The Daily Git Workflow

AiCE Software Engineering Foundations — The Developer Environment

With Git installed and your GitHub account ready, it's time for the workflow you'll repeat constantly for the rest of your career: getting a repository onto your machine, sending your changes back up to GitHub, and pulling down changes made by others. This lesson walks through cloning, the local/remote relationship, the add → commit → push cycle, and pulling.

1. Cloning a Repository

Cloning means downloading a copy of a GitHub repository to your local computer. Once cloned, you can create files, edit them, and send those changes back to GitHub whenever you're ready.

To clone, you need the repository's URL (found on GitHub under the green Code button — make sure HTTPS is selected) and a Personal Access Token.

Generating a Personal Access Token

A Personal Access Token (PAT) is a security credential that allows Git to authenticate with your GitHub account. To generate one:

1. Click your profile picture on GitHub and go to Settings.

2. Scroll to the bottom of the left sidebar and click Developer Settings.

3. Click Personal Access Tokens, then Tokens (Classic).

4. Click Generate New Token (Classic).

5. Give the token a name, and set the expiration to No expiration (recommended while learning).

6. Under scopes, select the repo checkbox — this grants access to repositories.

7. Click Generate Token.

8. Copy the token immediately — you will NOT be able to see it again after leaving the page.

> **Treat Your Token Like a Password**
> *A Personal Access Token is a security credential. Do not share it, commit it into a file, or post it anywhere public — anyone holding it can act on your GitHub account exactly as you can.*

Running the Clone

Navigate in your terminal to the directory where you want the repository to live, then run:

```
git clone
https://YOUR_TOKEN@github.com/yourusername/your-repo-name.git
```

Replace YOUR_TOKEN with your Personal Access Token, yourusername with your GitHub username, and your-repo-name with the actual repository name. The general shape of that URL is:

```
git clone
https://[token]@github.com/[username]/[repo-name].git
```

After pressing Enter, you'll see output confirming the clone is happening:

```
Cloning into 'your-repo-name'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
Receiving objects: 100% (3/3), done.
```

Run ls to confirm the repository folder now exists on your computer, then enter it with cd:

```
cd your-repo-name
```

2. The Local and Remote Repository

Once you clone a repository, you effectively have two copies of it:

Copy	Location	Name
Original	GitHub servers online	Remote repository
Duplicate	Your computer	Local repository

Any time you create, edit, or delete files in your local repository, the remote repository on GitHub is not automatically updated — you must explicitly send those changes, which is called pushing. Conversely, if someone changes the repository on GitHub (or you edit something directly through the GitHub interface), your local copy won't update automatically either — you must explicitly bring those changes down, which is called pulling.

> **The Two Directions**
> *Computer → GitHub = git push (send local changes to GitHub). GitHub → Computer = git pull (bring GitHub changes to your local copy).*

3. Git Add, Commit, and Push

This is the core Git workflow you will repeat hundreds of times throughout your career. Every time you make changes locally and want to send them to GitHub, you follow three steps:

Step	Command	What It Does
1. Add	git add .	Stages all changed files, marking them ready to be committed
2. Commit	git commit -m "message"	Takes a snapshot of the staged files and records it in history
3. Push	git push	Sends the committed snapshot to the remote repository on GitHub

Step 1 — git add

Before Git can record your changes, you must tell it which files to include. Using a dot (.) adds all modified files at once:

```
git add .
```

You can also add a specific file:

```
git add africa_report.txt
```

Step 2 — git commit

Committing creates a permanent snapshot of your staged changes. Every commit needs a message describing what changed:

```
git commit -m "Add initial report on African tech ecosystem"
```

Write commit messages in the present tense, as if describing what the commit does. Good messages are short, specific, and informative — poor ones make history impossible to understand later.

Poor Commit Message	Better Commit Message
fixed it	Fix broken login validation for Nigerian phone numbers
update	Add M-Pesa payment integration to checkout module
changes	Refactor database schema for African country codes
asdf	Remove deprecated endpoints from pan-African API

Step 3 — git push

```
git push
```

After pushing, go to your GitHub repository page and refresh — your files will now appear there.

A Complete Push Workflow, Start to Finish

```
# Create a file inside the repository
echo "African developers are building the future." >
vision.txt

# Stage the file
git add .

# Commit with a meaningful message
git commit -m "Add vision statement file"

# Push to GitHub
git push
```

4. Git Pull — Updating Your Local Repository

When changes exist on GitHub that aren't yet in your local repository, you bring them down with:

```
git pull
```

This is essential in three common situations:

You made changes directly on the GitHub website and want them on your computer.

A collaborator pushed changes to the shared repository and you want the latest version.

You're working across two different computers and need to sync.

> **Best Practice**
> *Always run git pull before you start working in a repository. This ensures you're building on the most recent version and meaningfully reduces the risk of conflicts.*

Key Takeaways

Clone once to get a repository onto your machine; from there, add → commit → push is the loop you'll run constantly to send your work to GitHub, and pull is how you bring in

everyone else's. Getting into the habit of pulling before you start, and committing in small, well-described steps, will save you real pain later — particularly once branching and merging, covered next, are part of your daily routine.