Resolving conflicts

In this section you will:

- encounter a merge conflict on GitHub
- encounter a merge conflict on the commandline
- resolve the conflict in a new temporary Git branch

Encountering a merge conflict on GitHub

Sometimes you'll discover that your GitHub fork and the upstream repository have changes that GitHub can't merge.

There's an *unmergeable-branch* at https://github.com/evildmp/afraid-to-commit. It's unmergeable because it deliberately contains changes that conflict with other changes made in *master*.

Using the GitHub interface, try creating a pull request from *unmergeable-branch* to your *master* on GitHub. If you do, GitHub will tell you:

```
We can't automatically merge this pull request.

Use the command line to resolve conflicts before continuing.
```

GitHub will in fact tell you the steps you need to take to solve this, but to understand what's actually happening, and to do it yourself when you need to, we need to cover some important concepts.

Merging changes from a remote branch

```
# make sure you have the latest data from upstream
$ git fetch upstream
# create and switch to a new branch based on master to explore the conflict
$ git checkout -b explore-conflict upstream/master
# now try merging the unmergeable-branch into it
$ git merge upstream/unmergeable-branch
Auto-merging attendees_and_learners.rst
CONFLICT (content): Merge conflict in attendees_and_learners.rst
Automatic merge failed; fix conflicts and then commit the result.
```

When there's a conflict, Git marks them for you in the files. You'll see sections like this:

```
<<<<<<< HEAD
* Daniel Pass <daniel.antony.pass@googlemail.com>
* Kieran Moore
=======
* Kermit the Frog
* Miss Piggy
>>>>>>> upstream/unmergeable-branch
```

The first section, `HEAD` is what you have in your version. The second section, `upstream/unmergeable-branch` is what Git found in the version you were trying to pull in.

You'll have to decide what the file should contain, and you'll need to edit it. If you decide you want the changes from both versions:

```
* Daniel Pass <daniel.antony.pass@googlemail.com>
* Kieran Moore
* Kermit the Frog
* Miss Piggy

$ git add attendees_and_learners.rst
$ git commit -m "fixed conflict"
[explore-conflict 91a45ac] fixed conflict
```

Note

Create new branches when resolving conflicts

> It's very sensible *not* to do merging work in a branch you have done valuable work in. In the example above, your *explore-conflict* branch is based on master and doesn't contain anything new, so it will be easy to re-create if it all goes wrong.
>
> If you had a branch that contained many complex changes however, you certainly wouldn't want to discover dozens of conflicts making a mess in the files containing all your hard work.
>
> So remember, **branches are cheap and disposable**. Rather than risk messing up the branch you've been working on, create a new one specially for the purpose of discovering what sort of conflicts arise, and to give you a place to work on resolving them without disturbing your work so far.
>
> You might have conflicts across dozens of files, if you were unlucky, so it's very important to be able to backout gracefully and at the very least leave things as they were.