Git and GitHub commands

1. Install Git on Windows

- *First, make sure Git is installed on your Windows system. If not:*
  - *Download and install Git from here.*
  - *During installation, you can choose the default settings. Just click "Next" through the installation steps.*

2. Set Up Git on Windows

- *After installing Git, you need to configure your Git with your username and email: Open Git Bash or Command Prompt (you can use Git Bash for a better experience).*

```
git config --global user.name "Your Name"
git config --global user.email "youremail@example.com"
```

3. Connect GitHub Account to Git on Windows

- *If you've already connected a GitHub account and want to switch, you need to remove the existing account and connect a new one.*

To remove old GitHub credentials:

- *In Git Bash, run:*

```
git credential-manager uninstall
```

To connect a new GitHub account:

- *Generate a Personal Access Token (PAT) from GitHub:*
  1. *Go to GitHub: GitHub Settings -> Developer Settings -> Personal Access Tokens*
  2. *Create a token with the necessary permissions.*
- *Now, when you push or pull from a repository, Git will prompt you for your GitHub username and the generated token as the password.*

4. Using GitHub with VS Code

- *To link GitHub with VS Code:*

1. *Install VS Code.*
2. *Install the Git extension (usually installed by default with VS Code).*
3. *Open your project in VS Code, and if it is a Git repository, VS Code will recognize it.*

VS Code Git Integration:

- *You can use the Source Control tab in VS Code to perform operations like commit, push, pull, and view changes.*

5. Basic Git Commands

Initialize a Repository:

- *If you're starting a new project and want to track it with Git:*

```
git init
```

Clone an Existing Repository:

- *To download a repository from GitHub:*

```
git clone https://github.com/username/repository.git
```

Check Repository Status:

- *To see changes and files that have been modified:*

```
git status
```

Adding Changes:

- *Stage the changes to commit:*

```
git add .  # Add all changes (use . to add everything)
```

```
git add <filename>  # Add a specific file
```

Commit Changes:

- *Commit the changes with a message:*

```
git commit -m "Your commit message"
```

Push Changes to GitHub:

- *Push the changes to your remote GitHub repository:*

```
• git push origin main  # For the main branch
• git push origin master  # For the master branch
```

Pull Changes from GitHub:

- *Pull changes from the remote repository:*

```
• git pull origin main  # For the main branch
```

View Commit History:

- *To view the commit log:*

```
• git log
```

Create a New Branch:

- *Create a new branch and switch to it:*

```
• git checkout -b new-branch-name
```

Switch Between Branches:

- *To switch to an existing branch:*

```
• git checkout branch-name
```

Merge Branches:

- *To merge a branch into the current branch:*

```
• git merge branch-name
```

Delete a Branch:

- *To delete a local branch after merging:*

```
• git branch -d branch-name
```

6. Create a New Repository on GitHub

1. *Go to your GitHub account, click the + icon in the upper right, and choose New repository.*
2. *Name your repository and create it.*
3. *Once created, you will see a URL like https://github.com/username/repository.git.*

Push Local Repository to GitHub:

- *If you have a local Git repository and want to push it to GitHub, use the following commands:*

  1. *Add the GitHub repository as a remote:*
  2. *git remote add origin https://github.com/username/repository.git*
  3. *Push your code to GitHub:*
  4. *git push -u origin main*

7. Delete a Repository (Locally and on GitHub)

Delete a Repository Locally:

- *To remove the local Git repository folder:*
- *rm -rf repository-name # Be careful, this deletes everything in the folder*

Delete a Repository on GitHub:

- *Go to your GitHub repository.*
- *Click on Settings > Scroll down to Danger Zone.*
- *Click Delete this repository and confirm.*

8. GitHub GUI Options (GitHub Desktop)

If you prefer not to use the command line, you can use the GitHub Desktop application:

- *Download GitHub Desktop: GitHub Desktop*
- *With GitHub Desktop, you can easily clone repositories, commit, push, pull, and manage branches without using the terminal.*

9. Team Collaboration

- *To collaborate with others on GitHub:*

  1. *Fork a Repository: Copy someone else's repository to your own GitHub account.*
  2. *git fork https://github.com/owner/repository.git*
  3. *Pull Requests (PR): After forking and making changes, submit a pull request to the original repository for review.*

10. Other Helpful Git Commands

- *Undo a commit: (if you want to undo the last commit but keep changes):*

```
git reset --soft HEAD~1
```

- *Remove untracked files:*

```
git clean -f
```

By using the above commands, you should be able to handle most of the tasks you'll encounter when working with Git and GitHub, from setting up to pushing, pulling, and collaborating with others.

Here are some additional

useful Git commands for various scenarios, from handling conflicts to working with remotes and undoing changes. These can help you in advanced Git operations or when dealing with certain challenges.

1. Working with Remotes

Add a New Remote Repository:

- *If you want to add another remote to your local repository, for example, to push to a second GitHub repository:*

```
git remote add <name> <url>
```

Example:

```
git remote add upstream https://github.com/another-user/repository.git
```

View Remote Repositories:

- *To check which remotes are connected to your repository:*

```
git remote -v
```

Change Remote URL:

- *If the URL of your remote repository has changed (for example, you move from HTTPS to SSH):*

```
• git remote set-url origin <new-url>
```

Remove a Remote:

- *To remove a remote repository:*

```
• git remote remove <name>
```

Fetch Updates from Remote:

- *To fetch changes from a remote without merging them:*

```
• git fetch origin
```

Push to a Specific Remote and Branch:

- *Push to a specific remote repository and branch:*

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

Example:

```
git push origin feature-branch
```

2. Handling Branches

List All Branches:

- *To list all branches (both local and remote):*

```
• git branch -a
```

Delete a Remote Branch:

- *To delete a branch from the remote repository:*

```
• git push origin --delete <branch-name>
```

Rename a Branch:

- *To rename your current branch:*

```
• git branch -m new-branch-name
```

Show Branch Merges:

- *To see the commit history of merged branches:*

```
• git log --graph --oneline --decorate
```

Track a Remote Branch:

- *To track a remote branch (i.e., to create a local branch that tracks the remote):*

```
git checkout --track origin/<branch-name>
```

3. Undoing Changes and Recovery

Undo a Commit (Soft Reset):

- *If you want to undo the last commit but keep your changes staged:*

```
git reset --soft HEAD~1
```

Undo a Commit (Hard Reset):

- *To remove the last commit and all changes made (this is dangerous as it discards all uncommitted changes):*

```
git reset --hard HEAD~1
```

Revert a Commit:

- *To create a new commit that undoes the changes made by a specific commit (without changing history):*

```
git revert <commit-hash>
```

Discard Local Changes in a File:

- *If you want to discard changes made to a specific file and revert it to the last committed state:*

```
git checkout -- <file-name>
```

Discard All Local Changes (Uncommitted Changes):

- *To discard all uncommitted changes in the working directory:*

```
git checkout -- .
```

Remove Staged Changes:

- *To unstage changes that were added by git add:*

```
git reset <file-name>
```

4. Merging and Resolving Conflicts

Merge a Different Branch into Your Current Branch:

- *To merge a branch into your current branch:*

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

Handle Merge Conflicts:

- *If there are conflicts after a merge, Git will mark the conflicting files. You need to manually resolve the conflicts in these files, then stage the resolved files:*

```
git add <file-name>
git commit
```

Abort a Merge:

- *If you want to cancel a merge and restore your branch to the state before the merge:*

```
git merge --abort
```

5. Stashing Changes

Stash Your Changes:

- *If you're in the middle of work and need to switch branches without committing, you can stash your changes:*

```
git stash
```

Apply Stashed Changes:

- *To apply the most recent stashed changes:*

```
git stash apply
```

List Stashed Changes:

- *To view a list of all stashes:*

```
git stash list
```

Drop a Stash:

- *To remove a stash entry after it's no longer needed:*

```
git stash drop <stash@{0}>
```

Pop a Stash (Apply and Remove It):

- *To apply and remove the most recent stash:*

```
• git stash pop
```

6. Working with Tags

Create a Tag:

- *To create a lightweight tag at the current commit:*

```
• git tag <tag-name>
```

Push a Tag to GitHub:

- *To push tags to the remote repository:*

```
• git push origin <tag-name>
```

List All Tags:

- *To list all tags in the repository:*

```
• git tag
```

Delete a Tag Locally:

- *To delete a tag locally:*

```
• git tag -d <tag-name>
```

Delete a Tag from Remote:

- *To delete a tag from the remote repository:*

```
• git push origin --delete <tag-name>
```

7. Viewing Git Information

Show Current Git Configuration:

- *To view the current global or local Git configuration:*

```
• git config --list
```

Show Details of a Commit:

- *To see the details of a specific commit:*

```
• git show <commit-hash>
```

Check the Current Branch:

- *To see the current active branch you're working on:*

```
• git branch
```

8. Working with Git Logs

Show Detailed Log (with Commit Changes):

- *To view commit history with changes in the commit:*

```
• git log -p
```

Show Log for Specific File:

- *To see the history of changes made to a specific file:*

```
• git log <file-name>
```

Show Log with One-Line Summary:

- *To view the commit history as one line per commit:*

```
• git log --oneline
```

Show a Graph of the Commit History:

- *To view a graphical representation of the commit history:*

```
• git log --graph --oneline --decorate
```

9. Working with Git Submodules

Add a Submodule:

- *If you want to add a submodule (another repository within your repository):*

```
• git submodule add <repository-url> <path>
```

Update Submodules:

- *To update the submodules to the latest commit:*

```
• git submodule update --remote
```

These additional Git commands cover a wide range of advanced operations, including handling remote repositories, branching, stashing, and managing history. You can use these commands depending on your project's needs, especially as you work with larger repositories or in collaborative environments.