**Documentation on Git and GitHub**

**1. Introduction to Git**

Git is a **distributed version control system (DVCS)** that helps developers track changes in their source code during software development. It is designed to handle everything from small to very large projects with speed and efficiency.

**Key Features:**

* Distributed development
* Fast performance
* Data integrity
* Branching and merging

**2. Introduction to GitHub**

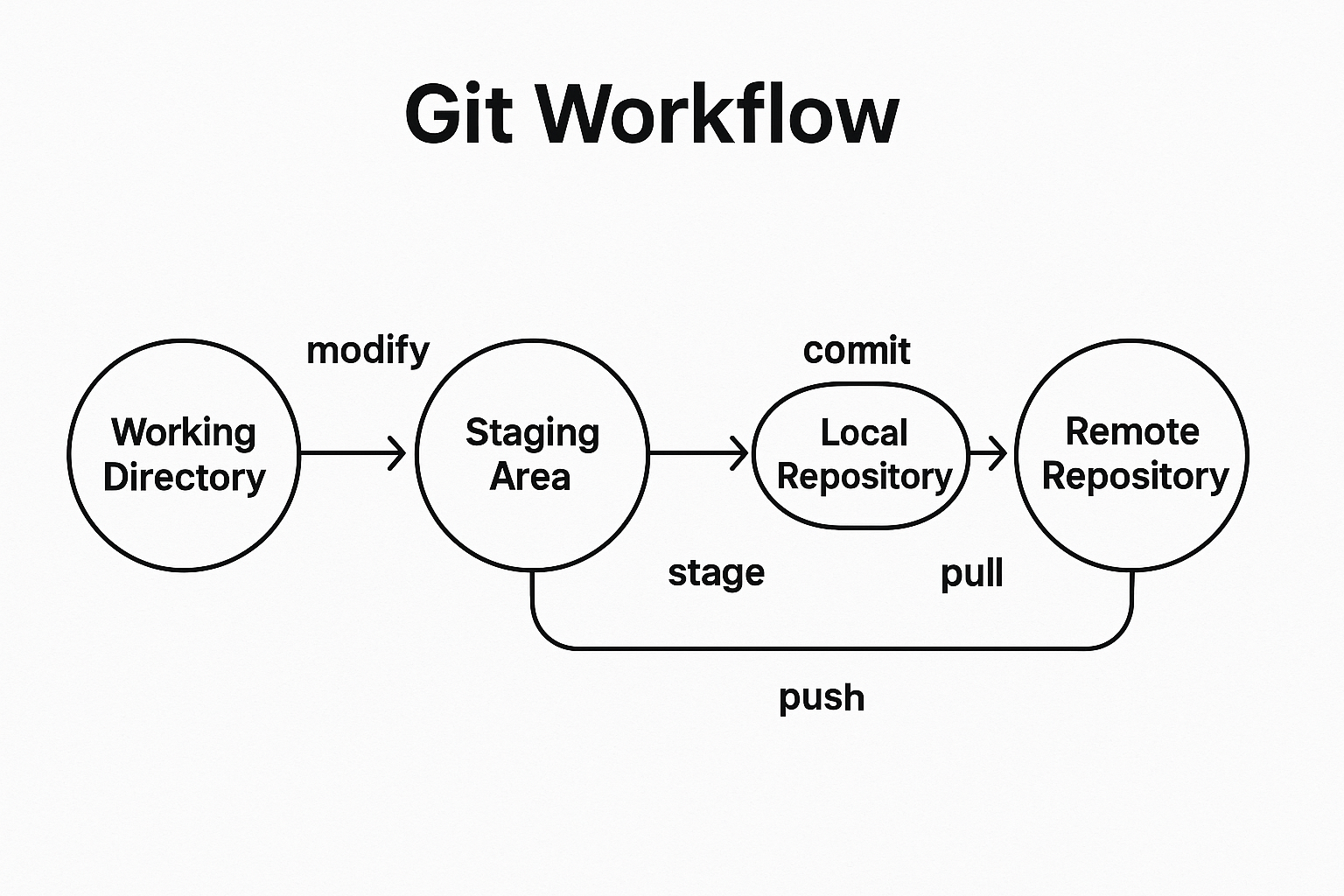
**GitHub** is a **web-based hosting service** for Git repositories. It provides a graphical interface and access control, as well as several collaboration features such as:

* Issue tracking
* Code review
* Project management

GitHub is not Git itself, but it makes using Git easier with cloud-based repository hosting.

**3. Common Git Terminology**

* **Repository (repo):** A project folder that Git tracks.
* **Commit:** A snapshot of changes.
* **Branch:** A separate line of development.
* **Merge:** Combining changes from different branches.
* **Clone:** Copying a remote repository locally.
* **Push:** Uploading local changes to a remote repository.
* **Pull:** Fetching and merging changes from a remote repo.
* **Staging area:** Area where changes are prepared before a commit.



**4. Git Commands and Explanations**

**4.1 Configuration**

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

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

* Sets user identity globally.

**4.2 Creating a Repository**

git init

* Initializes a new Git repository locally.

**4.3 Cloning a Repository**

git clone <repository\_url>

* Clones a remote repo to local machine.

**4.4 Checking Status**

git status

* Shows the current state of the working directory and staging area.

**4.5 Adding Files**

git add <file>

* Adds a file to the staging area.

git add .

* Adds all files to the staging area.

**4.6 Committing Changes**

git commit -m "Your commit message"

* Saves changes to the local repository.

**4.7 Viewing Commit History**

git log

* Shows the commit history.

**4.8 Branching**

git branch

* Lists all branches.

git branch <branch\_name>

* Creates a new branch.

git checkout <branch\_name>

* Switches to a branch.

git checkout -b <branch\_name>

* Creates and switches to a new branch.

**4.9 Merging**

git merge <branch\_name>

* Merges another branch into the current one.

**4.10 Pushing Changes**

git push origin <branch\_name>

* Pushes changes to the remote repository.

**4.11 Pulling Changes**

git pull

* Fetches and merges changes from the remote repository.

**4.12 Deleting Branches**

git branch -d <branch\_name>

* Deletes a local branch.

git push origin --delete <branch\_name>

* Deletes a remote branch.

**4.13 Stashing Changes**

git stash

* Temporarily stores uncommitted changes.

git stash pop

* Applies the stashed changes.

**5. Working with GitHub**

**Create a New Repo:**

* Go to GitHub > New repository > Name it > Create repository.

**Link Local Repo with GitHub:**

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

git push -u origin main

**Forking and Pull Request:**

* **Fork** creates a copy of someone else's repo.
* Make changes > Push > Create a **Pull Request** to contribute back.

**6. Best Practices**

* Commit frequently with meaningful messages.
* Use branches for features/bugs.
* Pull before you push.
* Use .gitignore to ignore unnecessary files.
* Review code before merging (PRs).

**7. Conclusion**

Git and GitHub are essential tools in modern software development, enabling collaboration, version control, and streamlined workflows. Mastering the commands and concepts ensures efficient and organized project management.

**✅ Tips to Avoid Conflicts (Merge Conflicts)**

* Pull latest changes before starting work: git pull
* Use smaller, focused branches.
* Communicate with teammates when editing shared files.
* Use git diff and git status often to track what’s changing.