Creating a Github Repository

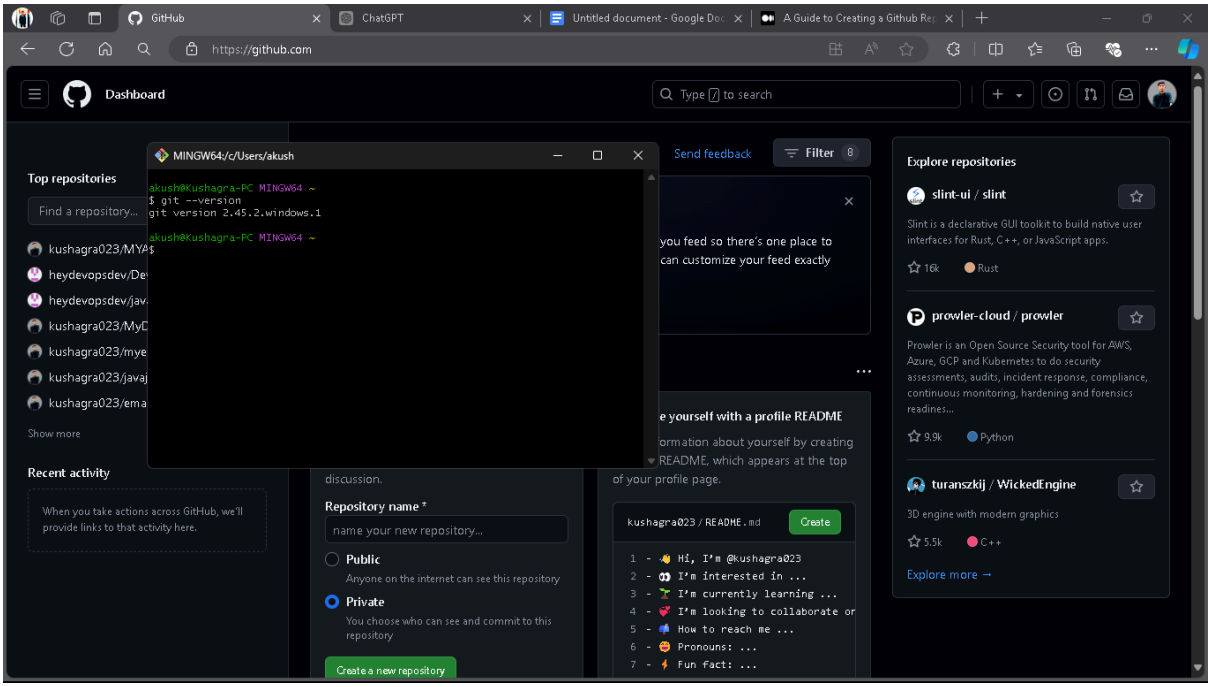
Install Git to the Terminal (Pre-Requiste)

Firstly, make sure that Git is installed on your terminal. This allows you to use the Git commands!

1. **Open the Terminal App on your System**

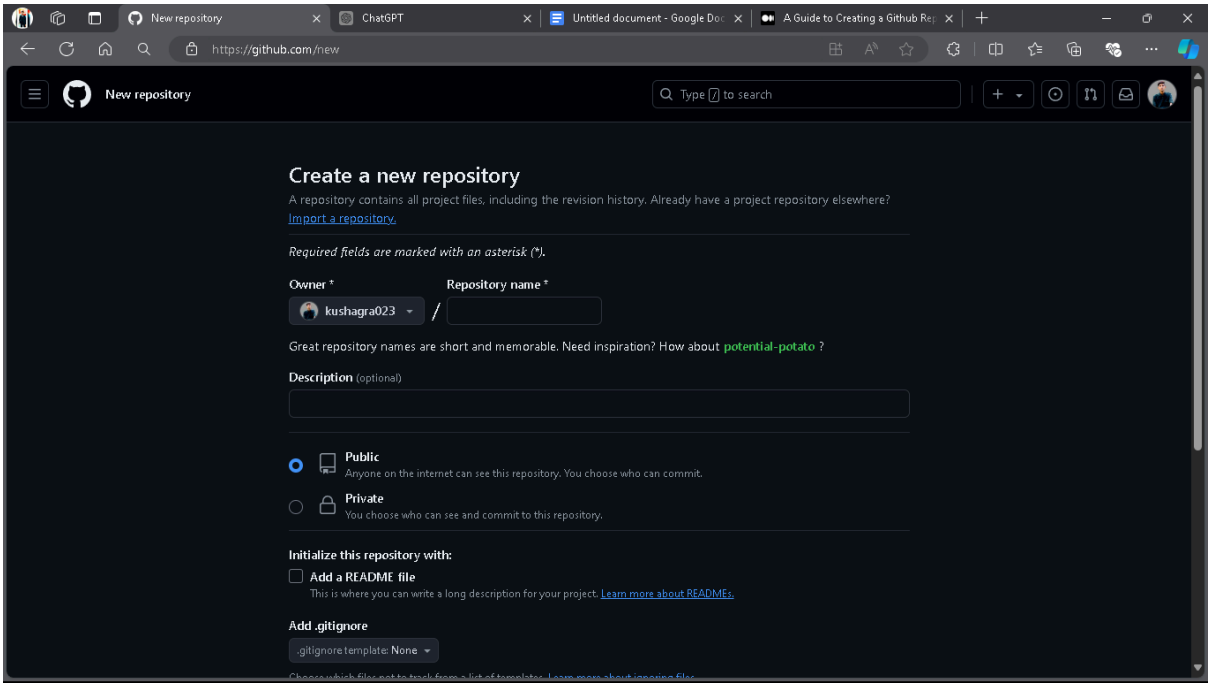
2.Type the following command into the Terminal

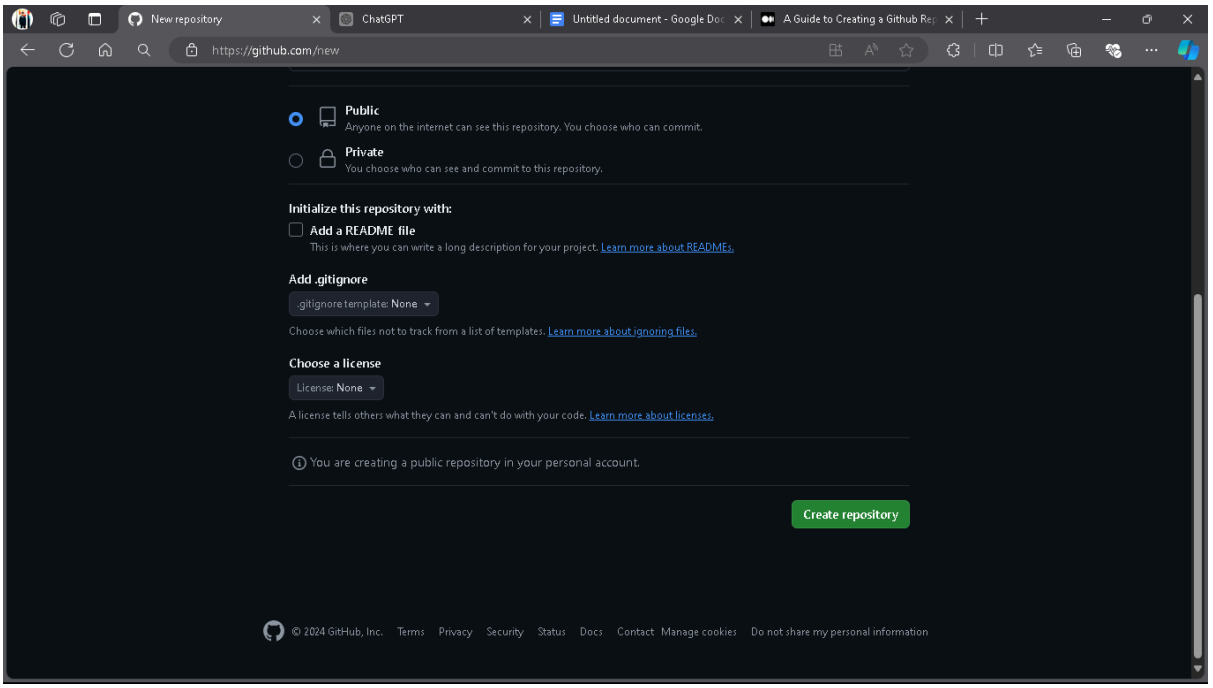
```
git --version
```



Create a Github Repository

Once you have installed ‘git’ to your terminal, we can get started by creating a repository, which is the place where your project will be stored.





To get started:

1.Go to Github (Create a account First) then .

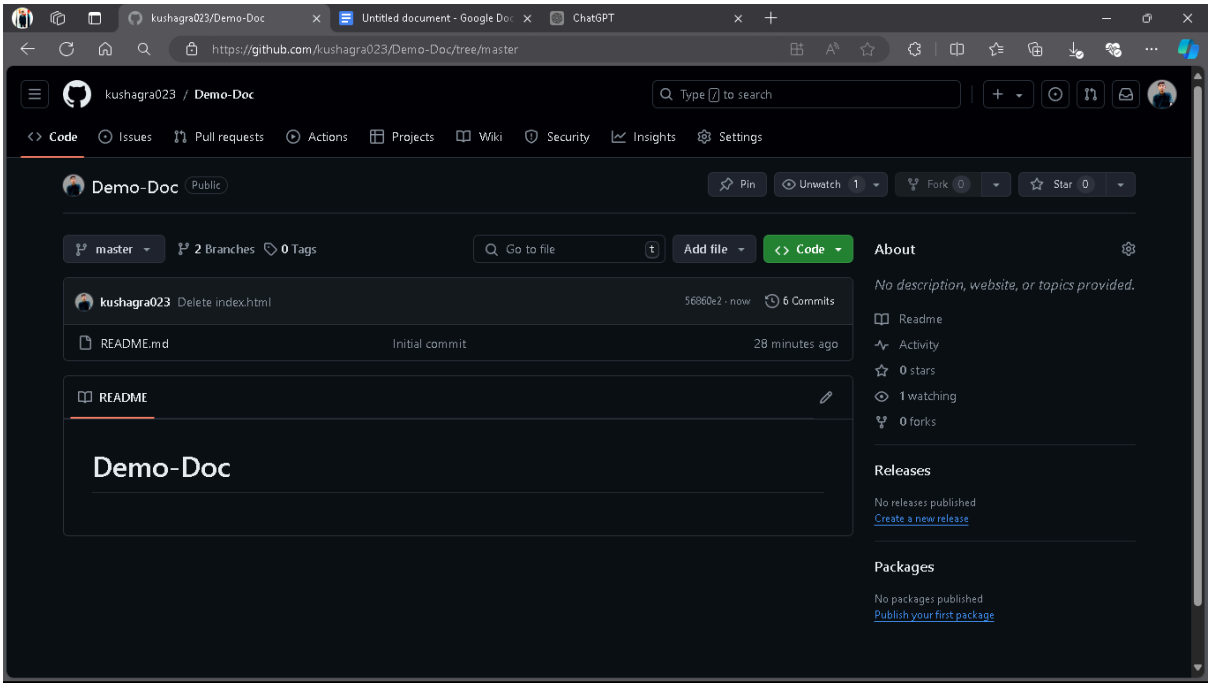
2. Click the “+” sign at the top right and click “New Repository”

3. Add the name of your project, along with a description.

4.Choose whether or not you want the Repository to be visible to others (Public Option) or just visible to you (Private Option)

5.I recommend initializing the repository with a README file

Note: The README File is where you can add notes/details about your project



Forking A Repository On Github

What is forking and how does it work? To Fork a project you are basically cloning someone else's original project from Github and adding it to your repository. This will allow you to modify the project without interfering with the owner's original project.

1)Head over to Github and create an account if you don't have one already.

2)Next search for the repository you would like to fork.

3)Click the fork button in the upper-righthand corner to add to your repository.

To add this forked project to our local machine, we will go back into our terminal and type the command

Clone The Forked Repo

git clone https://github.com/your-username/forked-repo.git

Make Changes :-

```
git add .            # To stage the changes
```

```
git commit -m "Enter Your commit message here"
git push origin main    # Replace 'main' with the branch you want to push to.
```

Updating Your Fork: If you wanna keep your forked repo up-to-date with the latest awesomeness from the original repo, no worries! Here's the secret sauce! First, add the original repo as an "upstream" remote with this command:

```
git remote add upstream
https://github.com/original-repo-owner/original-repo.git
```

Then, fetch all the juicy changes from the upstream repo using:

```
git fetch upstream
```

Now, it's time to merge those updates into your local repo! Just switch to the branch you wanna update (usually 'master') and run:

```
git checkout master
```

```
git merge upstream/master   # Replace 'master' with your branch name if it's different.
```

```
git push origin master
```

Git Commands

1. Git Init

The first step in using Git is to initialize a new Git repository. To do this, navigate to your project's directory and run the following command:

```
git init
```

This command creates a new Git repository in your current directory and initializes it with default settings.

2. Git Add

Once you've initialized a new Git repository, you need to tell Git which files to track. To do this, use the git add command:

```
git add <filename>
```

This command stages the specified file for the next commit. You can also use the git add command with a wildcard to add multiple files at once:

```
git add .
```

This command stages all files in your current directory for the next commit.

3. Git Commit

After you've staged your changes using git add, you need to create a new commit to save those changes to your Git repository. To do this, use the git commit command:

```
git commit -m "follow inkinsight"
```

This command creates a new commit with the *changes you've staged using git add. The -m flag allows you to specify a commit message* that describes the changes you've made.

4. Git Status

To see the current status of your Git repository, use the git status command:

```
git status
```

This command *shows you which files have been modified, staged, or committed since your last commit.*

5. Git Log

To view a history of all the commits in your Git repository, use the git log command:

```
git log
```

This command *shows you a list of all the commits in your repository, including the commit message, author, and date.*

6. Git Branch

To create a new branch in your Git repository, use the git branch command:

```
git branch <branch-name>
```

This command creates a new branch with the specified name. You can then switch to this branch using the git checkout command.

7. Git Checkout

To switch to a different branch in your Git repository, use the git checkout command:

```
git checkout <branch-name>
```

This command switches your working directory to the specified branch. *If the branch doesn't exist yet, you'll need to create it using the git branch command first.*

8. Git Merge

To merge changes from one branch into another, use the git merge command:

```
git merge <branch-name>
```

This command combines the changes from the specified branch into your current branch. If there are conflicts between the two branches, Git will prompt you to resolve them before the merge can be completed.

9. Git Pull

To update your local repository with changes from a remote repository, use the git pull command:

```
git pull <remote> <branch-name>
```

This command *fetches the latest changes from the specified remote repository and merges them into your current branch.*

10 . Git Push

To push your changes to a remote repository, use the git push command:

```
git push <remote> <branch-name>
```

This command sends your commits to the specified remote repository and updates it with your changes. *If you're pushing changes to a remote repository for the first time, you'll need to use the -u flag to set the upstream branch:*

```
git push -u <remote> <branch-name>
```

This command sets the upstream branch to the specified remote repository and branch, *so you can use git pull and git push without specifying the remote and branch every time.*

11. Git Clone

To download a copy of a remote repository to your local machine, use the git clone command:

```
git clone <repository-url>
```

12 . Git Remote

```
# Add a new remote
git remote add origin https://github.com/user/repo.git

# List all remotes
```

```
git remote -v


# Remove a remote
git remote remove origin
```

13. Git Revert

The git revert command is used to create a new commit that undoes the changes from a previous commit. It's a safe way to undo changes because it doesn't modify the commit history.

```
git revert [commit-hash]
```

14 . Git Rebase

The git rebase command is a powerful tool in Git that allows you to move or combine a sequence of commits to a new base commit. It's often used to maintain a clean and linear commit history. There are a few common scenarios where git rebase is useful, including integrating changes from one branch into another, cleaning up your commit history, and updating feature branches.

To rebase your current branch onto another branch, you can use:

```
git rebase [branch]
```

Example:

Assume you have two branches: main and feature-branch.

1. Switch to the feature-branch:

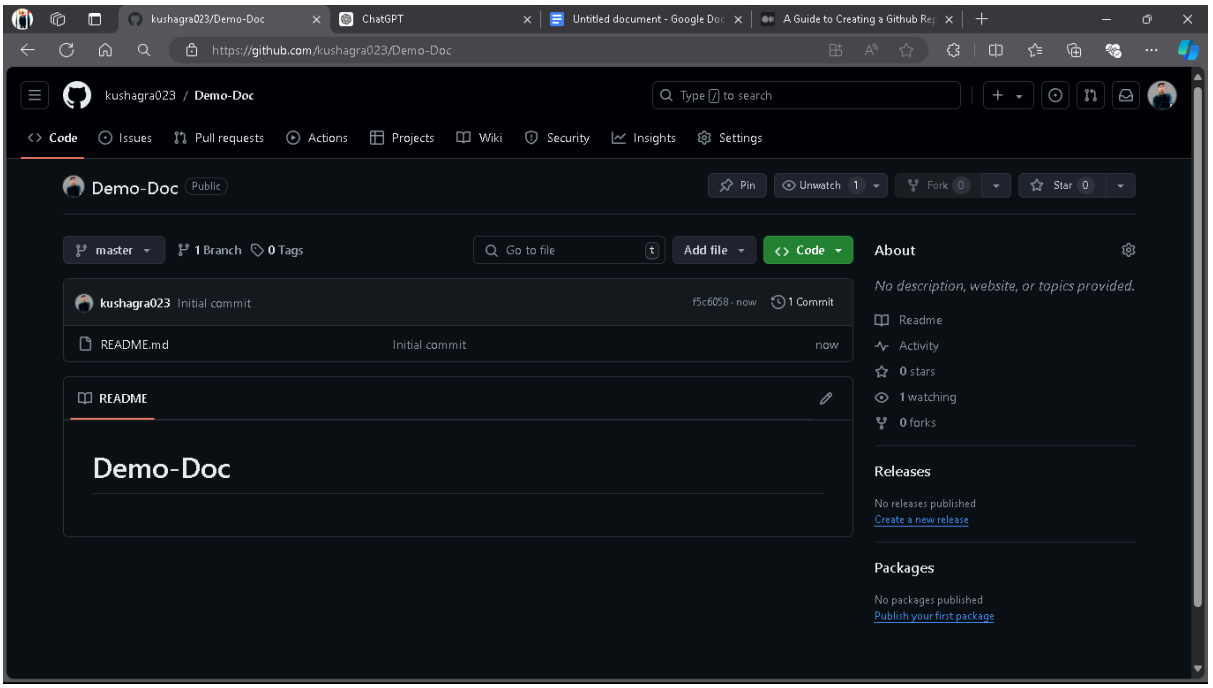
```
git checkout feature-branch
```

2. Rebase feature-branch onto main:

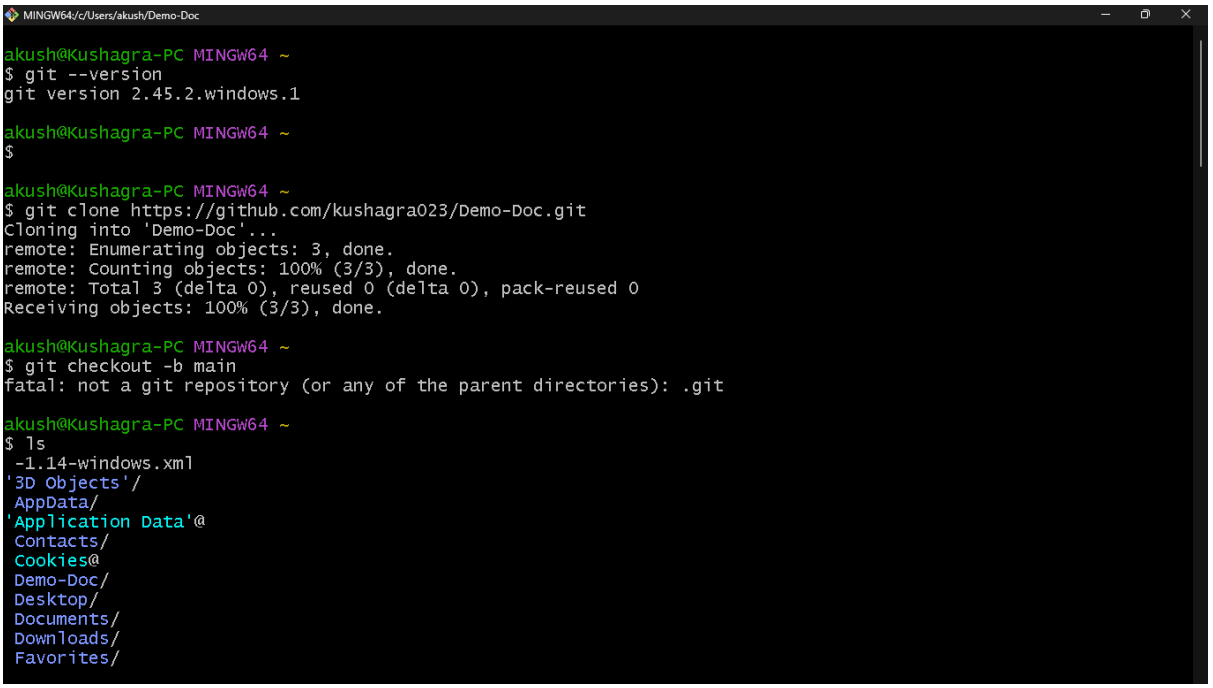
git rebase main

This will move all commits from feature-branch onto the tip of the main branch, effectively incorporating the latest changes from main into feature-branch.

After Creation Repo It Looks like This



Performing Some Basic Cmmand - Cloned the repo and Created a branch then pushed the code



```
MINGW64:/c/Users/akush/Demo-Doc
'Saved Games'/
 Searches/
 SendTo@
'Start Menu'@
 Templates@
 Untitled.ipynb
 Videos/
'VirtualBox VMs'/
 apache-maven.zip
 argo-cd/
 crd.yaml
 javacode-backend/
 ntuser.dat.LOG1
 ntuser.dat.LOG2
 ntuser.ini
 windowscicd/

akush@Kushagra-PC MINGW64 ~
$ cd Demo-Doc/

akush@Kushagra-PC MINGW64 ~/Demo-Doc (master)
$

akush@Kushagra-PC MINGW64 ~/Demo-Doc (master)
$ ls
README.md

akush@Kushagra-PC MINGW64 ~/Demo-Doc (master)
$ git checkout -b main
Switched to a new branch 'main'

akush@Kushagra-PC MINGW64 ~/Demo-Doc (main)
$ git add .

akush@Kushagra-PC MINGW64 ~/Demo-Doc (main)
$ git commit -m "code push to main"
On branch main
nothing to commit, working tree clean
```

```
MINGW64:/c/Users/akush/Demo-Doc
    --[no-]thin           use thin pack
    --[no-]receive-pack <receive-pack>
                          receive pack program
    --[no-]exec <receive-pack>
                          receive pack program
    -u, --[no-]set-upstream
                          set upstream for git pull/status
    --[no-]progress       force progress reporting
    --[no-]prune          prune locally removed refs
    --no-verify           bypass pre-push hook
    --verify              opposite of --no-verify
    --[no-]follow-tags    push missing but relevant tags
    --[no-]signed[=(yes|no|if-asked)]
                          GPG sign the push
    --[no-]atomic         request atomic transaction on remote side
    -o, --[no-]push-option <server-specific>
                          option to transmit
    -4, --ipv4            use IPv4 addresses only
    -6, --ipv6            use IPv6 addresses only


akush@Kushagra-PC MINGW64 ~/Demo-Doc (main)
$ git push origin main
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'main' on GitHub by visiting:
remote:      https://github.com/kushagra023/Demo-Doc/pull/new/main
remote:
To https://github.com/kushagra023/Demo-Doc.git
 * [new branch]      main -> main

akush@Kushagra-PC MINGW64 ~/Demo-Doc (main)
$ |
```