**Step-by-Step Guide for Using Git and GitHub**

**Step 1: Create a Repository on GitHub**

* Go to [GitHub](https://github.com).
* Click on **New Repository** and follow the prompts to create an empty repository (e.g., GitLearning).
* Do not initialize with a README file if you’re working with an existing local project.

**Step 2: Prepare Local Code**

* Create a folder in your desired location on your computer.
* Paste your project code inside this folder.

**Step 3: Open Git Bash**

* Open Git Bash in the folder where your code is located. You can do this by right-clicking inside the folder and selecting **Git Bash Here**.

**Step 4: Initialize Git**

Run the following command to initialize a new Git repository:

**--------------------------------------$ git init**

**Step 5: Check Git Status**

Check the status of your repository to see untracked files:

**-------------------------------------$ git status**

* **Reason**: This ensures that your files are being tracked correctly. Initially, they should appear in red, indicating they are untracked.

**Step 6: Stage Files**

Stage all files using the following command:

**-------------------------------------$ git add .**

* **Reason**: The . adds all untracked files to the staging area.

Check the status again to verify the files are staged:

**-------------------------------------$ git status**

* **Expected Result**: Files should appear in green, indicating they are staged for commit.

**Step 7: Commit Changes**

Commit the staged files with a message:

**------------------------------------$ git commit -m "Initial commit"**

**Step 8: Add Remote Repository**

Add the remote repository URL to link your local repository with GitHub:

**-------------$ git remote add origin https://github.com/njagades/GitLearning.git**

**Step 9: Push Code to GitHub**

Push your code to the remote repository for the first time:

**-------------------------$ git push -u origin master**

* **Reason**: The -u option sets the upstream reference, meaning future pushes can be done with just git push.

**Step 10: Creating a New Branch (Sub-Branch)**

To create a new branch, ensure you're in the master branch first:

**--------------------------------$ git checkout master**

Now create and switch to a new branch (e.g., develop):

**-------------------------------$ git checkout -b develop**

* **Note**: This command creates a new branch called develop and switches to it.

**Step 11: Check Current Branch**

Check which branch you are currently on:

**------------------------------------------$ git branch**

**Step 12: Write Code in the Local Branch**

Write or modify your code in your local branch (develop).

**Step 13: Stage the Changes**

Once your changes are done, stage them for commit. For example, if you've modified a specific file:

**-----------------$ git add src/test/java/project/testComponents/BaseTest.java**

**Step 14: Check Staged Files**

Check the status to confirm that the file has been staged correctly:

**---------------------------------$ git status**

**Step 15: Commit the Changes**

Commit your changes with a message:

**---------------------$ git commit -m "Second commit"**

**Step 16: Push Changes to the develop Branch**

Push the changes to the remote develop branch:

**----------------------------$ git push origin develop**

* **Note**: If the develop branch doesn’t exist on GitHub yet, it will be created.

**Step 17: Sync develop with master Before Merging**

Before merging your develop branch to master, make sure your develop branch is up to date with the latest changes in master:

**-------------------------------$ git pull origin master**

**Step 18: Resolve Any Merge Conflicts**

If there are any merge conflicts, Git will notify you. Resolve the conflicts in the affected files, then stage the resolved files:

**----------------------------------------$ git add .**

**Step 19: Commit the Resolved Files**

Once the conflicts are resolved and staged, commit the changes:

**------------------------$ git commit -m "Resolve conflicts"**

**Step 20: Push Resolved Changes to develop**

Push the resolved code to the develop branch:

**----------------------------$ git push origin develop**

**Step 21: Merge develop into master**

To merge the develop branch into master:

1. Switch to the master branch:

**--------------------------$ git checkout master**

1. Pull the latest code from develop into master:

**------------------------$ git pull origin develop**

1. If there are no conflicts, commit and push the changes:

**--------------------------$ git push origin master**

**Step 22: Create a Pull Request (PR) on GitHub**

1. Go to your GitHub repository.
2. Click on the **Pull Requests** tab.
3. Click **New Pull Request**.
4. Select the develop branch as the source branch and master as the target.
5. Review the changes and click **Create Pull Request**.

Finally, merge the pull request into master.

**Optional: Command-Line Merge (If No PR on GitHub)**

You can also merge the develop branch into master using the command line:

1. Switch to master:

**----------------------$ git switch master**

1. Pull changes from develop:

**--------------------$ git pull origin develop**

1. Resolve any conflicts if needed, stage, commit, and push:

**----------------------$ git add .**

**---------------------$ git commit -m "Merge develop into master"**

**--------------------$ git push origin master**