Branching is one of the most powerful features that Git offers. Creating different Git branches allows you to work on a particular feature or set of files independently from other “copies” of a repository. That way you and a friend can work on different parts of the same file on different branches, and then Git can help you elegantly merge your branches and changes together.

You can list all of the available branches with the command git branch:

```
git branch

## * master
```

The star (*) indicates which branch you’re currently on. The default branch that is created is always called *master*. Usually people use this branch as the working version of the software that they are writing, while they develop new and potentially unstable features on other branches.

To add a branch we’ll also use the git branch command, followed the name of the branch we want to create:

```
git branch my-new-feature
```

Now let’s enter git branch again to confirm that we’ve created the branch:

```
git branch

## * master
## my-new-feature
```

We can make my-new-feature the current branch using git checkout with the name of the branch:

```
git checkout my-new-feature

## Switched to branch 'my-new-feature'

git branch

##   master
## * my-new-feature
```

If we look at git status we can also see that it will tell us which branch we’re on:

```
git status

On branch my-new-feature
nothing to commit, working tree clean
```

We can switch back to the master branch using git checkout:

```
git checkout master

## Switched to branch 'master'

git branch

## * master
##   my-new-feature
```

Now we can delete a branch by using the -d flag with git branch and the name of the branch we want to delete:

```
git branch -d my-new-feature

## Deleted branch my-new-feature (was adef548).

git branch

## * master
```

Let's create a new branch for adding a section to the readme.txt in our repository. We can create a new branch and switch to that branch at the same time using the command git checkout -b and the name of the new branch we want to create:

```
git checkout -b update-readme

## Switched to a new branch 'update-readme'
```

Now that we've created and switched to a new branch, let's make some changes to a file. As you might be expecting right now we'll add a new line to readme.txt:

```
echo "I added this line in the update-readme branch." >> readme.txt
cat readme.txt

## Welcome to My First Repo
## Learning Git is going well so far.
## I added this line in the update-readme branch.
```

Now that we've added a new line let's commit these changes:

```
git add -A
git commit -m "added a third line to readme.txt"

## [update-readme 6e378a9] added a third line to readme.txt
##  1 file changed, 1 insertion(+)
```

Now that we've made a commit on the update-readme branch, let's switch back to the master branch, and then we'll take a look at readme.txt:

```
git checkout master

## Switched to branch 'master'
```

Now that we're on the master branch let's quickly glance at readme.txt:

```
cat readme.txt

## Welcome to My First Repo
## Learning Git is going well so far.
```

The third line that we added is gone! Don't fret, the line that we added isn't gone forever. We committed the change to this file while we were on the update-readme branch, so the updated file is safely in that branch. Let's switch back to that branch just to make sure:

```
git checkout update-readme
cat readme.txt

## Welcome to My First Repo
## Learning Git is going well so far.
## I added this line in the update-readme branch.
```

And the third line is back! Let's add and commit yet another line while we're on this branch:

```
echo "It's sunny outside today." >> readme.txt
git add -A
git commit -m "added weather info"

## [update-readme d7946e9] added weather info
##  1 file changed, 1 insertion(+)
```

Mark as completed