LOCUS 2026

22nd National Technological Festival

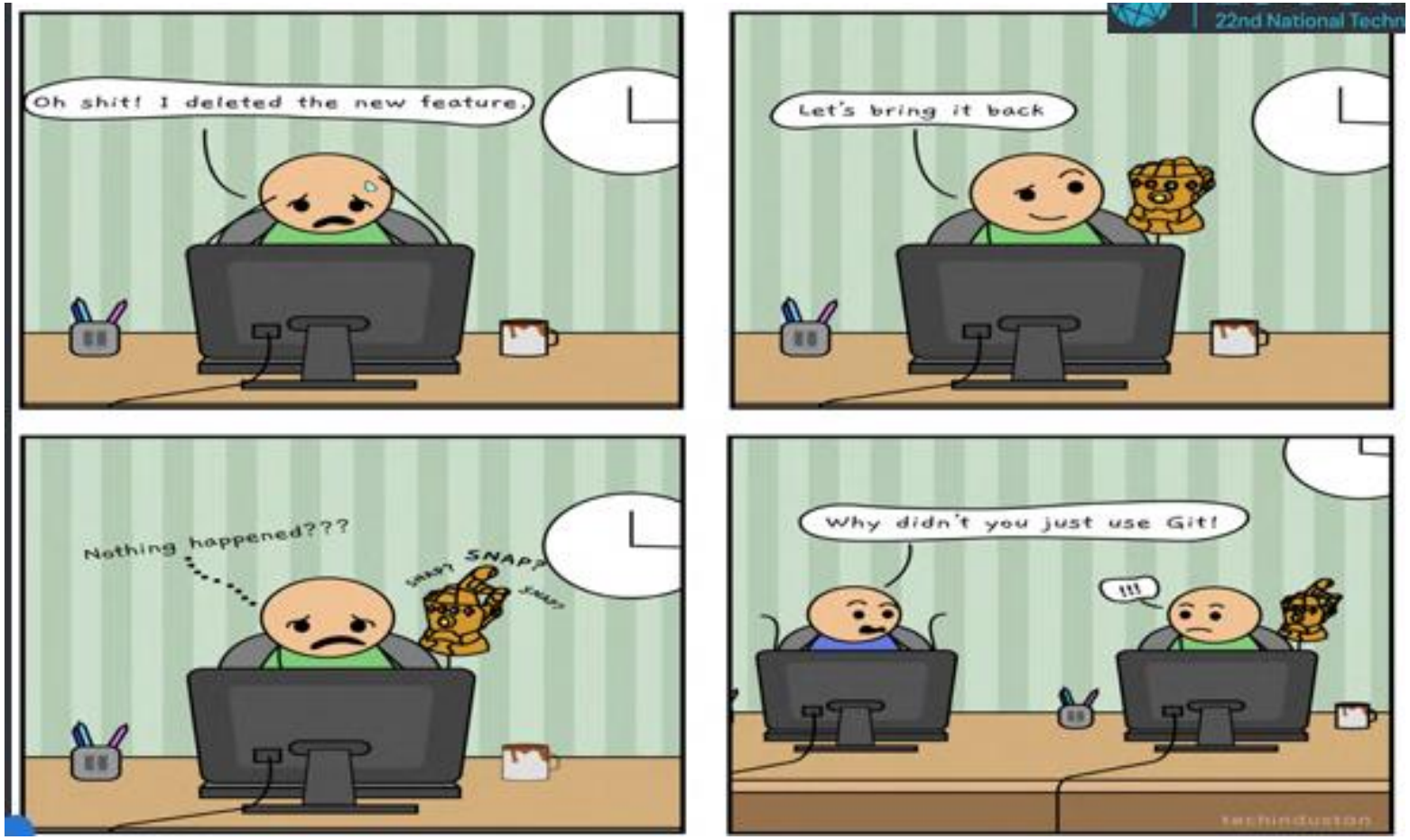
Git & Github

Software Fellowship Day 5

Objectives

- Initialize, stage, commit, and push with Git.
- Branch, merge, and resolve conflicts smoothly.
- Open pull requests and run concise code reviews.
- Connect front-end to back-end via fetch/AJAX calls.
- LAB: build a login page using the GitHub PR workflow.

Need of Version Control



Need of Version Control

managing changes: versions of files over time

collaborate: multiple users to on a project

history: track of the changes made

revert: go back to previous versions conflict management: multiple users working on same files

branching: multiple features at the same time

Setting up Git

Checking installation of git

```
$ git version
```

Setting up Git

Configuring Git with username and email

```
git config --global user.name “[firstname lastname]”
```

set a name that is identifiable for credit when review version history

```
git config --global user.email “[valid-email]”
```

set an email address that will be associated with each history marker

Example:

```
$ git config --global user.name "John Doe"
$ git config --global user.email johndoe@example.com
```

Checking your settings

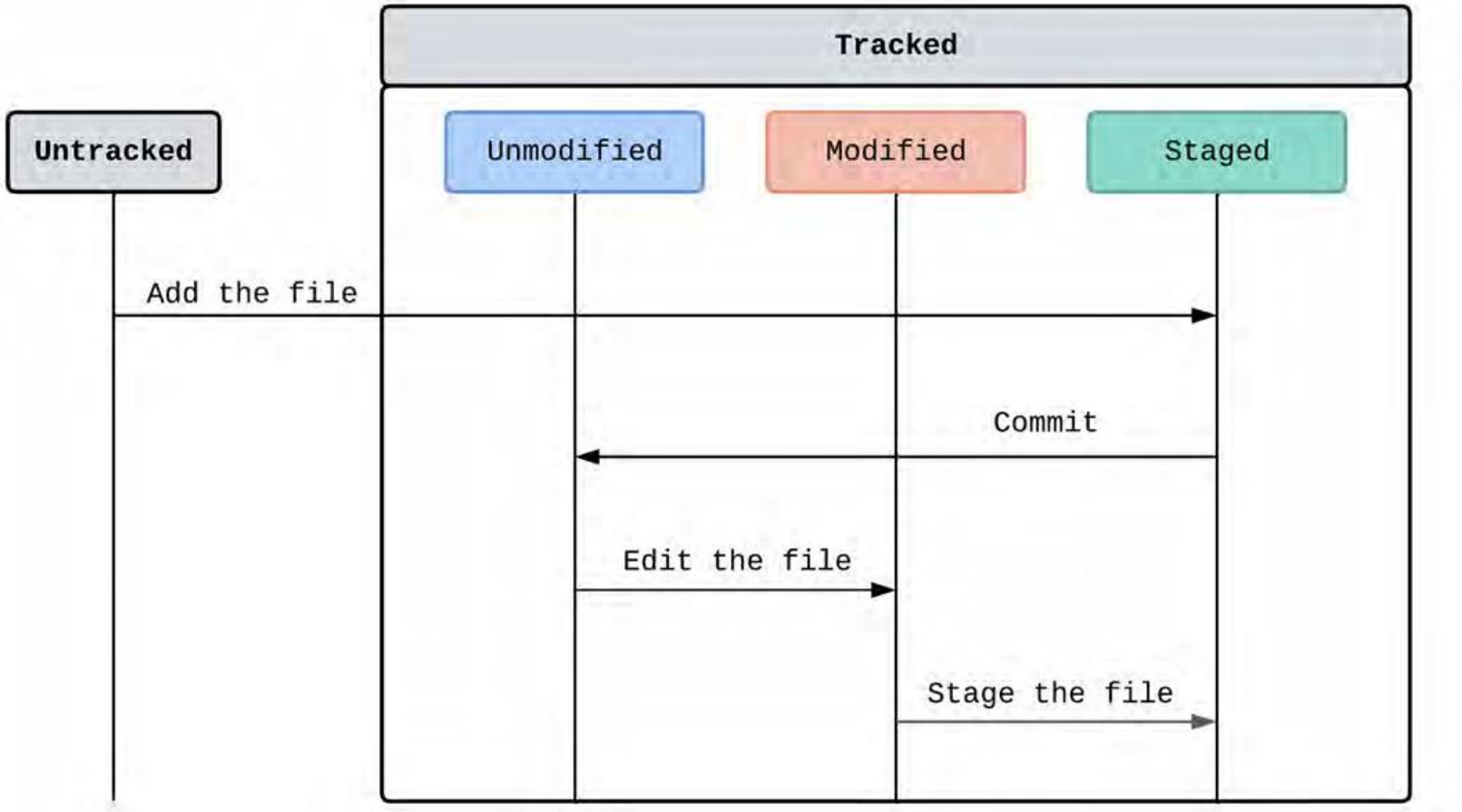

```
$ git config --list
```

list all the settings Git can find at that point

```
$ git config user.name
```

check specific key value for a setting

Lifecycle of status of file



Setting up Git

Creating a new git repository

```
git init
```

initialize an existing directory as a Git repository

or you can clone a repository but we'll get into it later.

- A hidden folder named **.git** is created which contains all the information of the repository such as change history, settings, compressed version of each file, etc.

Tracking your files


```
$ git add <filepath>
```

- Stage ***changes*** in the specified file(s), preparing them for the next commit
- Place the ***changes*** to the so called "staging area"

Example:

```
$ git add hangman.c
```

Batch Staging


```
$ git add .
```

- Stage ***changes*** in all the files in the current directory and its subdirectories.
- where **.** specifies the current working directory

Checking the status of your files


```
$ git status
```

- Shows the current status of your git repo
- Displays the information about tracked and untracked files in the current working directory
- Tracked files are files that Git is aware of and are already being version controlled.
- Untracked files are files that are not yet added to the Git repository.

Committing


```
$ git branch -M <new_name>
```

- change the branch

Committing


```
$ git commit –m ”Commit Message”
```

- used to save your changes to the local repository

Viewing Commit History

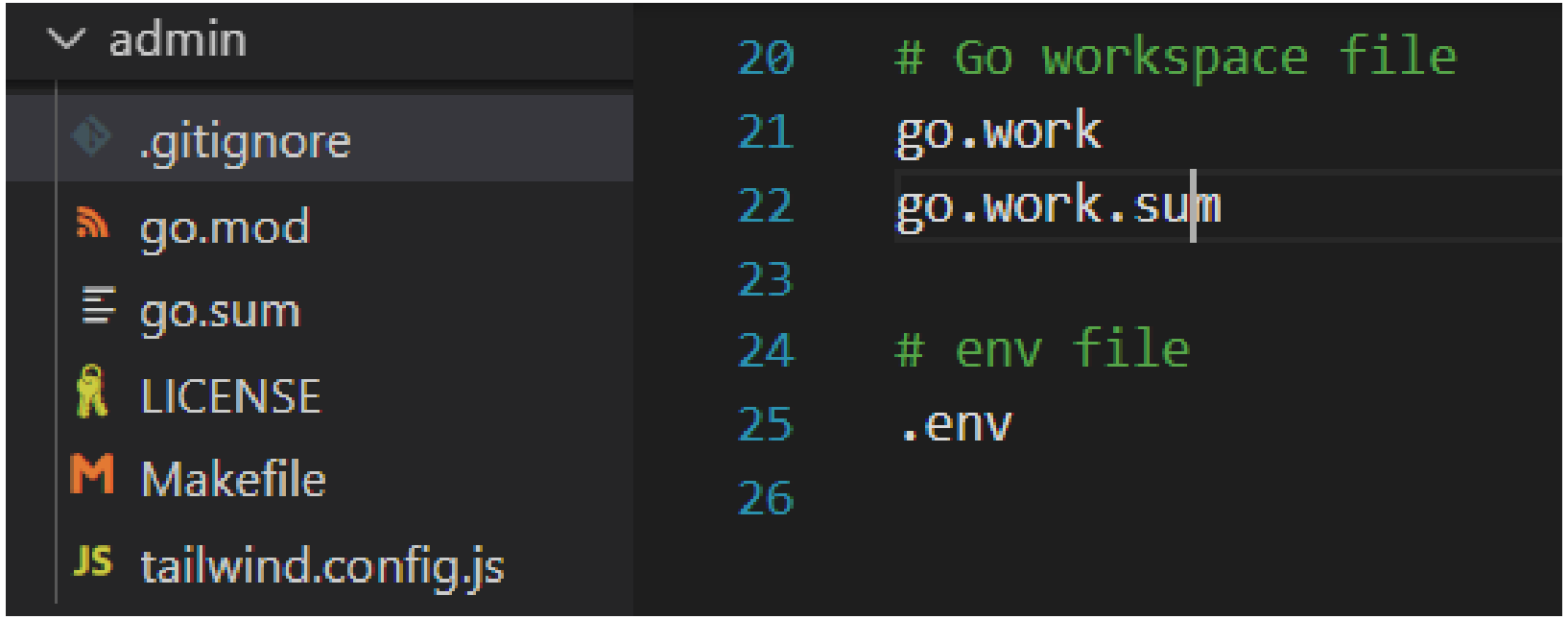

```
$ git log
```

- Displays the list of commits in reverse chronological order, showing the latest commits first
- Each commit in the log includes information such as the commit hash (SHA-1 checksum), author name, author email, commit date, the commit message, and the commit description (if any).
- Pressing the **Enter** key scrolls down through the log, displaying more commits if available. Press **q** to exit the log view.

.gitignore

- A .gitignore file is used in Git to specify files or directories that should be ignored by Git when committing changes.
- This is particularly useful for excluding sensitive information, temporary files, or build artifacts from your repository.

.gitignore


admin
.gitignore
go.mod
go.sum
LICENSE
Makefile
tailwind.config.js
20 # Go workspace file
21 go.work
22 go.work.sum
23
24 # env file
25 .env
26

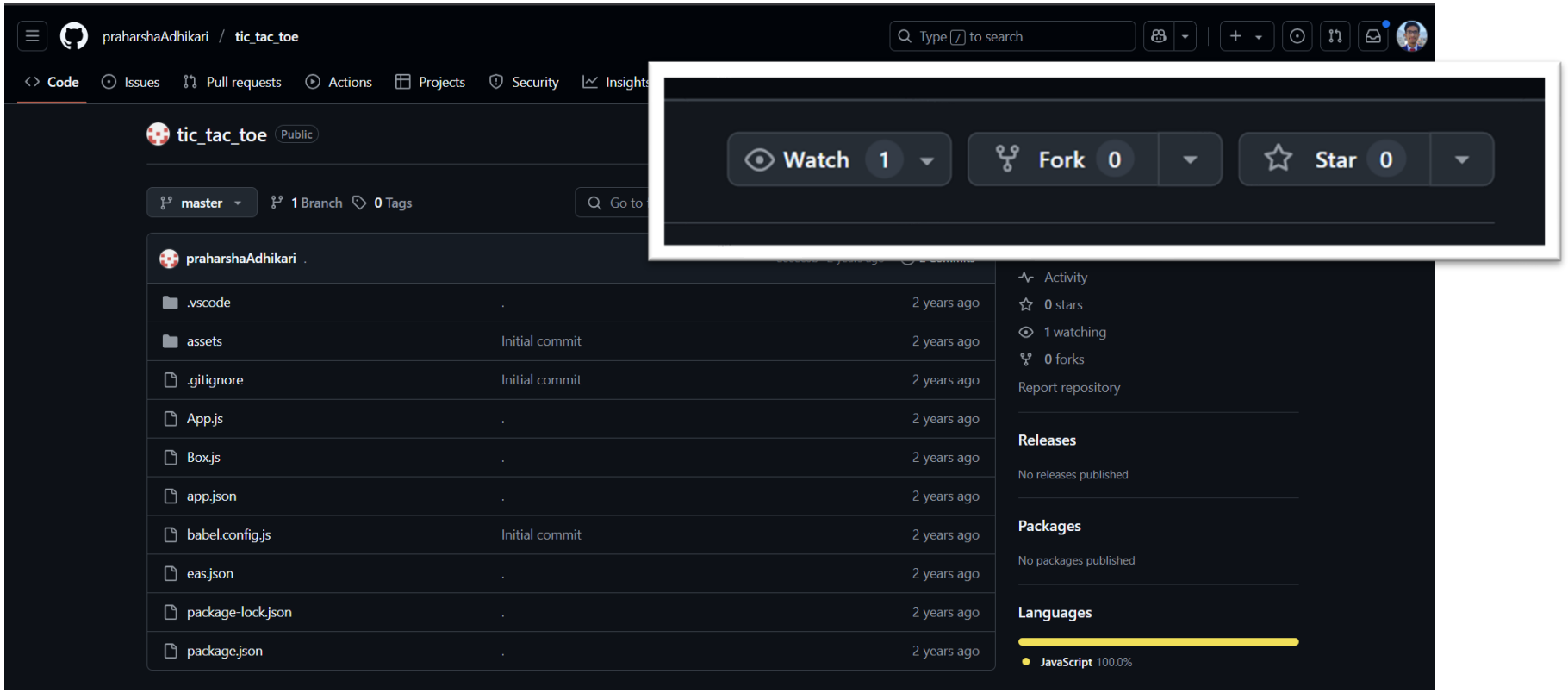
Pushing to remote repo


```
$ git push origin <branch_name>
```

- git push is used to send your local commits to a remote repository like GitHub.
- It updates the remote branch with your local branch’s changes.
- If someone else has pushed changes before you, you may need to pull and merge first.
- It's commonly used after you've committed changes locally and want to share them or back them up

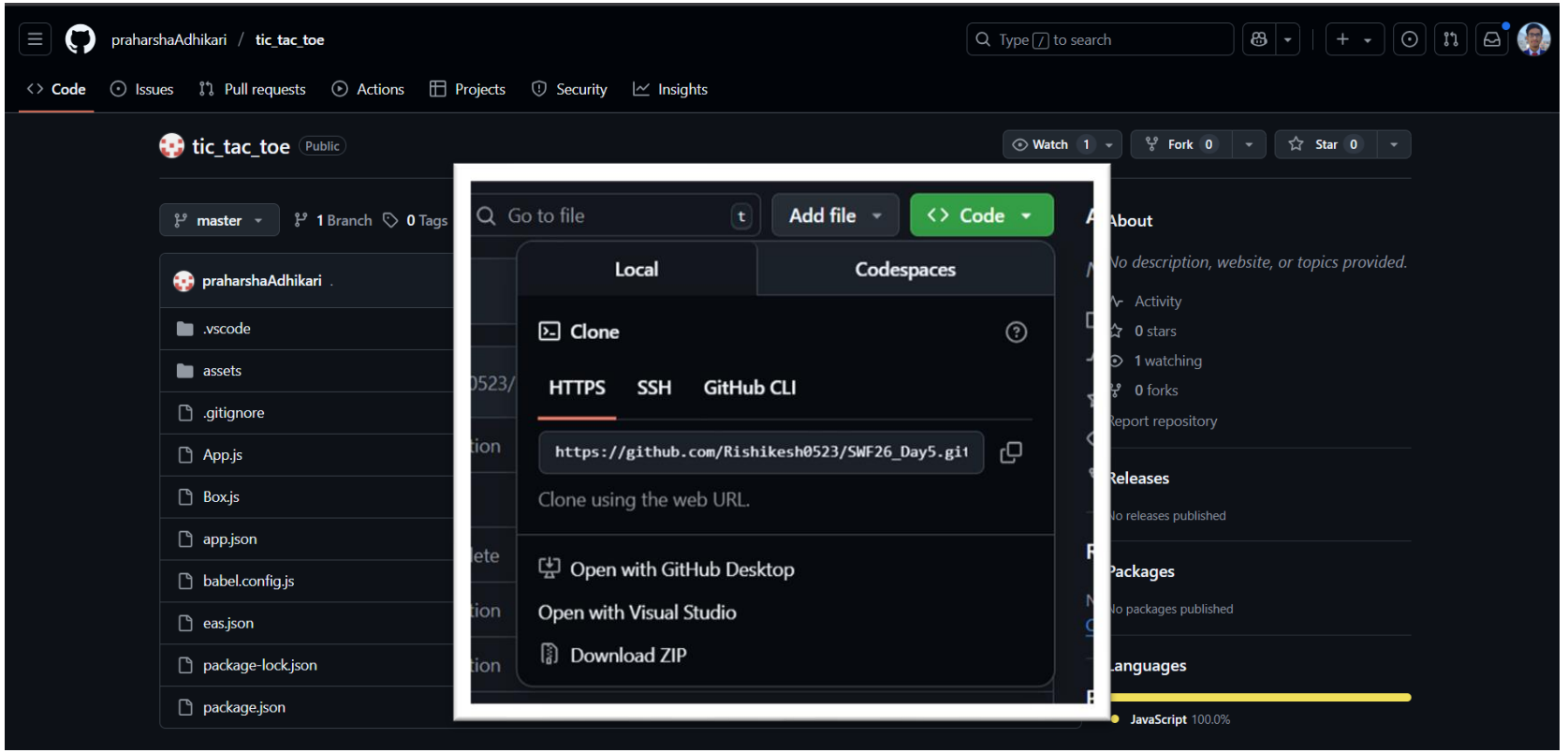
Forking

- Creating a replica of other's remote repository in your own remote repository



Cloning

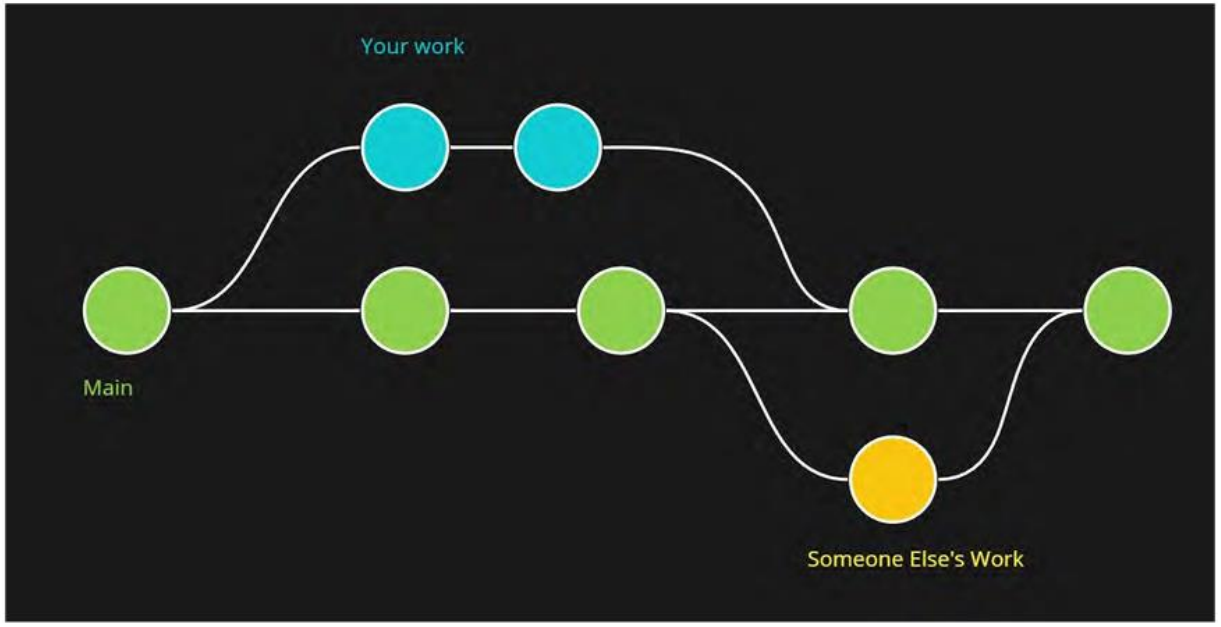
- Creating a replica of remote repository in your own local repository



Cloning


```
$ git clone <url>
```

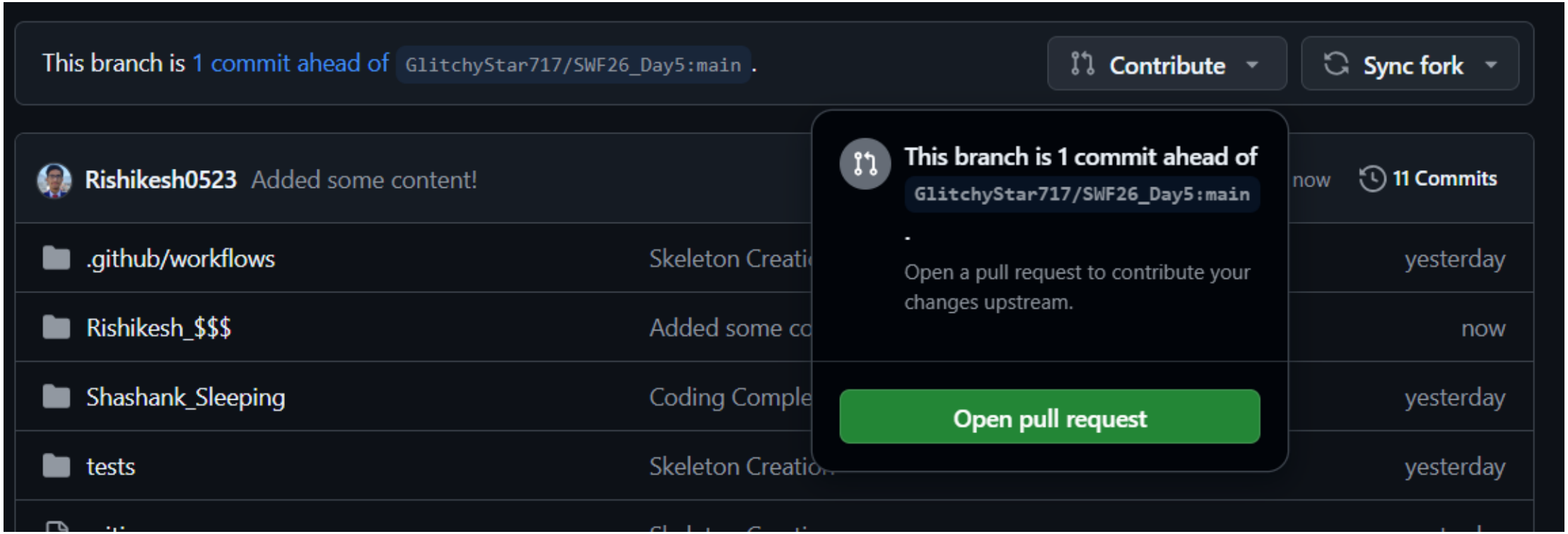
Branching and Merging




```
$ git checkout –b
<branch_name>
```

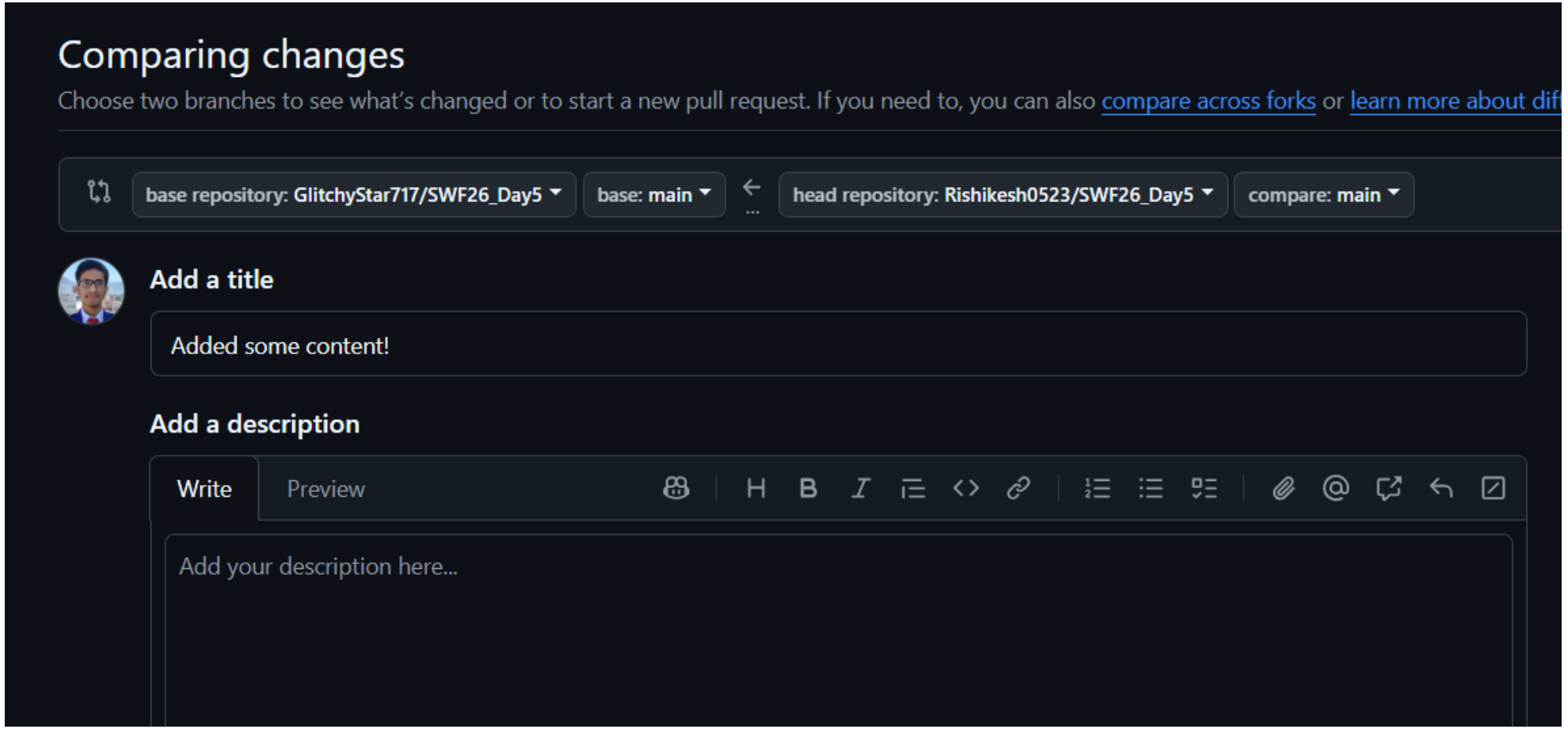
```
$ git merge
<branch_name>
```

Creating a Pull Request

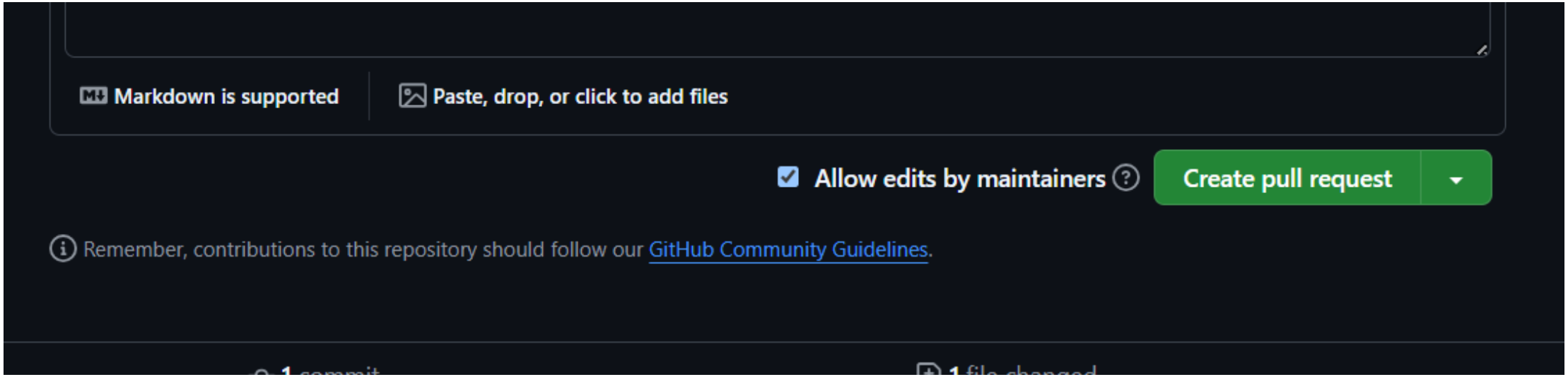


Creating a Pull Request

LOCUS 2026
22nd National Technological Festival



Creating a Pull Request



Linking frontend & backend (AJAX/fetch)

https://github.com/Rishikesh0523/ajax_demo

Lab: Team build login page with backend API integration via GitHub PR flow

https://github.com/GlitchyStar717/SWF26_Day5

Thank You!