GIT AND GITHUB NOTES.

**Introduction to Git**

Git is a **distributed version control system** that helps developers track changes in source code, collaborate with others, and maintain code history.

**Why Use Git?**

* Tracks changes in files over time.
* Allows multiple developers to work on a project simultaneously.
* Enables easy collaboration and code sharing.
* Provides a backup of the codebase.
* Supports branching and merging for better project management.

**Installing Git**

* **Windows**: Download from [git-scm.com](https://git-scm.com/) and install.
* **Linux**: Run sudo apt install git (Ubuntu/Debian) or sudo yum install git (CentOS/RHEL).
* **Mac**: Install using Homebrew: brew install git.

**Configuring Git**

After installing Git, configure your user details:

# Set your name

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

# Set your email

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

# Verify configuration

git config --list

**Basic Git Commands**

**Initializing a Repository**

git init

This creates a hidden .git directory, marking the folder as a Git repository.

**Checking Repository Status**

git status

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

**Adding Files to Staging Area**

git add filename # Add a single file

git add . # Add all modified files

**Committing Changes**

git commit -m "Initial commit"

This records the changes in the repository.

**Viewing Commit History**

git log

Shows the commit history.

**Checking Differences**

git diff # Show changes in unstaged files

git diff --staged # Show changes in staged files

**Viewing All Versions**

git log --oneline # Shows commit history in a single line format

git reflog # Shows all references, including HEAD movements

**Switching to a Previous Version**

git checkout <commit-hash>

To go back to the latest commit:

git checkout main

**Undoing Changes**

**Undo Changes in the Working Directory**

git checkout -- filename # Discard changes in a file

**Undo Changes in the Staging Area**

git reset HEAD filename # Unstage a file but keep changes

**Undo a Commit**

git reset --soft HEAD~1 # Undo last commit, keep changes staged

git reset --mixed HEAD~1 # Undo last commit, keep changes unstaged

git reset --hard HEAD~1 # Undo last commit, discard changes

**Reverting a Commit (without losing changes)**

git revert <commit-hash>

This creates a new commit that reverses the changes of the specified commit.

**Working with Branches**

Branches allow working on different features without affecting the main code.

**Creating a New Branch**

git branch feature-branch

**Switching Branches**

git checkout feature-branch

# OR

git switch feature-branch

**Creating and Switching to a New Branch**

git checkout -b new-branch

# OR

git switch -c new-branch

**Merging Branches**

git checkout main

git merge feature-branch

**Deleting a Branch**

git branch -d feature-branch

**GitHub**

GitHub is a cloud-based platform for hosting Git repositories, enabling collaboration.

**Creating a GitHub Repository**

1. Go to [GitHub](https://github.com/).
2. Click **New Repository**.
3. Provide a name, description, and visibility.
4. Click **Create Repository**.

**Connecting Local Repository to GitHub**

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

git branch -M main

git push -u origin main

**Cloning a Repository**

git clone https://github.com/username/repository.git

**Pulling Latest Changes from GitHub**

git pull origin main

**Pushing Changes to GitHub**

git push origin main

**Working with Collaborators**

* Invite team members under **Settings > Collaborators**.
* Contributors can clone, pull, and push changes based on access levels.

**Advanced Git Commands**

**Stashing Changes (Temporary Storage)**

git stash # Save changes

To retrieve the stashed changes:

git stash pop # Apply and remove stash

**Resolving Merge Conflicts**

1. Identify conflicting files using git status.
2. Open the file and resolve conflicts manually.
3. Add the resolved file using git add filename.
4. Commit the changes using git commit.

**Deleting Remote Branches**

git push origin --delete branch-name

**GitHub Forks & Pull Requests**

**Forking a Repository**

* Used to copy someone else’s repository to your GitHub account.
* Click **Fork** on GitHub.

**Creating a Pull Request (PR)**

1. Make changes in a forked repository.
2. Push changes and go to GitHub.
3. Click **New Pull Request**.
4. Describe changes and submit the PR.

**Reviewing & Merging PRs**

* Maintainers review PRs and merge them into the main branch.
* Use **GitHub Actions** for CI/CD integration.

**GitHub Issues & Projects**

* **Issues**: Used to report bugs, suggest features, or track tasks.
* **Projects**: Organize work using Kanban-style boards.

**Best Practices for Git & GitHub**

✅ Write clear commit messages.  
✅ Use branches for feature development.  
✅ Regularly push changes to GitHub for backup.  
✅ Review code via Pull Requests.  
✅ Use .gitignore to exclude unnecessary files.

**Conclusion**

Git and GitHub are essential tools for modern software development. Understanding how to use them efficiently improves collaboration, code management, and version control. 🚀