Day 9 : Deep Dive in Git

Kunal Maurya · Mar 3, 2023 · 6 min read

TABLE OF CONTENTS

Git is a powerful tool for version control and source code management. It allows developers to track changes made to their codebase, collaborate with others, and revert changes if necessary. In this blog, we'll go over some of the basic Git commands.

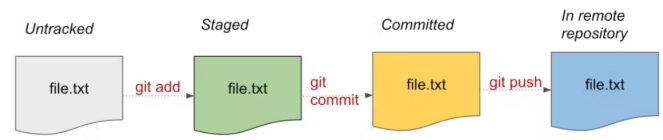
What is Repository?

In Git, a repository is a collection of files, folders, and the history of changes made to those files and folders. It is a central place where developers can store their code and collaborate with others. Git repositories are commonly used for version control, allowing developers to track changes, revert to previous versions, and collaborate on code.

There are two types of repositories in Git: local repositories and remote repositories.

1. **Local repositories**: A local repository is stored on your computer's hard drive. When you create a new Git repository using the `git init` command, Git creates a .git directory in your project folder. This directory contains all the metadata that Git uses to track changes to your files. You can use Git commands to manage your local repositories, such as `git add`, `git commit`, and `git log`.
2. **Remote repositories:** A remote repository is a repository that is hosted on a server, such as GitHub, GitLab, or Bitbucket. Remote repositories allow developers to collaborate on code by sharing their changes with others. You can use Git commands like `git push` and `git pull` to synchronize changes between your local repository and the remote repository.

Git Lifecycle



The Git lifecycle consists of four stages: the working directory, the staging area, the local repository, and the remote repository.

1. **Working Directory:** The working directory is the folder on your local machine where you keep your files. When you create a new file or modify an existing file, those changes are made in the working directory.
2. **Staging Area:** The staging area is a virtual space where you can prepare files for commit. You use the `git add` command to add files from the working directory to the staging area. When you add files to the staging area, Git is aware of the changes, but they have not yet been committed to the local repository.
3. **Local Repository:** The local repository is where Git stores the committed changes. When you use the `git commit` command, Git creates a new commit in the local repository that contains a snapshot of the changes in the staging area. The local repository is stored on your local machine in a .git directory that is located in the root of your project folder.
4. **Remote Repository:** The remote repository is a repository that is hosted on a remote server, such as GitHub or GitLab. You can use the `git push` command to push changes from your local repository to the remote repository. Conversely, you can use the `git pull` command to pull changes from the remote repository to your local repository.

Git Commands

- **Git config:** `git config` is a command-line tool that allows you to configure Git settings. Git uses configuration files to store settings like your name, email address, default branch name, and more. `git config` is a powerful tool that allows you to customize your Git environment and streamline your workflow.
  Example:
  Set your name and email address:

```
git config --global user.name "Kunal"
git config --global user.email Kunal@example.com
```

- **git init:** `git init` is used to initialize a new Git repository in the current directory. When you run this command, Git creates a new repository and adds a .git directory to your project. This directory contains all the necessary metadata that Git needs to track changes to your files.
  Syntax:

```
git init
```

- **git add:** `git add` is used to add files to the staging area. When you make changes to your codebase, Git will not automatically track these changes. You need to explicitly add them to the staging area before committing them.
  Syntax:

  ```
  git add <file>
  ```

  Example:

  ```
  git add index.html
  ```

  This command adds the `index.html` file to the staging area.
  You can also use `git add .` to add all changes to the staging area.

- **git commit:** `git commit` is used to commit changes to the local repository. When you commit changes, Git creates a new commit that contains a snapshot of your codebase at that point in time.
  Syntax:

  ```
  git commit -m "commit message"
  ```

  Example:

  ```
  git commit -m "Added login functionality"
  ```

  This command commits the changes to the local repository with a commit message that describes the changes.

- **git status:** `git status` is used to display the current status of the repository. It shows which files are tracked, untracked, or modified.
  Syntax:

  ```
  git status
  ```

- **git push:** `git push` is used to push changes to a remote repository.

Syntax:

```
git push <remote> <branch>
```

Example:

```
git push origin master
```

This command pushes the changes to the `master` branch of the `origin` remote.

- **git pull:** `git pull` is used to pull changes from a remote repository. If you're collaborating with others, you'll likely need to pull changes that others have made to the codebase.

Syntax:

```
git pull <remote> <branch>
```

Example:

```
git pull origin master
```

This command pulls changes from the `master` branch of the `origin` remote.

- **git clone:** The `git clone` command is used to create a copy of a Git repository on your local machine. Here's an example:

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

Output:

```
Cloning into 'repo'...
remote: Enumerating objects: 100, done.
remote: Counting objects: 100% (100/100), done.
remote: Compressing objects: 100% (84/84), done.
Receiving objects: 100% (100/100), 15.30 KiB | 5.10 MiB/s, done.
Resolving deltas: 100% (16/16), done.
```

- **git diff:** The `git diff` command is used to show the differences between two versions of a file.
  Here's an example:

```
git diff HEAD~2..HEAD~1 file.txt
```

Output:

```
diff --git a/file.txt b/file.txt
index e69de29..eb77e78 100644
--- a/file.txt
+++ b/file.txt
@@ -0,0 +1 @@
+This is a new line.
```

- **git log:** The `git log` command is used to show a history of commits.
  Here's an example:

```
git log --oneline
```

Output:

```
b7fbc12 Add new feature
32fd09c Fix bug
a0e45e9 Initial commit
```

- **git rm:** The `git rm` command is used to remove files from Git.
  Here's an example:

```
git rm file.txt
```

Output:

```
rm 'file.txt'
```

- **git mv:** The `git mv` command is used to move or rename files in Git.
  Here's an example:

```
git mv old-file.txt new-file.txt
```

Output:

```
rename old-file.txt => new-file.txt (100%)
```

- **git remote:** The `git remote` command is used to manage remote repositories.
  Here's an example:

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

Git GitHub Devops Gitcommands DevOps Journey

MORE ARTICLES

Kunal Maurya

Day 8 : Basic Git & GitHub for DevOps Engineers.

Git and GitHub are two closely related technologies that have revolutionized the way software develo...

Kunal Maurya

Day 7 : Understanding package manager and systemctl

Linux is a popular operating system that is known for its flexibility and customization. One of the ...

Kunal Maurya

Day 6: File Permissions and Access Control Lists

In Linux, file permissions are a crucial aspect of ensuring that your files are secure and can only ...