

Why Clone?

First, what is cloning?

to make an identical copy

What's the value of creating an identical copy of something, and how does this relate to Git and version control?

Why would you want to create an identical copy? Well, when I work on a new web project, I do the same set of steps:

- create an `index.html` file
- create a `js` directory
- create a `css` directory
- create an `img` directory
- create `app.css` in the `css` directory
- create `app.js` in the `js` directory
- add starter HTML code in `index.html`
- add configuration files for linting (validating code syntax)
 - [HTML linting](#)
 - [CSS linting](#)
 - [JavaScript linting](#)
- [configure my code editor](#)

...and I do this *every time* I create a new project...which is a lot of effort I'm putting in for each new project. I didn't want to keep doing these same steps over and over, so I did all of the steps listed above one last time and created a starter project for myself. Now when I create a new project, I just make an identical copy of that starter project!

The way that cloning relates to Git is that the command we'll be running on the terminal is `git clone`. You pass a path (usually a URL) of the Git repository you want to clone to the `git clone` command.

Wanna try cloning an existing project? Let's see how Git's `clone` command works!

Verify Terminal Location

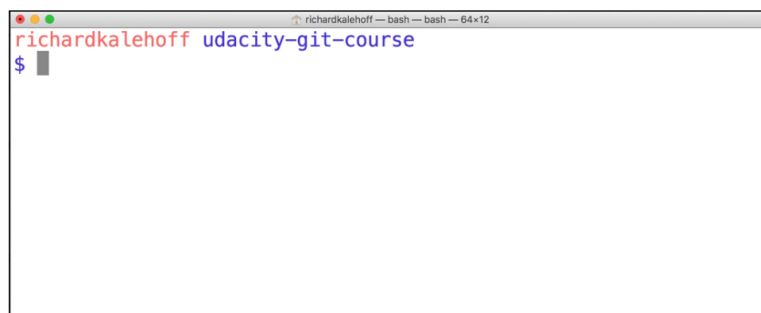
TIP: Now before you clone anything, make sure you are located in the correct directory on the command line. Cloning a project creates a new directory and places the cloned Git repository in it. The problem is that you can't have nested Git repositories. So make sure the terminal's current working directory isn't located in a Git repository. If your current working directory is not in your shell's prompt, type `pwd` to print the working directory.

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
```



The `git clone` command is used to copy the blog project repository into a `course-git-blog-project` folder in the current directory.

Git Clone Output Explanation

Let's look briefly at the output that `git clone` displays.

The first line says "Cloning into 'course-git-blog-project'...". Git is creating a directory (with the same name of the project we're cloning) and putting the repository in it...that's pretty cool!

The rest of the output is basically validation - it's counting the remote repository's number of objects, then it compresses and receives them, then it unpacks them.

Clone Project And Use Different Name

You just cloned the blog project for this course. Awesome job!

The command you ran in the terminal was:

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

...which created a directory named `course-git-blog-project`.

What if you want to use a different name instead of the default one? Yes, you could just run the command above and manually rename it in Finder/Windows Explorer or use `mv` on the terminal. But that's too many steps for us! Instead, we'd rather clone the project and have it use a different name all in one go! But how do we do that?

QUIZ QUESTION

Why don't you check out [the documentation](#) for `git clone` and pick the correct way to do it from the options below. The command should clone the blog project repo and store it in a directory named `blog-project`.

- ☐ `git clone-new-dir https://github.com/udacity/course-git-blog-project blog-project`
- ☐ `git clone https://github.com/udacity/course-git-blog-project --out blog-project`
- ☐ `git clone https://github.com/udacity/course-git-blog-project --rename blog-project`
- ☒ `git clone https://github.com/udacity/course-git-blog-project blog-project`

SUBMIT

Not In A Git Repository?

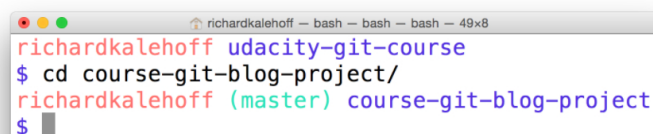
WARNING: Here's a very important step that often gets missed when you first start working with Git. When the `git clone` command is used to clone a repository, it creates a new directory for the repository...you already know this. But, it doesn't change your shell's working directory. It created the new repo inside the current working directory, which means that the current working directory is still outside of this new Git repo! Make sure you `cd` into the new repository.

Remember to use the Terminal's command prompt as an aid - if you're in a directory that is a Git repository, the command prompt will include a name in parentheses.



```
richardkalehoff udacity-git-course
$
```

Mac's Terminal application. The terminal shows the starting directory.



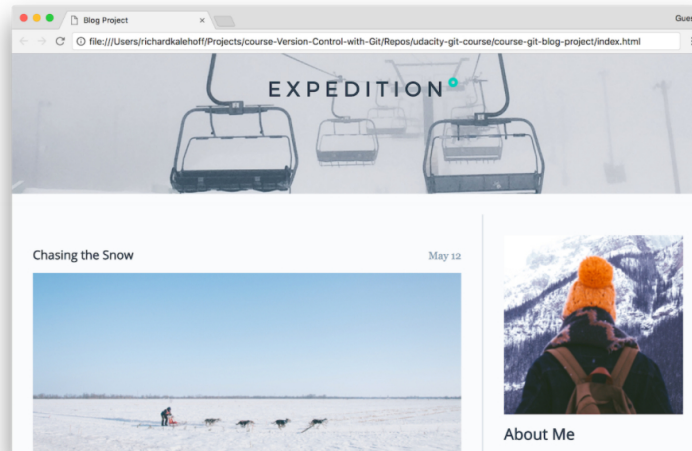
```
richardkalehoff udacity-git-course
$ cd course-git-blog-project/
richardkalehoff (master) course-git-blog-project
$
```

Mac's Terminal application. The terminal uses the `cd` command to move from the base directory into the `course-git-blog-project` which is a Git repository.

Look At The Project

So you've cloned the project to your computer, and you've `cd`ed into it. Don't you think it's time you checked it out in a browser to see what it looks like?

Open up the `index.html` file in your favorite browser.



The blog project loaded in Chrome.

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](#)