**Git & GitHub Command Tutorial**

**What’s Git All About?**

**Git** is a powerful, free, and open-source version control system. It acts like a time machine for your code, helping you keep track of changes and work together with others seamlessly. Pretty cool, right?

**What’s GitHub?**

**GitHub** is an online platform where you can host your Git repositories. It makes collaborating, reviewing code, and managing projects a breeze.

**Getting Started with Git**

**Setting Up Git:**

If you’re on **Linux**, use these commands:

bash

sudo apt update # Updates the package list

sudo apt install git # Installs Git

For **macOS** users:

bash

brew install git # Installs Git with Homebrew

**Want to check your Git version?**  
Run:

bash

git --version

**Basic Terminal Commands**

Here are some essential commands to get you moving in the terminal:

Table

| **Command** | **What It Does** |
| --- | --- |
| pwd | Shows you where you are in the directory |
| ls -a | Lists all files, even the hidden ones |
| mkdir foldername | Creates a new folder |
| cd foldername | Navigates into a folder |
| cat filename.txt | Displays the contents of a file |
| echo "message" > file.txt | Writes a message to a file |
| echo "add this" >> file.txt | Adds a message to the end of a file |

**Git Essentials**

**Ready to dive into Git? Here are the basics:**

**1. Initialize Git:**  
To start using Git in your directory, run:

bash

git init

**2. Set Your Configuration:**  
Do this once on your machine:

bash

git config --global user.name "Your Name"

git config --global user.email "your@email.com"

**3. Check Your Configurations:**  
See what you’ve set up so far with:

bash

git config --list

**4. Check the Status of Your Files:**  
To see what’s going on in your project:

bash

git status

**5. Add Files for Staging:**  
You can stage files like this:

bash

git add filename.txt # Adds a specific file

git add . # Adds all changes

**6. Commit Your Changes:**  
Don’t forget to save your progress with a message:

bash

git commit -m "Your commit message"

**7. View Your Commit History:**  
Keep track of your changes by checking:

bash

git log

**8. Clone a Repository:**  
To get a repository from GitHub to your machine:

bash

git clone <repository-url>

**9. Need Help?**  
If you’re stuck on a command, you can use:

bash

git commit --help

**Branching in Git**

Branches are great for working on different features without messing up the main code. Here’s how to manage them:

**1. List All Branches:**

bash

git branch

**2. Create a New Branch:**

bash

git branch new-feature

**3. Switch to Your Branch:**

bash

git checkout new-feature

**4. Merge Changes:**  
To bring your new feature back into the main branch:

bash

git merge new-feature

**5. Delete a Branch When You’re Done:**

bash

git branch -d new-feature

**6. Rename or Move a File:**  
Make changes to filenames easily:

bash

git mv oldname.txt newname.txt

**7. Remove a File:**  
To delete a file, just run:

bash

git rm filename.txt

**Helpful Tips**

* Write clear commit messages to remember what you did.
* Pull changes frequently when working with a team to stay updated.
* Avoid committing sensitive information like passwords.