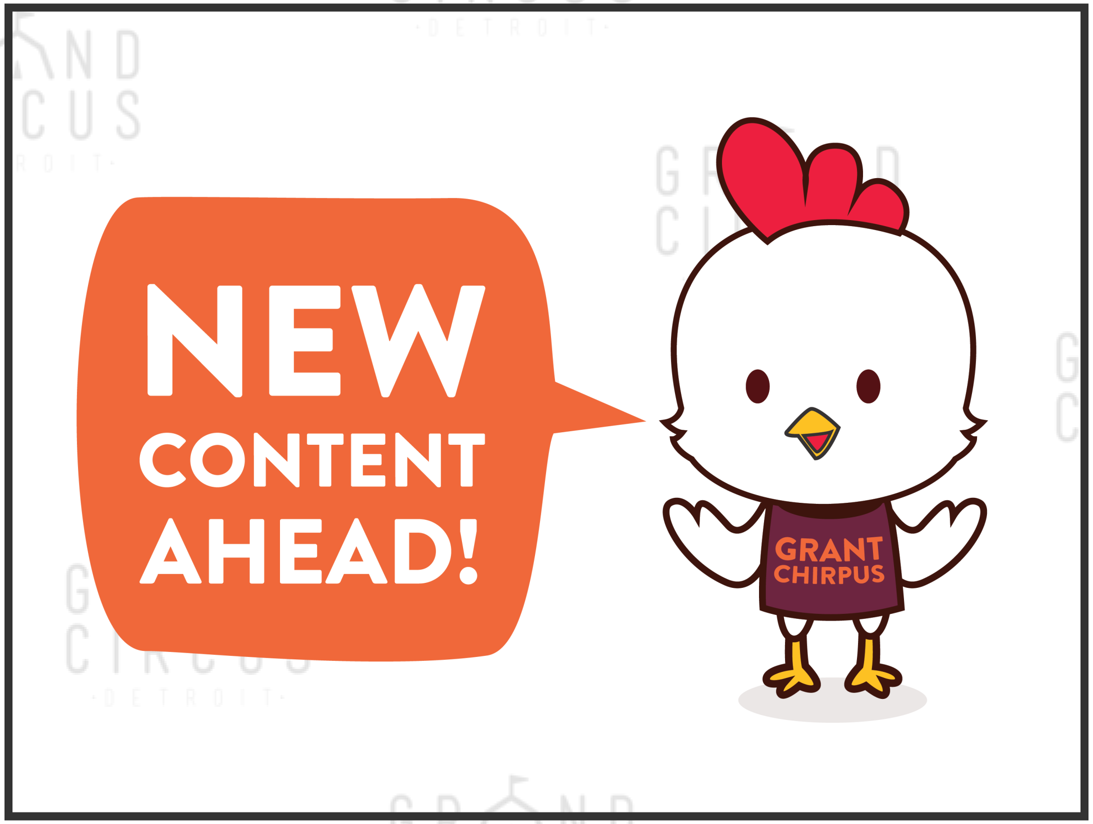

NEW CONTENT AHEAD!
GRANT CHIRPUS

BRANCHES

GO YOUR OWN WAY

There will be times that you want to make a major change to your code. It's generally best to keep these types of changes separate from your working version, especially if it's live!

HOW IT WORKS

Think of the history of your project like a tree with branches that can diverge and merge back in with the main trunk (or master branch). We can have branches for new features we're working on, branches to separate development and production code, even branches for different team members.

GIT BRANCH < BRANCH >

Create a new branch with the specified name.

```
$ git branch cool-feature
```

GIT CHECKOUT < BRANCH >

Switch over to the specified existing branch.

```
$ git checkout cool-feature
```

GIT CHECKOUT -B < BRANCH >

This is a shortcut! With just one line, we can create a new branch and switch over to it.

```
$ git checkout -b cool-feature
```

MERGING

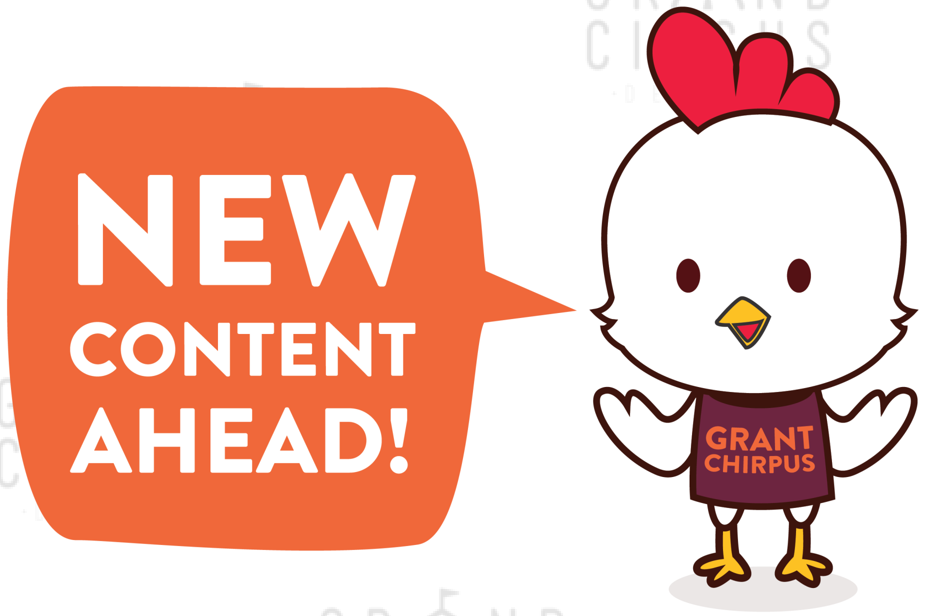
WHEN TO MERGE

At some point, you'll want to move all your glorious changes stored in the new branch to your master branch so that the world can benefit from your genius. This is the point at which you must merge.

GIT MERGE < BRANCH >

Merge the specified branch into the current branch.

```
$ git checkout master
$ git merge cool-feature
```

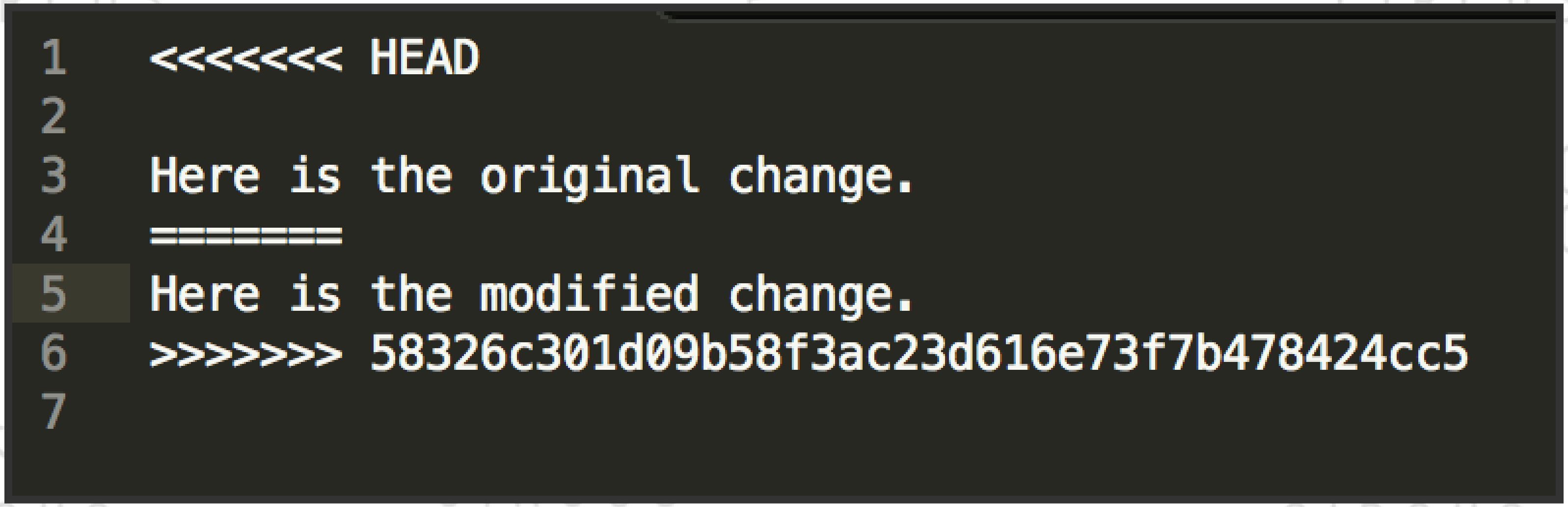

NEW CONTENT AHEAD!
GRANT CHIRPUS

CONFLICT RESOLUTION

Depending on which files were changed within the branch, merge conflicts may arise. If the same file has been changed in the same place on two different branches, Git won't know what to do and will defer to your judgment.

GETTING YOUR HANDS DIRTY

To fix a merge conflict, we need to dive into the code and manually delete the parts we don't want.



```
<<<<<<< HEAD

Here is the original change.
=======
Here is the modified change.
>>>>>>> 58326c301d09b58f3ac23d616e73f7b478424cc5

```

CLEANUP

Once we've made the hard decisions, we just need to add and commit the files we've changed just as in our normal Git workflow!

FURTHER READING

If you want more info on branches, check out (see what I did there?) this tutorial!

TOOLS LAB 4
BRANCHING OFF

INSTRUCTIONS

Break into groups and perform the following actions on the seat reservation app:

1. Each member creates their own branch.
2. Each member pushes a change to a different file. Merge all branches into the master.
3. Each member makes a change to the same file. Merge all branches into the master.
4. Each member makes a change to the same line. Merge all branches into the master.

Bonus: Fork the remote repo, commit changes, and make a pull request.