKING COPY
STAGING AREA
LOCAL REPOSITORY
STAGE CHANGES
COMMIT CHANGES
DEVELOPER A
DEVELOPER B


working directory
git add
staging area
git commit
repository

The Git workflow is quite simple. After initializing Git in a project, we usually don't commit changes directly. Instead, we **stage the changes first**, then commit them. Finally, we **push the commits to the remote repository**. This is the basic Git workflow.

Git Status Command

Now, let's get familiar with the `git status` command. In any project or repository where Git is initialized, running this command from the terminal allows us to check the current status of the project.

Stage Changes Using `git add` Command

As mentioned earlier, in the Git workflow, when we make changes in the project and are satisfied with those changes, we **stage them first** before committing. We **cannot directly commit changes without staging them first**.

To stage all changes in the project, we use the command:

```
git add .
```

This stages all the changes in the project.

Alternatively, if you want to stage a specific file, you can specify the file name after `add`, like this:

```
git add filename.extension
```

Make Your First Git Commit

After staging the files, you can check their status using the `git status` command. If everything seems fine and you are satisfied with the changes by using the `git diff` command, you can **finally commit your changes** with the following command:

```
git commit -m "Your commit message"
```

Congratulations! 🎉 You've just made your **first commit**! ✅

Viewing The Commit Snapshots/History

To check what changes have been made and what commits have been done in a repository, we can simply use the `git log` command.

If you want to see the **commit history in a minimal and concise way**, you can use:

```
git log --oneline
```

This is **very useful** for quickly tracking commit history! ✅

Reverting Your Mistakes

Since **humans make mistakes**, it's common to make errors while working with Git. But no worries! With `git revert`, `git reset`, and `git checkout` commands, we can easily fix those mistakes.

✅ `git checkout` is quite special because:

- You can **switch between branches**.
- You can **undo a specific commit**.
- Or even **restore a previous version** of your code.

If you want to **revert to a specific commit**, use:

```
git checkout <commit-hash>
```

The **commit hash (SHA-1)** can be found using:

```
git log --onelin
```

✅ Alternatively, with `git reset`:

```
git reset <commit-hash>
```

This will **remove the commits after the specified commit**.

✅ Or, the most recommended way is to use `git revert`:

```
git revert <commit-hash>
```

This will **undo the changes** but will **create a new commit**, which is safer and highly recommended in professional projects.

> 🎯 Pro Tip:
>
> `git restore` can also be used to restore specific files or changes if needed.

In short, **no need to panic if you make a mistake in Git**! 😊

Let's Create, Modify, Merge & Delete Branches

Creating a branch in Git is **very simple**.

✅ To create a new branch:

```
git branch branch_name
```

✅ To switch to that branch:

```
git checkout branch_name
```

✅ Or, create a new branch and switch to it at the same time:

```
git checkout -b branch_name
```

🎯 After making changes, staging, and committing, you can merge that branch into another branch.

- First, switch to the branch where you want to merge:

```
git checkout other_branch_name
```

- Then, merge the new branch:

```
git merge newly_created_branch_name
```

✅ To delete the branch locally:

```
git branch -d localBranchName
```

✅ To delete the branch from the remote repository:

```
git push origin --delete remoteBranchName
```

> In short:
>
> Create ➔ Switch ➔ Commit ➔ Merge ➔ Delete
>
> That's how **branching works in Git!** 🚀

Pushing Changes to Github

To **push changes to a remote Git repository**, you first need to **set the remote origin**.

✅ Step 1: Add Remote Origin

```
git remote add origin https://github.com/OWNER/REPOSITORY.git
```

- This sets the **remote origin URL** for the project.
- To verify if the remote origin is set correctly, use:

```
git remote -v
```

✅ Step 2: Push Changes to Remote Repository

```
git push origin main
```

or, if you're pushing to a specific branch:

```
git push origin branch_name
```

🔑 If Credentials Are Not Set Up:

- Set up **SSH Key** or **GitHub Credentials** from GitHub settings.
- Sometimes, Git Bash or VS Code will prompt you for a **username and password**. Just provide the correct credentials.

🔄 To Pull Changes from the Remote Repository:

```
git pull origin main
```

This will **fetch the latest changes and update your local repository**.

🎯 To Fetch Changes Without Merging:

```
git fetch origin
```

This will **only download the changes**, but won't merge them with your local branch.

> In short:
>
> `git pull` → **For getting updates**
>
> `git push` → **For sending updates**
>
> `git fetch` → **For checking updates without merging ✅**

GitHub Best Practices

Top 3 Github Alternatives

- https://gitlab.com/
- https://bitbucket.org/
- https://about.gitea.com/

Resources

- https://www.git-scm.com/doc
- https://docs.github.com/en
- https://www.w3schools.com/git/
- https://github.com/git-tips/tips
- https://with.zonayed.me/git-n-github/
- https://with.zonayed.me/book/git-n-github-at-glance/
- https://youtu.be/M-qSAdWFs9c
- https://youtu.be/oe21Nlq8GS4
- https://skyline.github.com/
- https://honzaap.github.io/GithubCity/