NAME:MIDHUNA R

ROLLNO:22BCS063

Experiment No. 2

Title: Implement GitHub Operations using Git.

Objective:

The objective of this experiment is to guide you through the process of using Git commands to interact with GitHub, from cloning a repository to collaborating with others through pull requests.

Introduction:

GitHub is a web-based platform that offers version control and collaboration services for software development projects. It provides a way for developers to work together, manage code, track changes, and collaborate on projects efficiently. GitHub is built on top of the Git version control system, which allows for distributed and decentralised development.

Key Features of GitHub:

- **Version Control:** GitHub uses Git, a distributed version control system, to track changes to source code over time. This allows developers to collaborate on projects while maintaining a history of changes and versions.
- **Repositories**: A repository (or repo) is a collection of files, folders, and the entire history of a project. Repositories on GitHub serve as the central place where code and project-related assets are stored.
- **Collaboration:** GitHub provides tools for team collaboration. Developers can work together on the same project, propose changes, review code, and discuss issues within the context of the project.
- **Pull Requests:** Pull requests (PRs) are proposals for changes to a repository. They allow developers to submit their changes for review, discuss the changes, and collaboratively improve the code before merging it into the main codebase.
- **Issues and Projects:** GitHub allows users to track and manage project-related issues, enhancements, and bugs. Projects and boards help organize tasks, track progress, and manage workflows.
- **Forks and Clones:** Developers can create copies (forks) of repositories to work on their own versions of a project. Cloning a repository allows developers to create a local copy of the project on their machine.
- **Branching and Merging:** GitHub supports branching, where developers can create separate lines of development for features or bug fixes. Changes made in branches can be merged back into the main codebase.
- **Actions and Workflows:** GitHub Actions enable developers to automate workflows, such as building, testing, and deploying applications, based on triggers like code pushes or pull requests.

- **GitHub Pages:** This feature allows users to publish web content directly from a GitHub repository, making it easy to create websites and documentation for projects.

Benefits of Using GitHub:

- **Collaboration:** GitHub facilitates collaborative development by providing tools for code review, commenting, and discussion on changes.
- **Version Control:** Git's version control features help track changes, revert to previous versions, and manage different branches of development.
- **Open Source:** GitHub is widely used for hosting open-source projects, making it easier for developers to contribute and for users to find and use software.
- **Community Engagement:** GitHub fosters a community around projects, enabling interaction between project maintainers and contributors.
- **Code Quality:** The code review process on GitHub helps maintain code quality and encourages best practices.
- **Documentation:** GitHub provides a platform to host project documentation and wikis, making it easier to document codebases and project processes.
- **Continuous Integration/Continuous Deployment (CI/CD):** GitHub Actions allows for automating testing, building, and deploying applications, streamlining the development process.

Materials:

- Computer with Git installed (https://git-scm.com/downloads)
- GitHub account (https://github.com/)
- Internet connection

Experiment Steps:

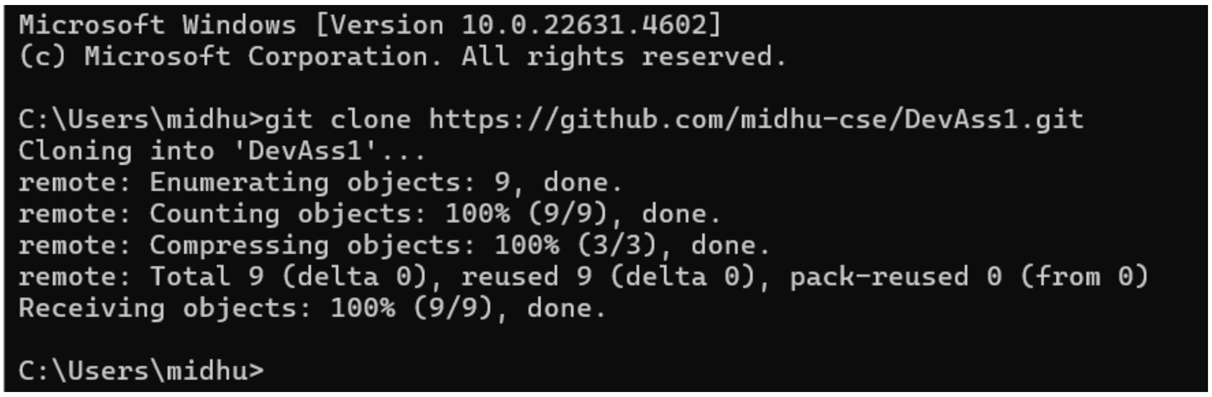
Step 1: Cloning a Repository

- Sign in to your GitHub account.
- Find a repository to clone (you can use a repository of your own or any public repository).
- Click the "Code" button and copy the repository URL.
- Open your terminal or command prompt.
- Navigate to the directory where you want to clone the repository.
- Run the following command:

git clone <repository_url>

- Replace <repository_url> with the URL you copied from GitHub.

- This will clone the repository to your local machine.



```
Microsoft Windows [Version 10.0.22631.4602]
(c) Microsoft Corporation. All rights reserved.

C:\Users\midhu>git clone https://github.com/midhu-cse/DevAss1.git
Cloning into 'DevAss1'...
remote: Enumerating objects: 9, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 9 (delta 0), reused 9 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (9/9), done.

C:\Users\midhu>
```

Step 2: Making Changes and Creating a Branch

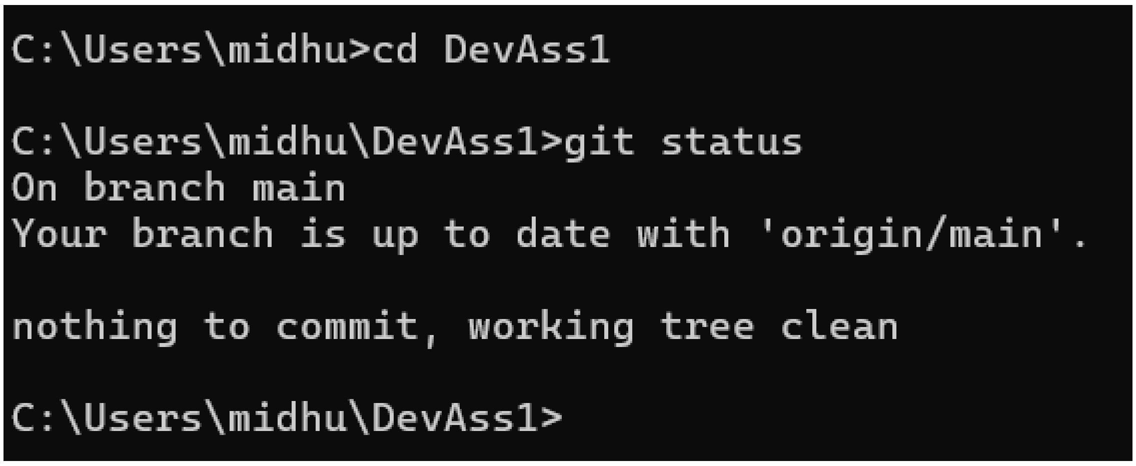
Navigate into the cloned repository:

cd <repository_name>

- Create a new text file named "example.txt" using a text editor.
- Add some content to the "example.txt" file.
- Save the file and return to the command line.
- Check the status of the repository:

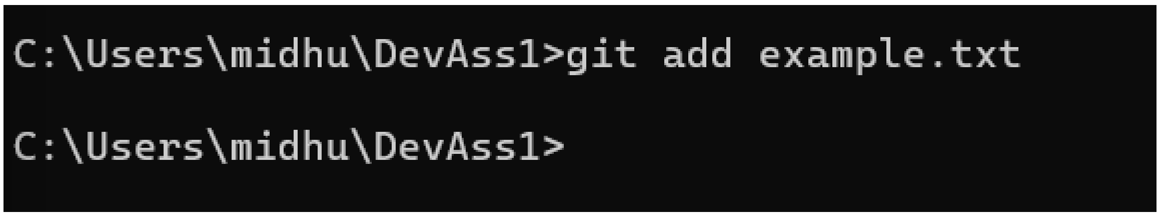
git status

- Stage the changes for commit:



```
C:\Users\midhu>cd DevAss1

C:\Users\midhu\DevAss1>git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

C:\Users\midhu\DevAss1>
```

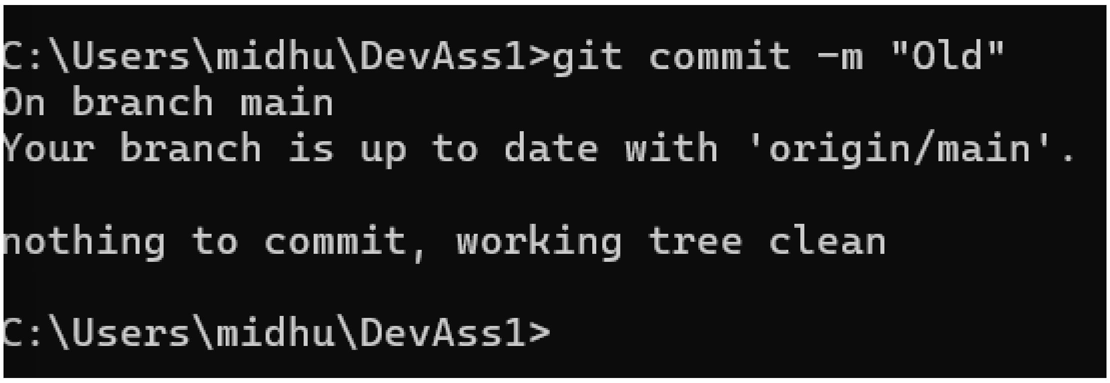
git add example.txt



```
C:\Users\midhu\DevAss1>git add example.txt

C:\Users\midhu\DevAss1>
```

- Commit the changes with a descriptive message:

git commit -m "Add content to example.txt"



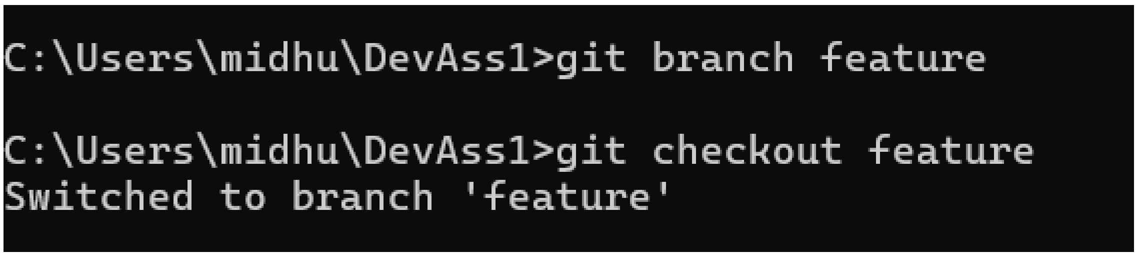
```
C:\Users\midhu\DevAss1>git commit -m "Old"
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

C:\Users\midhu\DevAss1>
```

- Create a new branch named "feature":

git branch feature

- Switch to the "feature" branch:

git checkout feature



```
C:\Users\midhu\DevAss1>git branch feature

C:\Users\midhu\DevAss1>git checkout feature
Switched to branch 'feature'
```

Step 3: Pushing Changes to GitHub

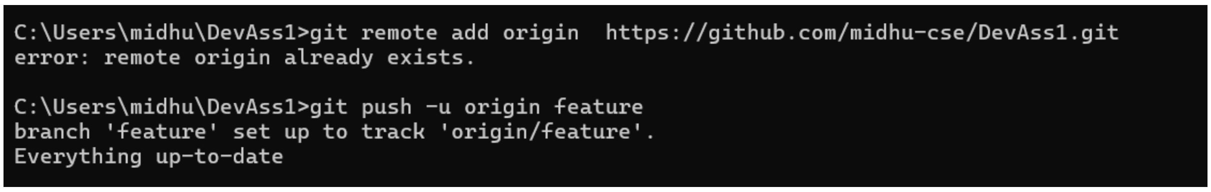
- Add Repository URL in a variable

git remote add origin <repository_url>

- Replace <repository_url> with the URL you copied from GitHub.
- Push the "feature" branch to GitHub:

git push origin feature

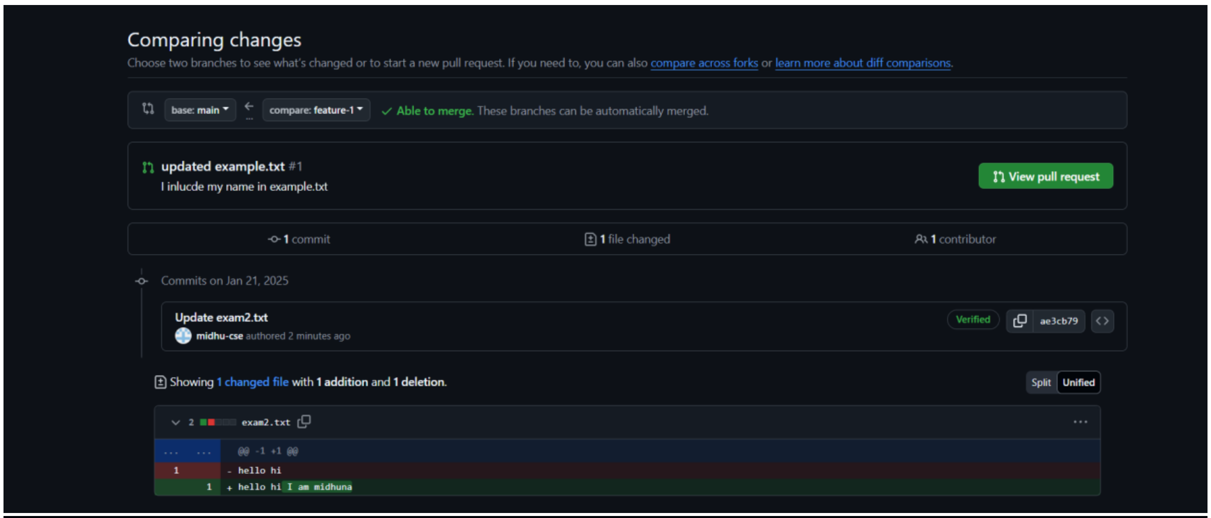
- Check your GitHub repository to confirm that the new branch "feature" is available.



```
C:\Users\midhu\DevAss1>git remote add origin  https://github.com/midhu-cse/DevAss1.git
error: remote origin already exists.

C:\Users\midhu\DevAss1>git push -u origin feature
branch 'feature' set up to track 'origin/feature'.
Everything up-to-date
```

Step 4: Collaborating through Pull Requests

- Create a pull request on GitHub:
- Go to the repository on GitHub.
- Click on "Pull Requests" and then "New Pull Request."
- Choose the base branch (usually "main" or "master") and the compare branch ("feature").
- Review the changes and click "Create Pull Request."
- Review and merge the pull request:
- Add a title and description for the pull request.
- Assign reviewers if needed.
- Once the pull request is approved, merge it into the base branch.



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 about diff comparisons.

base: main ← compare: feature-1 ✓ Able to merge. These branches can be automatically merged.

updated example.txt #1
I inlucde my name in example.txt

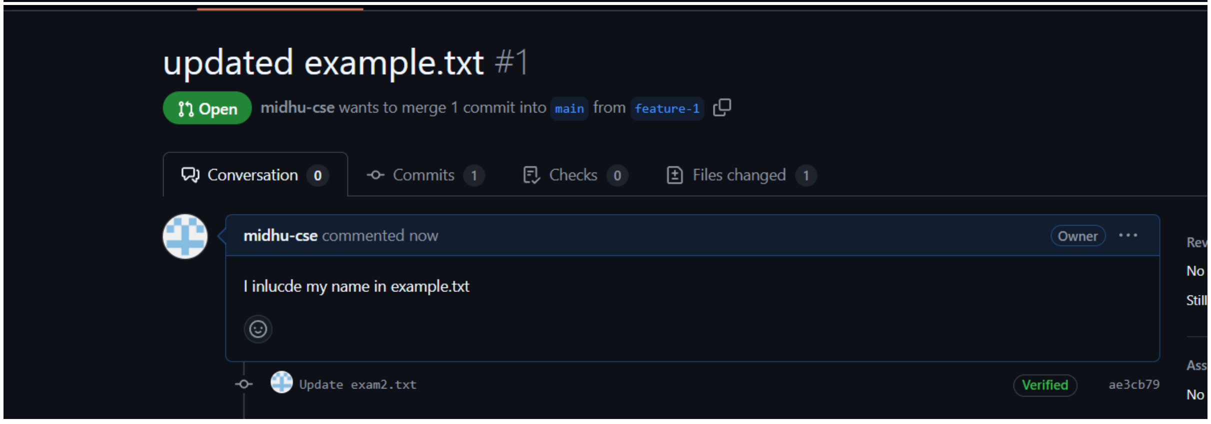
1 commit | 1 file changed | 1 contributor

Commits on Jan 21, 2025

Update exam2.txt
midhu-cse authored 2 minutes ago

Showing 1 changed file with 1 addition and 1 deletion.

```
2 exam2.txt
@@ -1 +1 @@
1     - hello hi
    1 + hello hi I am midhuna
```



updated example.txt #1

midhu-cse wants to merge 1 commit into main from feature-1

midhu-cse commented now

I inlucde my name in example.txt

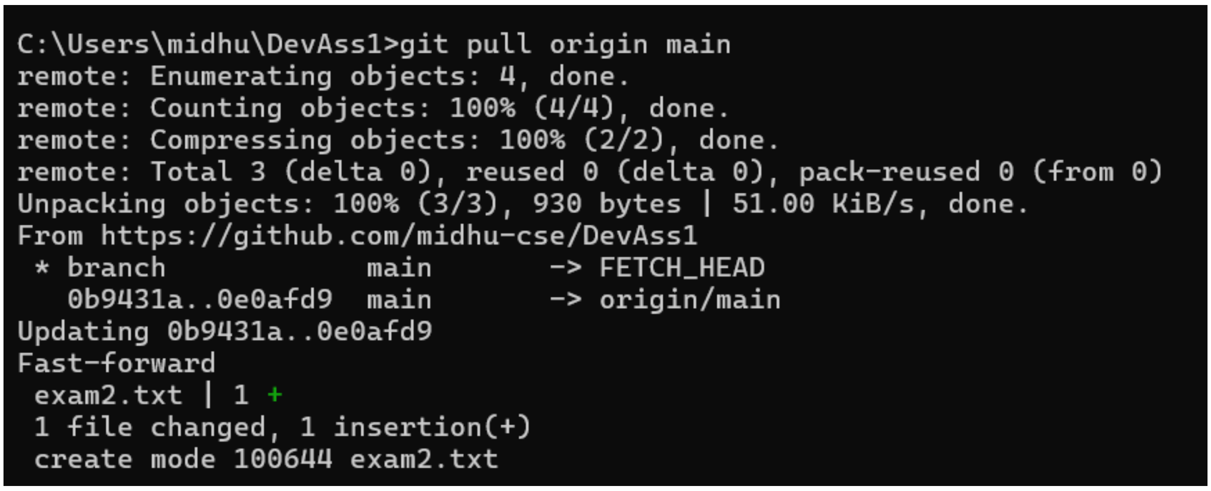
Update exam2.txt

Step 5: Syncing Changes

- After the pull request is merged, update your local repository:

git checkout main

git pull origin main



```
C:\Users\midhu\DevAss1>git pull origin main
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (3/3), 930 bytes | 51.00 KiB/s, done.
From https://github.com/midhu-cse/DevAss1
 * branch            main       -> FETCH_HEAD
   0b9431a..0e0afd9  main       -> origin/main
Updating 0b9431a..0e0afd9
Fast-forward
 exam2.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 exam2.txt
```

Conclusion:

This experiment provided you with practical experience in performing GitHub operations using Git commands. You learned how to clone repositories, make changes, create branches, push changes to GitHub, collaborate through pull requests, and synchronise changes with remote repositories. These skills are essential for effective collaboration and version control in software development projects using Git and GitHub.

Questions:

1. Explain the difference between Git and GitHub.

Aspect	Git	GitHub
Definition	A distributed version control system for tracking code changes.	A web-based platform for hosting and managing Git repositories.
Purpose	Manages code versions locally and facilitates collaboration.	Provides remote storage and collaboration tools for Git repositories.
Location	Operates on your local machine.	Accessible online through GitHub's web interface.
Collaboration	Limited to local or private networks.	Enables global collaboration via pull requests, issues, etc.
Features	Branching, merging, commits, and history tracking.	Additional features like pull requests, issue tracking, and CI/CD integrations.
Hosting	Not a hosting service.	Offers cloud-based hosting for repositories.
Usage	Command-line tool, IDE integrations.	Web interface, desktop apps, and API support.
Examples	Git installed on a developer's system.	Repository hosted on `github.com`.

2. What is a GitHub repository? How is it different from a Git repository?

Aspect	Git Repository	GitHub Repository
Definition	A collection of files and their version history managed by Git, stored locally on your computer.	A Git repository hosted on GitHub's cloud-based platform, accessible online.
Location	Stored locally on the user's machine.	Stored remotely on GitHub servers and accessible via the internet.
Purpose	Tracks changes to code and manages version history locally.	Facilitates collaboration, remote storage, and version control.
Collaboration	Limited to local or network sharing.	Provides tools like pull requests, issue tracking, and branch protection for global collaboration.
Access	Accessed via Git tools (e.g., command line or GUI).	Accessed through GitHub's web interface, APIs, or Git tools.
Visibility	Private to the user unless explicitly shared.	Can be public (visible to everyone) or private (restricted access).
Backup	Requires manual backup or setup for remote storage.	Automatically backed up on GitHub's servers.
Examples	Local repository on your computer (e.g., `my-project.git`).	A repository on GitHub (e.g., `github.com/username/my-project`).

3. Describe the purpose of a README.md file in a GitHub repository.

   A README.md file is a markdown file typically found in GitHub repositories.
   Purpose:

- Provides an introduction and overview of the project.
- Includes setup instructions, usage examples, and project goals.
- Serves as a reference for contributors and users.

4. How do you create a new repository on GitHub? What information is required during the creation process?

   Steps to create a repository:

1. Log into your GitHub account and navigate to your dashboard.
2. Click on the **New Repository** button or the "+" icon.
3. Fill in the following details:
   - **Repository Name**: Unique name for the repository.
   - **Description** (optional): Summary of the project.
   - **Visibility**: Public or private.
   - **Initialize Repository**: Optionally add a README file, .gitignore, or a license.

4. Click **Create Repository**.
5. Define what a pull request (PR) is on GitHub. How does it facilitate collaboration among developers?

   A **pull request** is a proposal to merge changes from one branch into another within a repository.
   Purpose:

- Facilitates collaboration by enabling developers to review and discuss changes before merging.
- Helps maintain code quality through peer reviews.

6. Describe the typical workflow of creating a pull request and having it merged into the main branch.

   - Create a new branch for your feature or bug fix.
   - Make changes and commit them to the new branch.
   - Push the branch to the remote repository on GitHub.
   - Open a pull request from the branch to the main branch.
   - Reviewers review the pull request, suggest changes, or approve it.
   - Once approved, the PR is merged into the main branch.

7. How can you address and resolve merge conflicts in a pull request?

   - **Identify the conflict**: GitHub highlights files with conflicts in the PR.
   - **Fetch latest changes**: Ensure your local branch has the latest updates from the main branch.
   - **Resolve conflicts**: Open conflicting files, decide which changes to keep, and edit the code accordingly.
   - **Mark as resolved**: After resolving conflicts, stage the changes.
   - **Push updates**: Push the resolved branch back to GitHub.

8. Explain the concept of forking a repository on GitHub. How does it differ from cloning a repository?

Aspect	**Forking a Repository**	**Cloning a Repository**
Definition	Creates a personal copy of someone else's repository on your GitHub account.	Downloads a repository (original or forked) to your local machine.
Purpose	To contribute to or modify a project you don’t own, with changes stored in your GitHub account.	To work on a repository locally, whether it's your own or someone else’s.
Location of Copy	Stored in your GitHub account.	Stored on your local computer.

Aspect	Forking a Repository	Cloning a Repository
Connection to Original	Remains linked to the original repository; you can create pull requests to propose changes.	Not directly linked to the original repository after cloning.
Use Case	Used to propose changes to projects you don’t own or to experiment with existing codebases.	Used to work offline or to manage your own repository locally.
Workflow	Requires forking on GitHub first, then optionally cloning the forked repository locally.	Directly clones the repository without creating a new copy on GitHub.