Git and GitHub Workshop

Objectives of the workshop:

- Understand the fundamentals of Version Control Systems (VCS).
- Understand Git and GitHub environments.
- Learn to push projects to GitHub and collaborate with fellow developers effectively.

What are Version Control Systems?

A version control system tracks any changes made to the project file, why these changes were made, and references to the problems fixed or enhancements introduced. It allows developer teams to manage and track changes in code over time. It will enable a person to switch to the previous states of the file, compare the versions and help identify issues in a file efficiently.

What Is Git?

- Git is one of the ways of implementing the idea of version control. It is a Distributed Version Control System (DVCS).
- Unlike a Centralized Version Control System that uses a central server to store all files and enables team collaboration, DVCS can be implemented with the help of just a desktop and a single software available via command line. So, the failure of the central server does not create any problems in DVCS. Additionally, most operations can be performed even when you are offline.

How to install Git?

Installing on Linux

If you want to install the basic Git tools on Linux via a binary installer, you can generally do so through the package management tool that comes with your distribution. If you're on Fedora (or any closely-related RPM-based distribution, such as RHEL or CentOS), you can use dnf:

```
$ sudo dnf install git-all
```

If you're on a Debian-based distribution, such as Ubuntu, try apt:

```
$ sudo apt install git-all
```

For more options, there are instructions for installing on several different Unix distributions on the Git website, at https://git-scm.com/download/linux.

Installing on macOS

There are several ways to install Git on macOS. The easiest is probably to install the Xcode Command Line Tools. On Mavericks (10.9) or above you can do this simply by trying to run git from the Terminal the very first time.

```
$ git --version
```

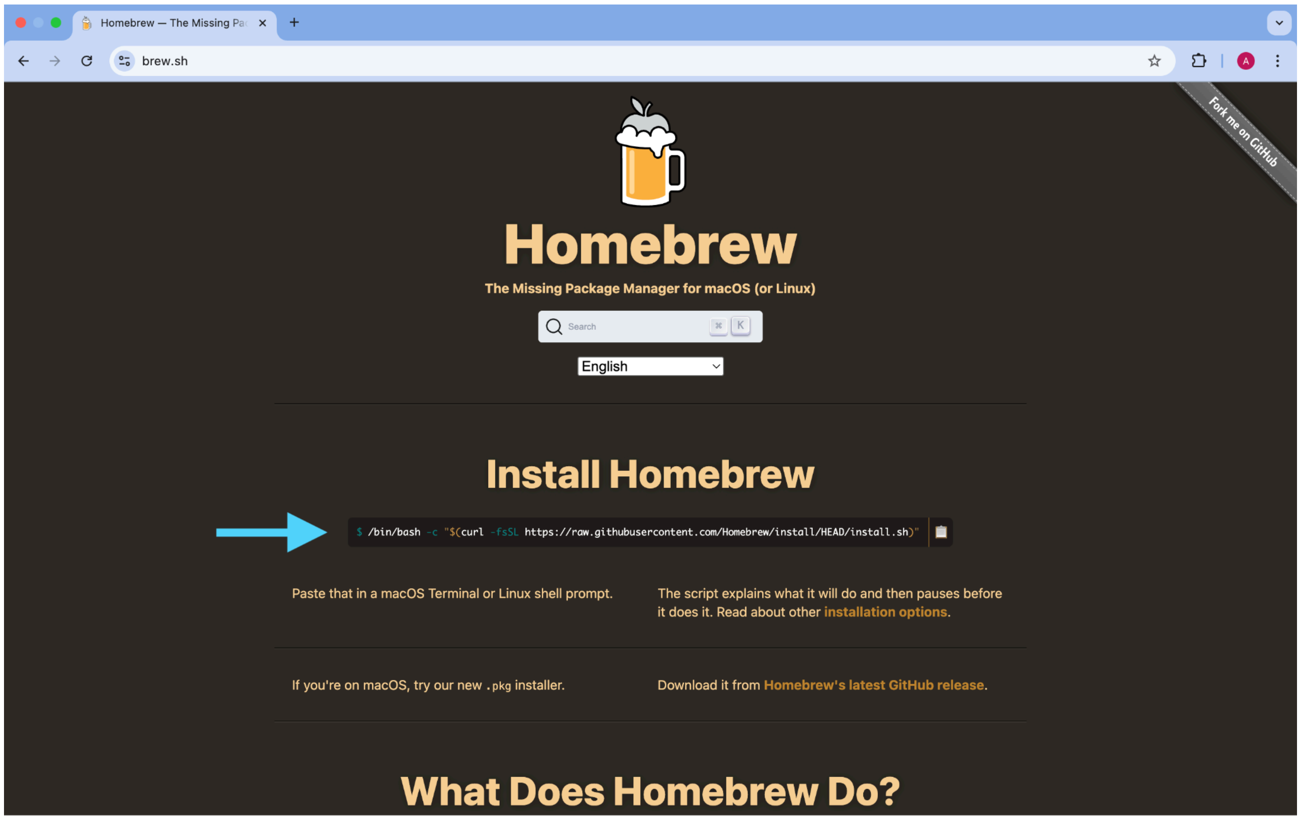
If you don’t have it installed already, it will prompt you to install it.

If you want a more up to date version, you can also install it via a binary installer. A macOS Git installer is maintained and available for download at the Git website, at https://git-scm.com/download/mac.

In the Git website for downloading Git for macOS, it will ask you to install Homebrew (if it is not already present).

```
$ brew --version
```

If you don’t have it installed already, it will prompt you to install it.
You can install it from the following website:
https://brew.sh/



Homebrew

The Missing Package Manager for macOS (or Linux)

Install Homebrew

```
$ /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

Paste that in a macOS Terminal or Linux shell prompt.

The script explains what it will do and then pauses before it does it. Read about other installation options.

If you're on macOS, try our new .pkg installer.

Download it from Homebrew's latest GitHub release.

What Does Homebrew Do?

Copy and Paste the command from the website where the arrow is pointing in the above Screenshot and run it in your terminal.

After successfully installing Homebrew, run the following command to install Git

```
$ brew install git
```

Installing on Windows

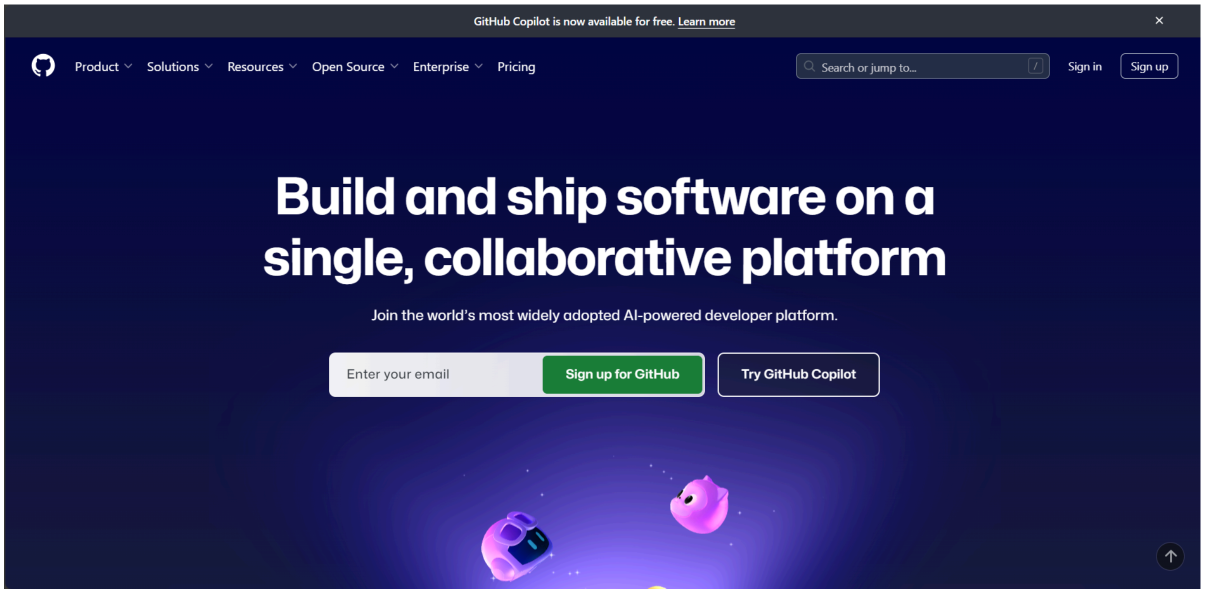
There are also a few ways to install Git on Windows. The most official build is available for download on the Git website. Just go to https://git-scm.com/download/win and the download will start automatically. Note that this is a project called Git for Windows, which is separate from Git itself; for more information on it, go to https://gitforwindows.org.

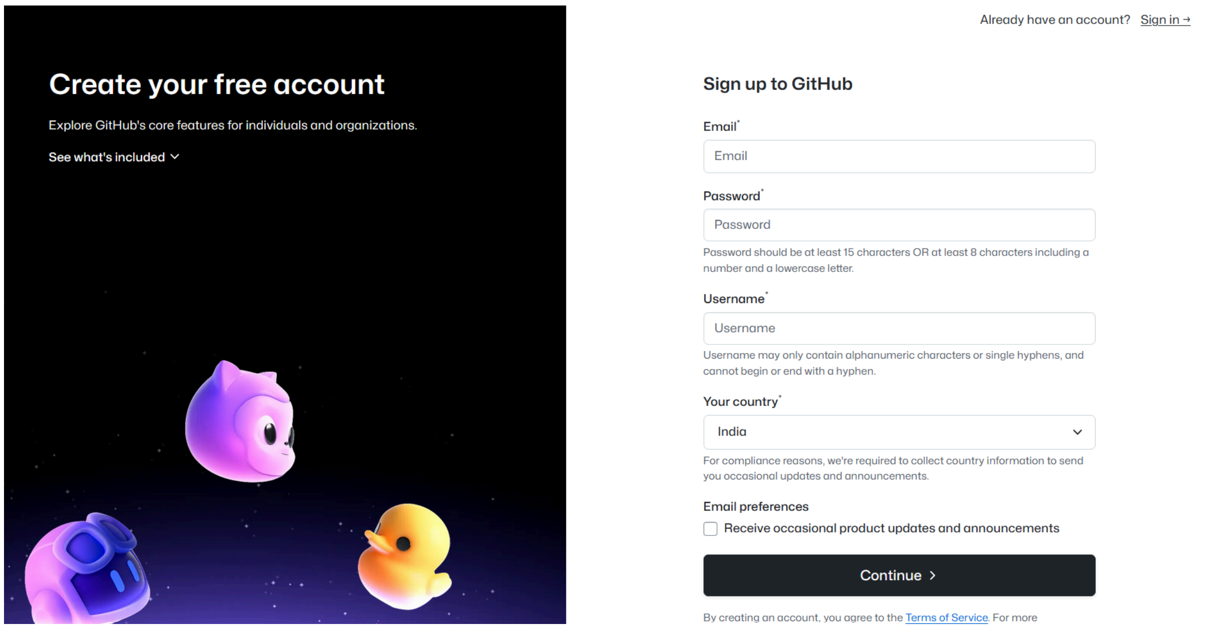
To get an automated installation, you can use the Git Chocolatey package. Note that the Chocolatey package is community-maintained.

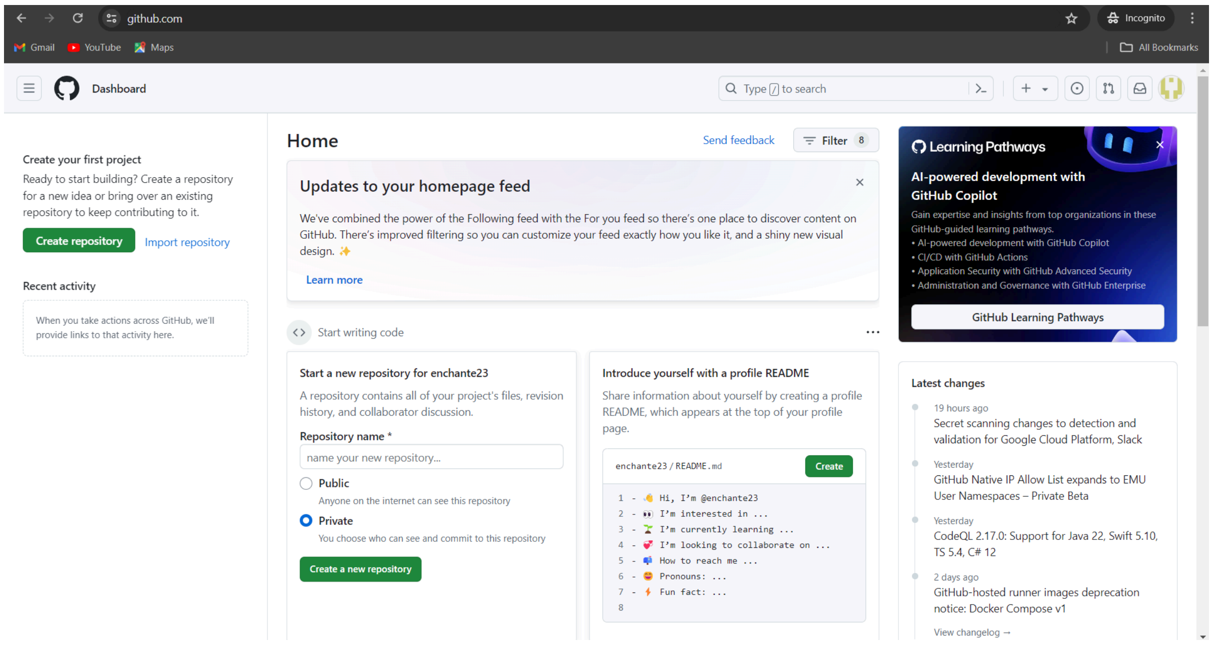
Creating an account on GitHub

To get started with GitHub, you'll need to create a free personal account on github.com and verify your email address.

Every person who uses GitHub.com signs in to a personal account. Your personal account is your identity on GitHub and has a username and profile.


GitHub Copilot is now available for free. Learn more
Product
Solutions
Resources
Open Source
Enterprise
Pricing
Search or jump to...
Sign in
Sign up
Build and ship software on a single, collaborative platform
Join the world's most widely adopted AI-powered developer platform.
Enter your email
Sign up for GitHub
Try GitHub Copilot


Already have an account? Sign in →
Create your free account
Explore GitHub's core features for individuals and organizations.
See what's included
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.
Your country*
India
For compliance reasons, we're required to collect country information to send you occasional updates and announcements.
Email preferences
Receive occasional product updates and announcements
Continue >
By creating an account, you agree to the Terms of Service. For more


github.com

Gmail YouTube Maps All Bookmarks

Dashboard

Type / to search

Create your first project

Ready to start building? Create a repository for a new idea or bring over an existing repository to keep contributing to it.

Create repository Import repository

Recent activity

When you take actions across GitHub, we'll provide links to that activity here.

Home

Send feedback Filter 8

Updates to your homepage feed

We've combined the power of the Following feed with the For you feed so there's one place to discover content on GitHub. There's improved filtering so you can customize your feed exactly how you like it, and a shiny new visual design. ✨

Learn more

Start writing code

Start a new repository for enchante23

A repository contains all of your project's files, revision history, and collaborator discussion.

Repository name *

name your new repository...

Public
Anyone on the internet can see this repository

Private
You choose who can see and commit to this repository

Create a new repository

Introduce yourself with a profile README

Share information about yourself by creating a profile README, which appears at the top of your profile page.

enchante23 / README.md Create

```
- 👋 Hi, I'm @enchante23
- 👀 I'm interested in ...
- 🌱 I'm currently learning ...
- 💞️ I'm looking to collaborate on ...
- 📫 How to reach me ...
- 😄 Pronouns: ...
- ⚡ Fun fact: ...
```

Learning Pathways

AI-powered development with GitHub Copilot

Gain expertise and insights from top organizations in these GitHub-guided learning pathways.
• AI-powered development with GitHub Copilot
• CI/CD with GitHub Actions
• Application Security with GitHub Advanced Security
• Administration and Governance with GitHub Enterprise

GitHub Learning Pathways

Latest changes

19 hours ago
Secret scanning changes to detection and validation for Google Cloud Platform, Slack

Yesterday
GitHub Native IP Allow List expands to EMU User Namespaces – Private Beta

Yesterday
CodeQL 2.17.0: Support for Java 22, Swift 5.10, TS 5.4, C# 12

2 days ago
GitHub-hosted runner images deprecation notice: Docker Compose v1

View changelog →

How to use Git?

Step 1: git config

→ git config --global user.name "My Name"
→ git config --global user.email "myemail@example.com"

In Git, the **git config** command is used to configure settings specific to Git operations. These settings are stored in three distinct scopes: **system, global, and local**.

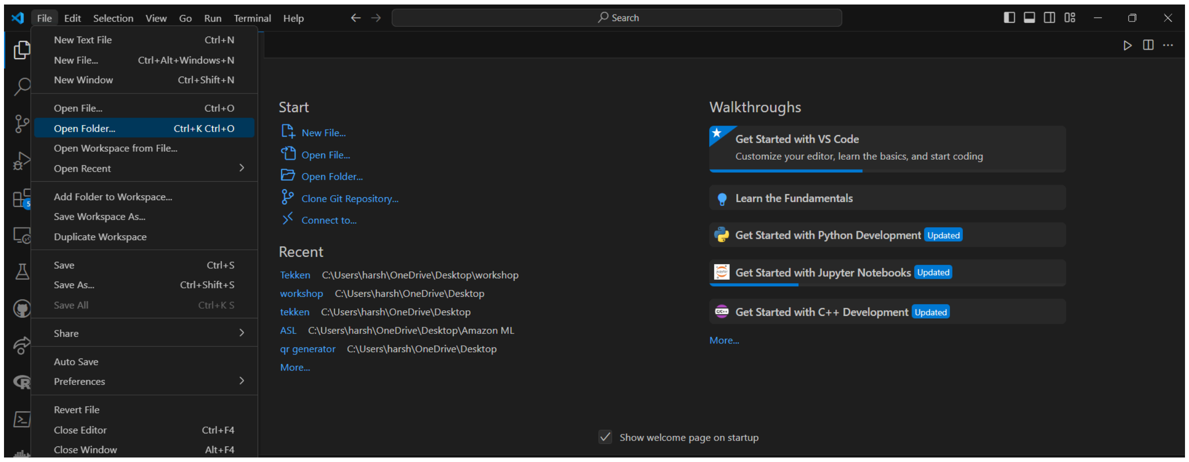
System Configuration: This scope applies system-wide, affecting all users and repositories on the system.
Configuration settings at this level are stored in the /etc/gitconfig file.

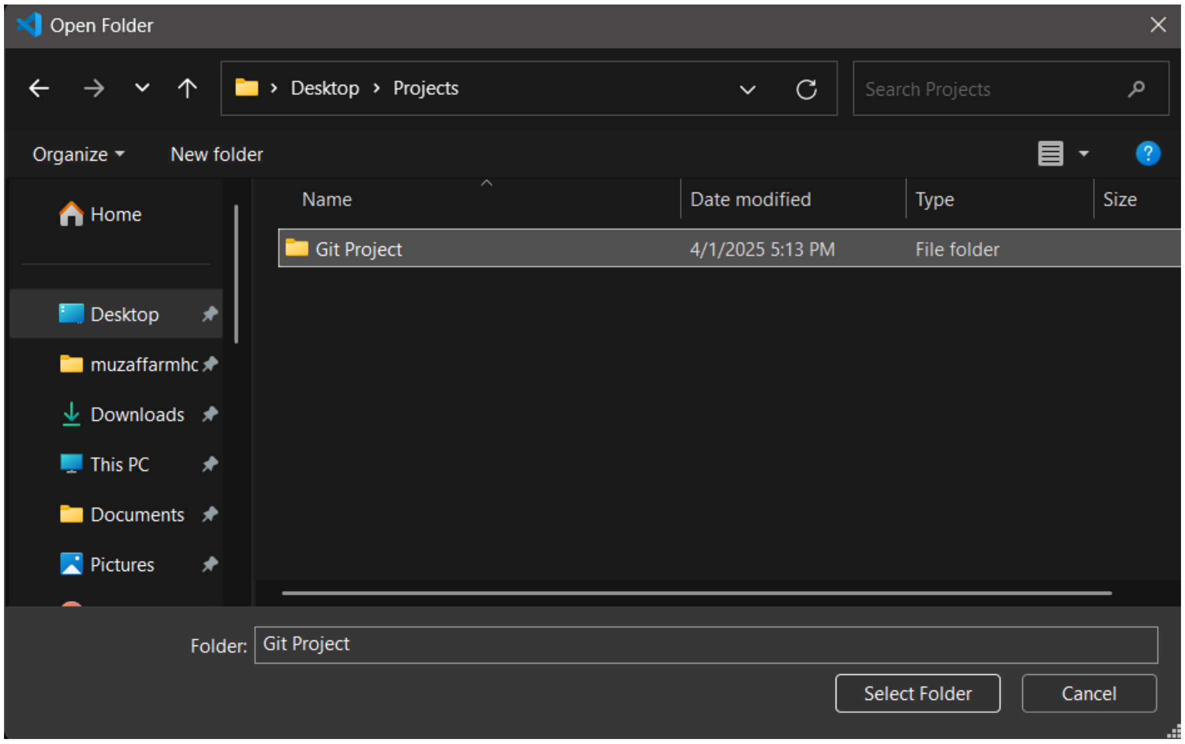
Global Configuration: Global configurations are specific to a single user and apply to all repositories associated with that user.

Local Configuration: Local configurations are specific to a single repository and override settings from higher levels.

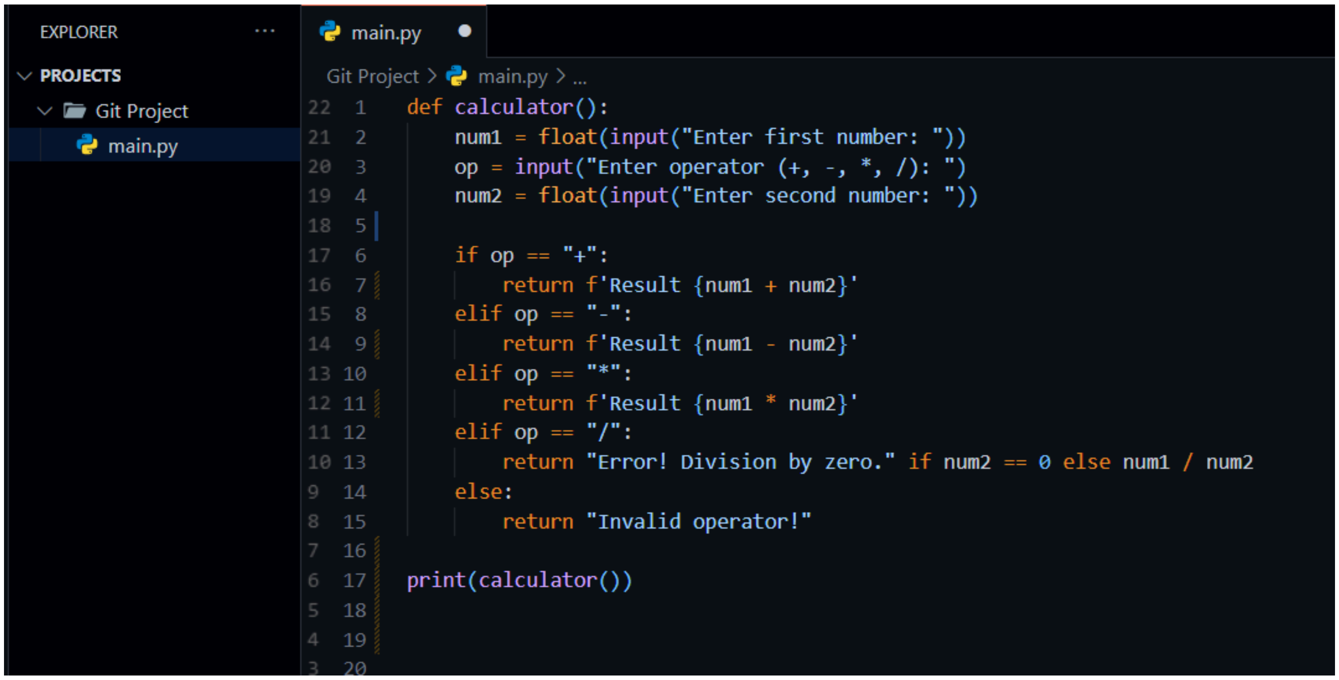
Step 2: Create a folder for your project and add the required files

- Create a new folder and open in on your code editor and initialize it using git init.


File Edit Selection View Go Run Terminal Help
Search
New Text File Ctrl+N
New File... Ctrl+Alt+Windows+N
New Window Ctrl+Shift+N
Open File... Ctrl+O
Open Folder... Ctrl+K Ctrl+O
Open Workspace from File...
Open Recent
Add Folder to Workspace...
Save Workspace As...
Duplicate Workspace
Save Ctrl+S
Save As... Ctrl+Shift+S
Save All Ctrl+K S
Share
Auto Save
Preferences
Revert File
Close Editor Ctrl+F4
Close Window Alt+F4
Start
New File...
Open File...
Open Folder...
Clone Git Repository...
Connect to...
Recent
Tekken C:\Users\harsh\OneDrive\Desktop\workshop
workshop C:\Users\harsh\OneDrive\Desktop
tekken C:\Users\harsh\OneDrive\Desktop
ASL C:\Users\harsh\OneDrive\Desktop\Amazon ML
qr generator C:\Users\harsh\OneDrive\Desktop
More...
Walkthroughs
Get Started with VS Code
Customize your editor, learn the basics, and start coding
Learn the Fundamentals
Get Started with Python Development Updated
Get Started with Jupyter Notebooks Updated
Get Started with C++ Development Updated
More...
Show welcome page on startup



- Add the necessary files for your project into the folder



EXPLORER

PROJECTS

Git Project

main.py

Git Project > main.py > ...

```python
def calculator():
    num1 = float(input("Enter first number: "))
    op = input("Enter operator (+, -, *, /): ")
    num2 = float(input("Enter second number: "))

    if op == "+":
        return f'Result {num1 + num2}'
    elif op == "-":
        return f'Result {num1 - num2}'
    elif op == "*":
        return f'Result {num1 * num2}'
    elif op == "/":
        return "Error! Division by zero." if num2 == 0 else num1 / num2
    else:
        return "Invalid operator!"

print(calculator())
```

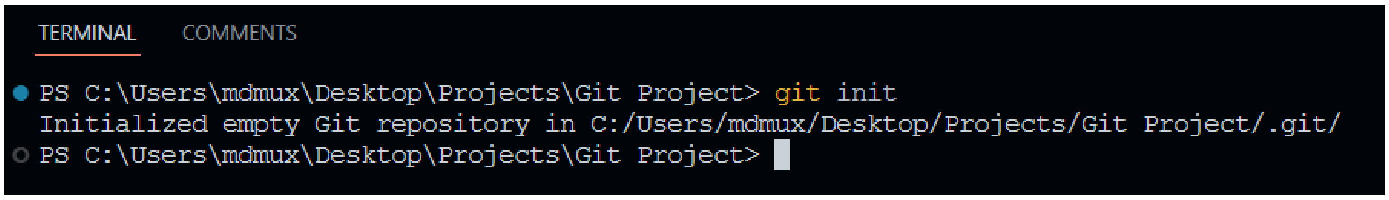
main.py

```python
def calculator():
    num1 = float(input("Enter first number: "))
    op = input("Enter operator (+, -, *, /): ")
    num2 = float(input("Enter second number: "))

    if op == "+":
        return f'Result {num1 + num2}'
    elif op == "-":
        return f'Result {num1 - num2}'
    elif op == "*":
        return f'Result {num1 * num2}'
    elif op == "/":
        return "Error! Division by zero." if num2 == 0 else num1 / num2
    else:
        return "Invalid operator!"

print(calculator())
```

Step 3: git init

→ git init

It is used to create an empty Git repository or reinitialize an existing one

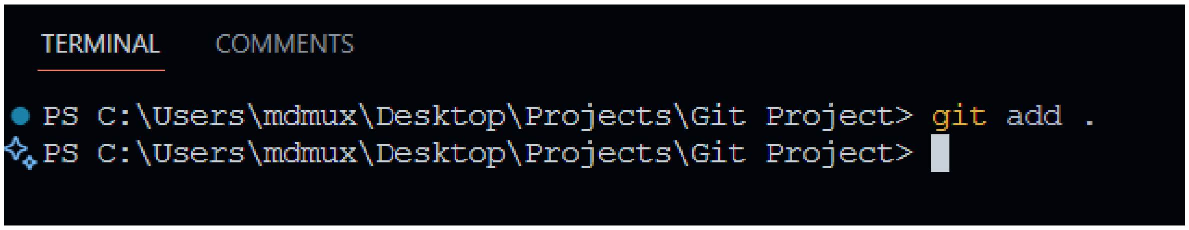


```
PS C:\Users\mdmux\Desktop\Projects\Git Project> git init
Initialized empty Git repository in C:/Users/mdmux/Desktop/Projects/Git Project/.git/
PS C:\Users\mdmux\Desktop\Projects\Git Project>
```

Step 4: git add

The git add command is used to stage files in your project directory for the next commit. It takes one or more paths as arguments and adds the current content of those paths to the staging area. The staging area is a file that Git uses to keep track of which files are going to be included in the next commit.

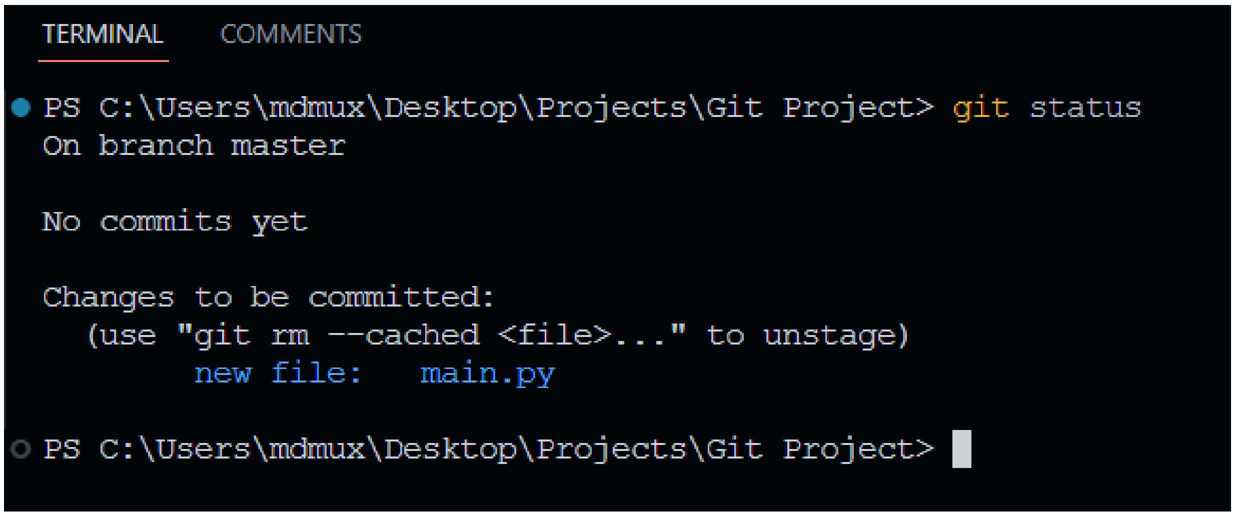
- **git add <file>** - Stages all changes in <file> for the next commit.
- **git add <directory>** - Stages all changes in <directory> for the next commit.
- **git add .** - Stages all changes in the current working directory and subdirectories, including new files and modifications for the next commit.
- **git add -A** - Stages all the changes in the entire repository and subdirectories, including new files, modifications, and *file deletions* for the next commit.



```
PS C:\Users\mdmux\Desktop\Projects\Git Project> git add .
PS C:\Users\mdmux\Desktop\Projects\Git Project>
```

Step 5: git status

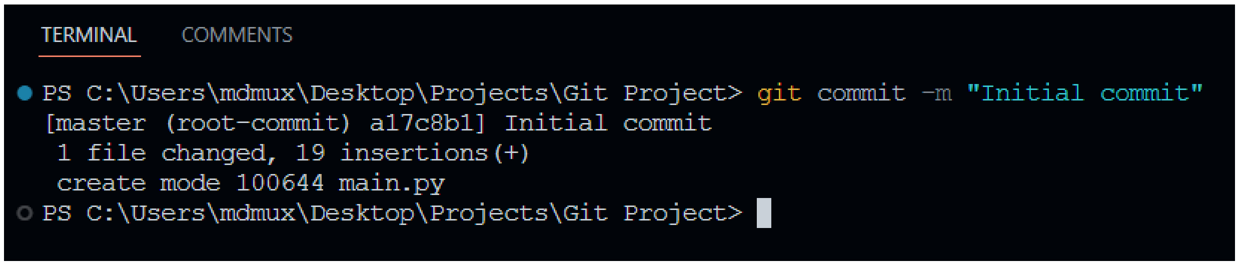
The git status command displays the state of the working directory and the staging area. It lets you see which changes have been staged, which haven't, and which files aren't being tracked by Git.



```
PS C:\Users\mdmux\Desktop\Projects\Git Project> git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   main.py

PS C:\Users\mdmux\Desktop\Projects\Git Project>
```

Step 6: git commit

A Git commit is a snapshot of your repository at a specific time. Commits are the building blocks of "save points" in Git's version control. The git commit command takes several options, but the most important one is the -m option, which allows you to specify a commit message.

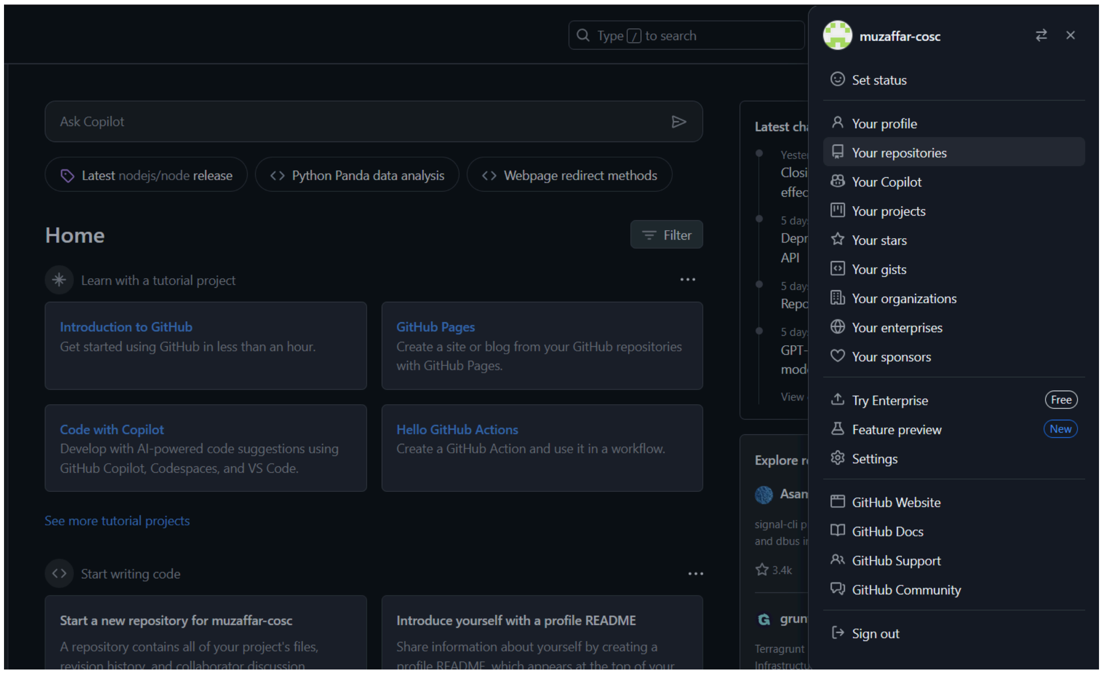


```
PS C:\Users\mdmux\Desktop\Projects\Git Project> git commit -m "Initial commit"
[master (root-commit) a17c8b1] Initial commit
 1 file changed, 19 insertions(+)
 create mode 100644 main.py
PS C:\Users\mdmux\Desktop\Projects\Git Project>
```

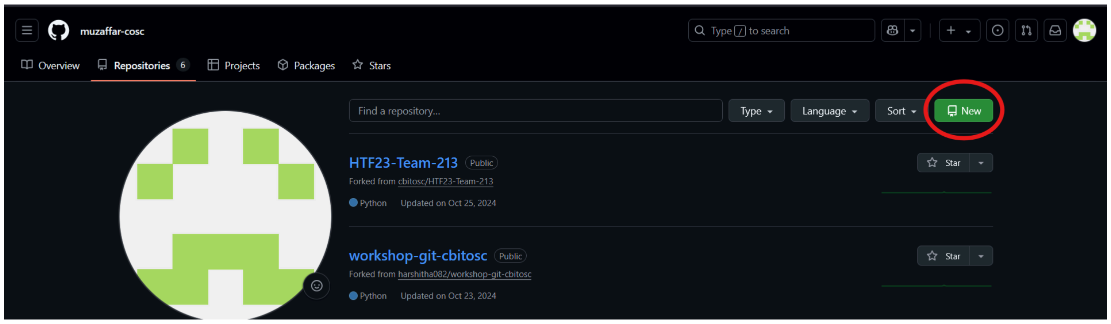
Step 7: Create a remote repository on GitHub

A remote repository on GitHub serves as a central hub for collaboration, version control, and project management.

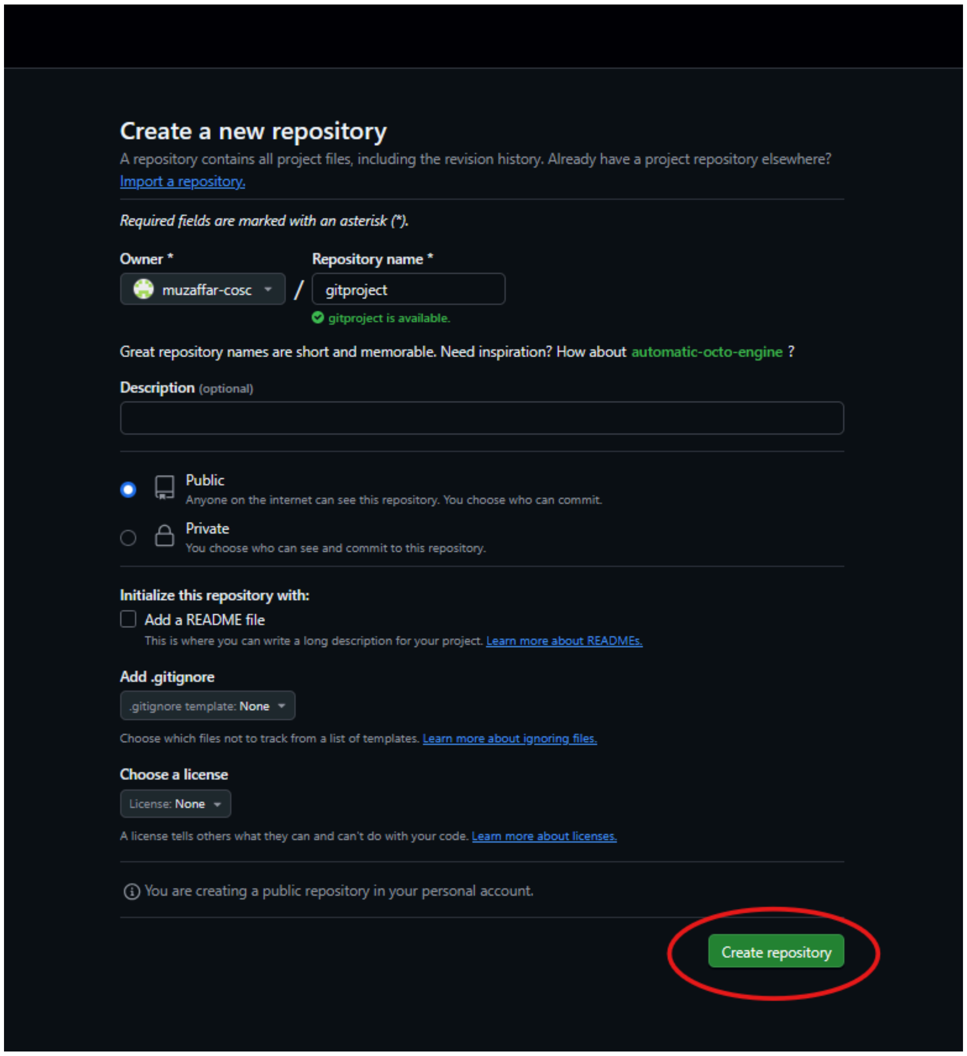
- Go to your GitHub account, click on your profile in the top right corner, and click on Your repositories.


Type / to search
Ask Copilot
Latest nodejs/node release
Python Panda data analysis
Webpage redirect methods
Home
Filter
Learn with a tutorial project
Introduction to GitHub
Get started using GitHub in less than an hour.
GitHub Pages
Create a site or blog from your GitHub repositories with GitHub Pages.
Code with Copilot
Develop with AI-powered code suggestions using GitHub Copilot, Codespaces, and VS Code.
Hello GitHub Actions
Create a GitHub Action and use it in a workflow.
See more tutorial projects
Start writing code
Start a new repository for muzaffar-cosc
Introduce yourself with a profile README
muzaffar-cosc
Set status
Your profile
Your repositories
Your Copilot
Your projects
Your stars
Your gists
Your organizations
Your enterprises
Your sponsors
Try Enterprise
Free
Feature preview
New
Settings
GitHub Website
GitHub Docs
GitHub Support
GitHub Community
Sign out

- Click on the new icon to create a new repository



- Create a new repository with the required specifications.



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 * muzaffar-cosc / Repository name * gitproject

gitproject is available.

Great repository names are short and memorable. Need inspiration? How about automatic-octo-engine ?

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.

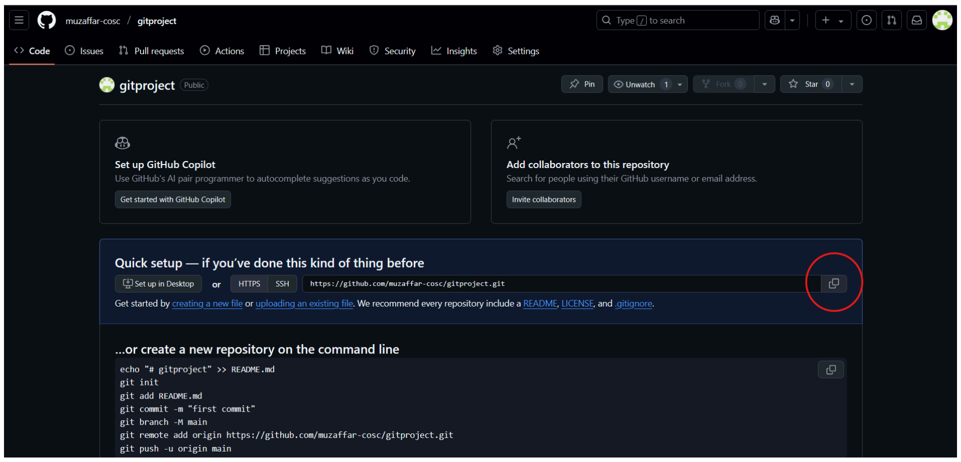
Choose a license

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.

- Once the repository is created, click on the Code icon and copy the HTTPS code, as we will use it to push your code.



Set up GitHub Copilot

Use GitHub's AI pair programmer to autocomplete suggestions as you code.

Add collaborators to this repository

Search for people using their GitHub username or email address.

Quick setup — if you've done this kind of thing before

```
https://github.com/muzaffar-cosc/gitproject.git
```

Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

...or create a new repository on the command line

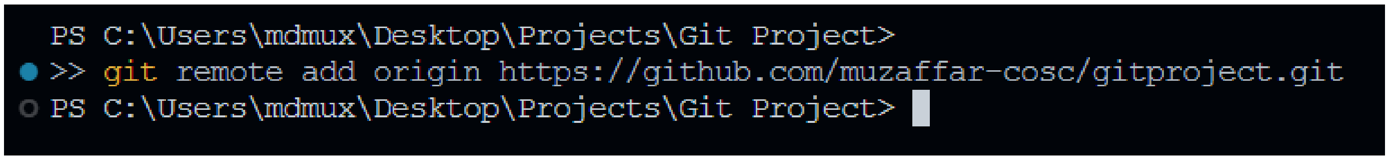
```
echo "# gitproject" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/muzaffar-cosc/gitproject.git
git push -u origin main
```

Step 8: Adding your new remote repository to your local git repository

The git remote command is essentially an interface for managing a list of remote entries that are stored in the repository's ./. git/config file.

git remote add <name> <url>

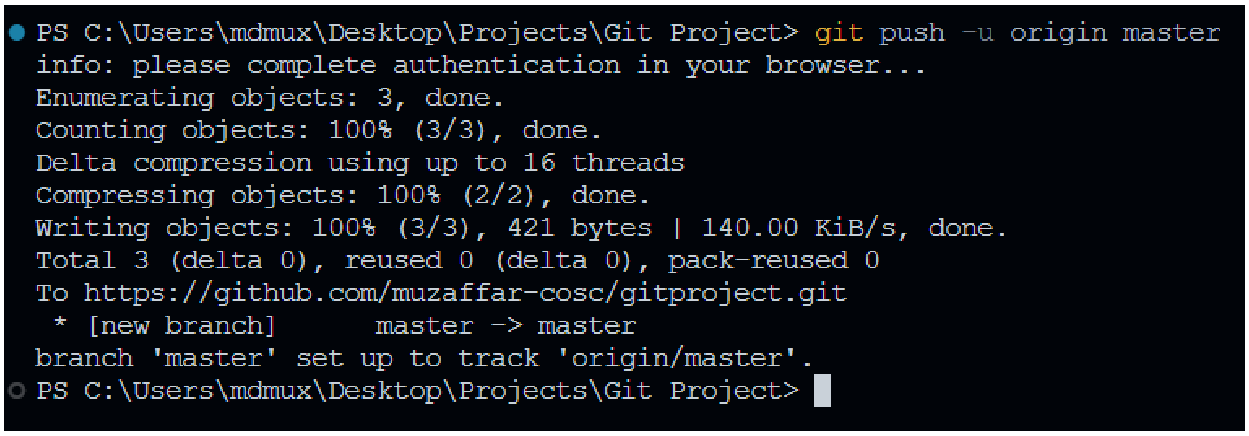
Create a new connection to a remote repository. After adding a remote, you'll be able to use <name> as a convenient shortcut for <url> in other Git commands.



```
 PS C:\Users\mdmux\Desktop\Projects\Git Project>
● >> git remote add origin https://github.com/muzaffar-cosc/gitproject.git
○ PS C:\Users\mdmux\Desktop\Projects\Git Project>
```

Step 9: git push

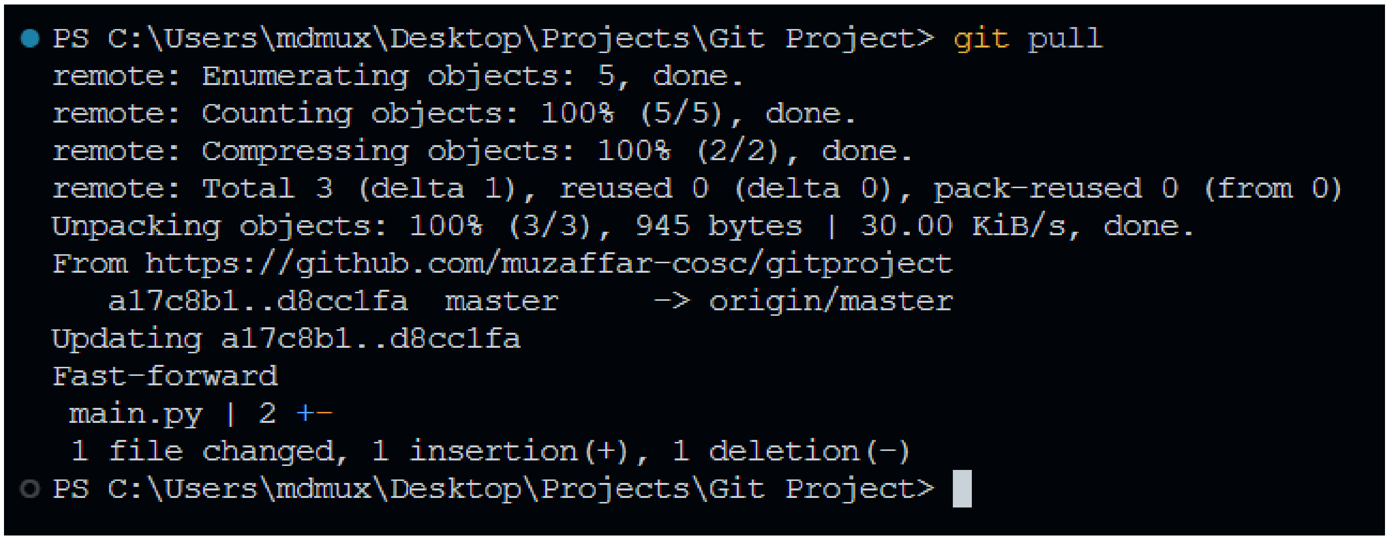
The git push command is used to upload local repository content to a remote repository. Pushing is how you transfer commits from your local repository to a remote repo. Pushing has the potential to overwrite changes, caution should be taken when pushing.



```
● PS C:\Users\mdmux\Desktop\Projects\Git Project> git push -u origin master
  info: please complete authentication in your browser...
  Enumerating objects: 3, done.
  Counting objects: 100% (3/3), done.
  Delta compression using up to 16 threads
  Compressing objects: 100% (2/2), done.
  Writing objects: 100% (3/3), 421 bytes | 140.00 KiB/s, done.
  Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
  To https://github.com/muzaffar-cosc/gitproject.git
   * [new branch]      master -> master
  branch 'master' set up to track 'origin/master'.
○ PS C:\Users\mdmux\Desktop\Projects\Git Project>
```

Step 10: git pull

The git pull command is used to fetch and download content from a remote repository and immediately update the local repository to match that content. Merging remote upstream changes into your local repository is a common task in Git-based collaboration workflows. The git pull command is a combination of two other commands, git fetch followed by git merge. In the first stage of operation, git pull will execute a git fetch scoped to the local branch that HEAD is pointed at. Once the content is downloaded, git pull will enter a merge workflow. A new merge commit will be created and HEAD updated to point at the new commit.

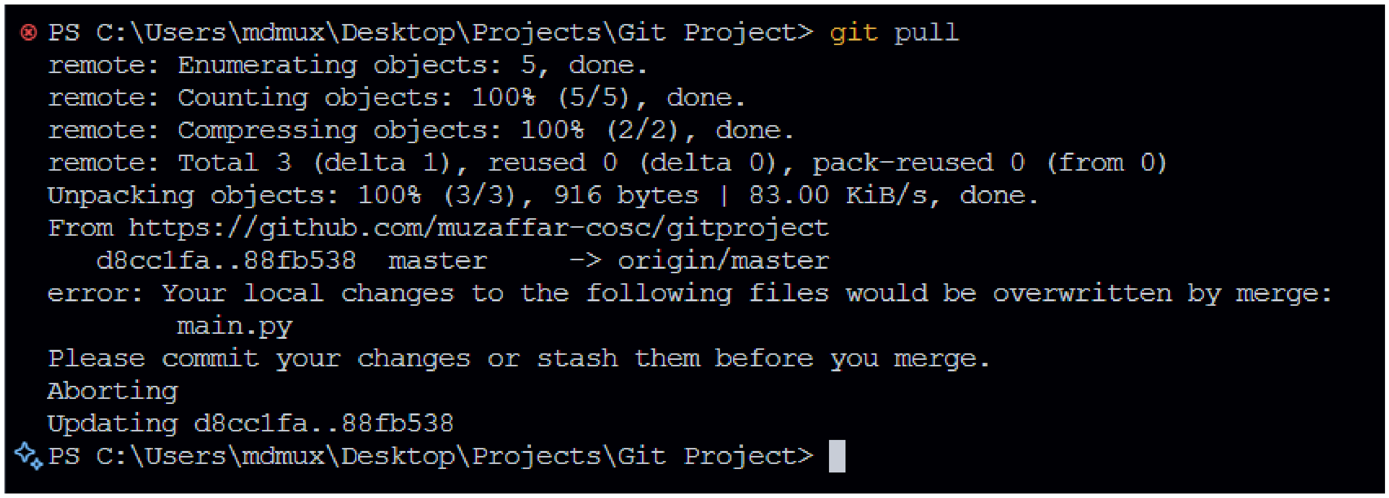


```
PS C:\Users\mdmux\Desktop\Projects\Git Project> git pull
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (3/3), 945 bytes | 30.00 KiB/s, done.
From https://github.com/muzaffar-cosc/gitproject
   a17c8b1..d8cc1fa  master     -> origin/master
Updating a17c8b1..d8cc1fa
Fast-forward
 main.py | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
PS C:\Users\mdmux\Desktop\Projects\Git Project>
```

Resolving Merge Conflicts

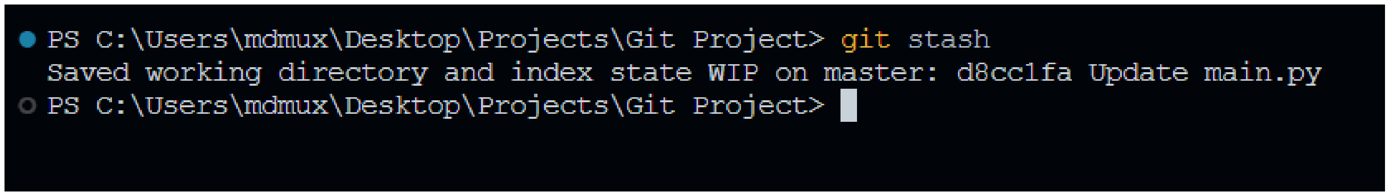
- When you pull changes, you are always prone to encountering Merge conflicts. Often, merge conflicts happen when people make different changes to the same line of the same file, or when one person edits a file and another person deletes the same file. You must resolve all merge conflicts before you can merge a pull request on GitHub.

- Here is an example of encountering a merge conflict.

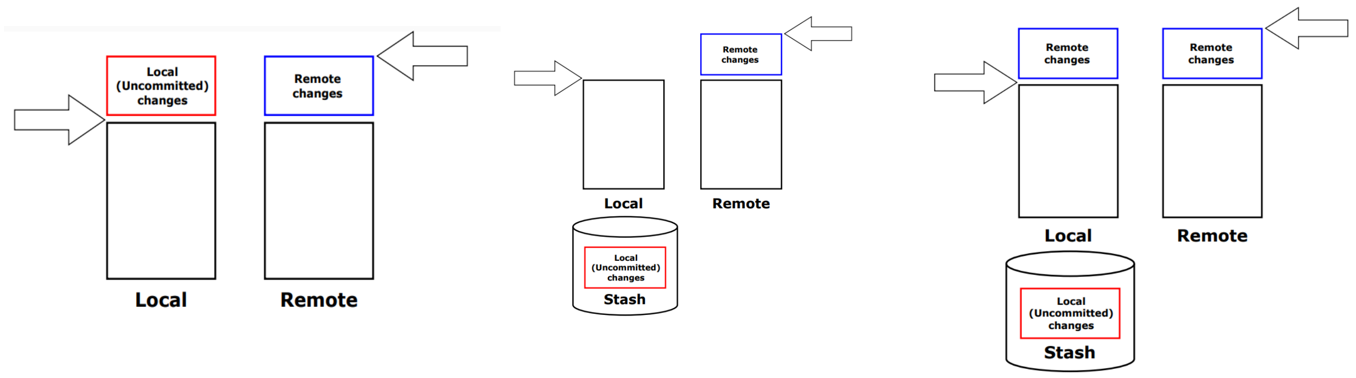


```
PS C:\Users\mdmux\Desktop\Projects\Git Project> git pull
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (3/3), 916 bytes | 83.00 KiB/s, done.
From https://github.com/muzaffar-cosc/gitproject
   d8cc1fa..88fb538  master     -> origin/master
error: Your local changes to the following files would be overwritten by merge:
        main.py
Please commit your changes or stash them before you merge.
Aborting
Updating d8cc1fa..88fb538
PS C:\Users\mdmux\Desktop\Projects\Git Project>
```

- You can use git stash to save all the uncommitted changes into a stack, the local becomes the same as the previous git.

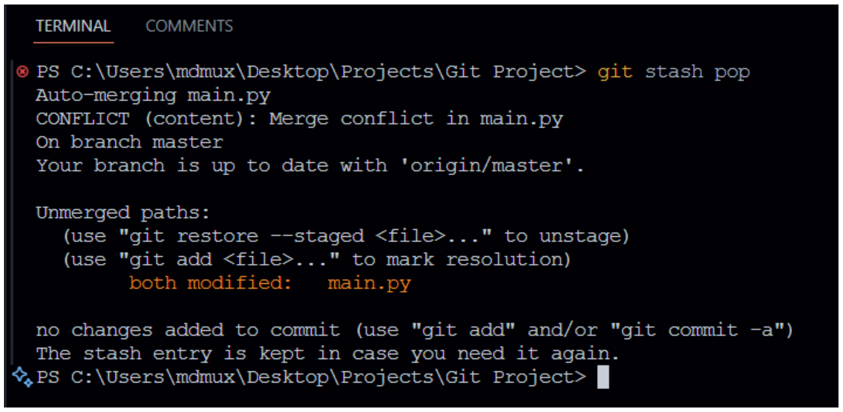


```
PS C:\Users\mdmux\Desktop\Projects\Git Project> git stash
Saved working directory and index state WIP on master: d8cc1fa Update main.py
PS C:\Users\mdmux\Desktop\Projects\Git Project>
```

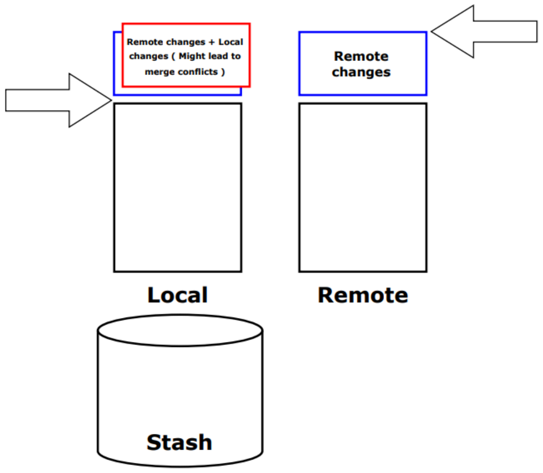


After stashing your changes, you can safely pull remote changes using command git pull and the reflected changes in the remote repository will appear in your local repo.

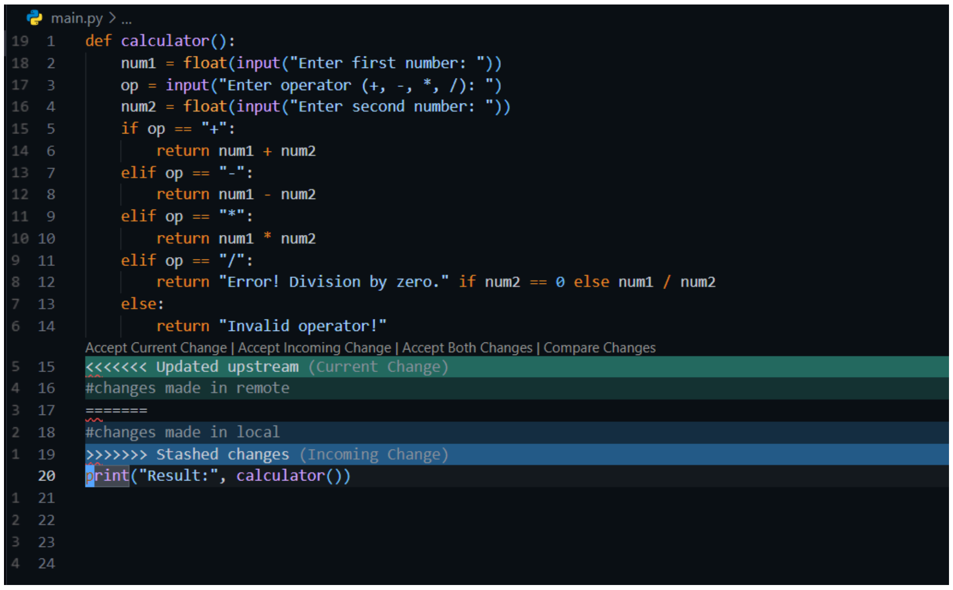
To reflect the stashed changes, which are present in a stack like data structure, you can simply pop the stash and the changes will be reflected. This is done using the command **git stash pop.**



```
TERMINAL    COMMENTS

PS C:\Users\mdmux\Desktop\Projects\Git Project> git stash pop
Auto-merging main.py
CONFLICT (content): Merge conflict in main.py
On branch master
Your branch is up to date with 'origin/master'.

Unmerged paths:
  (use "git restore --staged <file>..." to unstage)
  (use "git add <file>..." to mark resolution)
        both modified:   main.py

no changes added to commit (use "git add" and/or "git commit -a")
The stash entry is kept in case you need it again.
PS C:\Users\mdmux\Desktop\Projects\Git Project>
```

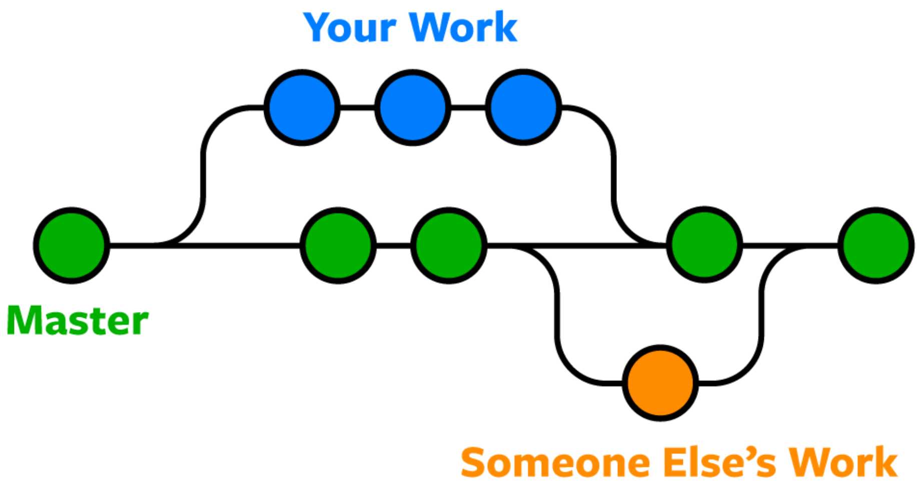

Remote changes + Local changes (Might lead to merge conflicts)
Remote changes
Local
Remote
Stash

- You can resolve the merge conflicts as shown below on your code editor.



```
main.py > ...
def calculator():
    num1 = float(input("Enter first number: "))
    op = input("Enter operator (+, -, *, /): ")
    num2 = float(input("Enter second number: "))
    if op == "+":
        return num1 + num2
    elif op == "-":
        return num1 - num2
    elif op == "*":
        return num1 * num2
    elif op == "/":
        return "Error! Division by zero." if num2 == 0 else num1 / num2
    else:
        return "Invalid operator!"
Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
<<<<<<< Updated upstream (Current Change)
#changes made in remote
=======
#changes made in local
>>>>>>> Stashed changes (Incoming Change)
print("Result:", calculator())
```

Branching



A branch represents an independent line of development. Branches serve as an abstraction for the edit/stage/commit process. You can think of them as a way to request a brand new working directory, staging area, and project history. New commits are recorded in the history for the current branch, which results in a fork in the history of the project.

The git branch command lets you create, list, rename, and delete branches. It doesn't let you switch between branches or put a forked history back together again. For this reason, git branch is tightly integrated with the git checkout and git merge commands.

Create a new branch called <branch>.

```
git branch <branch>
```

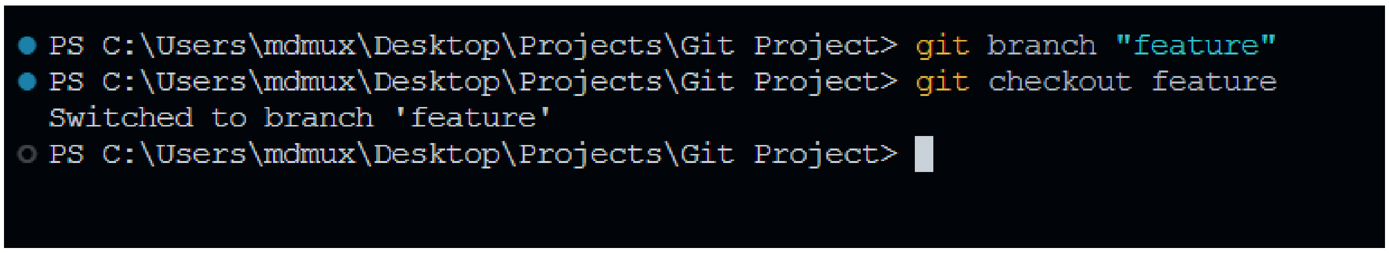
Switching branches is a straightforward operation. Executing the following will point HEAD to the tip of <branchname>.

```
git checkout <new-branch>
```

or

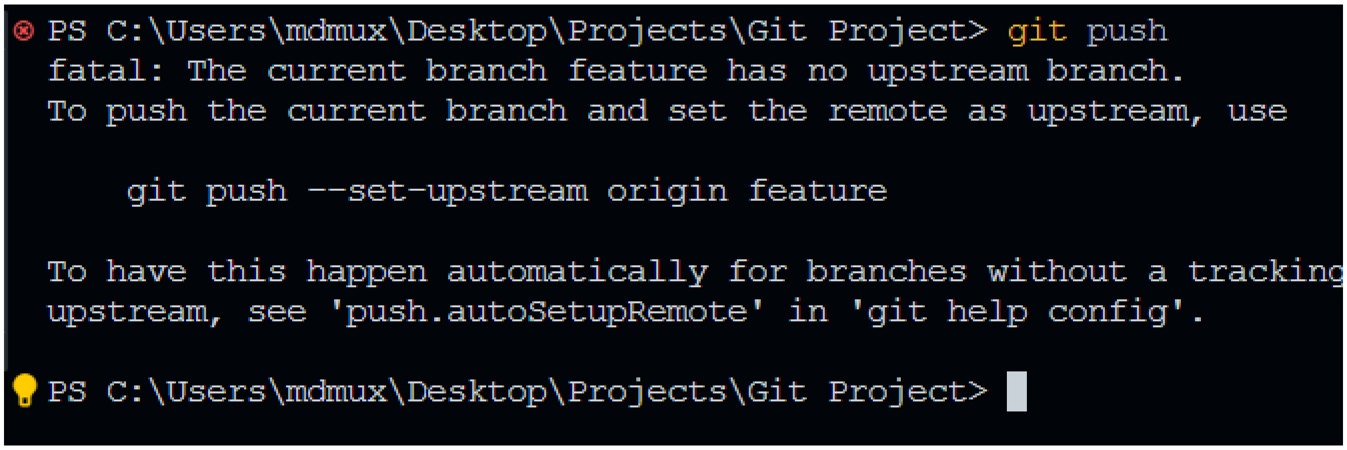
```
git checkout -b <new-branch>
```

The above example simultaneously creates and checks out <new-branch>.



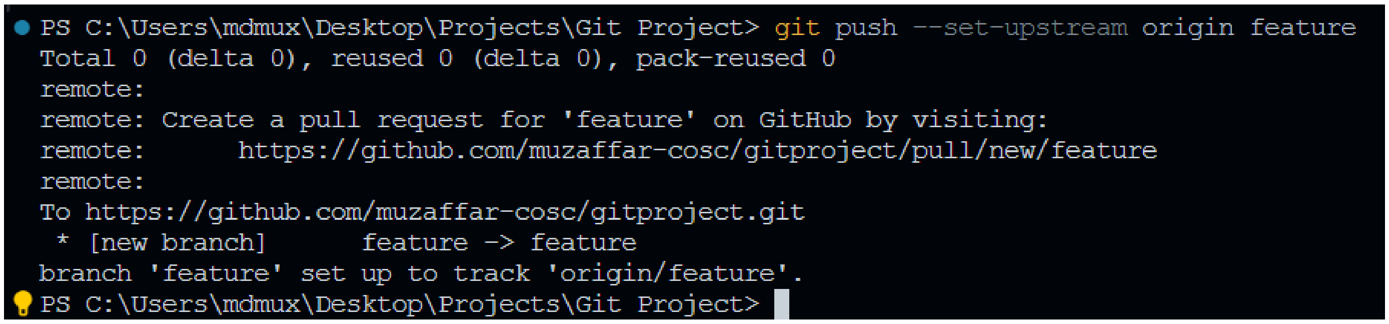
```
PS C:\Users\mdmux\Desktop\Projects\Git Project> git branch "feature"
PS C:\Users\mdmux\Desktop\Projects\Git Project> git checkout feature
Switched to branch 'feature'
PS C:\Users\mdmux\Desktop\Projects\Git Project>
```

- However, you can not directly push the changes made on a branch to your remote repository. Upon attempting to push, you will encounter the following error



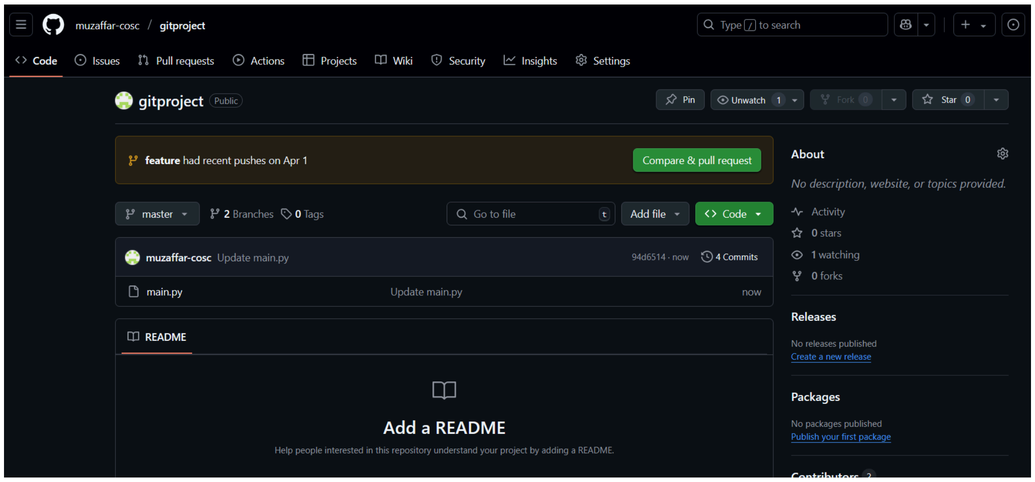
```
PS C:\Users\mdmux\Desktop\Projects\Git Project> git push
fatal: The current branch feature has no upstream branch.
To push the current branch and set the remote as upstream, use

    git push --set-upstream origin feature

To have this happen automatically for branches without a tracking
upstream, see 'push.autoSetupRemote' in 'git help config'.

PS C:\Users\mdmux\Desktop\Projects\Git Project>
```

- To push the branch changes, we need to set upstream as follows:
  git push --set-upstream origin <branch name>

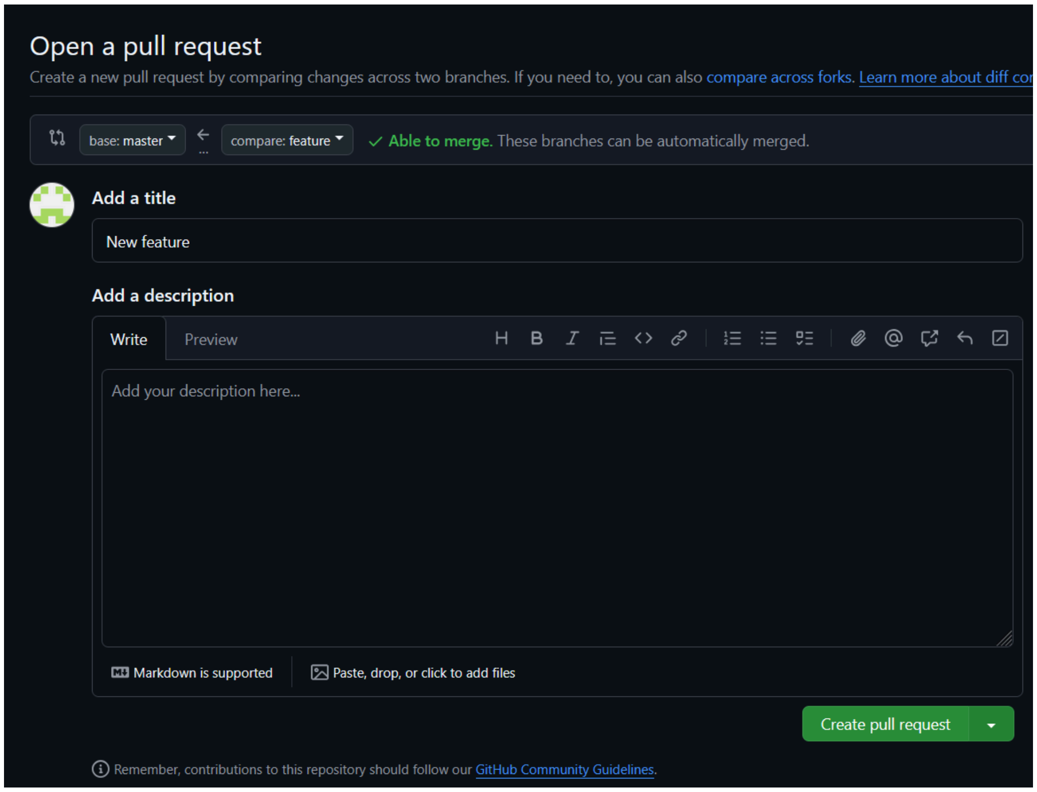


```
PS C:\Users\mdmux\Desktop\Projects\Git Project> git push --set-upstream origin feature
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'feature' on GitHub by visiting:
remote:      https://github.com/muzaffar-cosc/gitproject/pull/new/feature
remote:
To https://github.com/muzaffar-cosc/gitproject.git
 * [new branch]      feature -> feature
branch 'feature' set up to track 'origin/feature'.
PS C:\Users\mdmux\Desktop\Projects\Git Project>
```

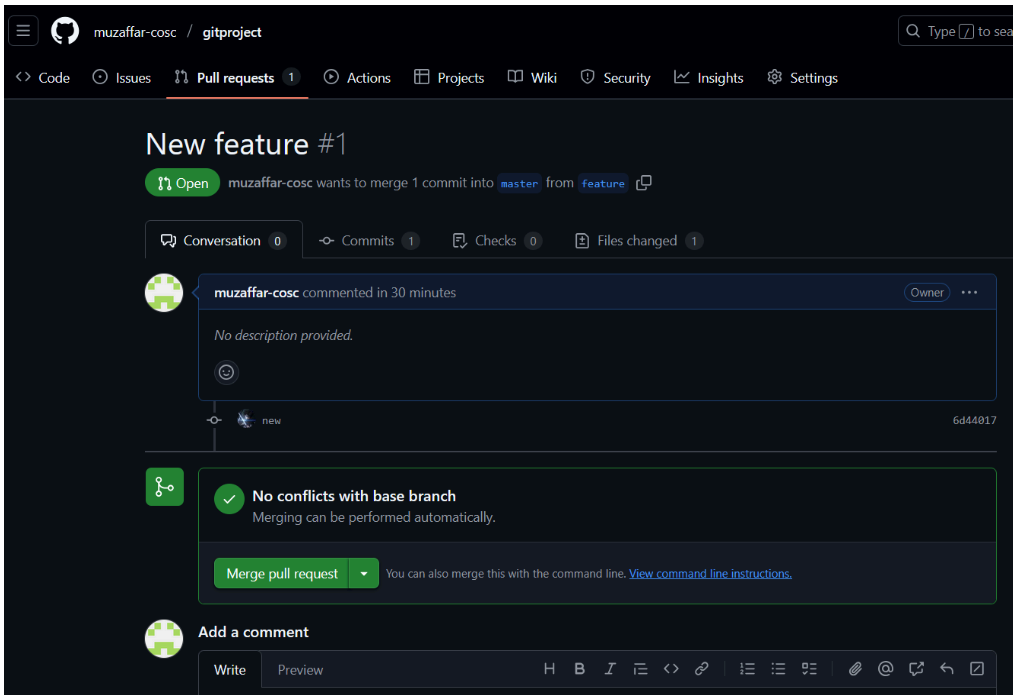
- Merging branches in Git is a fundamental operation that allows you to integrate changes from one branch into another. This is often used when you have been working on a feature or a bug fix in a separate branch and you want to bring those changes into another branch, typically the main branch.
- You can merge the branches you created by submitting a pull request on GitHub.



- Click on compare and pull request and add the necessary details to submit a pull request. Once the owners of the repository accept the Pull requests, your changes will be merged.



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 co
base: master
compare: feature
Able to merge. These branches can be automatically merged.
Add a title
New feature
Add a description
Write
Preview
Add your description here...
Markdown is supported
Paste, drop, or click to add files
Create pull request
Remember, contributions to this repository should follow our GitHub Community Guidelines.



New feature #1

Open muzaffar-cosc wants to merge 1 commit into `master` from `feature`

muzaffar-cosc commented in 30 minutes Owner

No description provided.

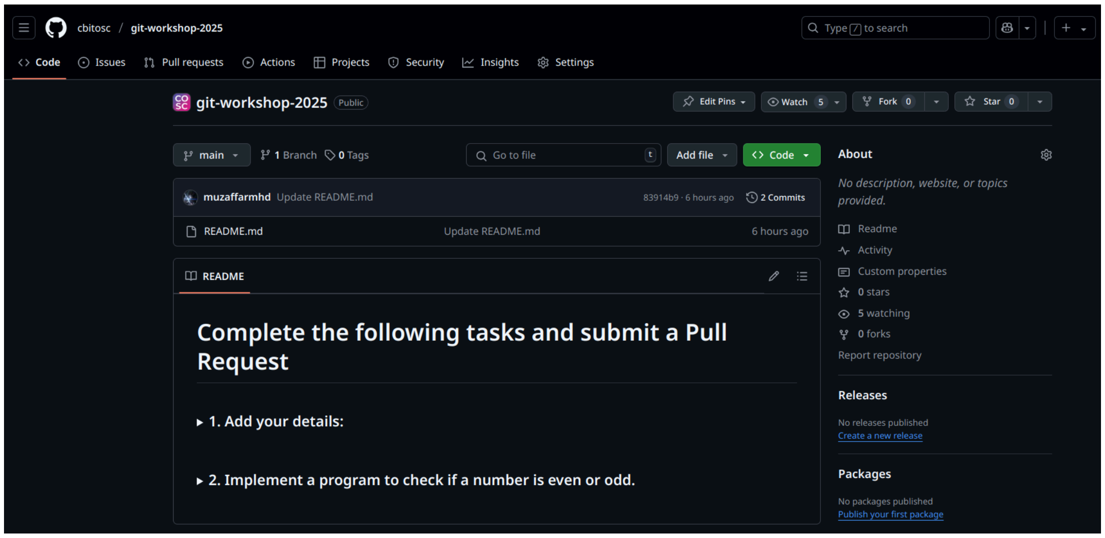
new 6d44017

No conflicts with base branch
Merging can be performed automatically.

You can also merge this with the command line. View command line instructions.

Contributing to Open Source Projects:

- GitHub is an excellent platform for finding and contributing to open-source projects. You can find many public repositories, such as below:
- https://github.com/cbitosc/git-workshop-2025



muzaffarmhd Update README.md 83914b9 · 6 hours ago 2 Commits

README.md Update README.md 6 hours ago

Complete the following tasks and submit a Pull Request

1. Add your details:

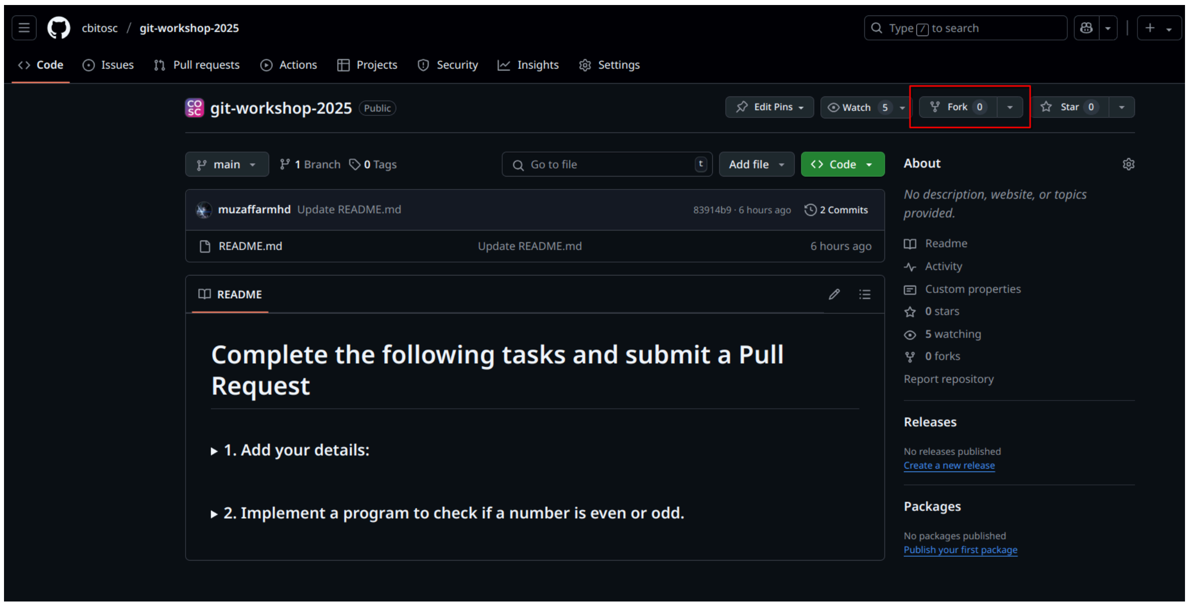
2. Implement a program to check if a number is even or odd.

About

No description, website, or topics provided.

Forking

- Forks let you make changes to a project without affecting the original repository, also known as the "upstream" repository. After you fork a repository, you can fetch updates from the upstream repository to keep your fork up to date, and you can propose changes from your fork to the upstream repository with pull requests. A fork can be owned by either a personal account or an organization.
- You can fork any repository by clicking on the fork button that's displayed when viewing a repository. You can not fork your own repository.



muzaffarmhd Update README.md 83914b9 · 6 hours ago 2 Commits

README.md Update README.md 6 hours ago

No description, website, or topics provided.

No releases published

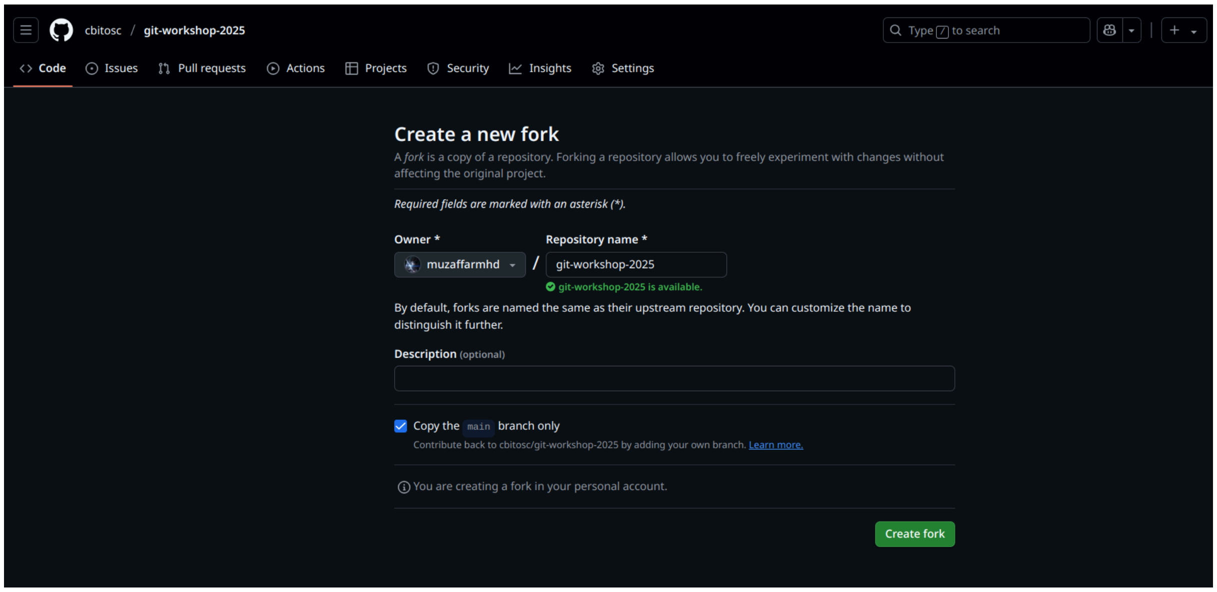
No packages published

Complete the following tasks and submit a Pull Request

▸ 1. Add your details:

▸ 2. Implement a program to check if a number is even or odd.

- After clicking on fork, specify the details of the fork and click on create fork.


cbitosc / git-workshop-2025

Code · Issues · Pull requests · Actions · Projects · Security · Insights · Settings

Create a new fork

A *fork* is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project.

Required fields are marked with an asterisk (*).

Owner *: muzaffarmhd / Repository name *: git-workshop-2025

git-workshop-2025 is available.

By default, forks are named the same as their upstream repository. You can customize the name to distinguish it further.

Description (optional)

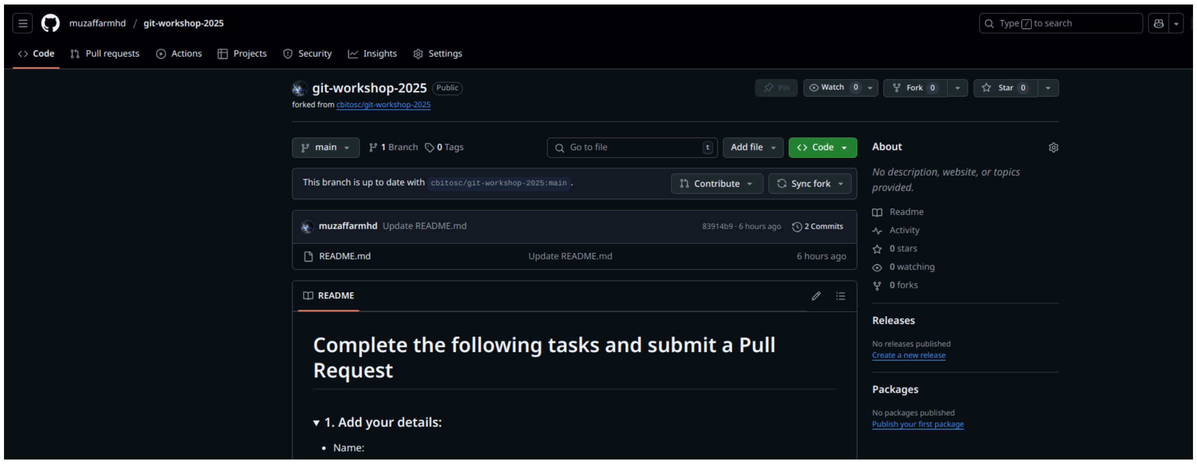
[✓] Copy the `main` branch only

Contribute back to cbitosc/git-workshop-2025 by adding your own branch. Learn more.

You are creating a fork in your personal account.

Create fork

- Once you have forked the repository, it will be reflected in your repositories.


muzaffarmhd / git-workshop-2025

Code · Pull requests · Actions · Projects · Security · Insights · Settings

git-workshop-2025 Public

forked from cbitosc/git-workshop-2025

Pin · Watch 0 · Fork 0 · Star 0

main · 1 Branch · 0 Tags · Go to file · Add file · Code

This branch is up to date with `cbitosc/git-workshop-2025:main` .

Contribute · Sync fork

muzaffarmhd Update README.md 83914b9 · 6 hours ago 2 Commits

README.md	Update README.md	6 hours ago

README

Complete the following tasks and submit a Pull Request

▾ 1. Add your details:

- Name:

About

No description, website, or topics provided.

Readme · Activity · 0 stars · 0 watching · 0 forks

Releases

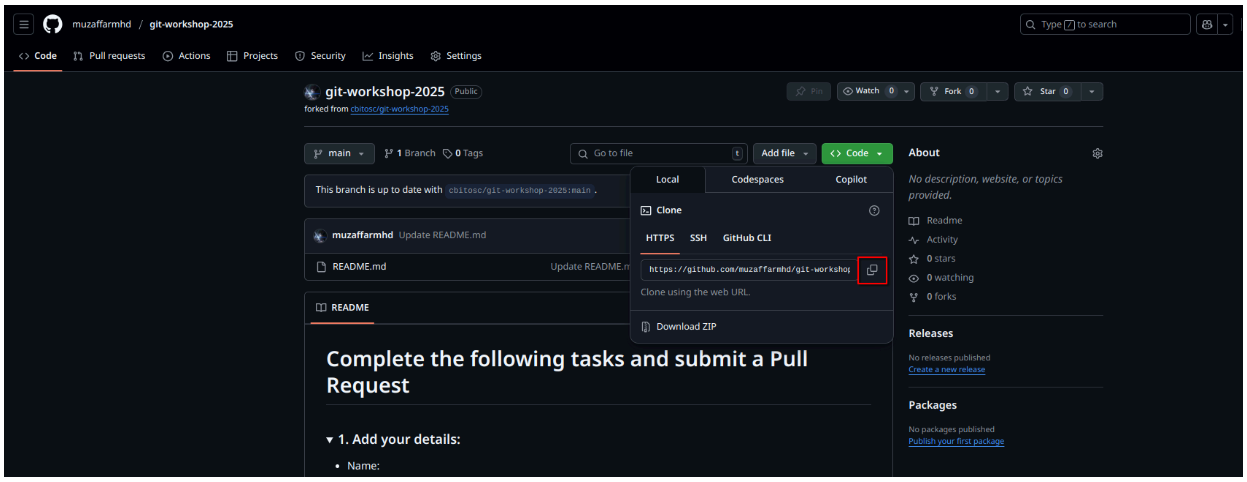
No releases published
Create a new release

Packages

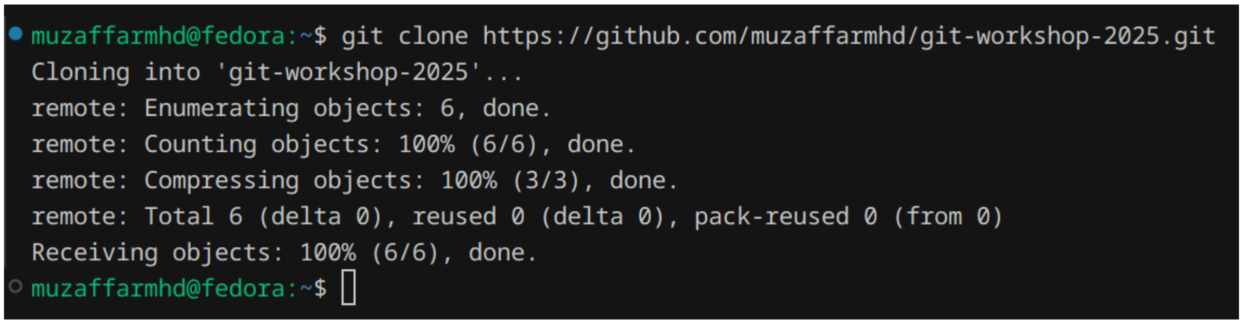
No packages published
Publish your first package

Cloning

git clone is a Git command line utility that is used to target an existing repository and create a clone, or copy of the target repository. In this page, we'll discuss extended configuration options and common use cases of git clone.



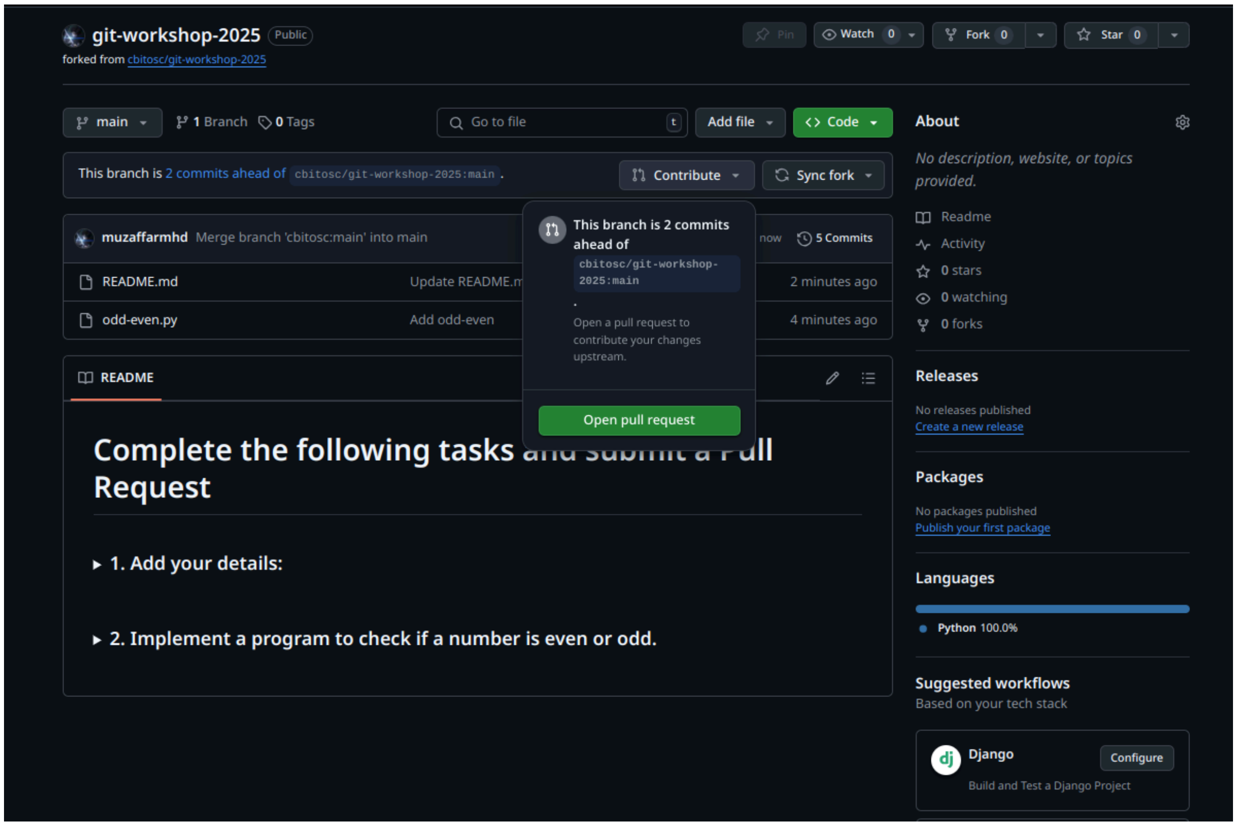
```
git clone <repo> <directory>
```



```
muzaffarmhd@fedora:~$ git clone https://github.com/muzaffarmhd/git-workshop-2025.git
Cloning into 'git-workshop-2025'...
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 6 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (6/6), done.
muzaffarmhd@fedora:~$ 
```

Submitting a Pull Request

- Once you have completed making changes to the cloned repository, to contribute these changes, you need to submit a Pull Request.
- You can do this by clicking on Contribute and then "Open pull request"


git-workshop-2025 Public
forked from cbitosc/git-workshop-2025

This branch is 2 commits ahead of `cbitosc/git-workshop-2025:main`.

muzaffarmhd Merge branch 'cbitosc:main' into main		now · 5 Commits
README.md	Update README.md	2 minutes ago
odd-even.py	Add odd-even	4 minutes ago

This branch is 2 commits ahead of `cbitosc/git-workshop-2025:main`.
Open a pull request to contribute your changes upstream.

Complete the following tasks and submit a Pull Request

▸ 1. Add your details:

▸ 2. Implement a program to check if a number is even or odd.

About
No description, website, or topics provided.

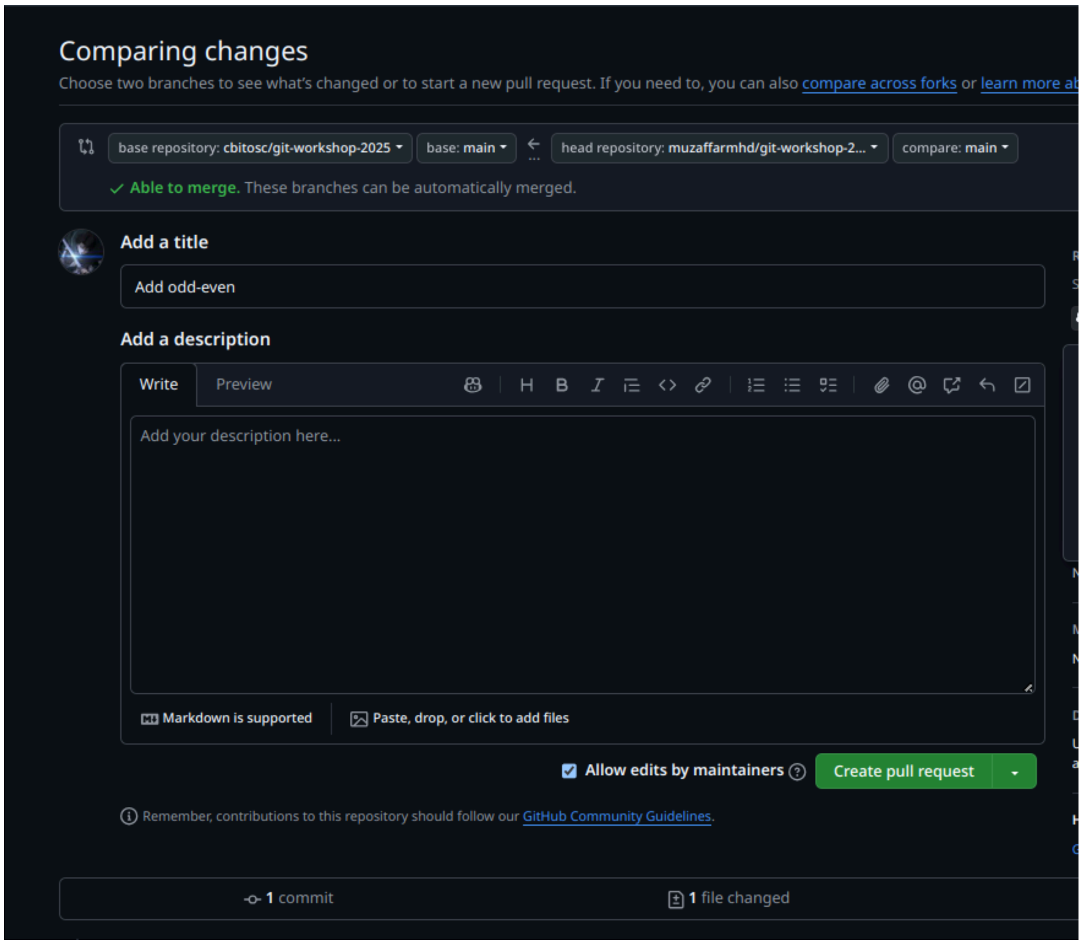
Releases
No releases published
Create a new release

Packages
No packages published
Publish your first package

Languages
Python 100.0%

Suggested workflows
Based on your tech stack
Django — Build and Test a Django Project

- Compare the changes and click on Create pull request. Once the owners of the repository approve of the pull request, your changes will be made to the original repository.


Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also compare across forks or learn more ab

base repository: cbitosc/git-workshop-2025 base: main head repository: muzaffarmhd/git-workshop-2... compare: main

✓ Able to merge. These branches can be automatically merged.

Add a title

Add odd-even

Add a description

Write
Preview
Add your description here...
Markdown is supported
Paste, drop, or click to add files
Allow edits by maintainers
Create pull request

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

1 commit 1 file changed

Additional Reading

1. Git Book - https://git-scm.com/book/en/v2