Git revert,rebase,clone,tracking remote repos,gitignore,stash,restore,reset,remote add repository,merge vs rebase,diff command and its types,

1. Git Revert

- **Purpose**: Used to undo changes in a commit by creating a new commit that reverses the changes made in the specified commit.
- **Command**:

  ```
  git revert <commit-hash>
  ```

- **Usage**: Ideal for undoing changes in a public history without rewriting commit history.
- **Example**:

  ```
  git revert a1b2c3d
  ```

2. Git Rebase

- **Purpose**: Integrates changes from one branch into another by moving or reapplying commits. It rewrites commit history.
- **Command**:

  ```
  git rebase <branch-name>
  ```

- **Usage**: Often used to maintain a linear project history.
- **Example**:

  ```
  git checkout feature-branch
  git rebase main
  ```

3. Git Clone

- **Purpose**: Creates a copy of an existing repository (including all branches, history, and tags).
- **Command**:

  ```
  git clone <repository-url>
  ```

- **Usage**: Used when you need to create a local copy of a remote repository.
- **Example**:

  ```
  git clone https://github.com/user/repo.git
  ```

4. Tracking Remote Repos

- **Purpose**: Allows a local branch to track a remote branch for easier pushing and pulling.
- **Usage**: When creating a new branch, use `--set-upstream` to track a remote branch.

- **Command**:

```
git branch --set-upstream-to=origin/<branch-name>
```

5. .gitignore

- **Purpose**: Specifies which files or directories Git should ignore in a repository.
- **Usage**: Create or modify `.gitignore` to list files and folders to be ignored.
- **Example**:

```
*.log
*.tmp
node_modules/
```

- **Note**: Changes to `.gitignore` won't remove files already tracked by Git.

6. Git Stash

- **Purpose**: Temporarily saves changes in the working directory that are not ready to commit.
- **Command**:

```
git stash
```

- **Usage**: Use when you need to switch contexts but aren't ready to commit your changes.
- **Example**:

```
git stash save "work-in-progress"
```

7. Git Restore

- **Purpose**: Restores changes in your working directory or staging area.
- **Command**:

```
git restore <file>
```

- **Usage**: Restores files to the state of the last commit (can be used with `--staged` to unstage files).
- **Example**:

```
git restore --staged file.txt
```

8. Git Reset

- **Purpose**: Resets the index (staging area) and working directory to a previous commit. Can affect commit history.
- **Command**:

```
git reset <commit-hash>
```

- **Types**:
  - `--soft`: Keeps changes in the working directory and staging area.
  - `--mixed`: Resets the index, keeps working directory changes.
  - `--hard`: Resets both the index and working directory (permanently removes changes).
- **Example**:

```
git reset --hard a1b2c3d
```

9. Git Remote Add Repository

- **Purpose**: Adds a remote repository URL to the local repository to enable syncing.
- **Command**:

```
git remote add <remote-name> <repository-url>
```

- **Usage**: Typically used to link your local repository with a remote (e.g., on GitHub).
- **Example**:

```
git remote add origin https://github.com/user/repo.git
```

10. Merge vs Rebase

- **Merge**:
  - Combines two branches' histories into one, maintaining the commit history of both branches.
  - Results in a merge commit.
  - **Use Case**: When you want to preserve the history and context of both branches.
  - **Command**:

```
git merge <branch-name>
```

- **Rebase**:
  - Reapplies commits from one branch onto another, resulting in a linear commit history.
  - Does not create a merge commit.
  - **Use Case**: When you want a clean, linear history without merge commits.
  - **Command**:

```
git rebase <branch-name>
```

11. Git Diff and Its Types

- **Purpose**: Compares changes in files between commits, branches, or working directory and staging area.
- **Command**:

```
git diff
```

- **Types**:

- **Unstaged changes**: Shows the difference between the working directory and the index (staging area).

- **Staged changes**: Shows the difference between the index and the last commit.

  ```
  git diff --staged
  ```

- **Comparing commits**: Compares two commits.

  ```
  git diff <commit-hash1> <commit-hash2>
  ```

- **Comparing branches**: Shows the difference between two branches.

  ```
  git diff <branch1>..<branch2>
  ```