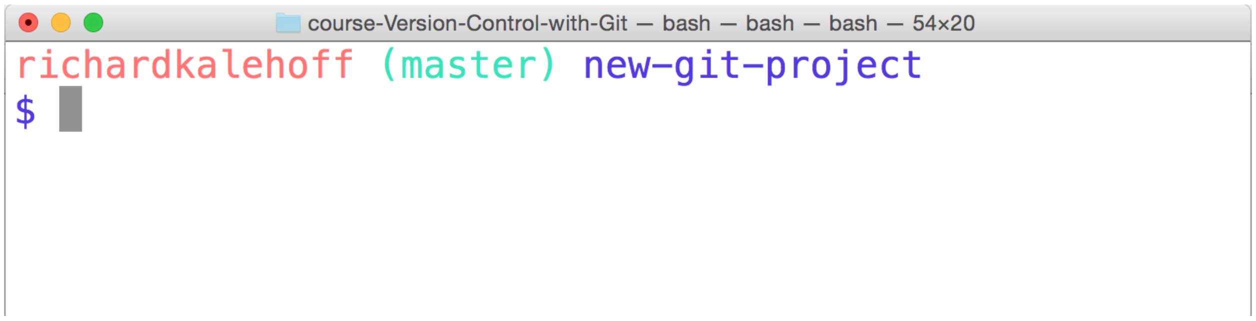
Branching

It's time to delve into the wonderful world of branches! Branches in Git are flexible and let you do some really powerful things. Before we get into the nitty gritty details of the commands, let's take another step back and look at the big picture of what branches are and how they function.

So that's the big picture of how branches work and how to switch between branches. Did you know that you've already seen the master branch on the command line? Because of the setup files you added in the first lesson, the current branch is displayed right in the command prompt.


course-Version-Control-with-Git — bash — bash — bash — 54×20

```
richardkalehoff (master) new-git-project
$ 
```

The Terminal application showing the current branch in the command prompt. The current branch is the "master" branch.

The git branch command

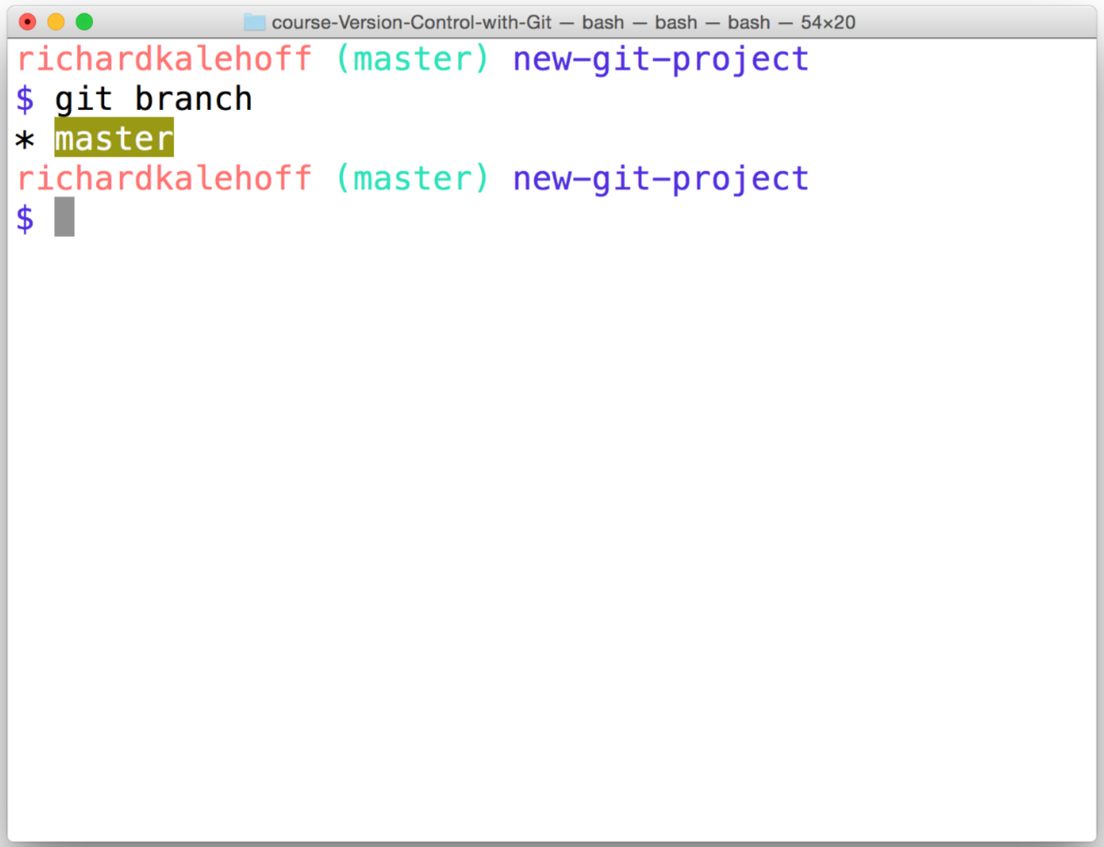
The git branch command is used to interact with Git's branches:

```
$ git branch
```

It can be used to:

- list all branch names in the repository
- create new branches
- delete branches

If we type out just git branch it will list out the branches in a repository:


course-Version-Control-with-Git — bash — bash — bash — 54×20

```
richardkalehoff (master) new-git-project
$ git branch
* master
richardkalehoff (master) new-git-project
$ 
```

The Terminal application showing the output of the git branch command. The master branch is displayed.

Create A Branch

To create a branch, all you have to do is use git branch and provide it the name of the branch you want it to create. So if you want a branch called "sidebar", you'd run this command:

```
$ git branch sidebar
```

The git checkout Command

Remember that when a commit is made that it will be added to the current branch. So even though we created the new sidebar, no new commits will be added to it since we haven't *switched to it*, yet. If we made a commit right now, that commit would be added to the master branch, *not* the sidebar branch. We've already seen this in the demo, but to switch between branches, we need to use Git's checkout command.

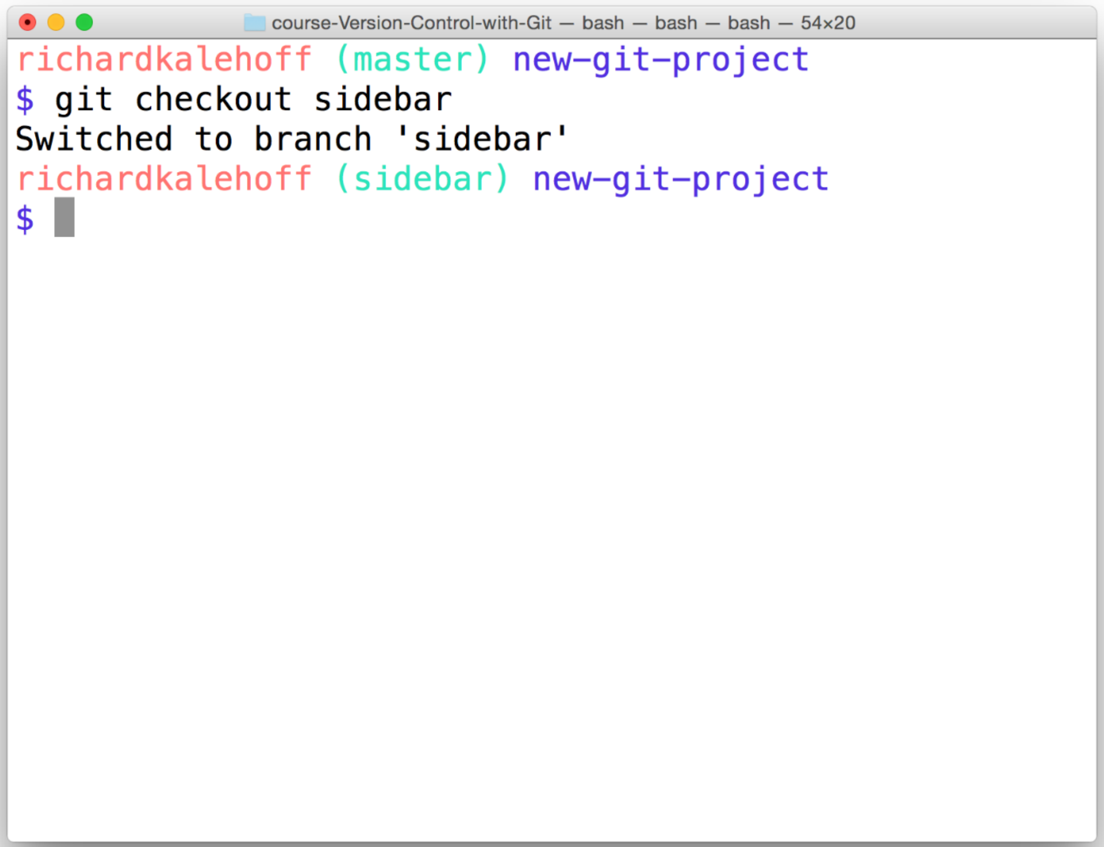
```
$ git checkout sidebar
```

It's important to understand how this command works. Running this command will:

- remove all files and directories from the Working Directory that Git is tracking
    - (files that Git tracks are stored in the repository, so nothing is lost)
- go into the repository and pull out all of the files and directories of the commit that the branch points to

So this will remove all of the files that are referenced by commits in the master branch. It will replace them with the files that are referenced by the commits in the sidebar branch. This is very important to understand, so go back and read these last two sentences.

The funny thing, though, is that both sidebar and master are pointing *at the same commit*, so it will look like nothing changes when you switch between them. But the command prompt will show "sidebar", now:


course-Version-Control-with-Git — bash — bash — bash — 54×20

```
richardkalehoff (master) new-git-project
$ git checkout sidebar
Switched to branch 'sidebar'
richardkalehoff (sidebar) new-git-project
$ 
```

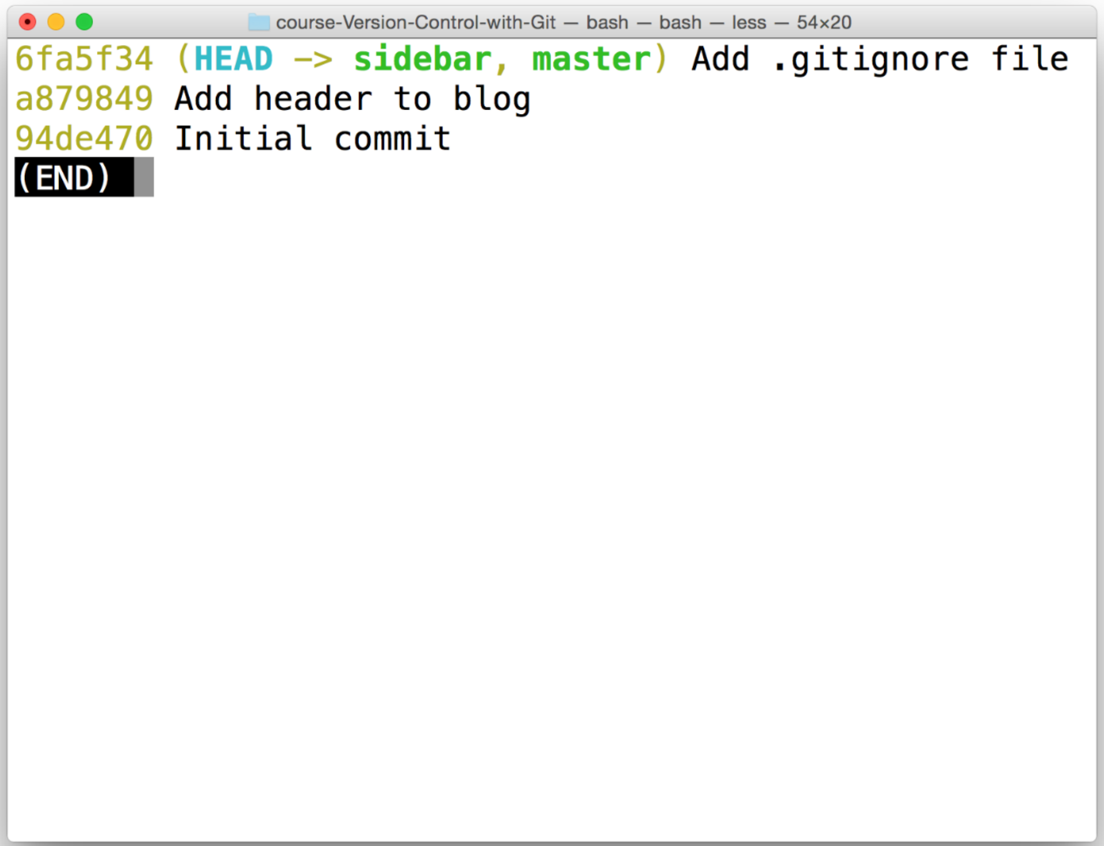
The Terminal application showing the "sidebar" branch in the command prompt.

Branches In The Log

The branch information in the command prompt is helpful, but the clearest way to see it is by looking at the output of git log. But just like we had to use the --decorate flag to display Git tags, we need it to display branches.

```
$ git log --oneline --decorate
```

This is what my log output displays (yours might look different depending on what commits you've made):



course-Version-Control-with-Git — bash — bash — less — 54×20

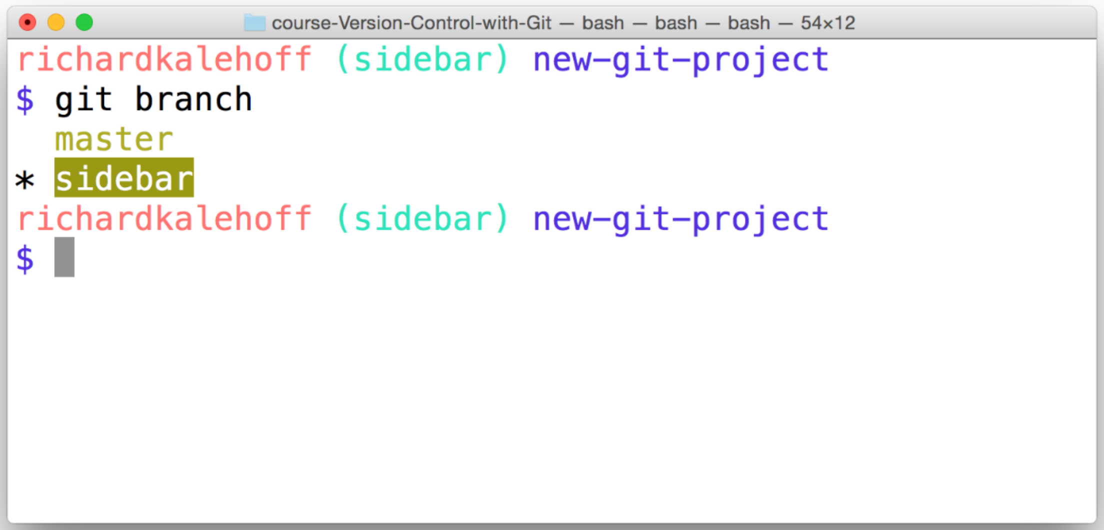
```
6fa5f34 (HEAD -> sidebar, master) Add .gitignore file
a879849 Add header to blog
94de470 Initial commit
(END)
```

The Terminal application showing the output of the git log --oneline --decorate command. The word "HEAD" has an arrow pointing to "sidebar" which is the active branch.

In the output above, notice how the special "HEAD" indicator we saw earlier has an arrow pointing to the sidebar branch. It's pointing to sidebar because the sidebar branch is the current branch, and any commits made right now will be added to the sidebar branch.

The Active Branch

The command prompt will display the *active* branch. But this is a special customization we made to our prompt. If you find yourself on a different computer, the *fastest* way to determine the active branch is to look at the output of the git branch command. An asterisk will appear next to the name of the active branch.


course-Version-Control-with-Git — bash — bash — bash — 54×12

```
richardkalehoff (sidebar) new-git-project
$ git branch
  master
* sidebar
richardkalehoff (sidebar) new-git-project
$
```

The Terminal application showing the output of the git branch command. The active branch (in this case, the "sidebar" branch) has an asterisk next to it.

Delete A Branch

A branch is used to do development or make a fix to the project that won't affect the project (since the changes are made on a branch). Once you make the change on the branch, you can combine that branch into the master branch (this "combining of branches" is called "merging" and we'll look at it shortly).

Now after a branch's changes have been merged, you probably won't need the branch anymore. If you want to delete the branch, you'd use the -d flag. The command below includes the -d flag which tells Git to *delete* the provided branch (in this case, the "sidebar" branch).

```
$ git branch -d sidebar
```

One thing to note is that you can't delete a branch that you're currently on. So to delete the sidebar branch, you'd have to switch to either the master branch or create and switch to a new branch.

Deleting something can be quite nerve-wracking. Don't worry, though. Git won't let you delete a branch if it has commits on it that aren't on any other branch (meaning the commits are unique to the branch that's about to be deleted). If you created the sidebar branch, added commits to it, and then tried to delete it with the git branch -d sidebar, Git wouldn't let you delete the branch because you can't delete a branch that you're currently on. If you switched to the master branch and tried to delete the sidebar branch, Git *also* wouldn't let you do that because those new commits on the sidebar branch would be lost! To force deletion, you need to use a capital D flag - git branch -D sidebar.

Git Branch Recap

To recap, the git branch command is used to manage branches in Git:

```
# to list all branches

$ git branch


# to create a new "footer-fix" branch

$ git branch footer-fix


# to delete the "footer-fix" branch

$ git branch -d footer-fix
```

This command is used to:

- list out local branches
- create new branches
- remove branches

Further Research

- Git Branching - Basic Branching and Merging from the Git Docs
- Learn Git Branching
- Git Branching Tutorial from the Atlassian Blog