Git Basics

A gentle introduction to the git command-line

History

Git was created by Linus Torvalds in 2005 specifically to version control and manage the Linux Kernel Project.

Written by Linux Kernel Developers for Linux Kernel Developers.

Let’s start by creating a new git repository

Open a terminal and run the following commands:

```
mkdir project

cd project

git init
```

What did we just do?

We just created an empty git repository.

Let’s take a look at .git directory:

```
ls -la .git
```

Also, let take a look at the state of the git workspace:

```
git status
```

Let’s add a file to our working directory

Type the following command:

```
echo ‘hello world’ > hello.txt
```

Let’s take a look at the status again:

```
git status
```

The status gives us a clue on what we need to do next:

```
git add .
```

Git knows we have a new file with the intention to add it to the system but we have not committed it yet.

Let’s commit our file to the repo

Type the following:

```
git commit -m “Commit first file to the repo”
```

Congratulations! We have made our first commit to master!

How do we know we are on master?

```
git branch
```

Curious about our .git directory?

Let’s see what our .git directory looks like now:

```
ls -la .git/objects
```

The objects directory contain the actual files that our stored. They are organized by subdirectories of the first 2 characters of the SHA1 of the file.

For example, look at subdirectory **3b** that contains the file: **18e512dba79e4c8300dd08aeb37f8e728b8dad**

For those who are curious, run: `git hash-object hello.txt`

Let’s create a new branch from master

Type the following in your terminal:

```
git checkout -b dev_llam
```

This will create a new branch and switch to that branch.

We should be able to see both branches now:

```
git branch
```

Let’s make a change on our new branch

Edit hello.txt by adding a 2nd line saying who you are. (Example: “I am Liz.”)

If you check the status, you will see git knows we made changes:

```
git status
```

We want to add our changes:

```
git add .
```

Now we want to commit them:

```
git commit -m “Saying who I am”
```

Let’s take a look at our history so far

Type in the terminal:

```
git log --oneline
```

We now have two commits in our history in our dev branch.

Let’s check on master:

```
git checkout master
git log --oneline
cat hello.txt
```

Time to merge!

We want to merge our changes from dev to master.

Remember, the target branch is always the branch you are on. Use ‘git branch’ if you are not sure where you are.

```
git merge dev_llam
```

Now we should see that master has the same changes as our dev branch.

```
git log --oneline
```

```
cat hello.txt
```

What about cloning?

Open another terminal window and clone the repo we just created:

```
git clone /path/to/git/project project2
```

You should see an exact copy of what we had in the original project,

```
cd project2

git log --oneline

cat hello.txt
```

What about pulling?

To keep our repo updated, we need to pull from the original/remote.

Go to the **first terminal** window with the original **project** open and make a change and commit it.

Go back to the **second terminal** window with **project2** open and check its status:

```
git status
```

Let’s pull!

Our cloned project (project2) does not know it is out of date with the original.

Let’s start by fetching the changes (note: fetching is different from pulling).

```
git fetch origin dev_llam

git status
```

Now our repo is updated but our working directory is not.

```
git pull

git log --oneline
```

Yay! We’re done!

You now know how to:

- Create a new git repository
- Add and commit files to a repo
- Create and merge branches
- Clone a repository
- Fetch and pull changes

CONGRATULATIONS!

Questions?

Feel free to reach out to me on Twitter @grepliz