**Git Deep Dive**

Git is a distributed version control system that enables developers to track changes made to their codebase over time. Here's a deep dive into Git:

Key Concepts:

1. Repository (Repo): The central location where all code changes are stored.

2. Commit: A snapshot of changes made to the codebase.

3. Branch: A separate line of development in the repo.

4. Merge: Combining changes from two branches.

5. Remote: A remote repository, often on a server, that syncs with the local repo.

Basic Git Workflow:

1. Clone: Create a local copy of the remote repo.

2. Pull: Fetch changes from the remote repo and merge into the local repo.

3. Edit: Make changes to the codebase.

4. Add: Stage changes for the next commit.

5. Commit: Create a new snapshot of changes.

6. Push: Send changes to the remote repo.

Advanced Git Concepts:

1. Branching Strategies: Feature branching, release branching, and Git Flow.

2. Merging: Fast-forward, no-fast-forward, and squash merging.

3. Rebase: Reapply commits on top of another branch.

4. Stash: Temporarily store changes and reapply later.

5. Submodules: Manage nested repositories.

6. Hooks: Custom scripts triggered by Git events.

7. Git Attributes: Customize behavior for specific files.

Git Commands:

1. git init: Initialize a new local repo.

2. git clone: Clone a remote repo.

3. git add: Stage changes.

4. git commit: Create a new snapshot.

5. git log: View commit history.

6. git branch: Create, list, or delete branches.

7. git merge: Combine changes from two branches.

8. git remote: Manage remote repositories.

9. git fetch: Fetch changes from a remote repo.

10. git pull: Fetch and merge changes.

11. git push: Send changes to a remote repo.

Best Practices:

1. Use meaningful commit messages.

2. Keep commits focused and small.

3. Use branches for feature development.

4. Regularly pull and merge changes.

5. Use Git hooks for automation.

6. Document your Git workflow.

Using Git

1. git init

- Usage: Initialize a new local repository

- Example: git init

2. git clone

- Usage: Clone a remote repository

- Example: `git clone (link unavailable)

3. git add

- Usage: Stage changes for the next commit

- Example: git add file.txt or git add . (stage all changes)

4. git commit

- Usage: Create a new snapshot of changes

- Example: git commit -m "Commit message"

5. git log

- Usage: View commit history

- Example: git log (view all commits) or git log -n 5 (view last 5 commits)

6. git branch

- Usage: Create, list, or delete branches

- Examples:

- git branch feature/new-feature (create new branch)

- git branch (list all branches)

- git branch -d feature/new-feature (delete branch)

7. git merge

- Usage: Combine changes from two branches

- Example: git merge feature/new-feature (merge into current branch)

8. git remote

- Usage: Manage remote repositories

- Examples:

- `git remote add origin (link unavailable) (add remote)

- git remote -v (list remotes)

9. git fetch

- Usage: Fetch changes from a remote repository

- Example: git fetch origin

10. git pull

- Usage: Fetch and merge changes from a remote repository

- Example: git pull origin master

11. git push

- Usage: Send changes to a remote repository

- Example: git push origin master

12. git status

- Usage: View the status of changes

- Example: git status

13. git diff

- Usage: View changes between commits or branches

- Examples:

- git diff (view changes between current and last commit)

- git diff feature/new-feature (view changes between current and feature branch)

14. git reset

- Usage: Reset changes or commits

- Examples:

- git reset file.txt (unstage file)

- git reset --hard (reset all changes and commits)

15. git stash

- Usage: Temporarily store changes and reapply later

- Examples:

- git stash (stash changes)

- git stash apply (reapply stashed changes)