Git and GitHub

Assignment 2A: Create version control account on GitHub and use Git commands to create repository and push your code to GitHub.

What is Git?

Git is a distributed version control system that helps developers track changes in source code efficiently. It enables multiple developers to collaborate on projects while maintaining version history.

Features of Git:

1. **Distributed System:** Every developer has a full copy of the repository, making collaboration more efficient and reducing dependency on a central server.
2. **Branching & Merging:** Developers can create separate branches to work on new features or bug fixes and later merge them into the main project.
3. **Lightweight & Fast:** Most operations are performed locally, making it efficient.
4. **Undo & Revert:** If a mistake is made, Git allows rolling back to previous versions without affecting the entire project.
5. **Staging Area:** Allows reviewing and modifying changes before committing.

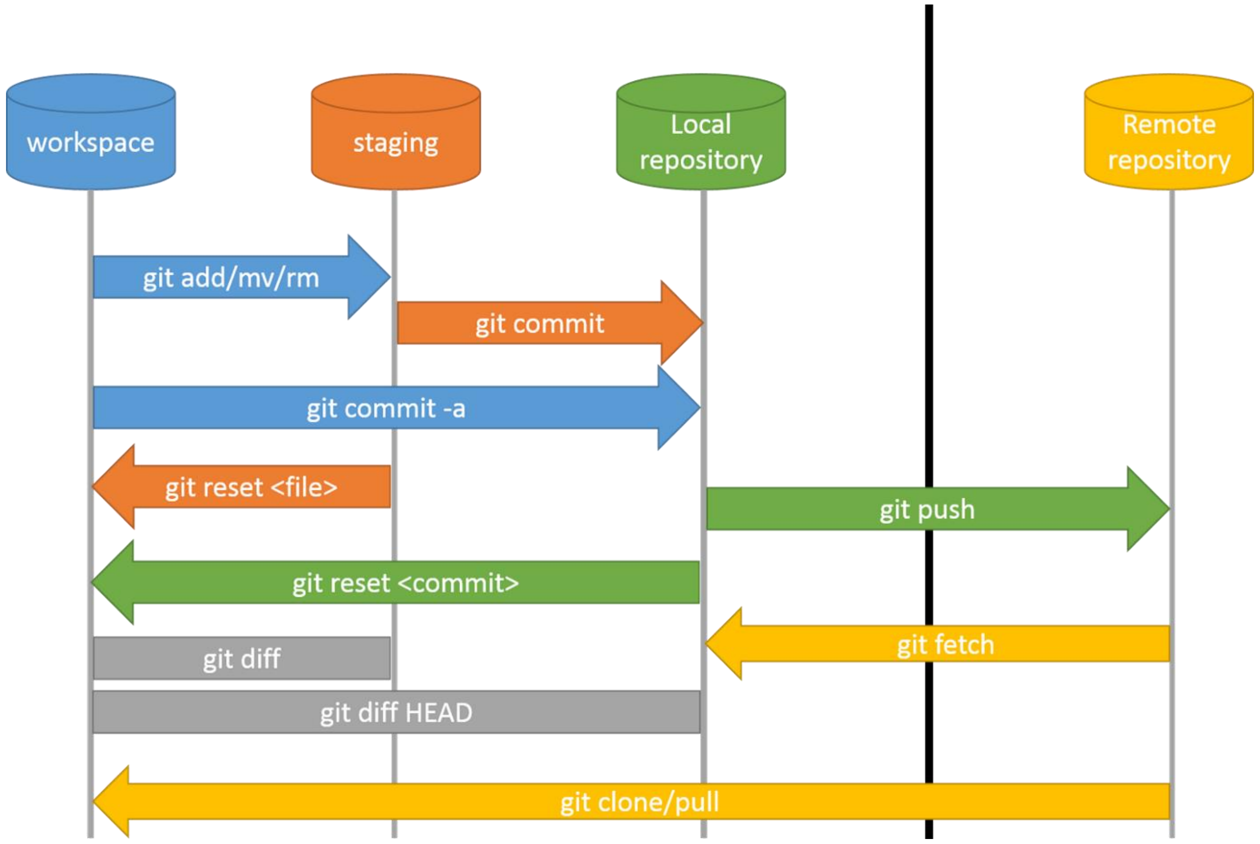
What is GitHub?

GitHub is a cloud-based platform that hosts Git repositories, providing tools for collaboration, project management, and security.

Features of GitHub:

1. **Remote Repository Hosting:** Stores Git repositories online, making them accessible from anywhere.
2. **Collaboration Tools:** Supports pull requests, issues, and code reviews.
3. **Version Control:** GitHub keeps track of all changes made to a project, showing a complete history of modifications.
4. **Security Features:** Includes branch protection, vulnerability scanning, secret detection, and two-factor authentication.
5. **Integration with Other Tools:** Works with third-party apps like Jira, Slack, VS Code, and CI/CD platforms for better productivity.

Git Architecture



Key Components of Git Architecture

1. Working Directory

- The actual directory on your system where you modify, create, or delete files.
- This is where all the current project files exist.

2. Staging Area (Index)

- A temporary storage where changes are added before committing them.
- It allows users to select specific changes they want to include in the next commit.
- Files move to the staging area using the `git add` command.

3. Local Repository (Git Directory)

- The version-controlled repository stored on the developer's computer.
- Contains the full project history and commits.
- Changes from the staging area are permanently saved here using `git commit.`

4. Remote Repository (GitHub, GitLab, Bitbucket, etc.)

- A cloud or remote server that stores the project repository.
- Developers push their changes from the local repository to the remote repository.
- Teams can collaborate by pulling updates from this repository.

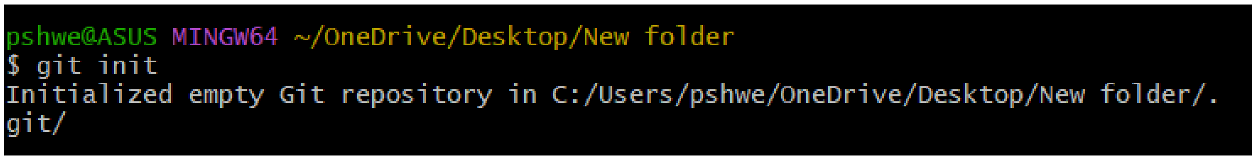
Git Workflow: Step-by-Step Implementation:

1. **Create a Folder:** Create a new folder for your project and navigate to it.

2. **Add a File:** Add any new file (e.g., index.html, README.md) to the folder.

3. **Open Git Bash:** Right-click inside the folder and open it with Git Bash.

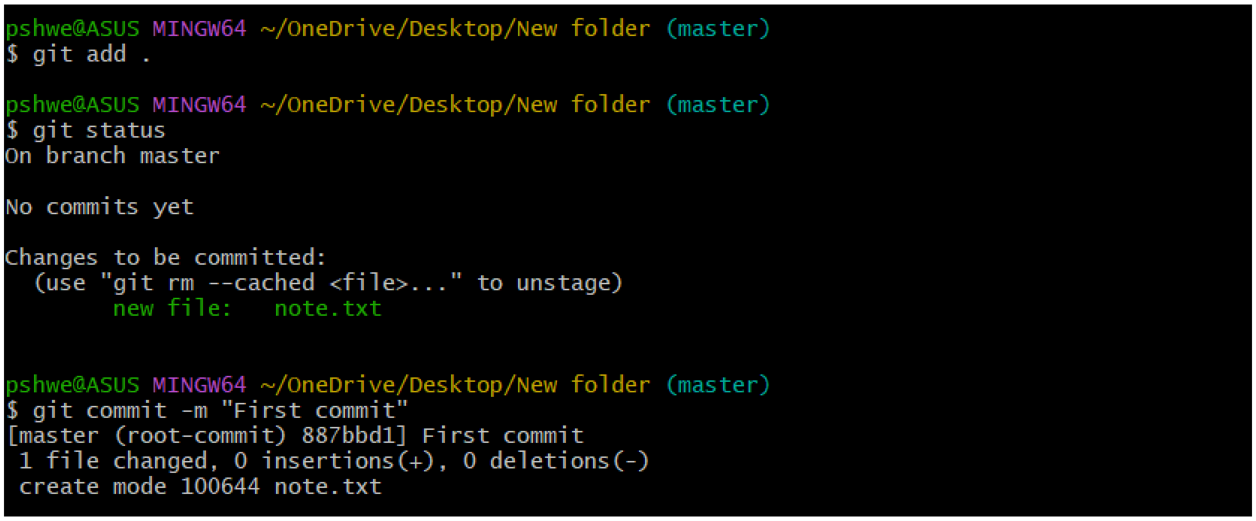
4. **Initialize Git:** Run `git init` to initialize the folder as a Git repository.



```
pshwe@ASUS MINGW64 ~/OneDrive/Desktop/New folder
$ git init
Initialized empty Git repository in C:/Users/pshwe/OneDrive/Desktop/New folder/.
git/
```

5. **Stage Changes:** Use “`git add .`” to add all files in the folder to the staging area.

6. **Commit Changes:** Run `git commit -m "Any message"` to save the staged changes to the local repository. Use git status to check status of file.

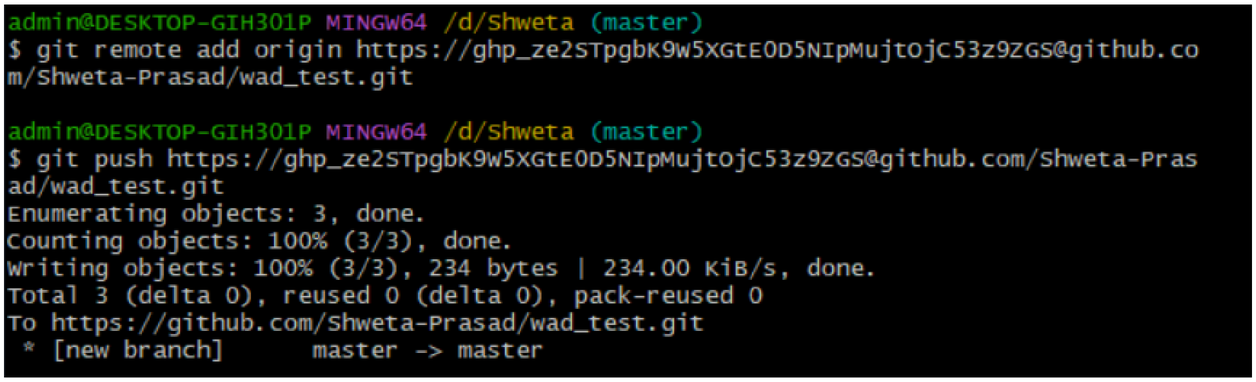


```
pshwe@ASUS MINGW64 ~/OneDrive/Desktop/New folder (master)
$ git add .

pshwe@ASUS MINGW64 ~/OneDrive/Desktop/New folder (master)
$ git status
On branch master

No commits yet

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


pshwe@ASUS MINGW64 ~/OneDrive/Desktop/New folder (master)
$ git commit -m "First commit"
[master (root-commit) 887bbd1] First commit
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 note.txt
```

7. **Create a GitHub Repository:** On GitHub, create a new repository but do not add a README. Copy the repository URL(HTTPS with .git at the end).

8. **Generate Personal Access Token:** Go to GitHub Settings > Developer Settings > Personal Access Tokens and generate a token with the required scopes (check all in this case). Copy it and store in a secure place.

9. **Add Remote Repository:** Adding a remote repository means linking your local Git repository to a repository hosted on a remote platform i.e. GitHub. Follow the structure of the URL as: https://<personal_access_token>@github.com/username/repository-name.git



```
admin@DESKTOP-GIH301P MINGW64 /d/Shweta (master)
$ git remote add origin https://ghp_ze2STpgbK9W5XGtEOD5NIpMujtOjC53z9ZGS@github.co
m/Shweta-Prasad/wad_test.git

admin@DESKTOP-GIH301P MINGW64 /d/Shweta (master)
$ git push https://ghp_ze2STpgbK9W5XGtEOD5NIpMujtOjC53z9ZGS@github.com/Shweta-Pras
ad/wad_test.git
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 234 bytes | 234.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Shweta-Prasad/wad_test.git
 * [new branch]      master -> master
```

10. **Perform Initial Push using PAT:** Above screenshot also shows the initial push from the local repository to the remote repository, done using PAT. After successful execution of the command, file should be visible on the GitHub repository as well, along with its respective commit message.

11. **Upstream option:** Use `git push -set-upstream <remote> <branch>.` What it does:

- Links the local master branch to the remote master branch in the origin remote repository.
- After this, you can simply run git push or git pull without specifying the branch name.

In case of any error of authentication, perform following commands:

```
1. git config -global user.name “GitHubUsername”

2. git config -global user.email “GitHubEmail”
```

Git Clone & Git Pull:

1. **Git Clone:** git clone is used to create a copy of a remote repository on your local machine. It also sets up the remote origin automatically, allowing you to easily push and pull changes from the remote repository.

Syntax: `git clone <repository-url>`

2. **Git Pull:** git pull fetches updates from the specified remote repository (e.g., origin) and the specified branch. It then merges the changes into your current local branch, keeping your local project up to date.

Syntax: `git pull <remote> <branch>`

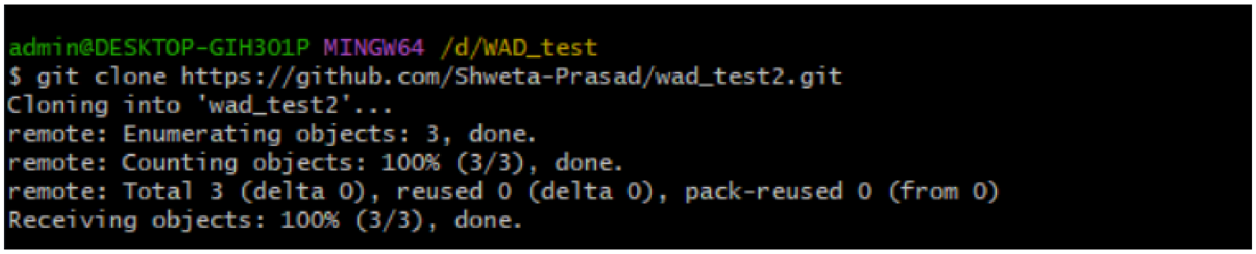
Example of Git Clone & Pull:

Step 1: Create a Repository and Add a File on GitHub

1. Navigate to GitHub and create a new repository.
2. Add a file (e.g., example.txt) to the repository and save changes.

Step 2: Clone the Repository

1. Create any new folder and open Git Bash from that location.
2. Run the git clone command.
3. This will create a local copy of the repository in a new folder named after the repository.



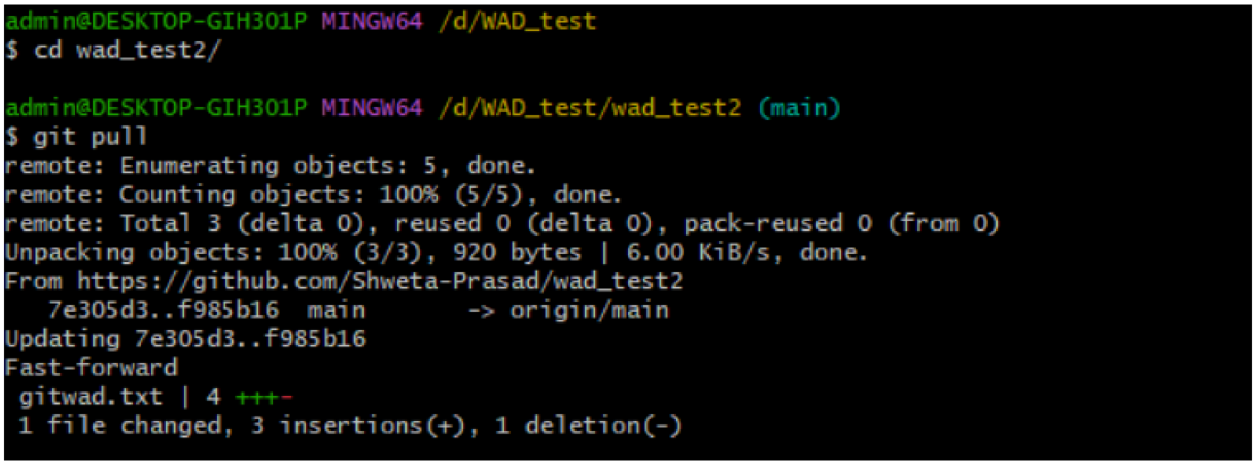
```
admin@DESKTOP-GIH301P MINGW64 /d/WAD_test
$ git clone https://github.com/Shweta-Prasad/wad_test2.git
Cloning into 'wad_test2'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.
```

Step 3: Edit the file on GitHub

1. Open the repository on GitHub and edit the file you added (e.g., example.txt).
2. Save the changes directly on GitHub.

Step 4: Navigate to the Cloned Repository on Your Local Machine

1. Use the cd command to navigate to the folder where the repository was cloned:



```
admin@DESKTOP-GIH301P MINGW64 /d/WAD_test
$ cd wad_test2/

admin@DESKTOP-GIH301P MINGW64 /d/WAD_test/wad_test2 (main)
$ git pull
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (3/3), 920 bytes | 6.00 KiB/s, done.
From https://github.com/Shweta-Prasad/wad_test2
   7e305d3..f985b16  main       -> origin/main
Updating 7e305d3..f985b16
Fast-forward
 gitwad.txt | 4 +++-
 1 file changed, 3 insertions(+), 1 deletion(-)
```

Step 5: Pull the Changes from GitHub

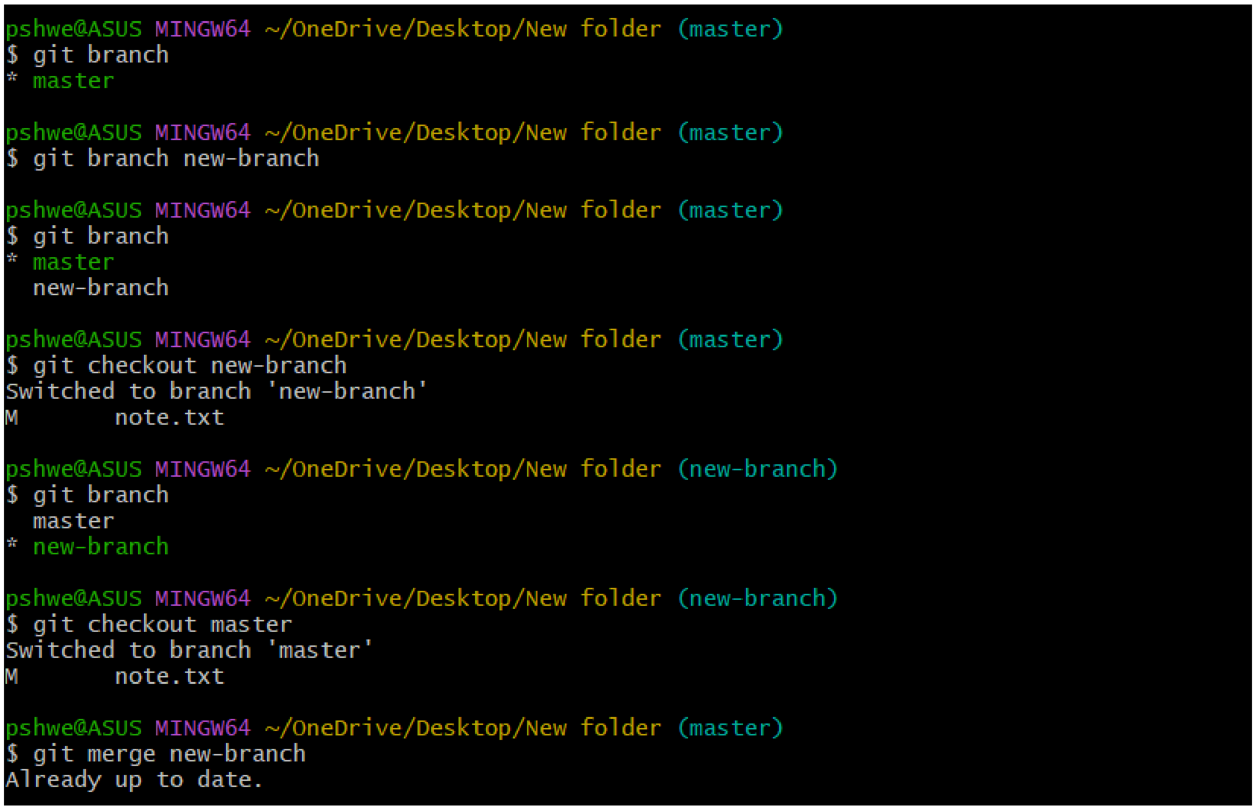
1. Run the git pull command to retrieve the latest changes made on GitHub(as shown above).
2. This will fetch and merge the updated file (e.g., gitwad.txt) into your local repository.

Git Branch

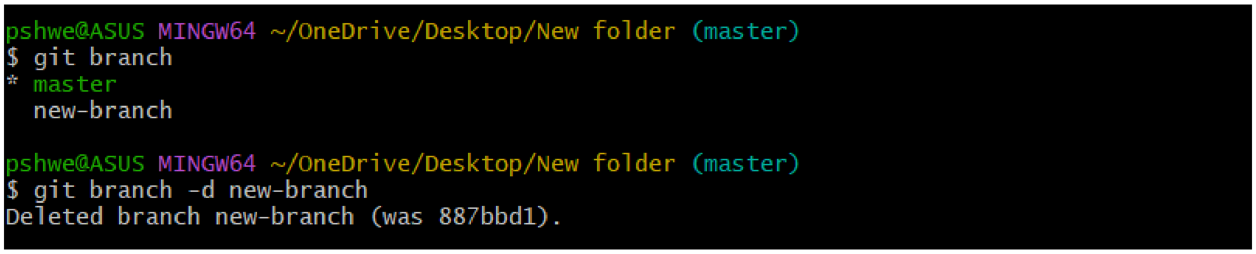
A branch in Git is a fundamental concept that allows developers to diverge from the main line of development, enabling them to work on new features, bug fixes, or experiments without affecting the stable version of the project.

Common commands for branching:

1. List all Branches:
    - `git branch`
2. Branch Creation:
    - Done using `git branch <branch-name>`
3. Branch Renaming:
    - `git checkout -m <new-name-of-branch>`
4. Branch Switching:
    - `git checkout <branch-name>`
5. Merging branches:
    - First switch to branch which will be on the receiving end of the merge operation.
    - `git merge <branch-name-which-is-to-be-merged-with-current>`
6. Deleting branch:
    - First navigate to another branch.
    - `git branch -d <branch-to-be-deleted>`



```
pshwe@ASUS MINGW64 ~/OneDrive/Desktop/New folder (master)
$ git branch
* master

pshwe@ASUS MINGW64 ~/OneDrive/Desktop/New folder (master)
$ git branch new-branch

pshwe@ASUS MINGW64 ~/OneDrive/Desktop/New folder (master)
$ git branch
* master
  new-branch

pshwe@ASUS MINGW64 ~/OneDrive/Desktop/New folder (master)
$ git checkout new-branch
Switched to branch 'new-branch'
M       note.txt

pshwe@ASUS MINGW64 ~/OneDrive/Desktop/New folder (new-branch)
$ git branch
  master
* new-branch

pshwe@ASUS MINGW64 ~/OneDrive/Desktop/New folder (new-branch)
$ git checkout master
Switched to branch 'master'
M       note.txt

pshwe@ASUS MINGW64 ~/OneDrive/Desktop/New folder (master)
$ git merge new-branch
Already up to date.
```



```
pshwe@ASUS MINGW64 ~/OneDrive/Desktop/New folder (master)
$ git branch
* master
  new-branch

pshwe@ASUS MINGW64 ~/OneDrive/Desktop/New folder (master)
$ git branch -d new-branch
Deleted branch new-branch (was 887bbd1).
```