Git & GitHub Command Notes (Beginner → Practical)

---

1. Basic Navigation Commands (Before Git)

pwd

Purpose:
Displays the current working directory.

Why it matters:
Git commands work only inside the correct project directory.

---

ls

Purpose:
Lists files and folders in the current directory.

ls -a

Option (-a – all):
Displays hidden files and folders such as .git and .gitignore.

---

cd <path>

Purpose:
Changes the current directory.

Example:

cd ~/Desktop/AWS\ Zero-to-Hero

---

2. Git Repository Setup (One-Time Configuration)

git init

Purpose:
Initializes a new Git repository in the current directory.

Effect:

- Creates a .git directory
- Enables version control tracking

---

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

Purpose:
Sets the author name for all commits.

---

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

Purpose:
Sets the author email for all commits.

---

git config --global --list

Purpose:
Displays all Git global configuration values to verify setup.

---

3. Repository Status & Tracking

git status

Purpose:
Shows:

- Modified files
- Untracked (new) files
- Staged files

Best practice:
Run this command before git add and git commit.

---

4. Adding Files to the Staging Area

git add .

Purpose:
Stages all modified and new files in the current directory and subdirectories.

Important:

- Does **not** upload files
- Only prepares files for commit

Add Only a Specific Folder

Use case: Changes made only in the EC2 folder.

git add EC2/

Effect:
Only files inside EC2/ are staged.

Add a Specific File

git add EC2/ec2-notes.docx

View Staged Files

git status

---

5. Committing Changes

git commit -m "message"

Purpose:
Creates a snapshot of staged changes.

Option (-m – message):
Adds the commit message inline.

Example:

git commit -m "EC2: added instance launch notes"

Rule:
One logical change = one commit.

---

6. Branch Management

git branch

Purpose:
Displays the current branch.

---

git branch -M main

Option (-M – move/force rename):
Renames the current branch to main.

---

7. Connecting Local Repository to GitHub (One-Time)

git remote add origin <repository-url>

Purpose:
Links the local repository to a GitHub repository.

Example:

git remote add origin https://github.com/Sunny-Batavale/AWS-Zero-to-Hero.git

---

git remote -v

Purpose:
Verifies remote repository URLs for fetch and push.

---

8. Uploading Code to GitHub

git push -u origin main

Purpose:
Uploads commits to GitHub.

Option (-u – upstream):
Sets origin/main as the default remote branch.

After first push:

git push

---

9. Daily Workflow (Correct Practice)

Case 1: Changes in All Folders

git status

git add .

git commit -m "Updated AWS notes"

git push

Case 2: Changes Only in EC2 Folder (Recommended)

git status

git add EC2/

git commit -m "EC2: updated security group notes"

git push

Result:
Clean, readable, and professional commit history.

---

10. Removing Files from Git Tracking (Keep Locally)

git rm --cached <file>

Purpose:
Removes a file from Git tracking without deleting it locally.

Common use cases:

- .pem
- .ppk
- Credential files

---

11. .gitignore (Security Control File)

Purpose:
Prevents sensitive or unnecessary files from being tracked by Git.

Example:

*.pem

*.ppk

*_credentials.csv

Effect:
Ignored files are never staged or uploaded.

---

12. Git Workflow (Conceptual Flow)

Working Directory

↓

Staging Area (git add)

↓

Local Repository (git commit)

↓

Remote Repository – GitHub (git push)

---

13. Key Rules to Remember

- git add . stages everything
- git add EC2/ stages only the EC2 folder
- No data is uploaded without git push
- .gitignore is mandatory for secrets
- Commit messages must be clear and specific

---

14. Recommended Commit Message Format

Format:

<service>: <change>

Examples:

- EC2: added instance lifecycle notes
- VPC: updated subnet explanation