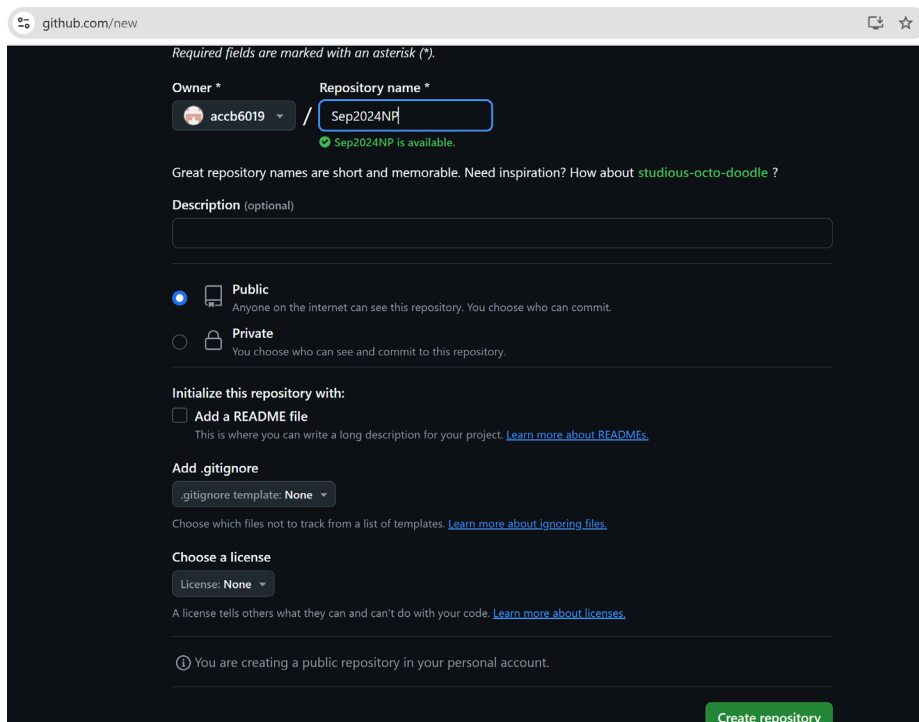
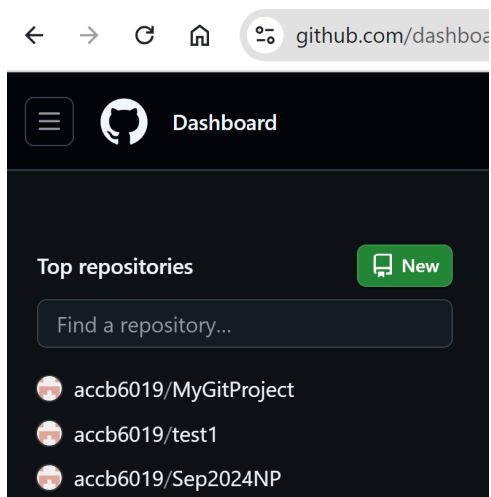


## Git and Git Hub:

### Login to your GitHub and create a Repository



The screenshot shows the GitHub 'new' repository page. At the top, the browser address bar shows 'github.com/new'. The page has a dark theme. A note at the top says 'Required fields are marked with an asterisk (\*)'. The 'Owner' field is set to 'accb6019'. The 'Repository name' field contains 'Sep2024NP' and has a green checkmark with the text 'Sep2024NP is available.' below it. A suggestion 'Great repository names are short and memorable. Need inspiration? How about [studious-octo-doodle](#)?' is shown. The 'Description (optional)' field is empty. The 'Visibility' section has 'Public' selected with a radio button and a lock icon, and 'Private' is unselected. Below this, the 'Initialize this repository with:' section has 'Add a README file' unselected. The 'Add .gitignore' section has a dropdown menu set to 'None'. The 'Choose a license' section has a dropdown menu set to 'None'. At the bottom, a note says 'You are creating a public repository in your personal account.' and a green 'Create repository' button is on the right.



```
mkdir MyGit1
```

```
cd MyGit1
```

```
dir
```

```
dir /a
```

```
git config --global user.name "Instructor"
```

```
git config --global user.email "dummy.demo4class@gmail.com"
git init
dir
dir /a
git status
dir
git add Sample1.txt
git commit -m "V1"
git branch team1
git checkout team1
git add Sample1.txt
git commit -m "Team1 update"
git checkout master
git merge team1
git remote add origin https://github.com/accb6019/Sep2024NP.git
git push -u origin master
doskey /history > C:\Users\n.kaliyamurthy\Documents\MyGit1\history2.txt
```

### Objective:

1. Create a folder called “MyGitFolder” on your workdrive Location.
2. Create some files and manage them using Git (with branches).
3. Upload the repository to GitHub.

#### 1. Install Git and Verify Installation

You’ve already installed Git version 2.47.0.windows.

Open your cmd prmt and type: git --version

```
C:\Users\n.kaliyamurthy>git --version
git version 2.47.0.windows.1
```

## 2. Create a Folder for Your Project

Create a working folder: "MyGitFolder"

```
mkdir C:\MyGit
```

## 3. Initialize Git in the Folder

```
C:\Users\n.kaliyamurthy\Documents\MyGit>git init
Initialized empty Git repository in C:/Users/n.kaliyamurthy/Documents/MyGit/.git/
```

What just happened?

You've turned your "MyGitFolder" folder into a Git repository.

Git created a hidden ".git" folder inside "MyGitFolder" that will track changes to files.

## 4. Create Files and Stage Them for Commit

a. Open Notepad/Doc/Working File and add inputs:

b. In a notepad, I wrote a text : Hello, this is my first Git project! And saved the file as "sample.txt" inside "MyGitFolder".

c. Add more files if you'd like, such as "notes.txt", "code.py", etc.

## 5. Track and Commit Changes with Git

a. Check the status of your files by running:

```
C:\Users\n.kaliyamurthy\Documents\MyGit>git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        history.txt

nothing added to commit but untracked files present (use "git add" to track)
```

You should see as an untracked file.

b. Let's tell Git to track it:

```
C:\Users\n.kaliyamurthy\Documents\MyGit>git add history.txt
```

We've staged the file. It's like saying, "Hey Git, I want to track changes to this file."

c. Now, commit the file to the repository:

```
C:\Users\n.kaliyamurthy\Documents\MyGit>git commit -m "Added the 1st version of history.txt"
[master (root-commit) 2e4d352] Added the 1st version of history.txt
1 file changed, 11 insertions(+)
create mode 100644 history.txt
```

A commit is like a snapshot of your project at a certain point in time.

The -m flag allows you to add a message describing the changes you've made.

## 6. Creating a New Branch

Branches are one of Git's coolest features. They allow you to create different versions of your project without affecting the main code.

Create a New Branch called Feature and Switch to the Feature

```
C:\Users\n.kaliyamurthy\Documents\MyGit>git branch feature  
  
C:\Users\n.kaliyamurthy\Documents\MyGit>git checkout feature  
Switched to branch 'feature'
```

Now, let's make changes on this new branch! Open `the doc` and add contents: Save the file, then stage and commit the changes:

```
C:\Users\n.kaliyamurthy\Documents\MyGit>git add history.txt  
  
C:\Users\n.kaliyamurthy\Documents\MyGit>git commit -m "Added new feature"  
[feature 3af05e3] Added new feature  
1 file changed, 3 insertions(+)
```

You can always switch back to the `main` branch:

## 7. Merging Branches

After completing work on your feature, you'll want to merge it into the main codebase.

```
C:\Users\n.kaliyamurthy\Documents\MyGit1>git checkout master  
Switched to branch 'master'  
  
C:\Users\n.kaliyamurthy\Documents\MyGit1>git merge team1  
Updating f317efd..d06ecfd  
Fast-forward  
 Sample1.txt | 3 ++-  
1 file changed, 2 insertions(+), 1 deletion(-)
```

## 8. Pushing to GitHub

Now that we have our local repository set up and changes committed, let's push it to GitHub.

- a. Login to your GitHub account.
- b. Create a new repository
  - i. Go to Repositories and click New.
    1. Name the repository, Leave it as **\*\*Public\*\*** (or Private if you prefer) and Click **\*\*Create Repository\*\***.
- c. Now, you can connect our local Git repository to GitHub.

In your command prompt, run

```
C:\Users\n.kaliyamurthy\Documents\MyGit1>git remote add origin https://github.com/accb6019/Sep2024NP.git
```

Replace YourUsername with your actual GitHub username.

Finally, push your code to GitHub:

```
C:\Users\n.kaliyamurthy\Documents\MyGit1>git push -u origin master
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 20 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (6/6), 484 bytes | 242.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/accb6019/Sep2024NP.git
 * [new branch]      master -> master
branch 'master' set up to track 'origin/master'.
```

If Git asks for your username and password, provide your GitHub credentials (or use an access token if you have set that up).

Head over to your GitHub account and check the repository you just created.

## Reflections

Git as a Distributed Version Control System allows you to track changes, create branches, and collaborate with others efficiently.

Branching is key to working on multiple features without affecting the main project.

Merging combines different branches and helps integrate features.

GitHub is a platform for sharing your work with the world!