Introduction to Git

Eric Jiang (@lorderikir)

Copyright © Eric Jiang 2018 - 2019 | Adapated from "Gitting Started at Hackathons" tech talk

Hi, I'm Eric Jiang 👋

- Currently Software Engineer, Digital Transformation, eSolutions.
- I founded MonPlan
- I also maintain other apps like GeckoDM and MARIE.js
- Also have worked at Localz too where I worked on the React Native Core App

Copyright © Eric Jiang 2018 - 2019 | Adapated from "Gitting Started at Hackathons" tech talk

A big thanks to our swag/ resource sponsor - GitHub

Copyright © Eric Jiang 2018 - 2019 | Adapated from "Gitting Started at Hackathons" tech talk

So what is today about?

Copyright © Eric Jiang 2018 - 2019 | Adapated from "Gitting Started at Hackathons" tech talk

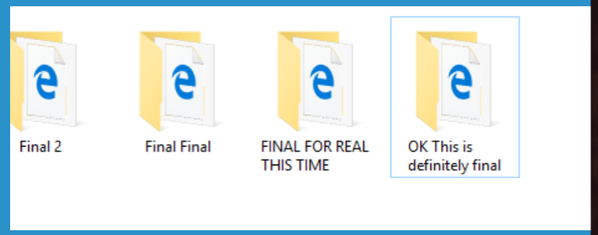
So, I love writing code & also love working in teams

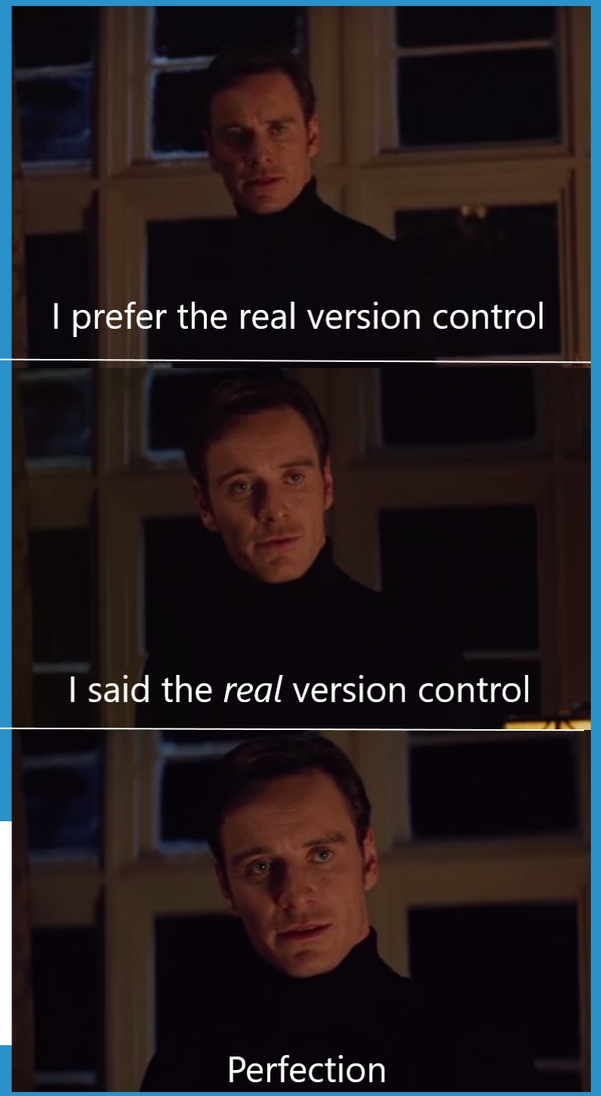
But what if there was a way that I could remember how the code look liked throughout its entire development lifecycle, for example if something went wrong and I want to go back to a previous version?

In comes Git

Copyright © Eric Jiang 2018 - 2019 | Adapated from "Gitting Started at Hackathons" tech talk

First of all, what is Git?





Copyright © Eric Jiang 2018 - 2019 | Adapated from "Gitting Started at Hackathons" tech talk

Git

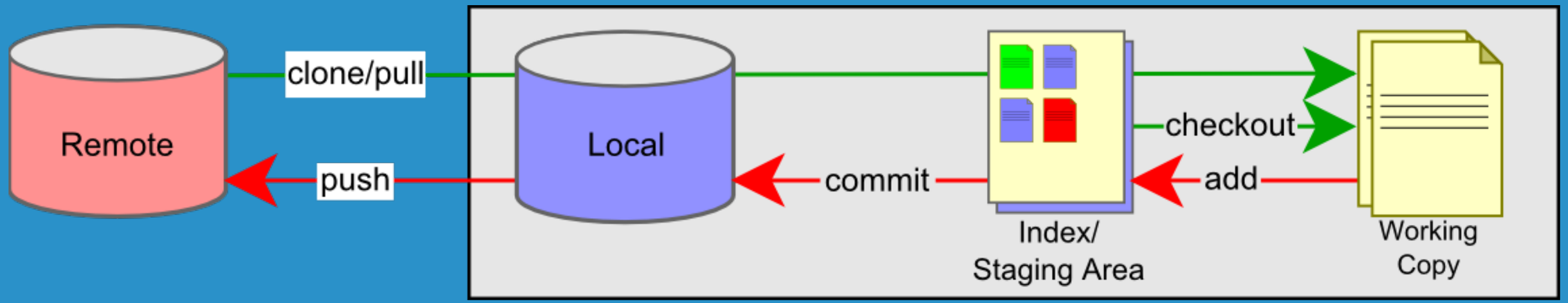
Git is a version control system for tracking changes in computer files and coordinating work on those files among multiple people

— Git-SCM Website

Also if you want to go into Software Development at some companies, you are bound to use Git or some kind of version control like SVN/Mecurial

Copyright © Eric Jiang 2018 - 2019 | Adapated from "Gitting Started at Hackathons" tech talk

How Git Works



Copyright © Eric Jiang 2018 - 2019 | Adapated from "Gitting Started at Hackathons" tech talk

Some Terminology

Repository

The Git repository is stored in the same directory as the project itself, in a subdirectory called .git. Note differences from central-repository systems like CVS or Subversion:

- **There is only one .git directory, in the root directory of the project.**
- **The repository is stored in files alongside the project. There is no central server repository.**

Copyright © Eric Jiang 2018 - 2019 | Adapated from "Gitting Started at Hackathons" tech talk

Some Terminology

Commit

The git commit command captures a snapshot of the project's currently staged changes

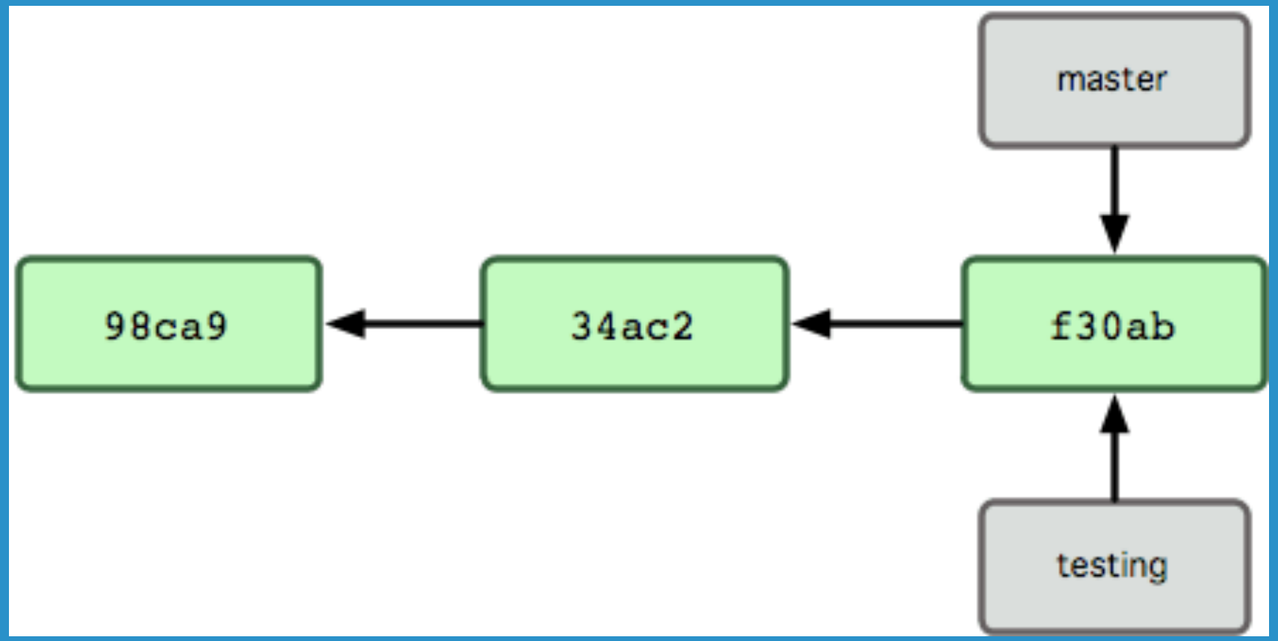
- **A reference to some changes to a file (removals or deletitions in files, creation/moving/deletition of files)**
- **Imagine the commit as a *save* of the changes to the file(s)**
- **A commit also contains the Commit Title/Name, and Commit Description**

Copyright © Eric Jiang 2018 - 2019 | Adapated from "Gitting Started at Hackathons" tech talk

Some Terminology

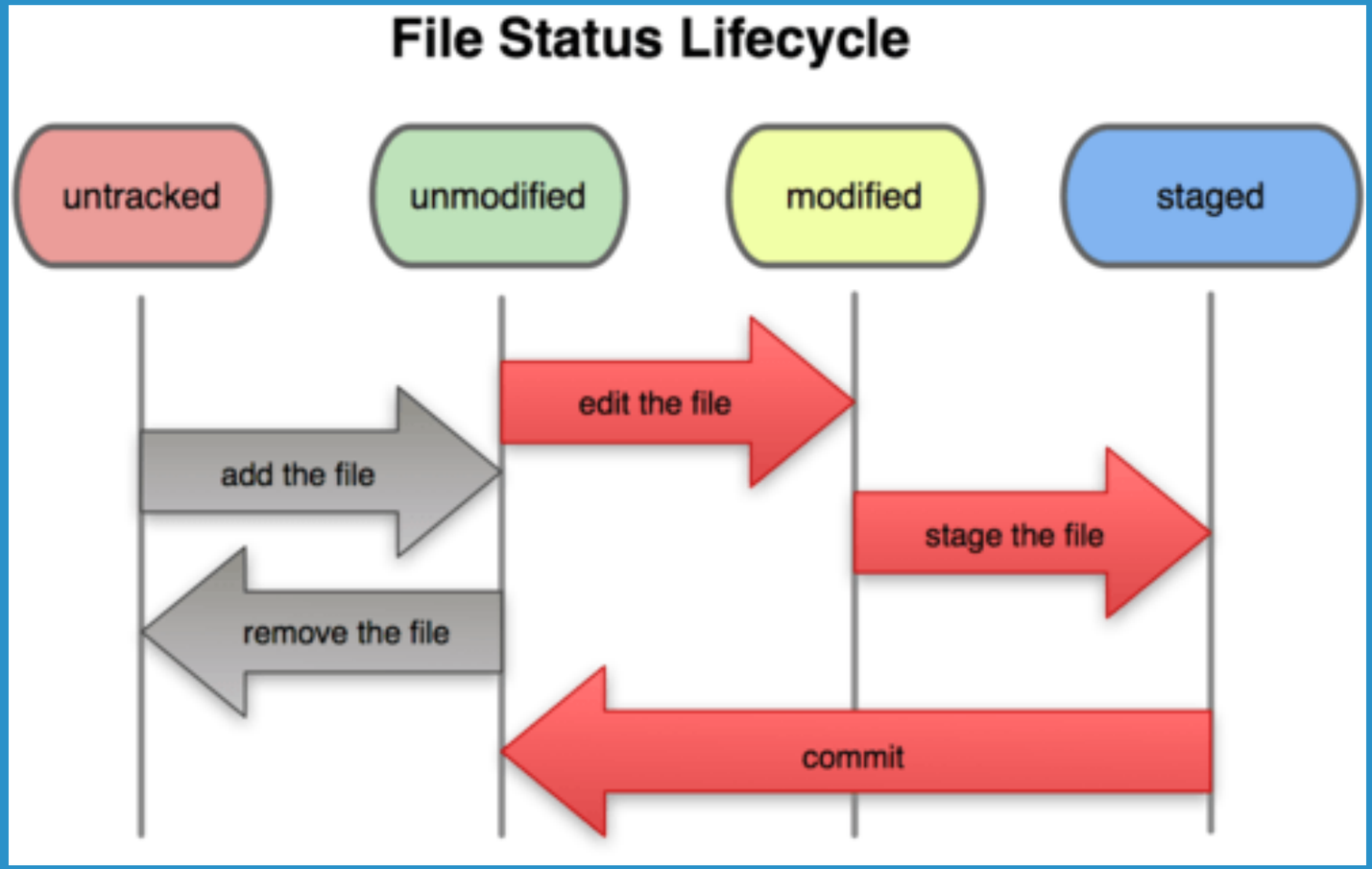
Branches

A branch in Git is simply a lightweight movable pointer to one of these commits. The default branch name in Git is master



Copyright © Eric Jiang 2018 - 2019 | Adapated from "Gitting Started at Hackathons" tech talk

Git File Lifecycle



Copyright © Eric Jiang 2018 - 2019 | Adapated from "Gitting Started at Hackathons" tech talk

Some Basic Commands

Command	Description
`git clone`	**Clones a repository locally**
`git add`	**Stages changes to file(s) for a commit**
`git commit`	**Creates a commit (set of changes)**
`git push`	**Push changes to the hosted repo**

Copyright © Eric Jiang 2018 - 2019 | Adapated from "Gitting Started at Hackathons" tech talk

Using Git within teams

Well, working with teams 👨‍👩‍👦 may be hard. There are generally two ways you can work off a repository.

- Using Branches
- Using Forks
- Open Source projects tend to use Forks, while:
- a lot of companies internally uses what is known as GitFlow which uses branches!

Copyright © Eric Jiang 2018 - 2019 | Adapated from "Gitting Started at Hackathons" tech talk

Use Branches 🌳 for Versioning Control (GitFlow)

1. Make a branch with the feature, bug, hotfix you are working on.
2. Every time you commit and push up
3. Make a Pull Request
4. Merge the pull request

One of the best workflows is known as *GitFlow*

Copyright © Eric Jiang 2018 - 2019 | Adapated from "Gitting Started at Hackathons" tech talk

In GitFlow, our Branches follow some naming conventions...

- `master`: **branch is the key branch, typically for our production/ public-facing version**
- `develop`: ***unstable*, most of the PRs should go here**
- `staging`: **this branch is occassionally but not always used, this matches our QA/Testing environment**
- `feat/*`, `fix/*`, **etc.: are 'for purpose' branches, these branches are for development**

This slide has been adapted from my CI-CD talk

Copyright © Eric Jiang 2018 - 2019 | Adapated from "Gitting Started at Hackathons" tech talk

So we know that development is done incrementally

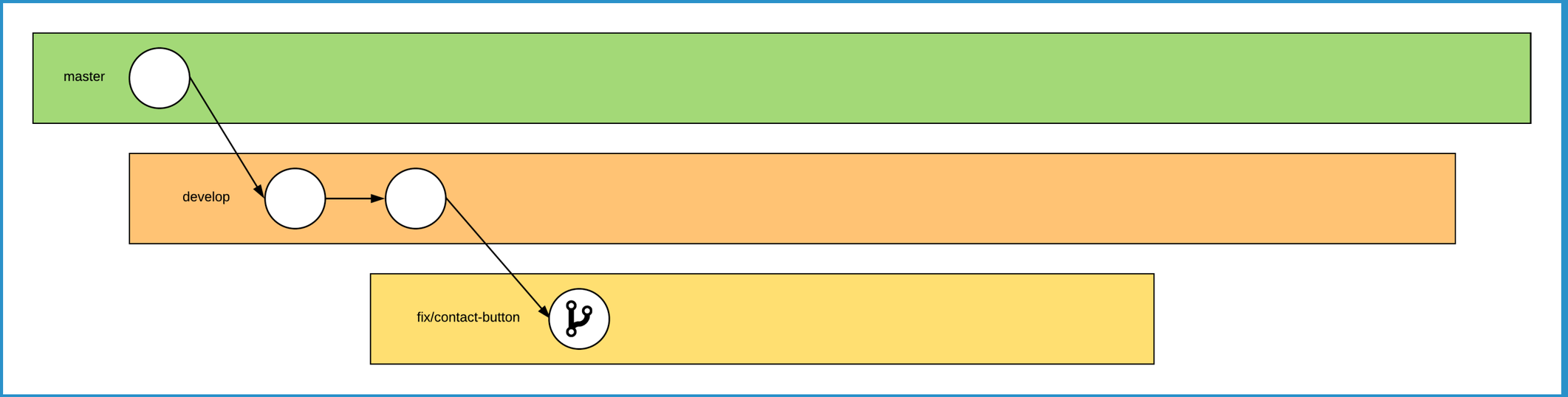
Copyright © Eric Jiang 2018 - 2019 | Adapated from "Gitting Started at Hackathons" tech talk

Imagine we using Git within our practices

And one of my team-mates, has found a bug within one of our buttons.

Copyright © Eric Jiang 2018 - 2019 | Adapated from "Gitting Started at Hackathons" tech talk

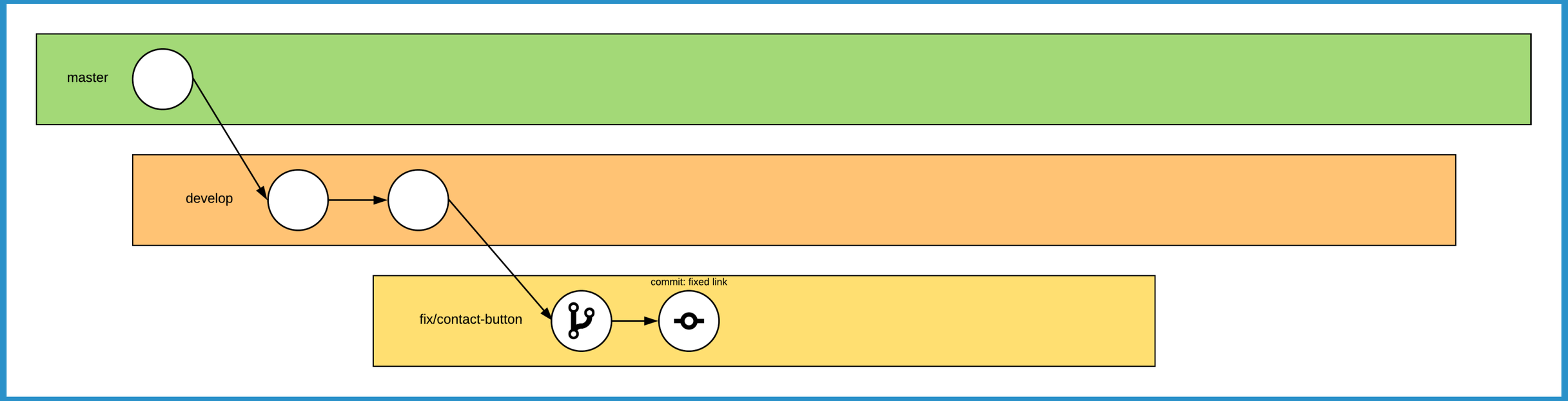
So, he creates a new branch to fix the bug




```
# update our develop branch
git checkout develop
git pull
# we create a new branch
git branch fix/contact-button
# we make the new branch the new working branch
git checkout fix/contact-button
```

Copyright © Eric Jiang 2018 - 2019 | Adapated from "Gitting Started at Hackathons" tech talk

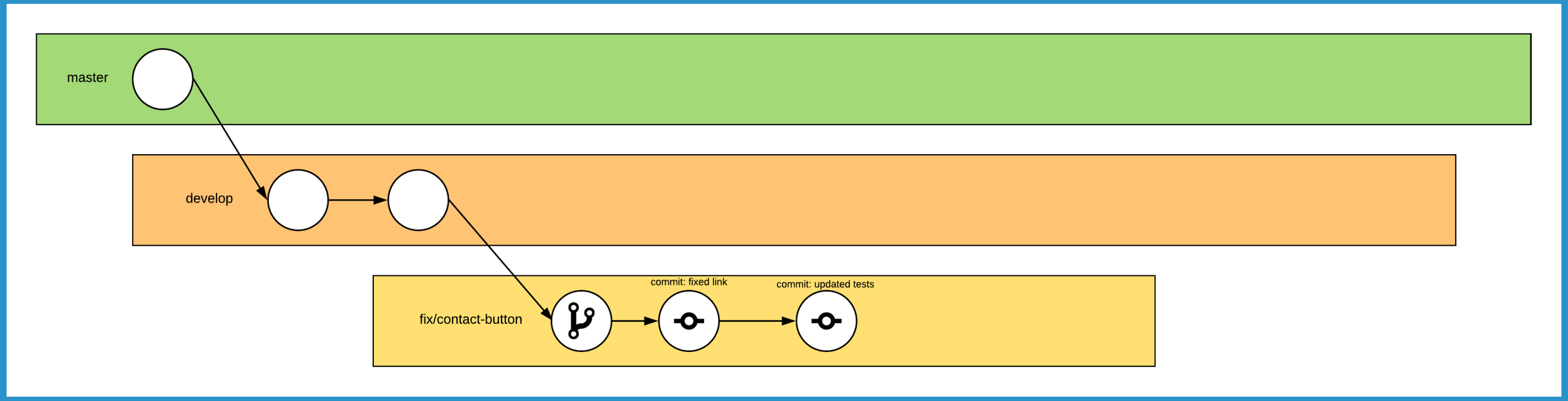
He fixes the code and stages the change in commits




```
git add .
git commit -m "new commit"
git push
```

Copyright © Eric Jiang 2018 - 2019 | Adapated from "Gitting Started at Hackathons" tech talk

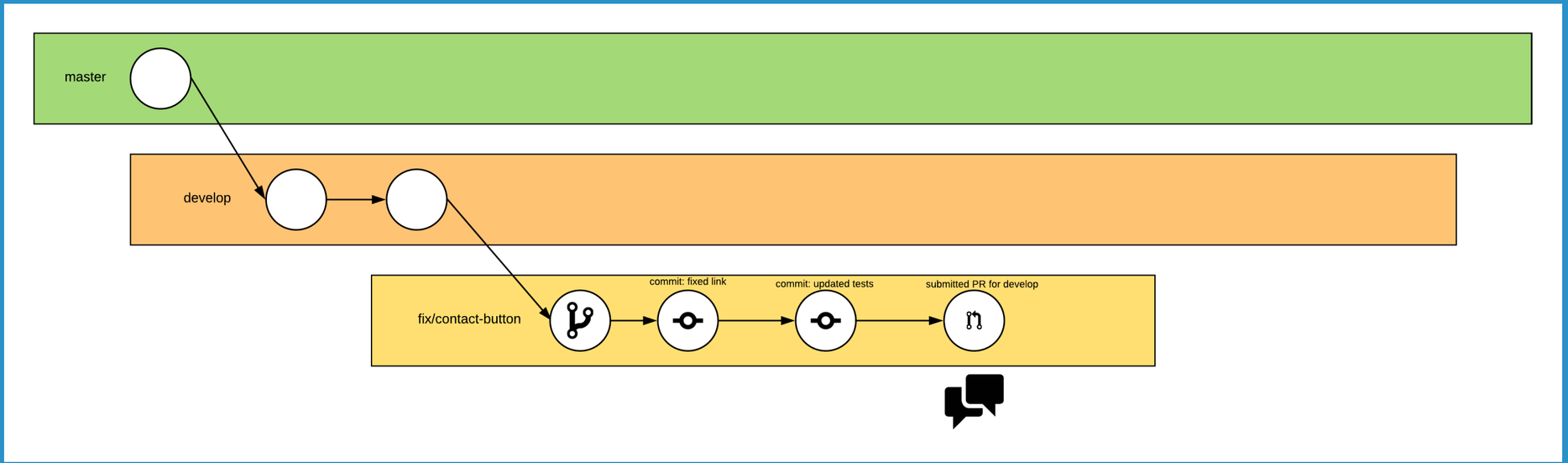
He fixes the code and stages the change in commits




```
git add .
git commit -m "new commit"
git push
```

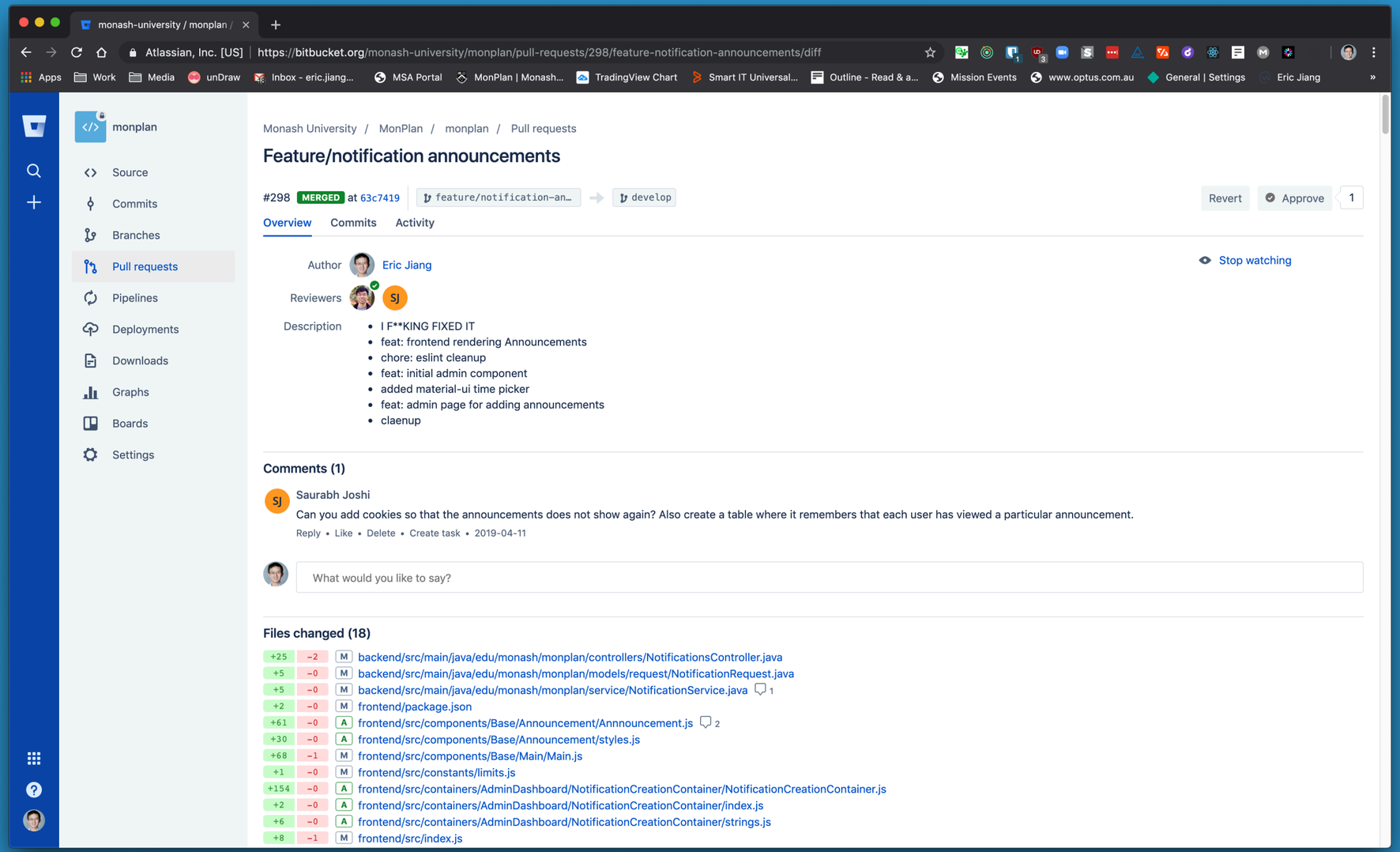
Copyright © Eric Jiang 2018 - 2019 | Adapated from "Gitting Started at Hackathons" tech talk

He then makes a PR into my develop or master branch



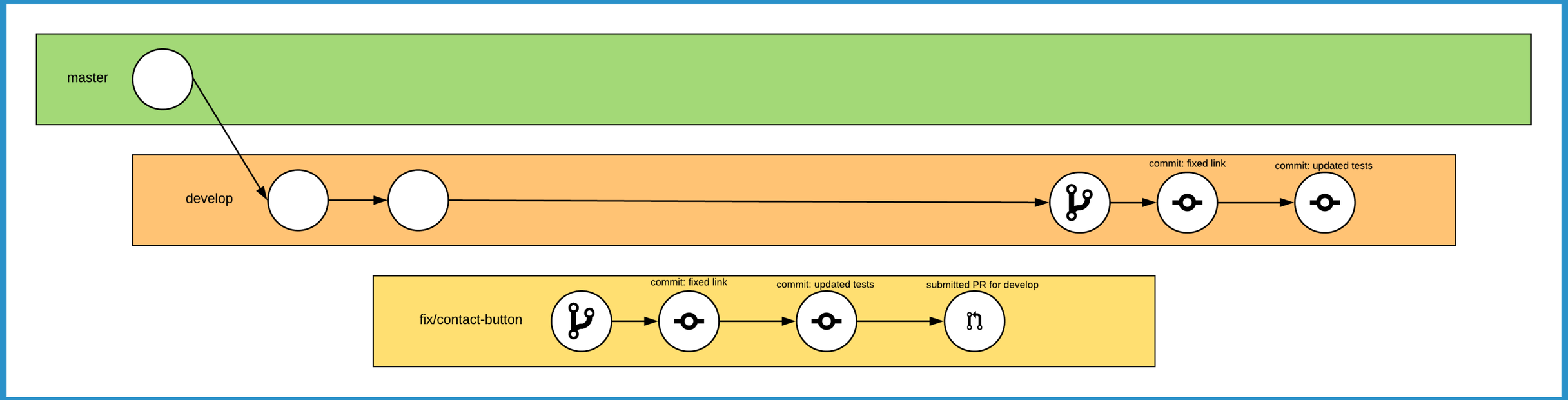
Where we discuss his proposed changes

Copyright © Eric Jiang 2018 - 2019 | Adapated from "Gitting Started at Hackathons" tech talk



Copyright © Eric Jiang 2018 - 2019 | Adapated from "Gitting Started at Hackathons" tech talk

We then merge the Changes



Copyright © Eric Jiang 2018 - 2019 | Adapated from "Gitting Started at Hackathons" tech talk

This would also work for...

— Upgrades to the codebase
— Refactoring our legacy code
— Upgrading frameworks to newer versions

Copyright © Eric Jiang 2018 - 2019 | Adapated from "Gitting Started at Hackathons" tech talk

Why is using GitFlow important?

- We seperate production code and our 'work-in-progress' (WIP) code.
- We have a clearer understanding of what each developer is working on
- We can branch off WIP branches and merge changes in
- Relatively easier (not always) to fix merge conflicts
- Some CI/CD tools only run off branches (not PRs)
- We can set our CI/CD to deployment so that it can deploy off branches (i.e. `develop` to ***dev***, `master` to ***staging*** or ***qat*** and `deploy` to ***prod***)

Copyright © Eric Jiang 2018 - 2019 | Adapated from "Gitting Started at Hackathons" tech talk

Key notes 📝

- Version Control over Development is really important as it helps keep 'backups' and you can see the changes
  - You can always see who pushed out the broken code with `git blame` 😈
- Git is always useful as you can always revert broken code or changes
- Branching and forking is basically the same,
  - when working we typically use branches over forks as we can solve merge conflicts more easily (and locally)

Copyright © Eric Jiang 2018 - 2019 | Adapated from "Gitting Started at Hackathons" tech talk

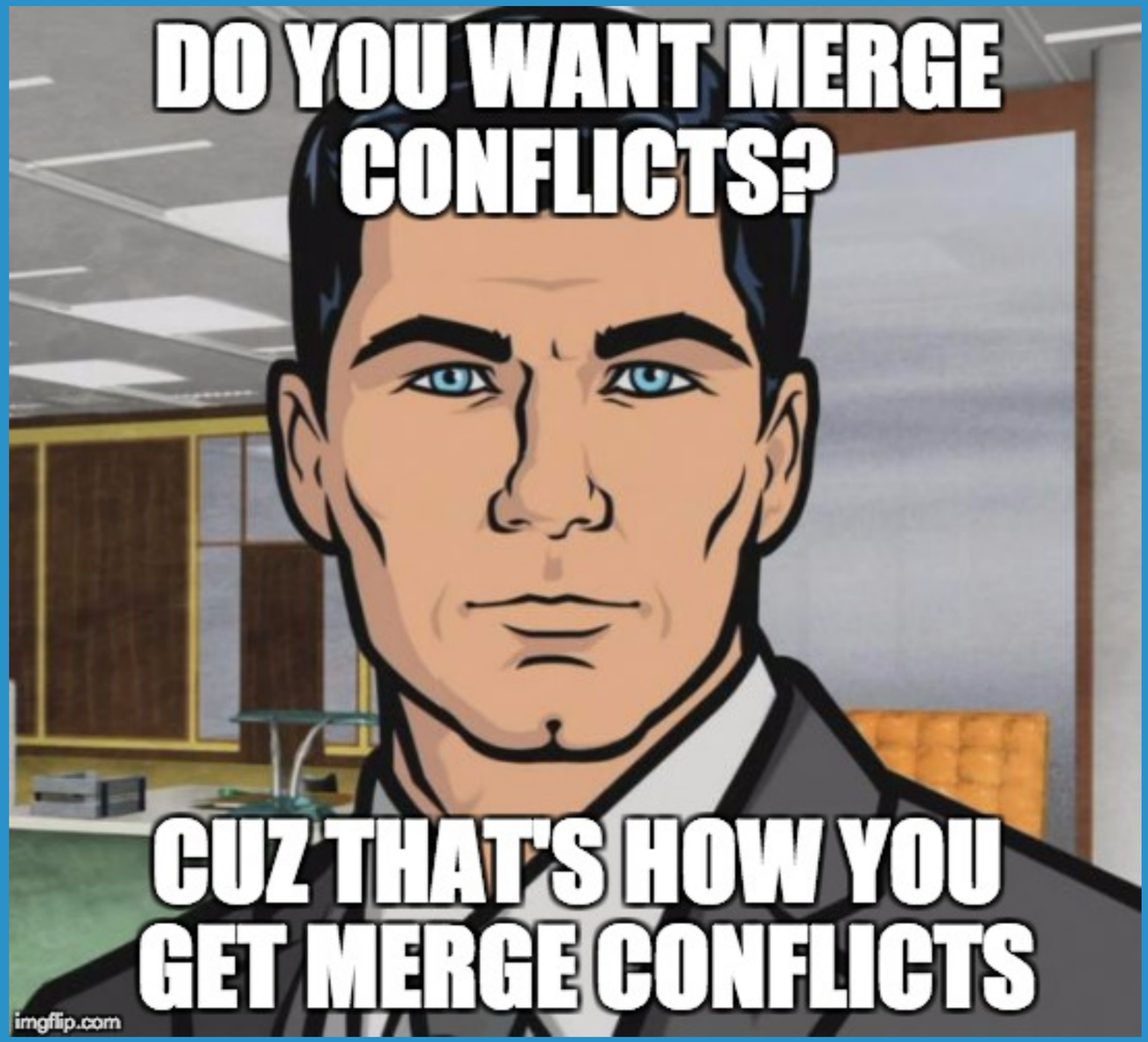
Please DO NOT ever `git push --force`

Copyright © Eric Jiang 2018 - 2019 | Adapated from "Gitting Started at Hackathons" tech talk

Key things to look 🔭 out for.

- Merge conflicts are always the hardest part
- Be careful of `git merge` and `git rebase` commands.
  - This is because rebase always applies your changes last (assumes you are always correct)
  - When merging between branches and fixing conflicts always work with a team-mate

Copyright © Eric Jiang 2018 - 2019 | Adapated from "Gitting Started at Hackathons" tech talk



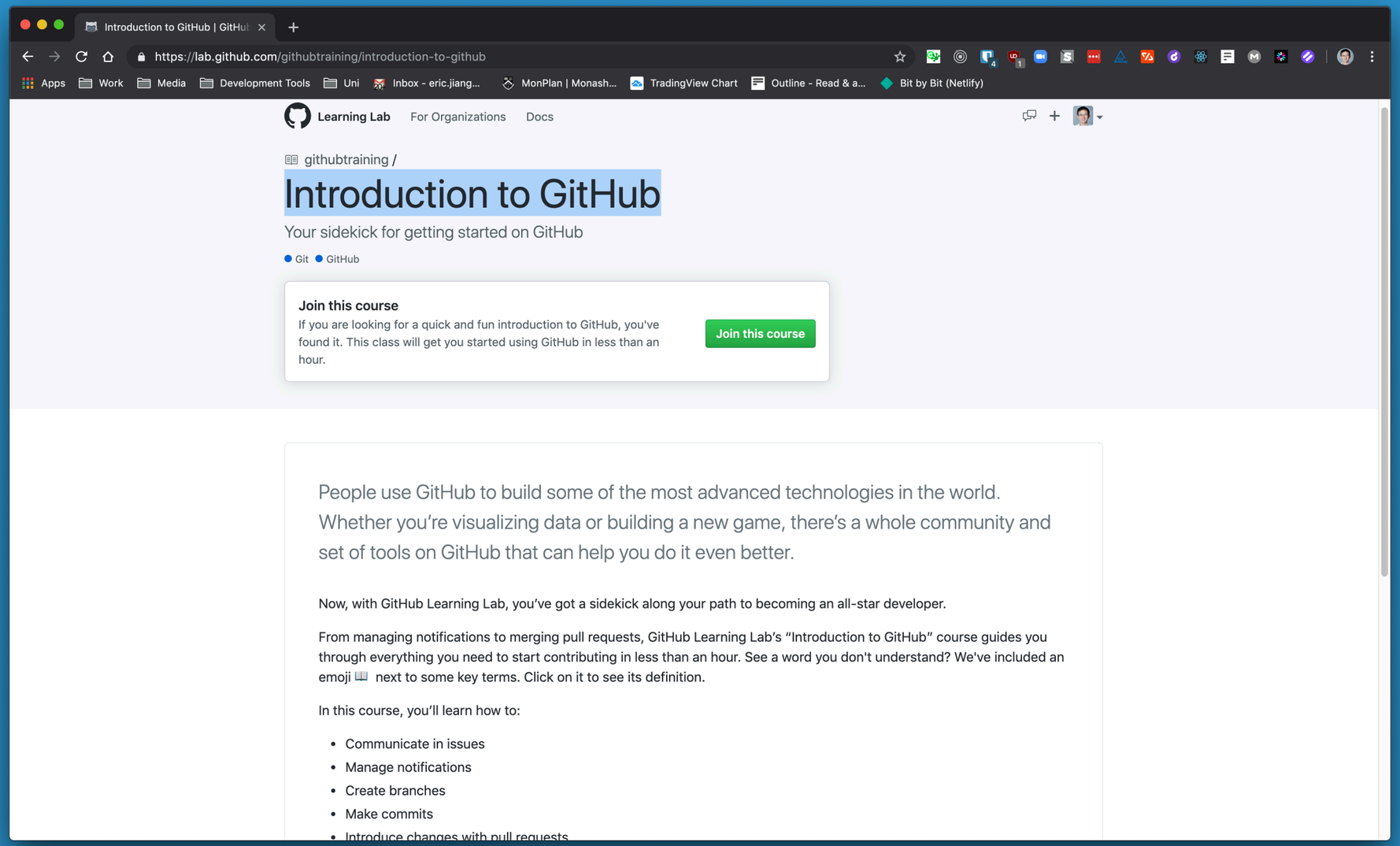
Copyright © Eric Jiang 2018 - 2019 | Adapated from "Gitting Started at Hackathons" tech talk

Got it? ¯_(ツ)_/¯

Let's do some interactive exercises

1. Go to https://github.com to create an account if you don't have one (use your student email address as the primary email)
2. Go to https://lab.github.com and sign up to GitHub Learning Lab
3. We're going to do the *Introduction to GitHub* course

Copyright © Eric Jiang 2018 - 2019 | Adapated from "Gitting Started at Hackathons" tech talk



Copyright © Eric Jiang 2018 - 2019 | Adapated from "Gitting Started at Hackathons" tech talk

So merge conflicts, what are they and how do we resolve them?

Copyright © Eric Jiang 2018 - 2019 | Adapated from "Gitting Started at Hackathons" tech talk

Merge Conflicts

- essentially, when we are merging changes to code and *Git* sees different changes on the same file
- this most likely happens when someone already committed to the branch from earlier on, and hasn't applied their changes on to the branch that we are using
- therefore, the Git history (of our changes) 'diverges' as each commit is a different hash

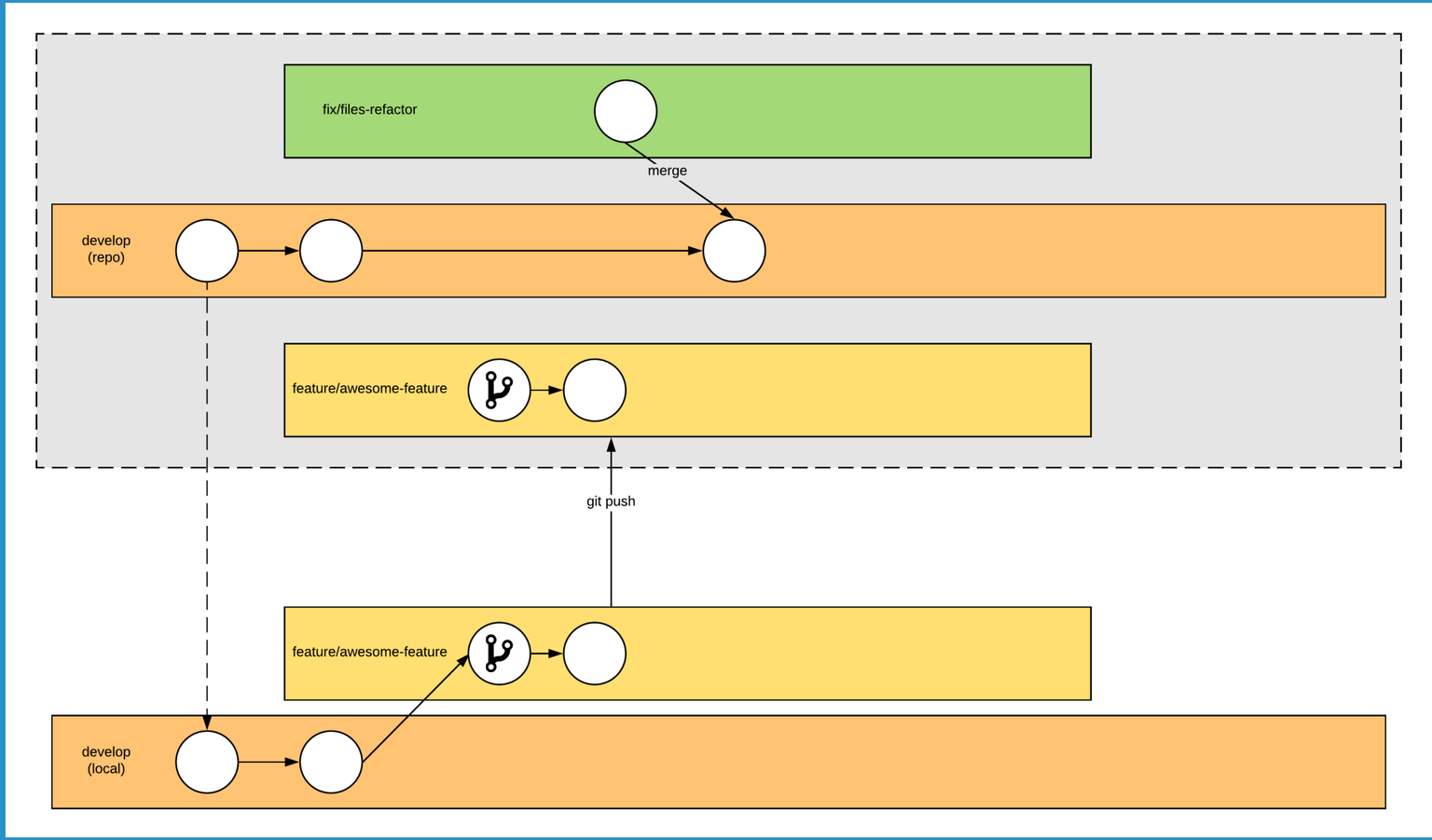
Copyright © Eric Jiang 2018 - 2019 | Adapated from "Gitting Started at Hackathons" tech talk

Here's one of the best and easiest ways to resolve a conflict

1. We go to the target branch and pull down the latest changes
2. We then 'checkout' our current working branch and create a new branch off the working branch
3. We then attempt to merge our target branch into our new branch
4. Resolve Conflicts (by choosing the right pieces of code you want), VSCode makes this really easier
5. Merge the new branch into our current branch
6. Merge the current branch into the target branch

Copyright © Eric Jiang 2018 - 2019 | Adapated from "Gitting Started at Hackathons" tech talk

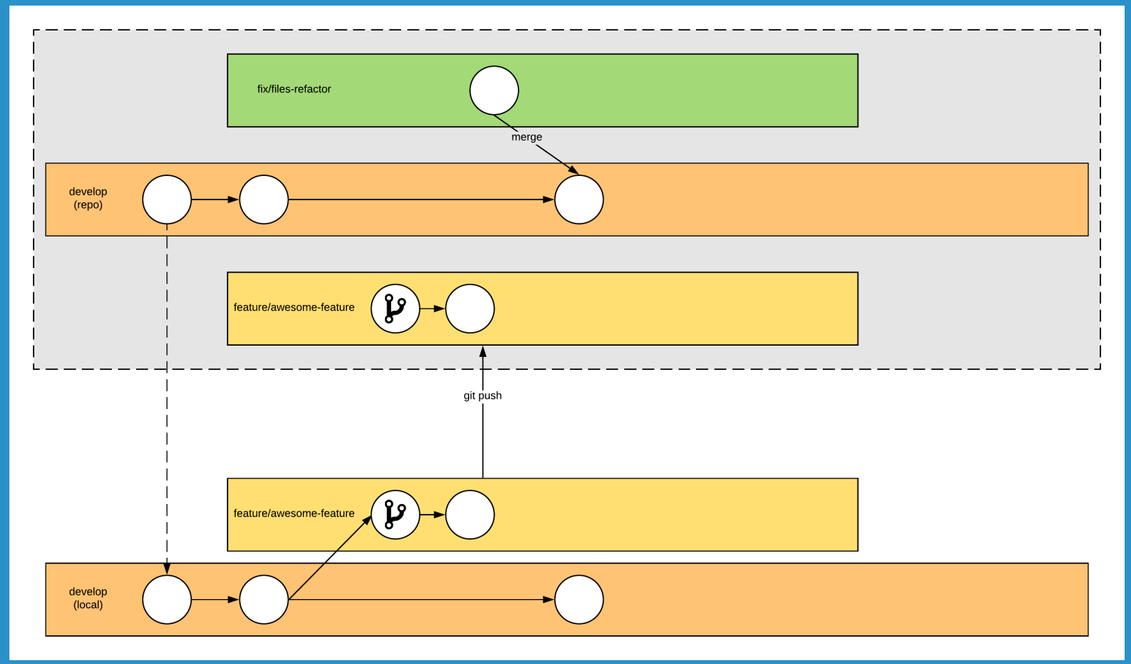
This is the current state of the branches



Copyright © Eric Jiang 2018 - 2019 | Adapated from "Gitting Started at Hackathons" tech talk

We go to the target branch and pull down the latest changes

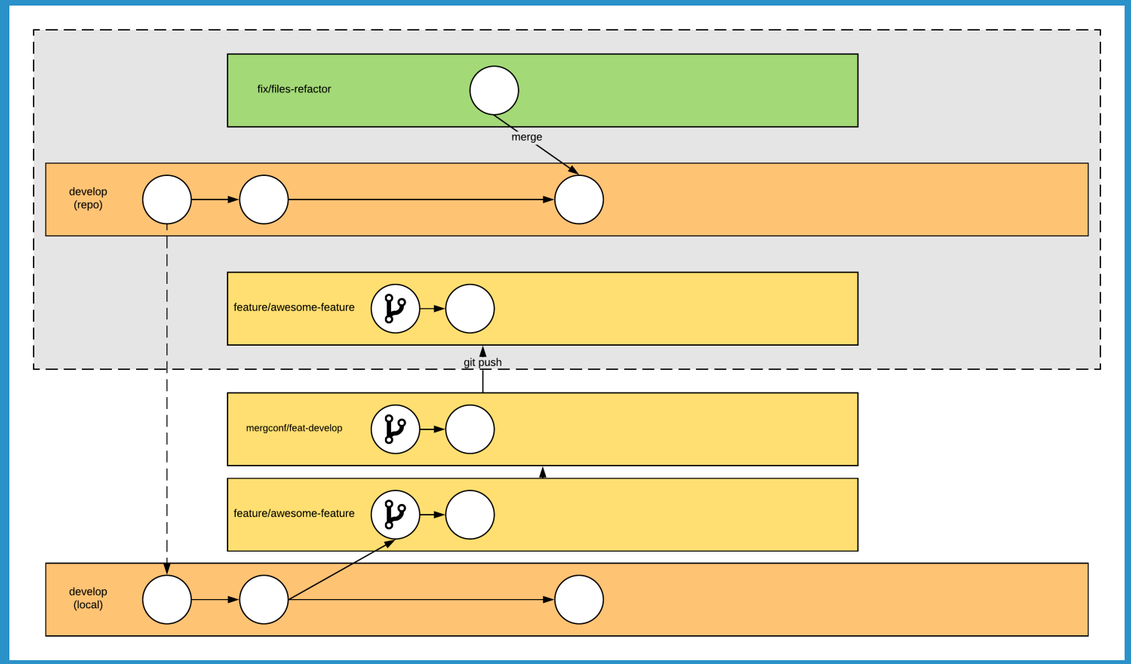
```
git checkout develop
git pull
git checkout feature/awesome-feature
git checkout mergconf/feat-develop
# ...
```



Copyright © Eric Jiang 2018 - 2019 | Adapated from "Gitting Started at Hackathons" tech talk

We then 'checkout' our current working branch and create a new branch off the working branch

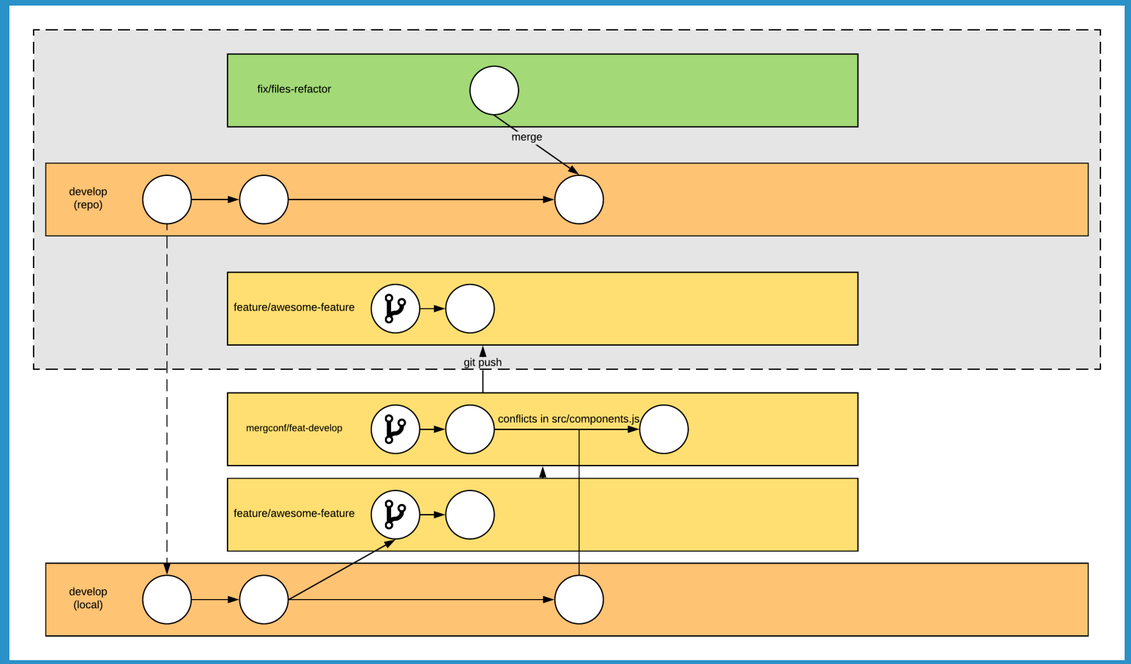
```
git pull
git checkout feature/awesome-feature
git checkout mergconf/feat-develop
git merge develop
# ...
```



Copyright © Eric Jiang 2018 - 2019 | Adapated from "Gitting Started at Hackathons" tech talk

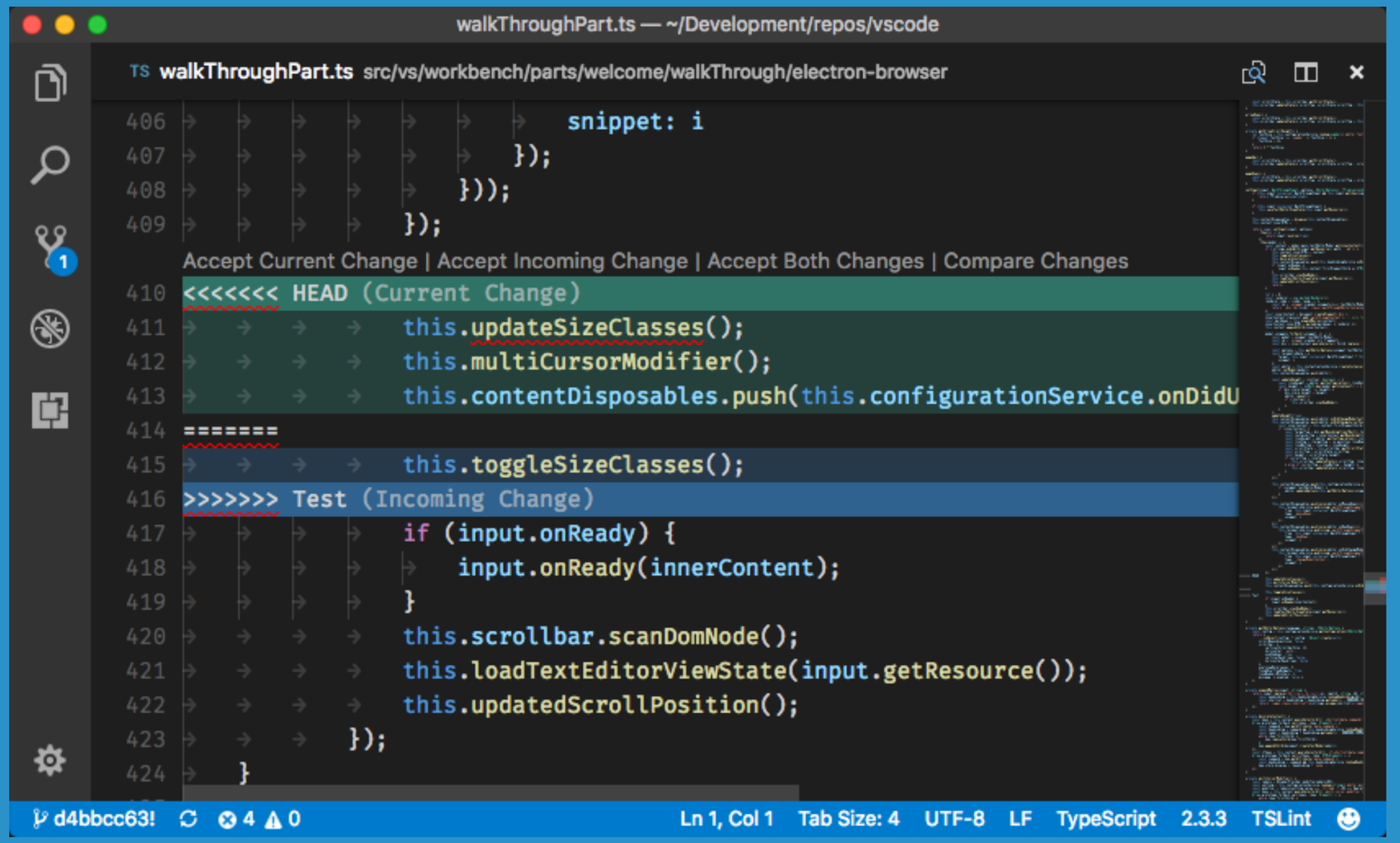
We then attempt to merge our target branch into our new branch

```
git pull
git checkout feature/awesome-feature
git checkout mergconf/feat-develop
git merge develop
# ...
```



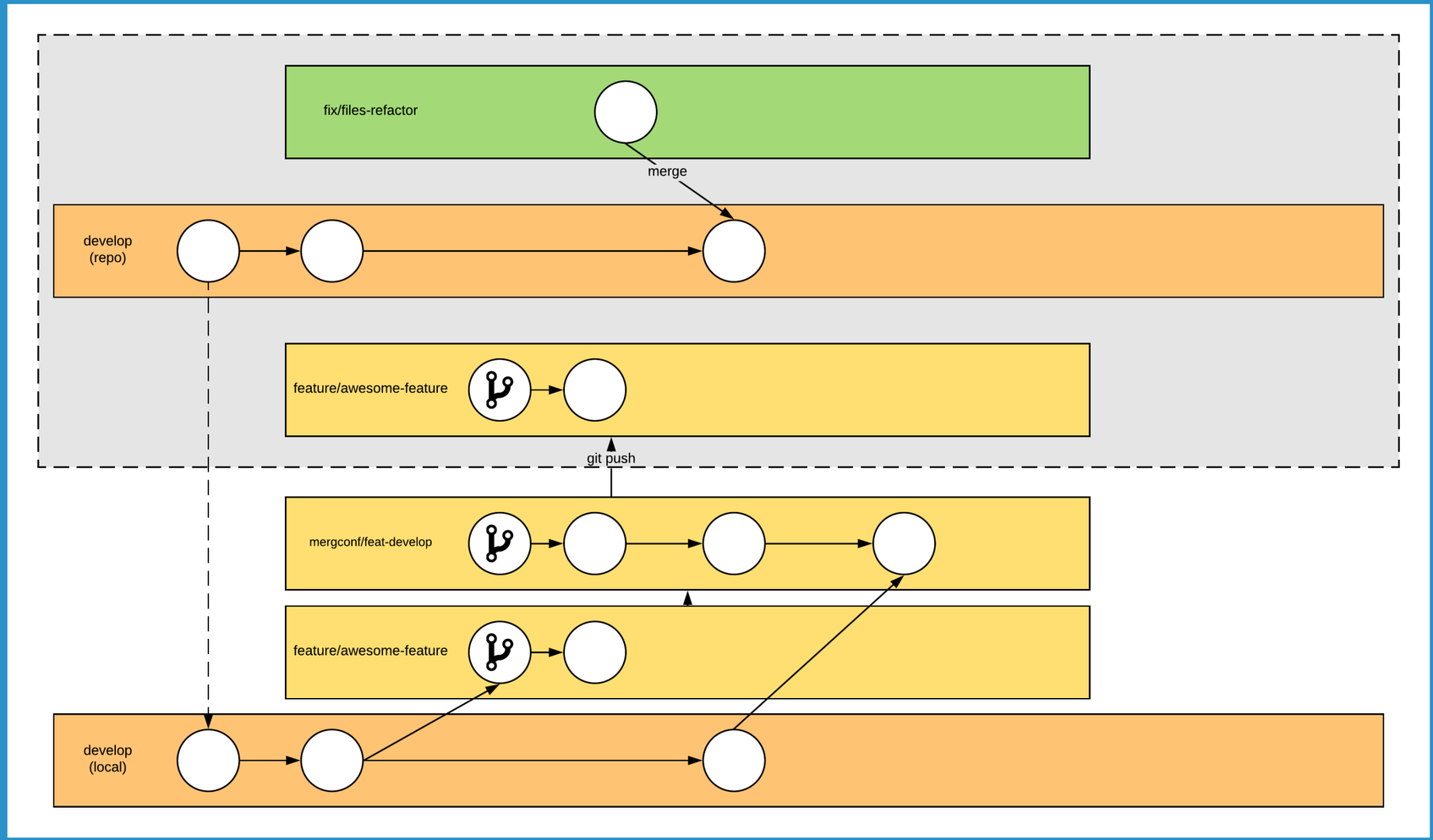
Copyright © Eric Jiang 2018 - 2019 | Adapated from "Gitting Started at Hackathons" tech talk

Resolve Conflicts (by choosing the right pieces of code you want), VSCode makes this really easier



Copyright © Eric Jiang 2018 - 2019 | Adapated from "Gitting Started at Hackathons" tech talk

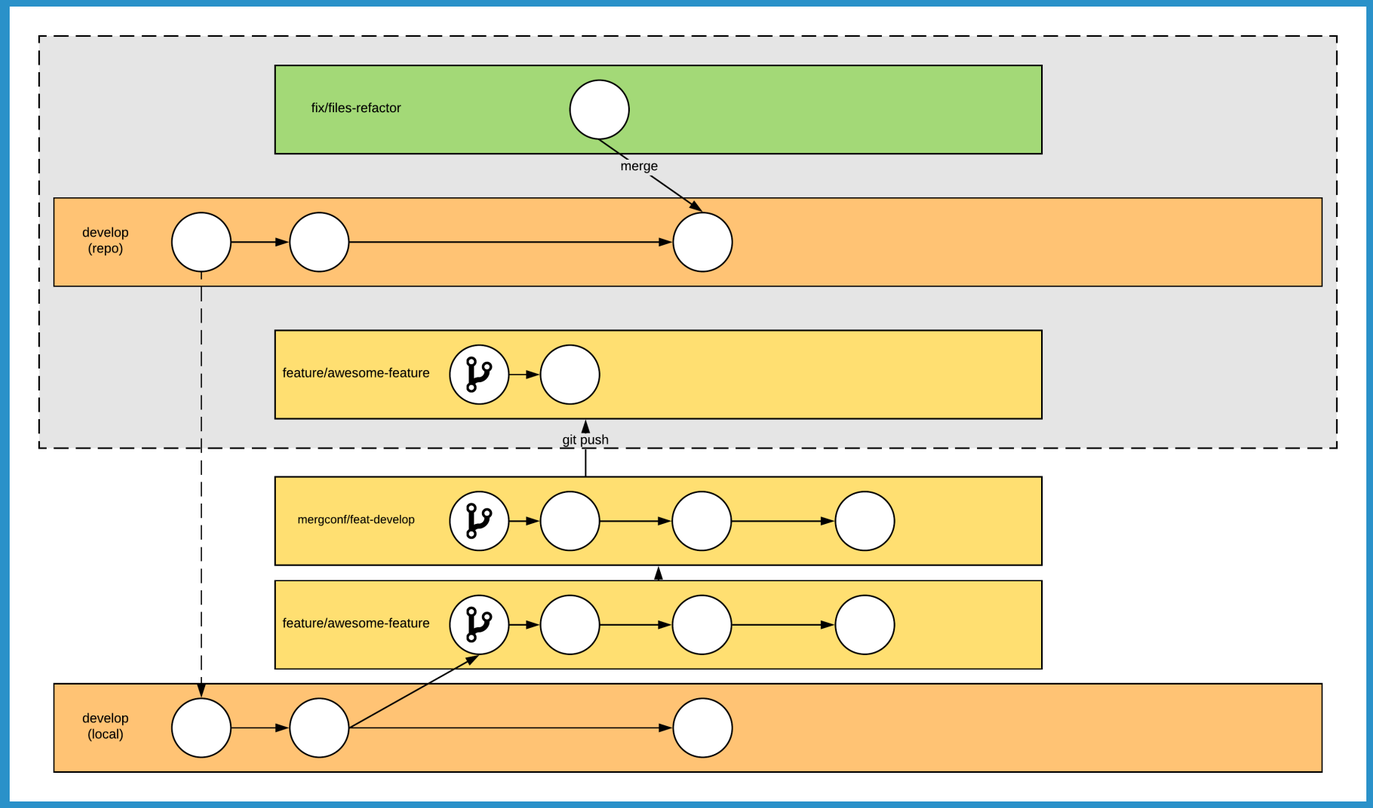
Here's how the state of the branches are:



Copyright © Eric Jiang 2018 - 2019 | Adapated from "Gitting Started at Hackathons" tech talk

Merge the new branch into our current branch

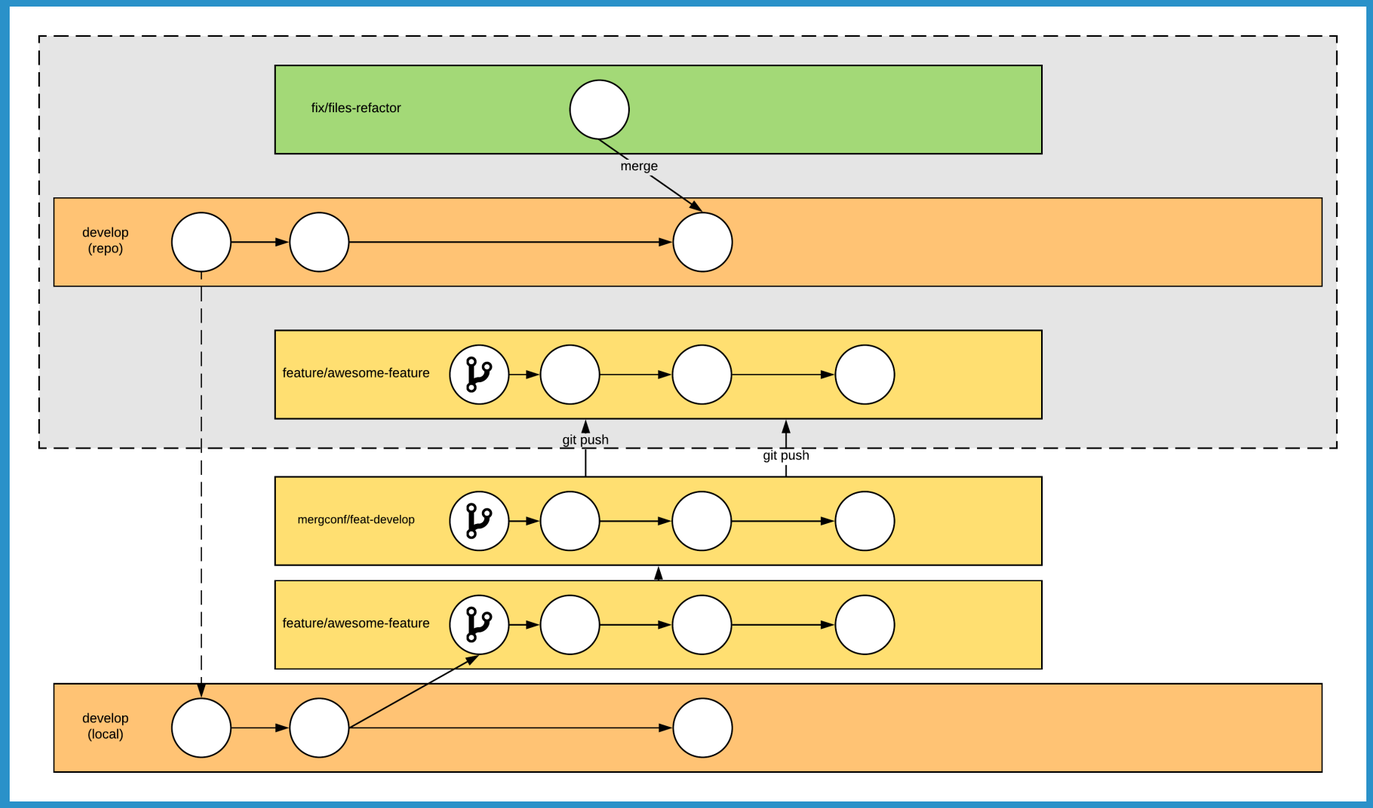
```
git checkout feature/awesome-feature #get to current working branch
git merge mergconf/feat-develop
git push # push to repo
```



Copyright © Eric Jiang 2018 - 2019 | Adapated from "Gitting Started at Hackathons" tech talk

Push changes to repository

```
git checkout feature/awesome-feature #get to current working branch
git merge mergconf/feat-develop
git push # push to repo
```



Copyright © Eric Jiang 2018 - 2019 | Adapated from "Gitting Started at Hackathons" tech talk

Our Conflicts would have been solved now! 

Copyright © Eric Jiang 2018 - 2019 | Adapated from "Gitting Started at Hackathons" tech talk

Questions?

Copyright © Eric Jiang 2018 - 2019 | Adapated from "Gitting Started at Hackathons" tech talk

Additional Resources

Make sure to check out the GitHub Education Pack at https://education.github.com/pack, which includes:

- **free GitHub Pro for students**
- **AWS + credits, free one year algolia,**
- **free 2 year DataDog,**
- **one year free domain from namecheap,**
- **and heaps more!**

Copyright © Eric Jiang 2018 - 2019 | Adapated from "Gitting Started at Hackathons" tech talk

Thank You!!

You can follow me on my social accounts:

— Twitter: @lorderikir
— GitHub: @lorderikir
— LinkedIn @ericjiang97

Copyright © Eric Jiang 2018 - 2019 | Adapated from "Gitting Started at Hackathons" tech talk