## ➢ Introduction to Git

Git is a version control system that helps you manage and track changes to files over time. It is especially helpful when working on coding or writing projects, as it allows you to save different versions, collaborate with others, and prevent data loss.

## ➢ Why Use Git?

Git helps in :

- **Tracking changes**: Know who changed what and when.
- **Version control:** Revert back to previous versions easily.
- **Collaboration:** Work together with others without conflict.
- **Backup:** Sync your work to platforms like GitHub.
- **Experiment safely:** Try new ideas without affecting the main project.

## ➢ Getting Started

### ❖ How to Install Git

- **On Windows:**
  - Go to https://git-scm.com/
  - Download and run the installer.
  - Follow the installation steps using default options.

- **Verify Installation:**

  Open terminal or command prompt and run:

```
git –version
```

```
This will show the installed Git version.
```

### ❖ Setting Up Git (Initial Setup)

You only need to do this once after installing Git:

```
git config --global user.name "Your Name"
git config --global user.email your@email.com
```

These settings will be used in your commits.

## ➢ Creating and Managing Repositories

- ### What is a Repository?
  A repository (repo) is a folder where Git tracks all your project files and changes.

- ### Create a New Git Repository:

  ```
  git init
  ```

  This turns your folder into a Git project.

- ### Clone a Remote Repository:

If you are working on an existing project(like from GitHub):

```
git clone https://github.com/username/repo-name.git
```

`This creates a local copy of the project on your computer.`

## ➢ Tracking Changes in Files

- ### Check File Status:

  ```
  git status
  ```

`Displays current changes, staged files, and untracked files.`

- ### Add Files to Staging:

  ```
  git add file.txt
  ```

`Adds a specific file to the staging area.`

```
git add .
```

`Adds all modified and new files in the current folder.`

### ➢ Save Changes (Commit):

```
git commit -m "Write a clear message here"
```

`Saves a snapshot of the staged changes with a message.`

## ➢ Branching and Merging

- **Create a Branch:**

```
git branch new-feature
```

`Creates a new branch for separate development.`

- **Switch to a Branch:**

  ```
  git checkout new-feature
  ```

Switches to the branch named new-feature.

- **Create and Switch Together:**

  ```
  git checkout -b new-feature
  ```

Shortcut to create and switch to a branch.

- **Merge a Branch into Main:**

  ```
  git checkout main
  ```

Switch to the main branch.

  ```
  git merge new-feature
  ```

Merge changes from new-feature into main.

## ➢ Working with Remote Repositories

- **Add a Remote :**

  ```
  git remote add origin https://github.com/username/project.git
  ```

Adds a remote connection called origin.

- **Push Your Code Online:**

  ```
  git push origin main
  ```

Sends your local commits to the remote repository.

- **Pull Updates from Online:**

  ```
  git pull origin main
  ```

Fetches and merges changes from the online version.

## ➢ Undoing and Fixing Mistakes

- **Unstage a File:**

  ```
  git reset file.txt
  ```

Removes file.txt from the staging area.

- **Revert changes to a File:**

  ```
  git checkout -- file.txt
  ```

Restores the last committed version of the file.

- **Undo a Commit:**

```
git revert <commit-id>
```

Creates a new commit that reverses changes made in the specified commit.

- Hard Reset to Previous Version:

```
git reset --hard <commit-id>
```

Moves the branch pointer to an older commit and deletes all changes after it. Use with caution.

## Viewing Commit History

- **View Full History:**

```
git log
```

Shows detailed information about each commit.

- **View in One Line:**

```
git log --oneline
```

Displays each commit on one line for quick viewing.

- View with Visual Graph:

```
git log --oneline --graph –all
```

Shows all branches and commits in a visual structure.

## .gitignore File

.gitignore tells Git what files and folders to skip.

**Sample .gitignore:**

```
*.log
node_modules/
.env
__pycache__/
```

These files will not be tracked by Git.

## Stashing Temporary Changes

- **Save Changes Without Committing:**

```
git stash
```

Temporarily saves uncommitted changes.

- **View All Stashes:**

  ```
  git stash list
  ```

  ```
  Lists all previously stashed items.
  ```

- **Apply Last Stash:**

  ```
  git stash apply
  ```

  ```
  Restores the most recent stash.
  ```

- **Remove Last Stash:**

  ```
  git stash drop
  ```

  ```
  Deletes the most recent stash.
  ```

## ➢ Best Practices for Using Git

- Write clear, short commit messages.
- Commit frequently after small changes.
- Use branches for new features or testing.
- Pull updates before pushing.
- Review changes with git status and git diff.
- Use .gitignore to avoid committing temporary or sensitive files.

## ➢ Git Command Summary Table with Explanation

| Task | Command | Explanation |
|---|---|---|
| Initialize Git | git init | Starts a new Git repository in your folder. |
| Clone Repository | git clone <url> | Copies an online repository to your local machine. |
| Add Files to Stage | git add file.txt or git add . | Prepares changes for the next commit. . adds all files. |
| Commit Changes | git commit -m "message" | Saves a snapshot of staged files with a message. |
| Check Status | git status | Shows which files are changed, staged, or untracked. |
| View Commit Log | git log | Displays the full history of commits. |
| Short Log View | git log --oneline | One-line summary of each commit. |
| Visual Commit Graph | git log --oneline --graph --all | Visual representation of branches and commits. |
| Create New Branch | git branch branch-name | Makes a new branch. Useful for feature development. |

| | | |
|---|---|---|
| Switch Branch | git checkout branch-name | Moves to the specified branch. |
| Create and Switch Branch | git checkout -b branch-name | Creates a new branch and switches to it in one step. |
| Merge Branch | git merge branch-name | Merges changes from another branch into the current one. |
| Add Remote Repo | git remote add origin <url> | Connects your local Git to an online repository. |
| Push to Remote | git push origin branch-name | Uploads local commits to the online repository. |
| Pull from Remote | git pull origin branch-name | Fetches and merges updates from the online repo. |
| Unstage File | git reset file.txt | Removes file from staging but keeps the changes. |
| Revert File Changes | git checkout -- file.txt | Restores last committed version of the file. |
| Revert a Commit | git revert <commit-id> | Undoes a commit by creating an opposite commit. |
| Reset to Old Commit | git reset --hard <commit-id> | Rewinds your project to a past state. Use carefully. |
| Save Temporary Work | git stash | Saves changes without committing, clears working area. |
| See Stashed Work | git stash list | Lists all saved stashes. |
| Restore Last Stash | git stash apply | Brings back the most recently stashed work. |
| Delete Last Stash | git stash drop | Removes the top stash from the list. |

## ➢ Useful Resources

- Git Official Documentation: https://git-scm.com/doc
- GitHub Docs: https://docs.github.com/
- Learn Git Visually: https://learngitbranching.js.org/