Git Init, Git Clone, Git Status

Git Init:

If you're all set up project-directory, then your terminal should be "inside" the project-directory, then we can init our git repo:

```
$ git init
```

Git Init's Effect

Running the git init command sets up all of the necessary files and directories that Git will use to keep track of everything. All of these files are stored in a directory called .git. This .git directory *is the "repo"*! This is where git records all of the commits and keeps track of everything!

Let's take a brief look at the contents of the .git directory.

.Git Directory Contents

We're about to take a look at the .git directory...it's not vital for this course, though, so don't worry about memorizing anything, it's here if you want to dig a little deeper into how Git works under the hood.

Here's a brief synopsis on each of the items in the .git directory:

- **config file** - where all *project specific* configuration settings are stored.
  From the Git Book:

Git looks for configuration values in the configuration file in the Git directory (.git/config) of whatever repository you're currently using. These values are specific to that single repository.

For example, let's say you set that the global configuration for Git uses your personal email address. If you want your work email to be used for a specific project rather than your personal email, that change would be added to this file.

- **description file** - this file is only used by the GitWeb program, so we can ignore it
- **hooks directory** - this is where we could place client-side or server-side scripts that we can use to hook into Git's different lifecycle events
- **info directory** - contains the global excludes file
- **objects directory** - this directory will store all of the commits we make
- **refs directory** - this directory holds pointers to commits (basically the "branches" and "tags")

Remember, other than the "hooks" directory, you shouldn't mess with pretty much any of the content in here. The "hooks" directory *can* be used to hook into different parts or events of Git's workflow, but that's a more advanced topic that we won't be getting into in this course.

Further Research

- Git Internals - Plumbing and Porcelain (advanced - bookmark this and check it out later)
- Customizing Git - Git Hooks

Git Init Recap

Use the git init command to create a new, empty repository in the current directory.

```
$ git init
```

Running this command creates a hidden .git directory. This .git directory is the brain/storage center for the repository. It holds all of the configuration files and directories and is where all of the commits are stored.

Helpful Links

- Initializing a Repository in an Existing Directory
- git init docs
- git init Tutorial

Git Clone:

Cloning The Blog Repository

Ready? Let's get cloning!

The command is git clone and then you pass the path to the Git repository that you want to clone. The project that we'll be using throughout this course is located at this URL: https://github.com/udacity/course-git-blog-project So using this URL, the full command to clone blog project is:

```
$ git clone https://github.com/udacity/course-git-blog-project
```

Git Clone Recap

The git clone command is used to create an identical copy of an existing repository.

$ git clone <path-to-repository-to-clone>

This command:

- takes the path to an existing repository

- by default will create a directory with the same name as the repository that's being cloned
- can be given a second argument that will be used as the name of the directory
- will create the new repository inside of the current working directory

Helpful Links

- Cloning an Existing Repository
- git clone docs
- git clone Tutorial

Status Update

At this point, we have two Git repositories:

- the empty one that we created with the git init command
- the one we cloned with the git clone command

How can we find any information about these repositories? Git's controlling them, but how can we find out what Git knows about our repos? To figure out what's going on with a repository, we use the git status command. Knowing the status of a Git repository is *extremely* important, so head straight on over to the next concept: Determine A Repo's Status.

Supporting Materials

- Git Repository .zip file

Git Status Recap

The git status command will display the current status of the repository.

$ git status

I can't stress enough how important it is to use this command *all the time* as you're first learning Git. This command will:

- tell us about new files that have been created in the Working Directory that Git hasn't started tracking, yet
- files that Git *is* tracking that have been modified
- a whole bunch of other things that we'll be learning about throughout the rest of the course ;-)

Helpful Links

- Checking the Status of Your Files
- git status docs

- git status Tutorial