Getting Started at Hackathons

Track 1: Gitting Started

Hi, I'm **Eric Jiang** 👋

- Currently, the Project Lead for monPlan
- Co-founded GeckoDM and MARIE.js
- Co-founded and Pitched FutureYou to Marketing
- @lorderikir
- https://lorderikir.me
- eric.jiang@monash.edu
- github.com/lorderikir

So, I love coding 👨‍💻 and I love working in teams 👨‍👩‍👦

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

First of all, what is git?

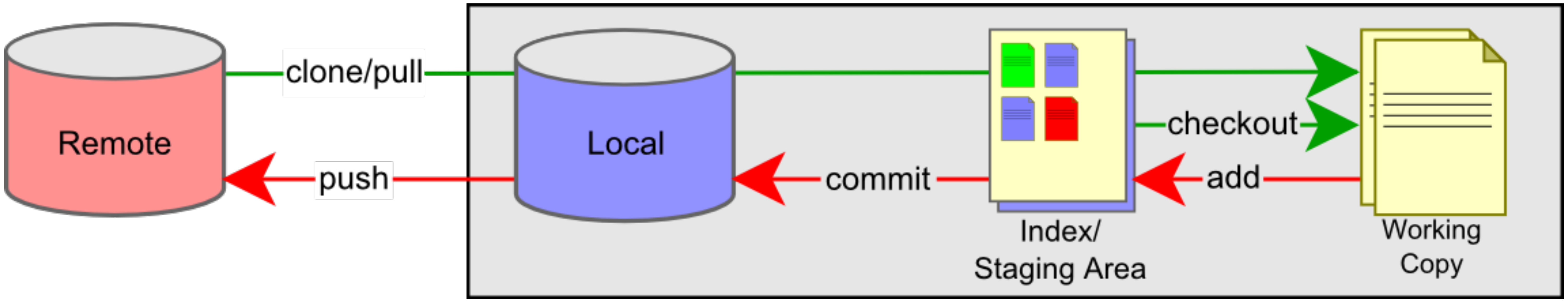
git
SUBVERSION
Final 2
Final Final
FINAL FOR REAL THIS TIME
OK This is definitely final
I prefer the real version control
I said the *real* version control
Perfection

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

How Git Works



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

Using Git within teams

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

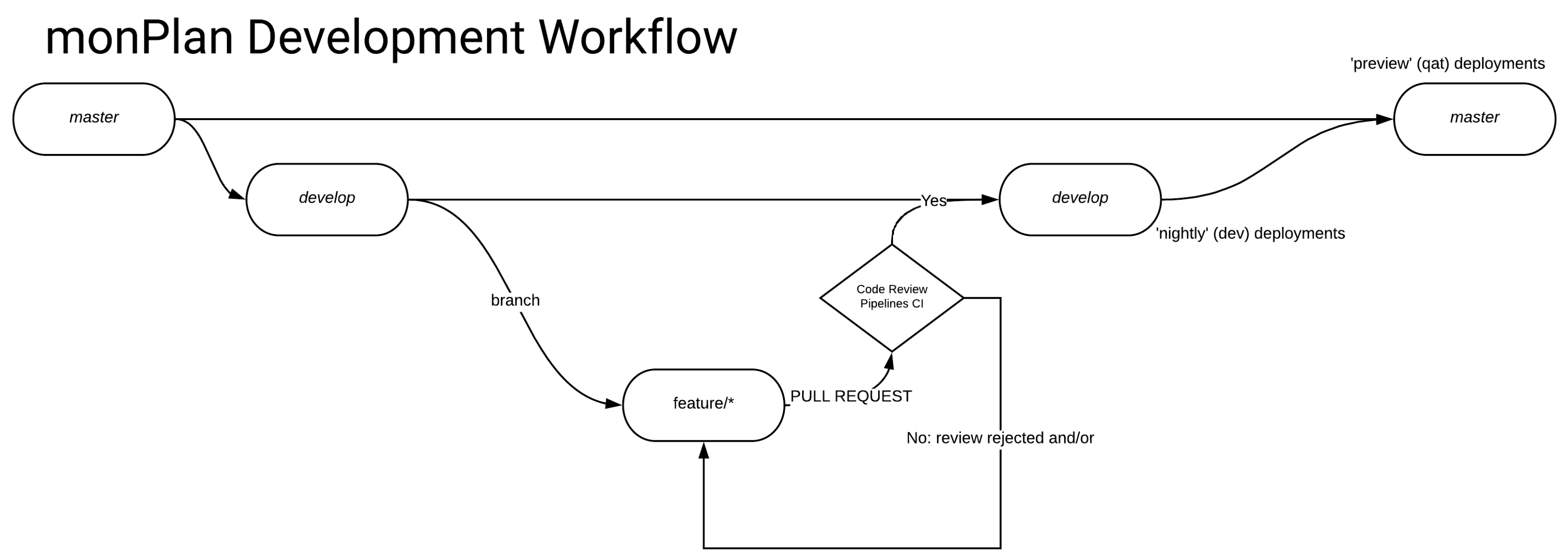
- Using Branches
- Using Forks

Option 1: Use Branches 🌳 for Versioning Control

1. Make a branch with the feature name or your own username
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*

GitFlow - Used with monPlan Git Workflow



monPlan Development Workflow
'preview' (qat) deployments
master
develop
Yes
develop
master
'nightly' (dev) deployments
branch
Code Review Pipelines CI
feature/*
PULL REQUEST
No: review rejected and/or

- **master**: branch is the key branch, everytime for release
- **develop**: *unstable*, most of the PRs should go here
- **'feature/*'**, **'fix/***, etc.: are 'for purpose' branches, these branches are for development
- **deploy** (not shown), is for **manual** deployments to prod

This slide has been adapted from my ***CI-CD talk***

MONASH University

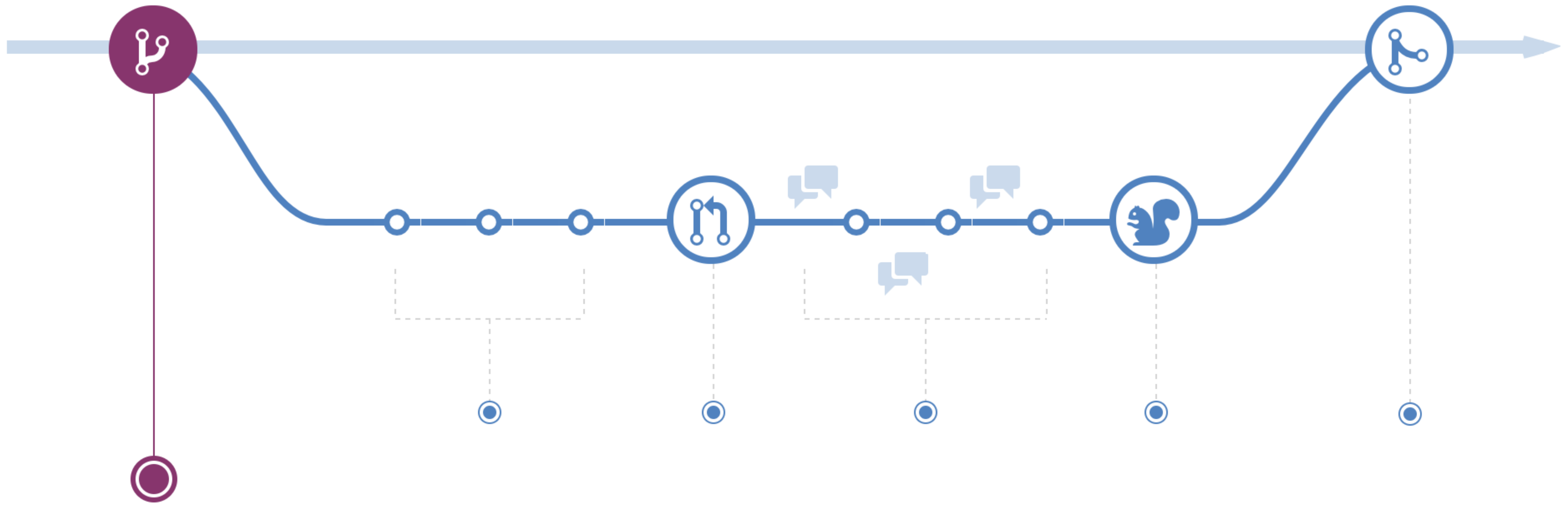
So we know that development is done incrementally

Imagine we using Git within our practices

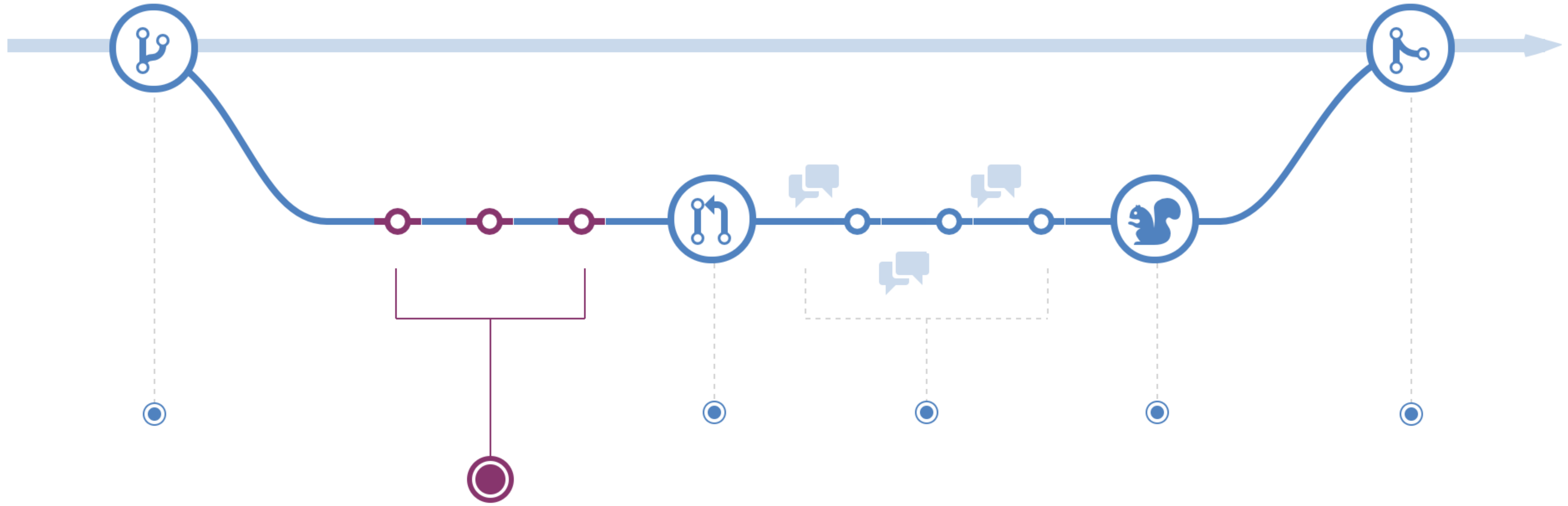
And one of my team mates, Nicholas has found a bug within one of our buttons.

So, he creates a new branch to fix the bug

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

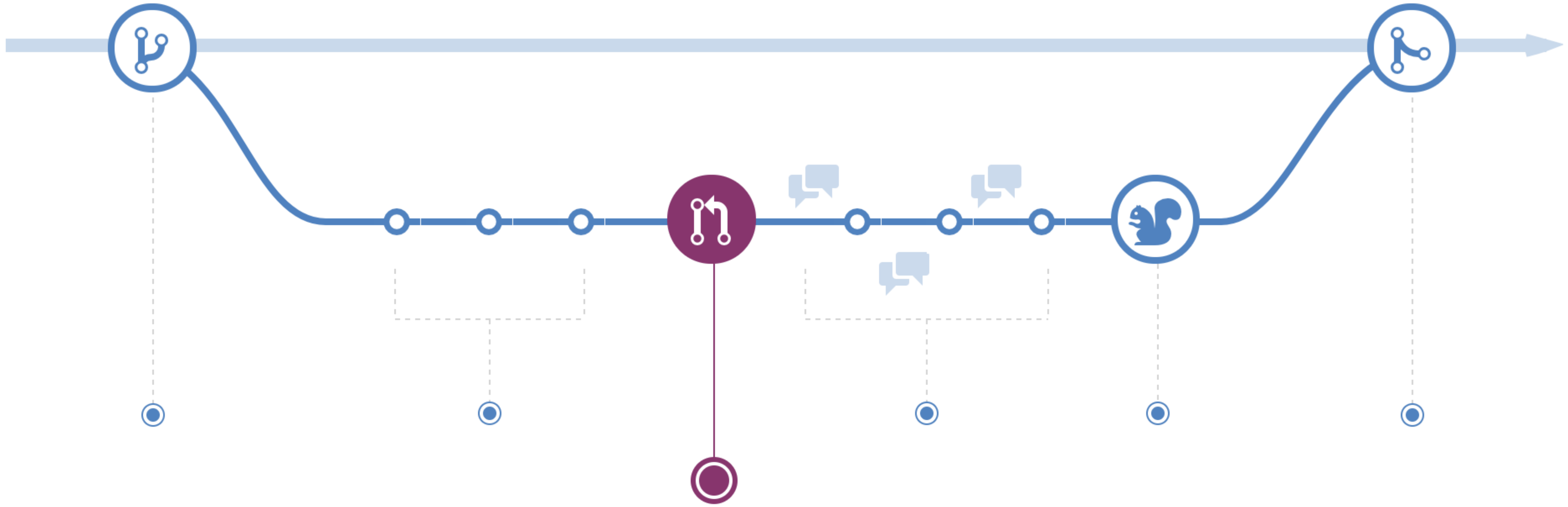


He fixes the code and stages the change in commits

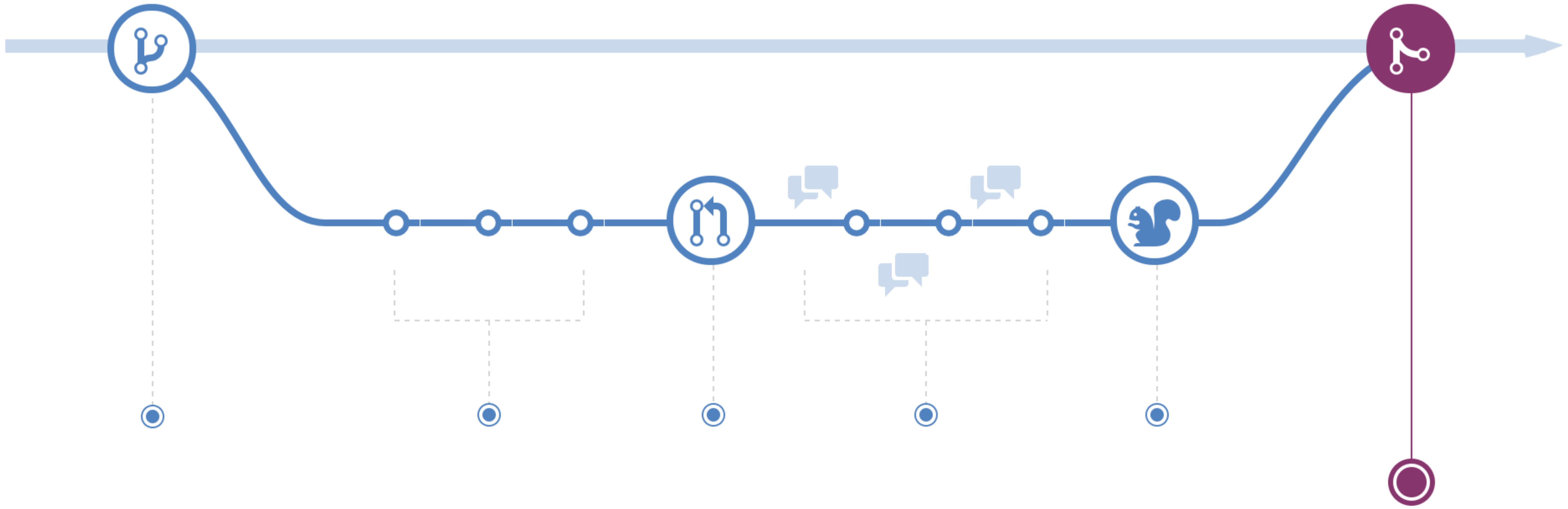


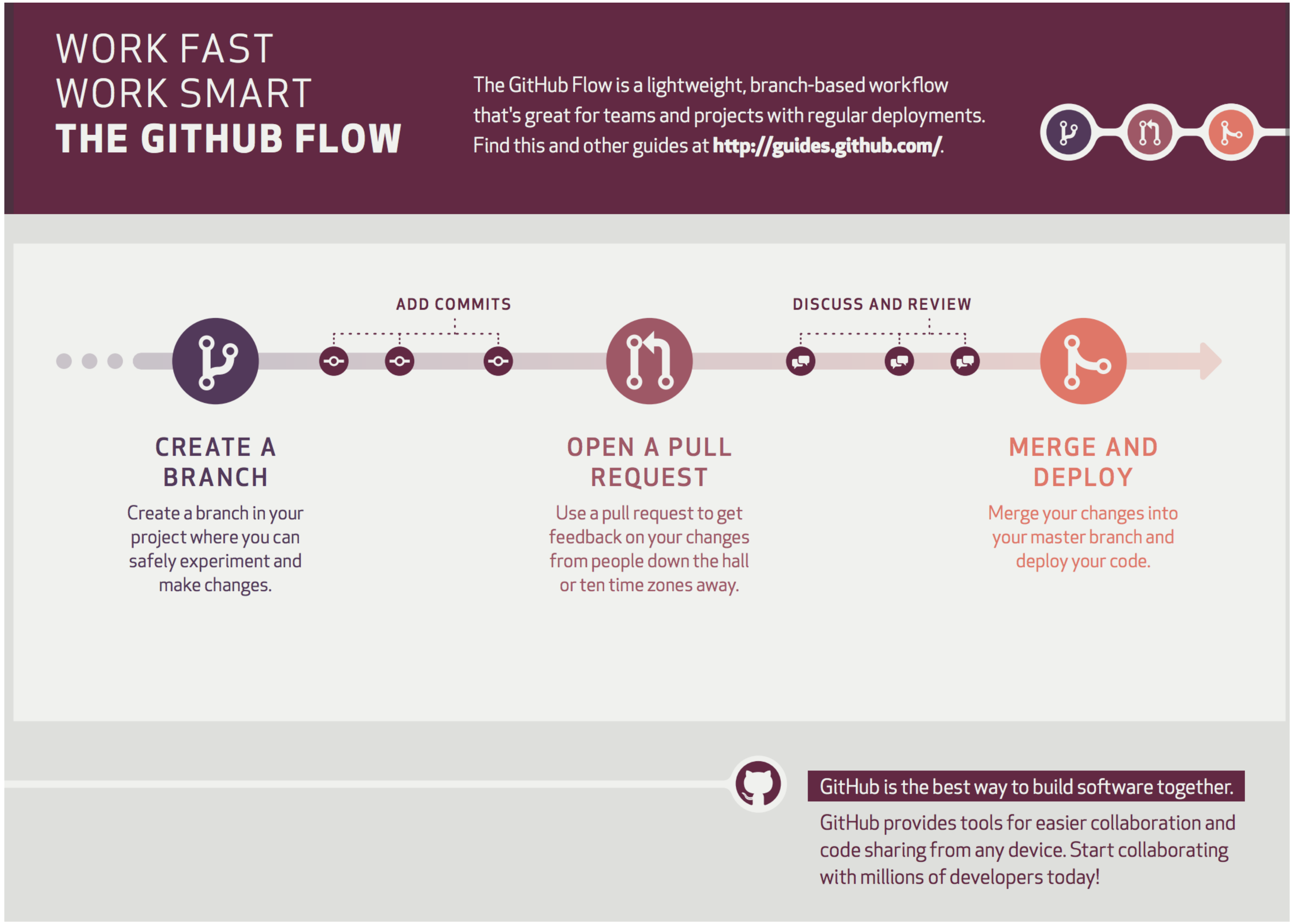
```
git add .
git commit -m "new commit"
git push
```

He then makes a PR into my develop or master branch



We then Review and Discuss the Changes and Merge the Changes





WORK FAST
WORK SMART
THE GITHUB FLOW

The GitHub Flow is a lightweight, branch-based workflow that's great for teams and projects with regular deployments. Find this and other guides at **http://guides.github.com/**.

CREATE A BRANCH

Create a branch in your project where you can safely experiment and make changes.

OPEN A PULL REQUEST

Use a pull request to get feedback on your changes from people down the hall or ten time zones away.

MERGE AND DEPLOY

Merge your changes into your master branch and deploy your code.

GitHub is the best way to build software together.

GitHub provides tools for easier collaboration and code sharing from any device. Start collaborating with millions of developers today!

This would also work for...

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

Unfortunately we won't go into fixing merge conflicts in this talk

Option 2: Using Forks 🍴 for Versioning Control

The best way to image a fork, is image a copy of the main repository that you own that you can *pull*, *merge* and apply changes to.

(We won't go into much detail here.)

Questions?

🤔 🎤 📣

Goodbye 👋

Track 2: Firebase + ReactJS for Hacks coming soon