In the exercises that follow, you'll start using Git by adding a simple HTML file to your working tree. Then, you'll make some changes in the directory and learn how to commit the changes.

## Create and add (stage) a file

Git doesn't do much with empty directories, so let's add a file to the working tree to serve as the home page for the cat photo website.

Make sure your session is still active and that you're in your repo folder named Cats.

Use a touch command to create a file named index.html:

*touch index.html*

touch updates a file's last-modified time if the file exists. If the file doesn't exist, Git creates an empty file with that file name.

Now, use git status to get the status of the working tree:

*git status*

Git responds by informing you that nothing has been committed, but the directory does contain a new file:

Output

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)

Notice that git status gives you hints about what you can do next. Git can be configured to be less wordy, but at this stage, more is better.

Use git add to add the new file to Git's index, followed by git status to check the status. Don't forget the period at the end of the command. It tells Git to index all the files in the current directory that have been added or modified.

*git add .*

A commit has now been staged. Git's index is a staging area for commits. It's a list of all the file versions that will be part of the next commit you make.

Rather than use git add ., you could have used git add index.html because index.html was the only new file in the directory. But if several files had been added, git add . would have covered them all.

Finally, use git status again to make sure your changes were staged properly:

git status

You should see output like this example:

Output

On branch main

## Initial commit

Changes to be committed:

*(use "git rm --cached <file>..." to unstage)*

*new file: index.html*

Make your first commit

Now that index.html has been added to the index, the next step is to commit it.

Use the following command to create a commit:

git commit index.html -m "Create an empty index.html file"

The -m flag in this command tells Git that you're providing a commit message.

There are many different ways to phrase commit messages, but a good guideline is to write the first line so that it says what the commit does to the tree. It's also common to capitalize the first letter, and to leave off the closing period to save space. Imagine that the first line of the message completes the sentence starting with "When pushed, this commit will..."

A commit message can have multiple lines. The first line should have no more than 50 characters and should be followed by a blank line. Subsequent lines should have no more than 72 characters. These requirements aren't firm, and they harken back to the days of punch cards and "dumb" terminals, but they do make git log output look better.

Git responds with a confirmation of what you did:

[main (root-commit) 87e874c] Create an empty index.html file

1 file changed, 0 insertions(+), 0 deletions(-)

create mode 100644 index.html

Follow up with a git status command and confirm that the working tree is clean—that is, the working tree contains no changes that haven't been committed.

Now, use a git log command to show information about the commit:

Bash

*git log*

Git's response should be something like this example:

Output

commit 87e874c4aeeb3f9692ae5d9875235353708d7dd5

Author: User Name <user-name@contoso.com>

Date: Fri Nov 15 20:47:05 2019 +0000

## Create an empty index.html file

Modify index.html and commit the change

index.html was created to serve as the website's home page, but it's currently empty. The next step is to add some HTML to it. We'll start simple by using the editor to add a single line of HTML.

Open index.html in the online editor by typing code index.html at the terminal prompt:

code index.html

Type or paste the following statements in the editor:

HTML

<h1>Our Feline Friends</h1>

Save the file and close the editor.

Use a git status command to check the status of the working tree:

*git status*

You can see that Git is aware of the changes you made:

On branch main

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")

Now, commit the changes:

*git commit -a -m "Add a heading to index.html"*

Note that we didn't run the git add command this time to stage our changes. Instead, we used the -a flag in the git commit command. The -a option adds all the files you modified since the last commit. It won't add new files. To add new files, you still need git add.

Check the output. It should look like this example:

[main 8c9143a] Add a heading to index.html

1 file changed, 1 insertion(+)

The change to index.html has been committed. There are now two versions of the file in the repo, although you see only one of them (the current one). One of the benefits of using Git is that you can roll back the changes you have made, or you can go backward in time and see previous versions. More on this important topic later.