**Web Application Development Lab**

**Git and GitHub**

**What is Git?**

Git is a **distributed version control system (DVCS)** designed to handle everything from small to large projects with speed and efficiency. Created by **Linus Torvalds in 2005**, Git allows multiple developers to collaborate by tracking code changes, managing different versions, and ensuring smooth teamwork.

**Why use Git?**

* **Version Control:** Tracks modifications to files, enabling rollback to previous states.
* **Collaboration:** Multiple developers can work on the same project simultaneously without conflicts.
* **Branching & Merging:** Developers can work on features separately and later merge them into the main project.
* **Local & Distributed:** Every developer has a complete copy of the project’s history, ensuring reliability even without a network connection.

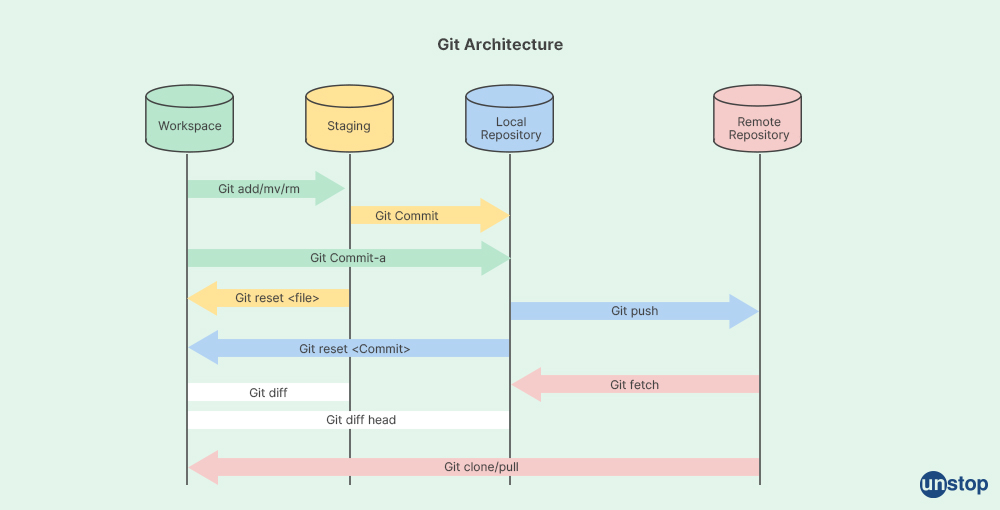
**What is GitHub?**

GitHub is a **web-based platform** for hosting Git repositories. It extends Git’s capabilities by providing a **remote repository**, making it easy to share projects, collaborate, and manage code with version control in the cloud.

**Why use GitHub?**

* **Remote Repository:** Stores and shares Git repositories on the cloud.
* **Collaboration Tools:** Features like pull requests, issues, and discussions improve teamwork.
* **Integration & Automation:** Supports CI/CD, code reviews, and integrations with various tools.
* **Security & Access Control:** Provides role-based access to repositories for secure collaboration.

**Git Architecture**



**The architecture consists of four main components:**

**1. Workspace (Working Directory)**

* This is where you modify files in your project.
* You can create, edit, delete, move, or rename files here.
* Changes made in the workspace need to be staged before committing.

🔹 Common Commands:

* git add <file> → Adds changes to the staging area.
* git rm <file> → Removes a file.
* git mv <file> → Moves or renames a file.

**2. Staging Area (Index)**

* A temporary storage area where Git tracks changes before committing them.
* Changes must be staged before being committed to the local repository.

🔹 Common Commands:

* git commit -a → Stages and commits changes directly.
* git reset <file> → Unstages a file (removes it from staging without deleting changes).

**3. Local Repository (.git Directory)**

* A hidden directory inside your project that contains the entire history of the repository.
* Every commit you make is stored here.

🔹 Common Commands:

* git commit -m "message" → Saves changes to the local repository.
* git reset <commit> → Reverts to a previous commit.
* git diff → Shows differences between versions.

**4. Remote Repository (GitHub, GitLab, Bitbucket, etc.)**

* A version of your repository stored on a remote server.
* Used for collaboration with other developers.

**Step-by-Step Guide: Creating & Pushing a Git Repository to GitHub**

**1. Create a Local Repository**

First, open a terminal and navigate to your project folder or create a new one:

mkdir my-project # Create a new folder

cd my-project # Navigate into the folder

Now, initialize Git in this directory:

git init

This will create a hidden .git directory, marking this as a Git repository.

**2️. Create and Track a File**

Create a simple file (README.md) and add some content:

echo "# My Project" > README.md

Then, check the status of your repo:

git status

You’ll see that README.md is **untracked**.

**3️ Stage & Commit Changes**

Add the file to the **staging area**:

git add README.md

Commit the file to the local repository:

git commit -m "Initial commit"

Now, the changes are stored in the local repository.

**4️ Create a Repository on GitHub**

1. Go to [GitHub](https://github.com/) and log in.
2. Click the **+ (New Repository)** button.
3. Enter a repository name (e.g., my-project).
4. Keep it **public** or **private**, as needed.
5. **Do NOT** check “Initialize this repository with a README” (we already have one).
6. Click **Create Repository**.
7. Copy the remote URL (it will look something like this):
8. https://github.com/your-username/my-project.git

**5️ Connect Local Repository to GitHub**

Now, link your local repo to the remote repo:

git remote add origin https://github.com/your-username/my-project.git

Verify that the remote was added correctly:

git remote -v

You should see:

origin https://github.com/your-username/my-project.git (fetch)

origin https://github.com/your-username/my-project.git (push)

**6️ Push Code to GitHub**

Push the local repository to GitHub:

git branch -M main # Ensure you're on the main branch

git push -u origin main

This sends your local files to GitHub. You can now see them in your GitHub repository.

**7️ Make Additional Changes & Push Again**

If you modify a file (e.g., README.md), repeat these steps:

git add .

git commit -m "Updated README"

git push origin main

This updates your repository on GitHub.