This is a small example of how to use Git branching, but you can see how you can make incremental edits to plain text (usually code files) without effecting the master branch (the tested and working copy of your software) and without effecting any other branches. You can imagine how this system could be used for multiple people to work on the same codebase at the same time, or how you could develop and test multiple software features without them interfering with each other. Now that we've made a couple of changes to readme.txt, let's combine those changes with what we have in the master branch. This is made possible by a Git merge. Merging allows you to elegantly combine the changes that have been made between two branches. Let's merge the changes we made in the update-readme branch with the master branch. Git incorporates other branches into the current branch by default. When you're merging, the current branch is also called the base branch. Let's switch to the master branch so we can merge in the changes from the update-readme branch:

```
git checkout master

## Switched to branch 'master'
```

To merge in the changes from another branch we need to use git merge and the name of the branch:

```
git merge update-readme

## Updating adef548..d7946e9
## Fast-forward
##  readme.txt | 2 ++
##  1 file changed, 2 insertions(+)

cat readme.txt

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

It looks like you've merged your first branch in Git! Branching is part of what makes Git so powerful since it enables parallel developments on the same code base. But what if there are two commits in two separate branches that make different edits to the same line of text? When this occurs it is called a **conflict**. Let's create a conflict so we can learn how they can be resolved.

First we'll switch to the update-readme branch. Use nano to edit the last line of readme.txt, then commit your changes:

```
git checkout update-readme
nano 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.
## It's cloudy outside today.
```

Notice that we changed “sunny” to “cloudy” in the last line.

```
git add -A
git commit -m "changed sunny to cloudy"
```

Now that our changes are committed on the update-readme branch, let’s switch back to master:

```
git checkout master
```

Let’s change the same line of code using nano:

```
nano 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.
## It's windy outside today.
```

Now let’s commit these changes:

```
git add -A
git commit -m "changed sunny to windy"
```

We’ve now created two commits that directly conflict with each other. On the update-readme branch the last line says It's cloudy outside today., while on the master branch the last line says It's windy outside today.. Let’s see what happens when we try to merge update-readme into master.

```
git merge update-readme

## Auto-merging readme.txt
## CONFLICT (content): Merge conflict in readme.txt
## Automatic merge failed; fix conflicts and then commit the result.
```

Uh-oh, there’s a conflict! Let’s check the status of the repo right now:

```
git status

## On branch master
## You have unmerged paths.
##   (fix conflicts and run "git commit")
##   (use "git merge --abort" to abort the merge)
##
## Unmerged paths:
##   (use "git add <file>..." to mark resolution)
##
##  both modified:   readme.txt
##
## no changes added to commit (use "git add" and/or "git commit -a")

```

If you're getting used to reading the result of git status, you can see that it often offers suggestions about what steps you should take next. Git is indicating that both versions of readme.txt have modified the same text. Let's take a look at readme.txt to see what's going on there:

```
cat readme.txt

## Welcome to My First Repo
## Learning Git is going well so far.
## I added this line in the update-readme branch.
## <<<<<<< HEAD
## It's windy outside today.
## =======
## It's cloudy outside today.
## >>>>>>> update-readme
```

The first three lines of this file look normal, then things get interesting! The line between <<<<<<< HEAD and ======= shows the version of the conflicted line on the current branch. In Git terminology the HEAD represents the most recent commit on the branch which is currently checked out (which is master in this case). The line between ======= and >>>>>>> update-readme shows the version of the line on the update-readme branch. In order to resolve this conflict, all we need to do is open readme.txt with nano so we can delete the lines we want to get rid of. In this case let's keep the "cloudy" version.

```
nano 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.
## It's cloudy outside today.
```

Now we can commit the resolution of this conflict.

```
git add -A
git commit -m "resolved conflict"
```

You're now familiar with this basics of Git! If you want to go into further depth with your study of Git I highly recommend the free and open source book Pro Git.

Summary

- Git branching allows you and others to work on the same code base together.
- You can create a branch with the command git branch [name of branch].
- To switch to a branch use git checkout [name of branch].
- You can combine a branch with your current branch by using git merge.

Mark as completed