Accidental Deletion

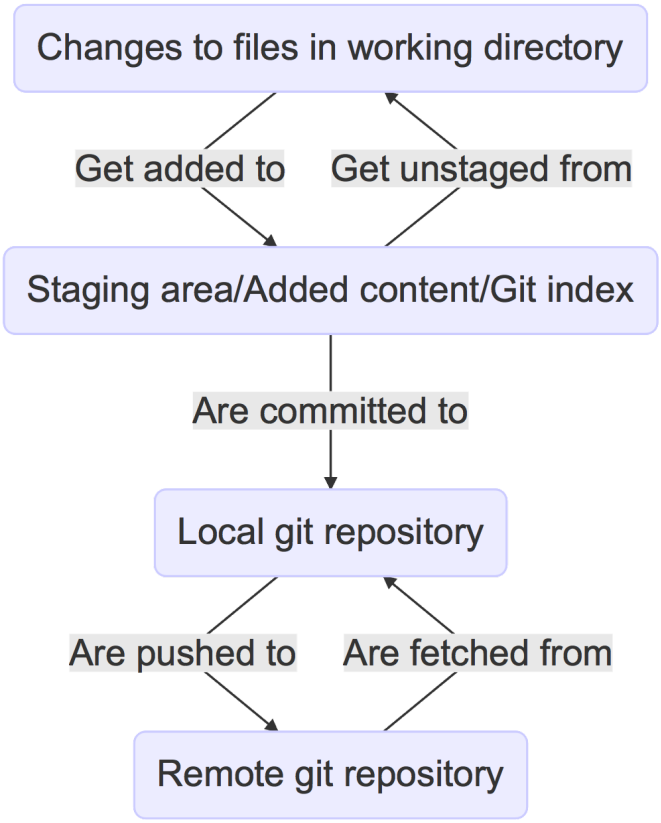
See what happens when you accidentally delete a file.

We'll cover the following

- The four stages of Git repositories
- Visualizing the repository's history
- Cloning the repository for testing purposes
- Making a "mistake"

The four stages of Git repositories

Recall the four stages of data in Git repositories:



The four phases of Git content

Visualizing the repository's history

Run these commands in which you look at the repository, and then make a disastrous mistake.

```
1   git log
```

This shows you a default history of the repository. Page through it a few times by hitting space or down. You will see how far it goes back. Hit `q` to stop viewing it, and return to the command line.

```
2   git log --oneline
```

Another way to view the log is one line per commit, which is much more concise and useful for many purposes. Obviously, some information is lost here.

```
3   git log --oneline --graph
```

You've added the `--graph` flag, and now you get a visual representation of the history. Parsing this graph can be tricky; don't worry about understanding it exactly. But keep it in mind. It is helpful if you ever have to figure out what went on in a repository's past.

Using `git log` to view the history on the terminal is one of the most powerful tools you will have at your disposal if you are responsible for a shared repository.

Cloning the repository for testing purposes

Now you're going to clone the repository so that you've got a copy to mess around with.

```
4   cd ..
5   git clone shutit cloned_shutit
6   cd cloned_shutit
```

Terminal 1

Terminal

It's worth, at this point, reminding yourself of the `.git` folder.

```
7   ls .git
```

Have a look around, and see how it differs from the repository you originally cloned into the `shutit` folder.

Making a "mistake"

Now you get distracted and type:

```
8   rm -rf ../cloned_shutit/*
```

Terminal 1

Terminal

Oops. You just deleted your cloned repository.

```
9   ls .git
```

Hmmm, why is the `.git` folder still there?

This has nothing to do with Git, rather it's due to how BASH interprets the `*` character. The `*` character does not match files beginning with '.' as the command `ls` ignores them similarly. This is just a convention of

Linux (and other) operating systems.

You have cloned the repository and “accidentally” deleted all the files under Git’s control in this cloned repository. This kind of disaster happens all the time. Most often, it happens when you start to use Git and more often when you are on a tight deadline.

Back

Cloning a Remote Repository

Next

Recover Your Repository

Mark as Completed

Report an Issue

Ask a Question (https://discuss.educative.io/tag/accidental-deletion__clone-a-repository__learn-git-the-hard-way)