**Git Commands for Data Science with Practical Tasks**

### **git init**

* **Purpose:** Initializes a new Git repository.
* **Example:** You start working on a new project and want to track its changes using Git.
* In your project directory, you run:

$git init

This command creates a .git directory, setting up the project for version control.

### **git clone**

* **Purpose:** Copies an existing Git repository (from a remote server, like GitHub) to your local machine.
* **Example:** You find a useful open-source project on GitHub that you want to contribute to. You can download (clone) the repository to your local machine by running:

$git clone <https://github.com/username/repository.git>

### **git add**

* **Purpose:** Stages changes (files or modifications) to be included in the next commit.
* **Example:** You’ve modified a file named app.py and want to include it in the next commit. You run:

$git add app.py

Alternatively, to stage all modified files, you can use:

**$git add .**

### **git commit**

* **Purpose:** Saves the staged changes to the repository with a descriptive message.
* **Example:** After staging your changes, you save them with a commit message:

$git commit -m "Fixed the login bug"

### **git log**

* **Purpose:** Shows the commit history.
* **Example:** You want to see the history of commits in your project. Run:

$git log

You can also use flags like --oneline to see a more compact

view:

$git log –oneline

### **git push**

* **Purpose:** Uploads local commits to a remote repository.
* **Example:** After committing your changes locally, you want to share them with your team on GitHub:

$git push origin main

### **git config**

* **Purpose:** Sets configuration options for Git, like username and email.
* **Example:** Before making any commits, you should set your user information:

$git config --global user.name "Your Name"  
$git config --global user.email "[your.email@example.com](mailto:your.email@example.com)"

### **git help**

* **Purpose:** Displays help information about Git commands.
* **Example:** If you’re unsure how to use a specific Git command, you can ask for help:

$git help commit

Or to list all Git commands:

$git help –a

\*NOTE : :wq(write and quit) saves and closes the file it breaks the command

**Task:**

**Move the existing local code to a new repository in Git.**

# Initialize a new Git repository

$ git init

# Create a README file

$ echo "# My New Repository" > README.md

# Check the status of your repository

$ git status

# Stage the README file for commit

$ git add README.md

# Stage all files for commit

$ git add .

# Set up global Git username

$ git config --global user.name "YOUR USER NAME"

# Set up global Git email

$ git config --global user.email "YOUR EMAIL"

# Create a repository on GitHub (do this through the GitHub interface)

# Saves the staged changes to the repository with a descriptive message

$ git commit -m "initial commit"

# Link your local repository to the remote GitHub repository

$ git remote add origin https://github.com/username/test-repo.git

# Push your code to GitHub

$ git push -u origin “Branch name”

### **git diff**

* **Purpose:** To check the changes in the working area and the staged area.
* **Example:** Modify a file in your working directory, say file.txt.

$ git diff

### **git diff --staged**

* **Purpose:** To check the changes in the staged area and the repositary area.
* **Example:**

Stage the changes to file.txt using:

$git add file.txt

Run the following command to see the changes that are staged for the next commit:

$git diff --staged

This will show the differences between the staged changes and the last commit. If file.txt was modified and staged, this command will display those differences.

\*NOTE : vi createdfile.txt it will create a text file after adding the content press esc and :wq

### **git checkout and git switch**

* **Purpose:** This command is used to switch branches or restore working tree files.
* **Example:** You’re working on the feature branch but need to switch to the main branch:

$git checkout main

$git switch main

The git checkout -b command is used to create and switch to a new branch in a Git repository.

Here’s a breakdown of what it does:

* **-b**: This flag is used to create a new branch before switching to it.

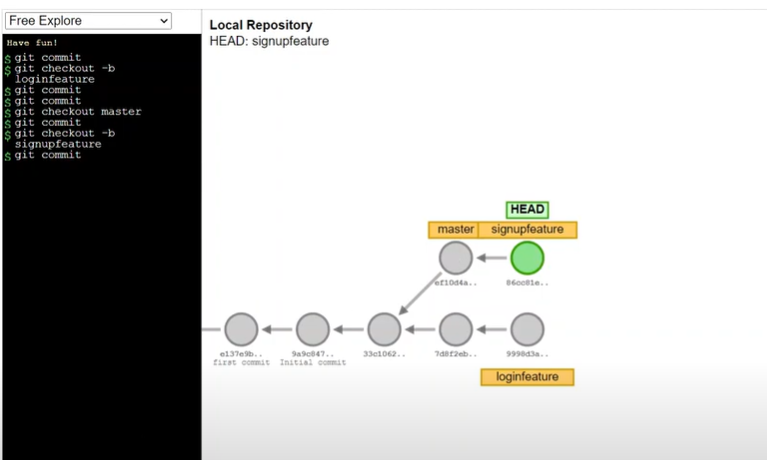
### **Example:**

$git checkout -b new-branch-name

$git switch -c new-branch-name

* **- :** If you want to quickly switch back to the branch you were previously on, you can use the - (dash)

$ git switch -



### **git branch**

The git branch command is used to manage branches in a Git repository. Branches are an essential part of Git, allowing you to work on different parts of a project simultaneously without affecting the main codebase.

#### **1. List All Branches**

* To list all the branches in your repository, both local and remote:

$git branch

This command will display the branches, with the current branch highlighted by an asterisk (\*).

* To list all branches, including remote branches:

$git branch -a

#### **2. Create a New Branch**

* To create a new branch called new-branch:

$git branch new-branch

* This creates a new branch but does not switch to it. You can use git checkout or git switch to switch to the new branch:

$git checkout new-branch

$git switch new-branch

#### **3. Delete a Branch**

* To delete a branch named old-branch:

$git branch -d old-branch

The -d flag deletes the branch, but only if it has been fully merged. If the branch has unmerged changes, you can use -D to force delete:

$git branch -D old-branch

#### **4. Rename a Branch**

* To rename the current branch to new-name:

$git branch -m new-name

* If you want to rename a branch that you are not currently on, specify both the old and new names:

$git branch -m old-branch new-name

#### **5. See the Last Commit on Each Branch**

* To see the last commit on each branch:

$git branch –v

\*NOTE: $git mv old\_file\_name new\_file\_name to rename the file and automatically move to stage.

### **git restore**

#### **1. Restore a File to Its Last Committed State**

* If you made changes to a file but haven’t staged or committed them yet, and you want to discard those changes:

$git restore <file>

Example:

$git restore README.md

This command will undo any changes you made to README.md and revert it back to how it was in the last commit.

#### **2. Unstage a File**

* If you’ve staged a file (using git add) but decide you don’t want to include it in the next commit:

$git restore --staged <file>

Example:

$git restore --staged README.md

This command removes README.md from the staging area but keeps your changes in the working directory.

#### **3. Restore All Files**

* To restore all files in the working directory to their last committed state:

$git restore .

This command will discard all changes in your working directory.

### **git revert**

To undo a specific commit by creating a new commit that negates the changes made in the original commit.

#### **1. Revert a Single Commit**

* To undo the changes made in a specific commit, you use:

$git revert <commit-hash>

Example:

$git revert 66b095827fab87fc9083a0b0e1a2564303732d69

This command creates a new commit that reverses the changes introduced by the commit with the hash 66b095827fab87fc9083a0b0e1a2564303732d69. Your project will look as if that commit was never made, but the commit itself remains in the history.

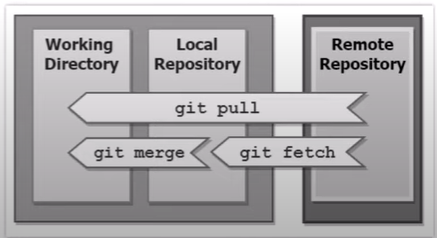
#### **2. Revert Multiple Commits**

* You can also revert a range of commits by specifying the commit hashes:

$git revert <start-commit-hash>..<end-commit-hash>

This will create a new commit for each commit in the specified range, effectively undoing all of them.

### **git pull , git fetch, git merge**



**git pull**

The git pull command is used to update your local repository with changes from a remote repository. It combines two actions: fetching new commits from the remote repository and merging those changes into your current branch.

#### **Pull Updates from a Specific Remote and Branch**

* To specify a remote and branch to pull from:

$git pull <remote> <branch>

For example, to pull changes from the develop branch on the origin remote:

$git pull origin develop

**git fetch**

The git fetch command is used to retrieve updates from a remote repository without altering your working directory

#### **Fetch All Updates from the Remote Repository**

* To fetch all branches and updates from the remote repository:

$git fetch

This command retrieves new commits and branches from the remote repository but does not automatically merge them into your current branch.

**git merge**

The git merge command is used after fetching changes using git fetch, you can merge those changes into your local branch.

$ git merge <branch-name>