

1.1.5 Get Git

This is a good place to stop and introduce another important aspect when it comes to managing a project: **version control**.

Git is a system that allows you to create "save points" (or **commits**) of your work. It's good practice to commit your work whenever you reach a good stopping point. This creates a history of changes and allows you to revert to an earlier version if necessary.

ON THE JOB

As a software developer, you'll hear the term **production** a lot. This refers to the live version of your app or website that users are currently using. Sometimes bad code can make it to production. When this happens, the best course of action is to immediately undo! With Git, an older (working) version of the codebase can be pushed to production while developers investigate what went wrong with the new version.

Git allows you to push commits to a remote location so you won't lose any work if your computer gets run over by a bus or struck by lightning. This also lets you easily switch between computers while working on the same code.

Git also facilitates working on a team, which is common in software development. Without Git, it would be extremely tedious for developers to share code and work on the same app without accidentally losing or overwriting each other's code.

Using Git, developers can create alternate versions of the same codebase (called **branches**). When they're ready to merge these branches, Git will point out any conflicting lines of code and give developers a chance to fix the overlap. Pretty nifty stuff!

DEEP DIVE ▼

We'll be honest—learning Git is tough. It's okay if things don't click right away. You'll have plenty of opportunities to practice Git in the coming weeks. By the time you complete the boot camp, you'll be a Git master!

Create a Run Buddy Git Repository

Let's turn Run Buddy into a Git **repository**, which is basically a project folder with version control capabilities.

Open your command and make sure you're in the in the `run-buddy` directory and run the following command: `git init`. The terminal should print something like `Initialized empty Git repository`.

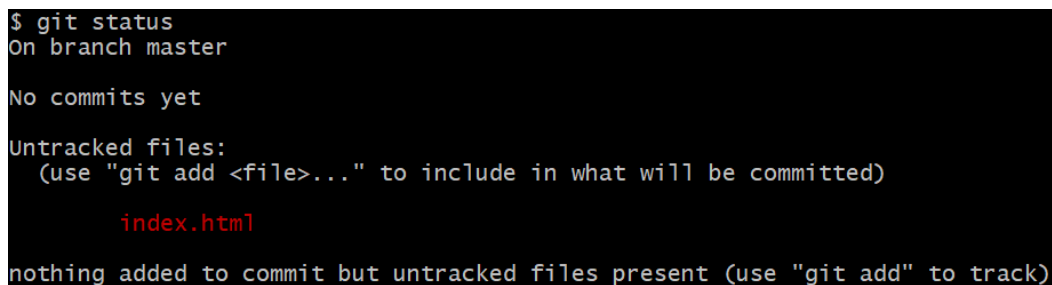
IMPORTANT

If your computer doesn't recognize the `git` command, make sure you've installed Git from the [Git website](https://git-scm.com/downloads) `(https://git-scm.com/downloads)`.

It might not seem like much happened, but this created a hidden `.git` folder in the `run-buddy` directory that designates it as a Git repository. If you want to verify for yourself, run the command `ls -a`, which will list any hidden files and folders as well as normal files. A file or folder starting with `.` is hidden. The `.git` folder is marked as hidden because it's probably not something we should be messing with!

Try Some Git Commands

Now that we have a Git repository, we can now start doing all things Git while in this folder. First, try running the command `git status`, which should display the following information that this image shows:



```
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        index.html

nothing added to commit but untracked files present (use "git add" to track)
```

IMPORTANT

macOS users, if you see a `.DS_Store` file in there as well, ignore it for now. We'll talk more about this file later in the lesson.

`git status` is a great way to quickly check what files have changed since the last time you saved, or committed, your work. Granted, we're just starting out, so there's not much to see yet. But notice that Git has listed `index.html` under `Untracked files`. The thing about Git is that it only cares about files you tell it to care about—in other words, files you tell it to track.

Well, we definitely care about `index.html`, so let's add it to Git's tracking. To do that, type the following in the command line:

```
git add index.html  
git status
```

`git add` is another important command that moves any new files or changes to **staging**. Think of it like the process of getting actors ready to go out on stage. The show hasn't started yet, but we need to round up who is going. In Git terms, we haven't saved/committed anything yet; we've just prepped Git on what could be committed.

IMPORTANT

The `git add` command shouldn't display any output and just return you back to the command line prompt, so don't worry about not seeing any feedback from the command. If something goes wrong, Git will let you know.

Make Your First Git Commit

If you run `git status` again, you'll see that `index.html` now falls under `Changes to be committed`.

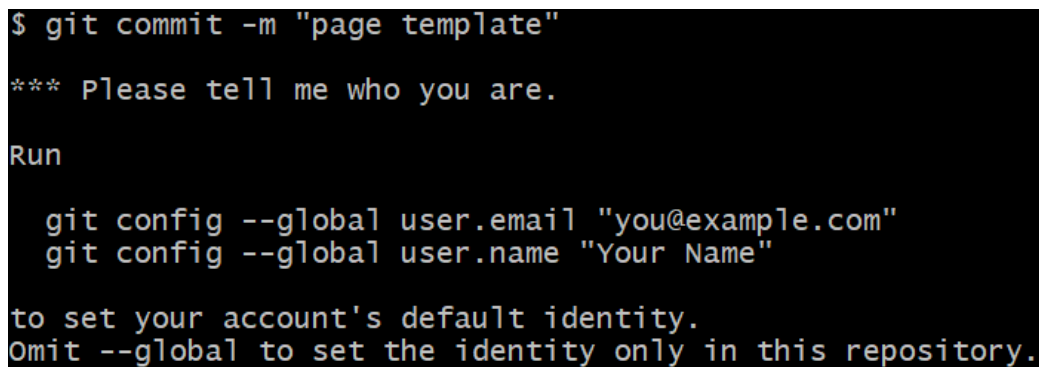
Make that commit now by typing the following:

```
git commit -m "page template"
```

`git commit` takes everything in staging and commits it. The `-m "message"` part of the command contains a short description of the commit.

SHOW PRO TIP

On your first commit, Git may ask you to identify yourself, as shown here in this image:



```
$ git commit -m "page template"

*** Please tell me who you are.

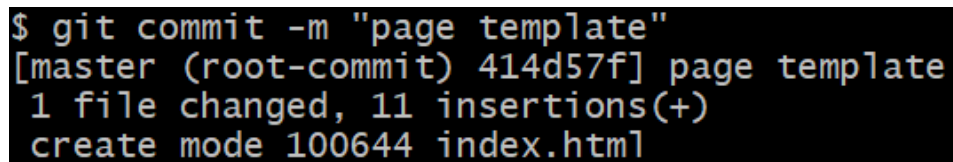
Run

  git config --global user.email "you@example.com"
  git config --global user.name "Your Name"

to set your account's default identity.
Omit --global to set the identity only in this repository.
```

This is normal. Just run the commands that the terminal is suggesting (`git config --global user.name "Your Name"`).

If you needed to do this step, you'll also need to run the `git commit -m "page template"` command again. You'll know you succeeded if the terminal prints something like this image shows us:



```
$ git commit -m "page template"
[master (root-commit) 414d57f] page template
1 file changed, 11 insertions(+)
create mode 100644 index.html
```

Great! You made your first commit! That was a lot to take in, so let's go through the process again.

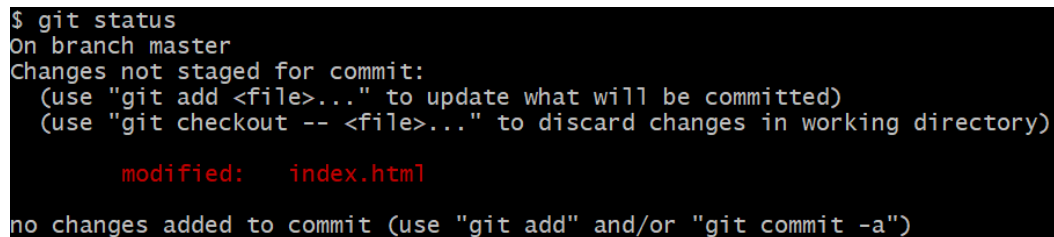
Make Another Commit

Now let's try making a change to the `index.html` file. Change the content in the `<body>` element to look like this:

```
<body>
  <h1>RUN BUDDY</h1>
  What We Do What You Do Your Trainers Reach Out
</body>
```

Now run `git status`. Git will recognize that the `index.html` file has been modified but that the changes are not staged for commit.

You should see something like this image in the command line:



```
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   index.html

no changes added to commit (use "git add" and/or "git commit -a")
```

If we want to commit this change to Git, we'll need to `add` it to staging first. To do that, type the following:

```
git add index.html
git commit -m "add more text"
```

With two commits under our belt, run the command `git log`. This shows a history of all of the commits made, including the author of the commit and the message that was provided. The output should resemble this image:

```
$ git log
commit afd6314f332f68b83af59d9dd9523b97455f0e4d (HEAD -> master)
Author: Clark <test@test.com>
Date: Tue Jul 2 09:35:12 2019 -0700

    add more link text

commit 42c4176a7e0e67b2c45e62591b34c0fa4ce57498
Author: Clark <test@test.com>
Date: Tue Jul 2 09:34:39 2019 -0700

    page template
```

IMPORTANT

To quit out of the window `git log` takes you into, simply press the `q` key on your keyboard!

Can you imagine how useful this will be when you start working on larger apps with other developers? Very useful indeed!

Take a quick assessment to help make this new knowledge stick.



Please Wait...