Git Workflow

1. Create a repository: git init
2. Add changes: git add
3. Commit changes: git commit
4. Download commits from remote repo (if necessary): git pull
5. Upload commits to the remote repo: git push
6. Repeat steps 2-5 as required

Basic Git Commands with Examples

**Git** is a powerful **version control system** that helps developers**manage and track changes** to their codebase over time. It enables**collaboration**, **allowing multiple developers** to work on the same project without overwriting each other’s work. **Git tracks**every modification, making it easy to **roll back, track bugs, and collaborate efficiently.**

Here, we’ll go through some essential**Git commands**and setup instructions that will help you get started with Git and GitHub.

**Prerequisites**

Before you begin using Git, you’ll need to have it installed on your system. Follow the instructions below based on your operating system:

**1. Checking Git Version**

Before starting with any Git operations, it’s essential to check if Git is installed correctly on your system. The following command will show you the version of Git installed:

git --version

**2. Initializing a Git Repository (git init)**

To start tracking a project using [Git](https://www.geeksforgeeks.org/git-tutorial/), you need to initialize a new Git repository in your project directory. This can be done with the git init command.

* **Navigating to your Project Directory:** Before using Git, navigate to your project directory: cd <filePath> command
* **Initialize Git Repository:** Once you’re in the project directory, run the following command to initialize a Git repository: git init

**2. Git Configuration**

After that, configure your username and email:

git config --global user.name "your Name"  
git config --global user.email "your.email@example.com"

**3. Forking and Cloning a Repository**

[Forking](https://www.geeksforgeeks.org/git-fork/) is a repository means creating a copy of an existing repository in your GitHub account so that we can make changes without affecting the original repository.

**How to Fork a Repository**

* Go to the repository we want to contribute to on GitHub.
* Click the Fork button at the top right.
* Once forked, navigate to your GitHub profile and open the forked repository

A screenshot of a computer

AI-generated content may be incorrect.

* **Copy URL:** Then a copy of real repository will be created in your local repository. After that, we have to copy the URL from your local repo. For doing that click to code and copy the URL.

**4. Cloning the Repository Locally**

After forking, [clone the repository](https://www.geeksforgeeks.org/how-to-git-clone-a-local-repository/) to your local machine

A screenshot of a computer

AI-generated content may be incorrect.

* Create a folder on your desktop where you want to store the project files.
* Open Git Bash and navigate to the newly created folder using the cd command:

cd <file\_directory>

* Copy the repository URL from GitHub.
* In [Git Bash](https://www.geeksforgeeks.org/working-on-git-bash/), type the following command and press Enter

git clone <copied\_url>

* The repository will be cloned into your desktop folder, making the project files available on your system.

**5. Checking the Status**

After making code changes, check which files are not added using:

git status

* This command displays the current state of your working directory, indicating whether files are untracked, staged, or committed. Files in red are untracked or modified but not staged, while files in green are staged and ready to be committed.

**Note:**

* make sure the parent directory is right.
* Renaming the main branch : git branch -m master

**6. Adding Files to Staging Area**

When we get to know which files are not added by typing git status(red-colored files are not added).

Stages all changes: git add -A

Stages new and modified files only: git add .

Stages modified and deleted files only: git add -u

Stages a specific fil: git add <fileName>

And then

Checking the Status: git status

**7. Committing Changes**

* To save your changes in the local repository, first check the status using:

git status

* Files displayed in green are staged but not yet committed. To commit these changes, use:

git commit -m "Your commit message"

And then

Checking the Status: git status

**Git Commit Guidelines**

* Create meaningful commits that represent a single task or related changes (e.g. adding a feature or bug fix)
* Commit early and often
* Break down larger tasks into sub-tasks
* Make commits based on a time period
* Write clear commit messages
  + "Added base CSS styles"
  + "Project status at the end of the day"
* When collaborating, decide on a commit strategy together

**8. Pushing Changes to GitHub**

To upload commits to your forked repository

git push origin <branchName>

**9. Making a Pull Request**

Once your changes are pushed, you need to create a [Pull Request](https://www.geeksforgeeks.org/git-pull-request/)(PR):

* Go to your forked repository on [GitHub](https://www.geeksforgeeks.org/introduction-to-github/).
* Click on Compare & pull request.
* Add a title and description for your changes.
* Click Create pull request to submit your changes for review.

A screenshot of a computer

AI-generated content may be incorrect.

**10. Merging Branches**

To[merge a branch](https://www.geeksforgeeks.org/how-to-merge-a-git-branch-into-master/) into the current branch:

git merge <branchName>

If there are conflicts, Git will prompt you to resolve them manually before completing the merge. Once resolved, commit the changes using:

git commit -m "Merged <branch\_name> into <current\_branch>"

To abort a merge in case of issues:

git merge --abort

**11. Deleting a Branch**

To delete a local branch:

git branch -d <branch\_name>

To force delete

git branch -D <branch\_name>