

# GITHUB TUTORIAL

Creative Software Architectures for  
Collaborative Projects

CS 130

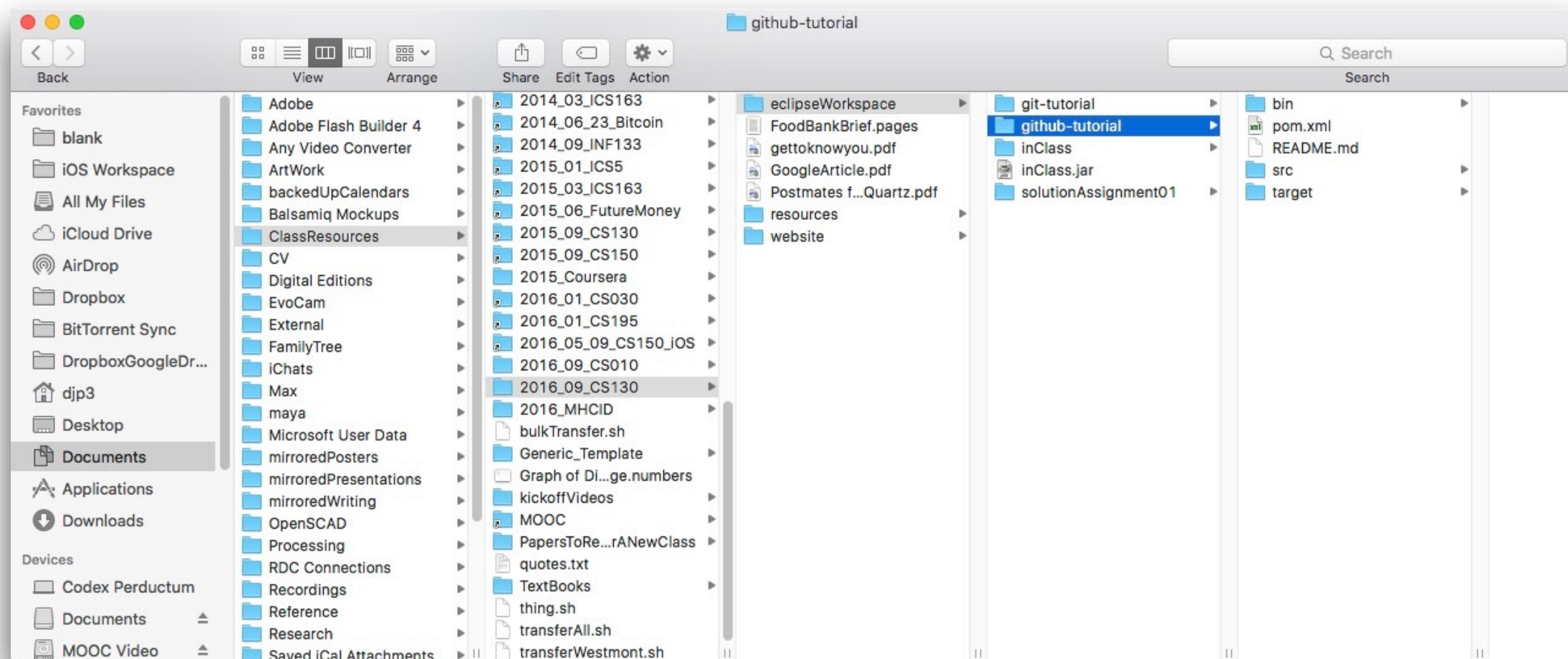
Donald J. Patterson



# GET A COPY OF THE REPOSITORY

## PREPARE YOUR LOCAL SYSTEM TO RECEIVE

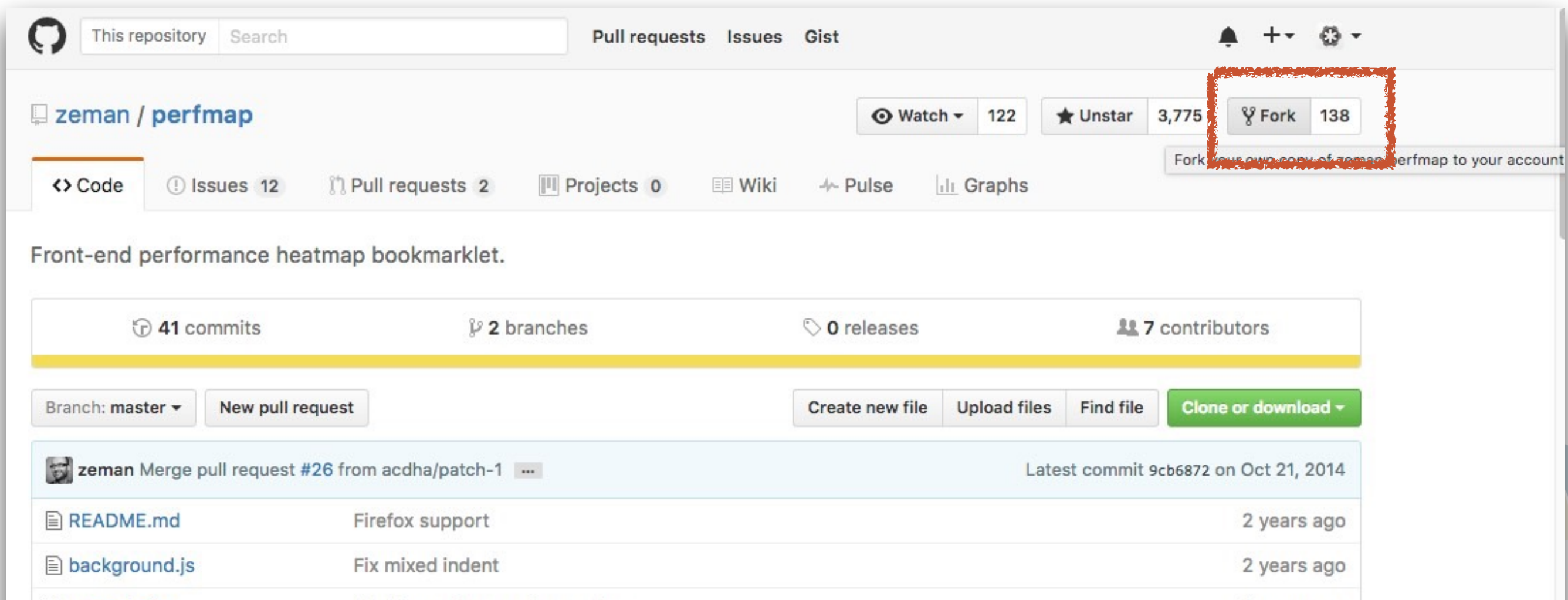
- Identify the workspace that you want to use for development on your hard drive
- “Where do you want to put the files?”



# GET A COPY OF THE REPOSITORY

## FORK THE REPOSITORY ON GITHUB

- Log in to Github
- Find the correct repository and “fork it”



The screenshot shows the GitHub interface for the repository 'zeman/perfmap'. The 'Fork' button is highlighted with a red box. Below the repository name, there are tabs for 'Code', 'Issues', 'Pull requests', 'Projects', 'Wiki', 'Pulse', and 'Graphs'. The repository description is 'Front-end performance heatmap bookmarklet.' Below this, there are statistics: 41 commits, 2 branches, 0 releases, and 7 contributors. At the bottom, there is a list of recent commits, including 'Firefox support' and 'Fix mixed indent'.

GitHub repository page for **zeman / perfmap**.

Buttons: Watch (122), Unstar (3,775), **Fork (138)** (highlighted with a red box).

Repository description: Front-end performance heatmap bookmarklet.

Statistics: 41 commits, 2 branches, 0 releases, 7 contributors.

Actions: Branch: master, New pull request, Create new file, Upload files, Find file, Clone or download.

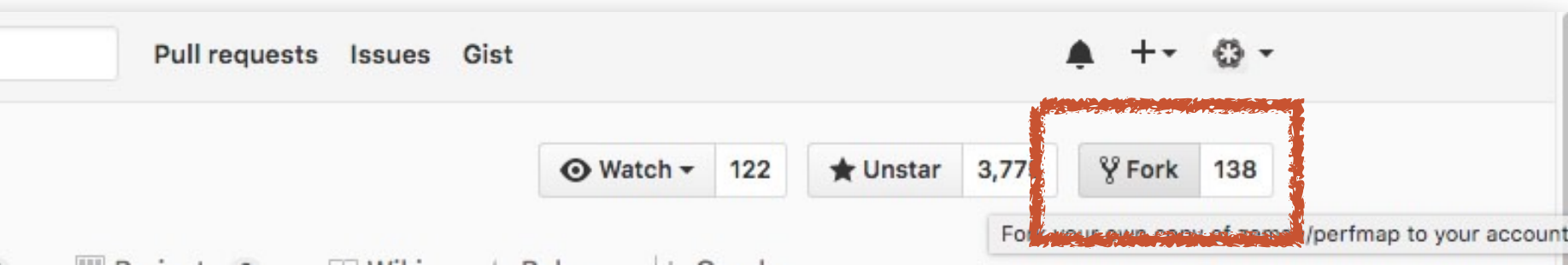
Recent commits:

- zeman Merge pull request #26 from acdha/patch-1 (Latest commit 9cb6872 on Oct 21, 2014)
- README.md: Firefox support (2 years ago)
- background.js: Fix mixed indent (2 years ago)

# GET A COPY OF THE REPOSITORY

## FORK THE REPOSITORY ON GITHUB

- Log in to Github
- Find the correct repository and “fork it”
- This is going to create a copy of this repository in your own github account



# GET A COPY OF THE REPOSITORY

## CLONE YOUR FORK


- On your computer, using git (or SourceTree), clone your github forked project from URL
- Make the destination path a directory in your eclipse workspace
- Make the name something that you can remember for importing in Eclipse
- `/Users/djp3/Documents/ClassResources/  
2016_09_CS130/eclipseWorkspace/github-tutorial`





# GET A COPY OF THE REPOSITORY

## CLONE YOUR FORK

**SourceTree**  
  

### Clone a repository


Cloning is even easier if you set up a [remote account](#).

Source URL:

Destination Path:

Name:

► Advanced Options

 This is a Git repository

# SETUP ECLIPSE

## IMPORT THE REPO INTO ECLIPSE

- Open Eclipse
- Create a new Java project at the location where you checked out the remote repository.
- This will cause Eclipse to use the same files that git (via SourceTree) is keeping up to date





## New Java Project

### Create a Java Project

Create a Java project in the workspace or in an external location.



Project name:

☒ Use default location

Location:

[Browse...](#)

#### JRE

☒ Use an execution environment JRE:



☐ Use a project specific JRE:



☐ Use default JRE (currently 'Java SE 8 [1.8.0\_45]')

[Configure JREs...](#)

#### Project layout

☐ Use project folder as root for sources and class files

☒ Create separate folders for sources and class files

[Configure default...](#)


#### Working sets

☐ Add project to working sets

Working sets:



[Select...](#)

 The wizard will automatically configure the JRE and the project layout based on the existing source.



< Back

Next >

Cancel

Finish

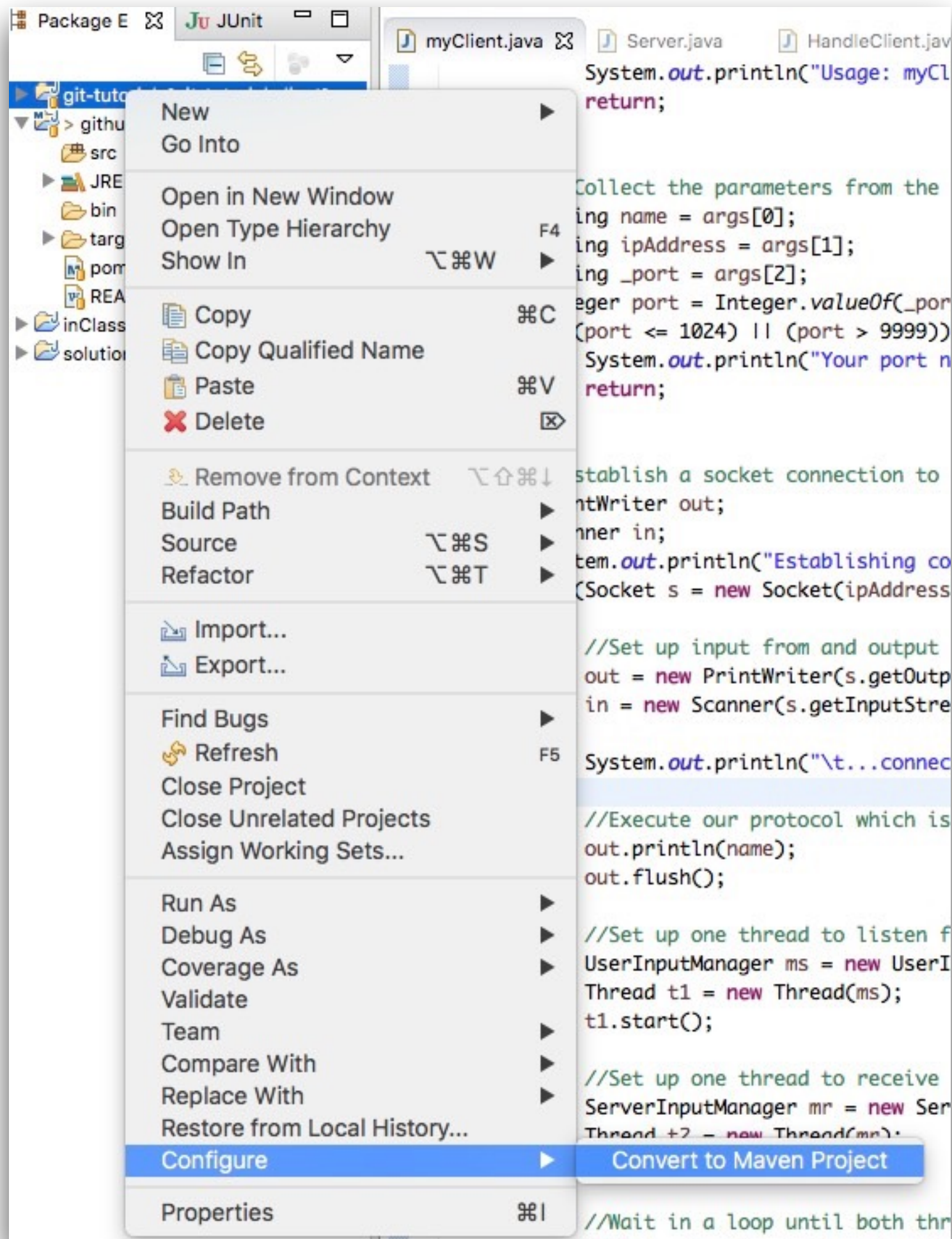


# SETUP ECLIPSE

## CONVERT TO A MAVEN PROJECT

- Maven is a system for compiling java projects from the command line
- We need it to work with some other tools





# GET THE WORKFLOW READY

## ADD THE ORIGINAL GITHUB REPO

- In SourceTree add the original project as a remote repository
  - the one you forked from



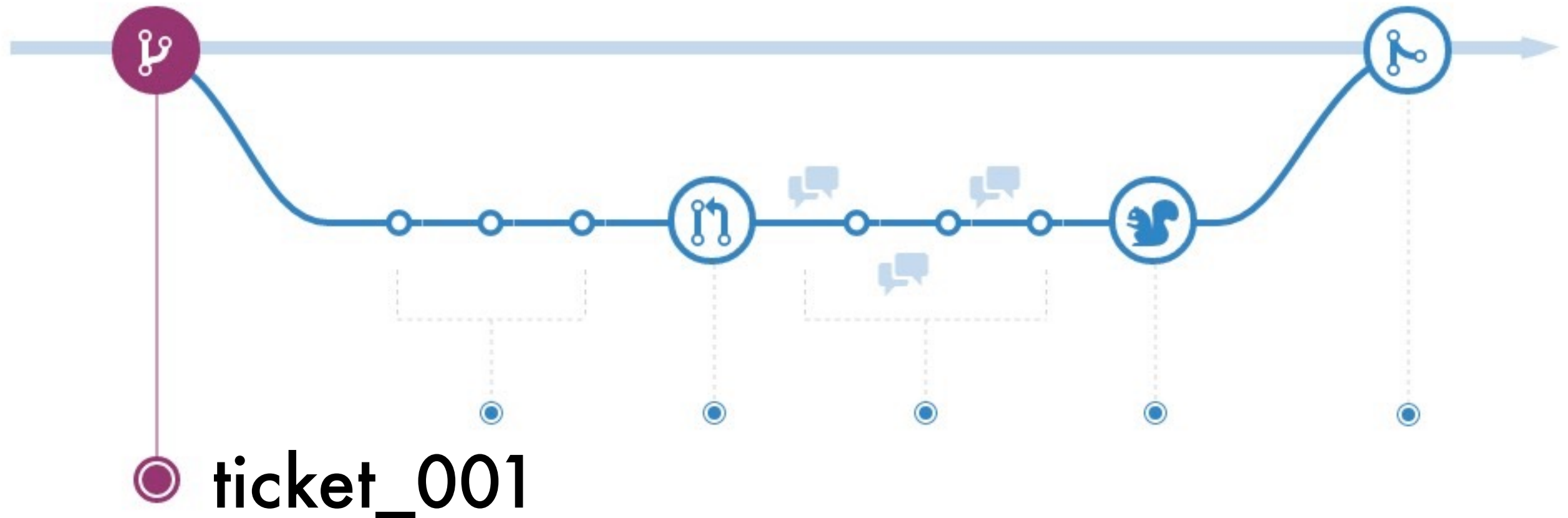
# GET THE WORKFLOW READY

## START CODING

- Use the GitHub workflow



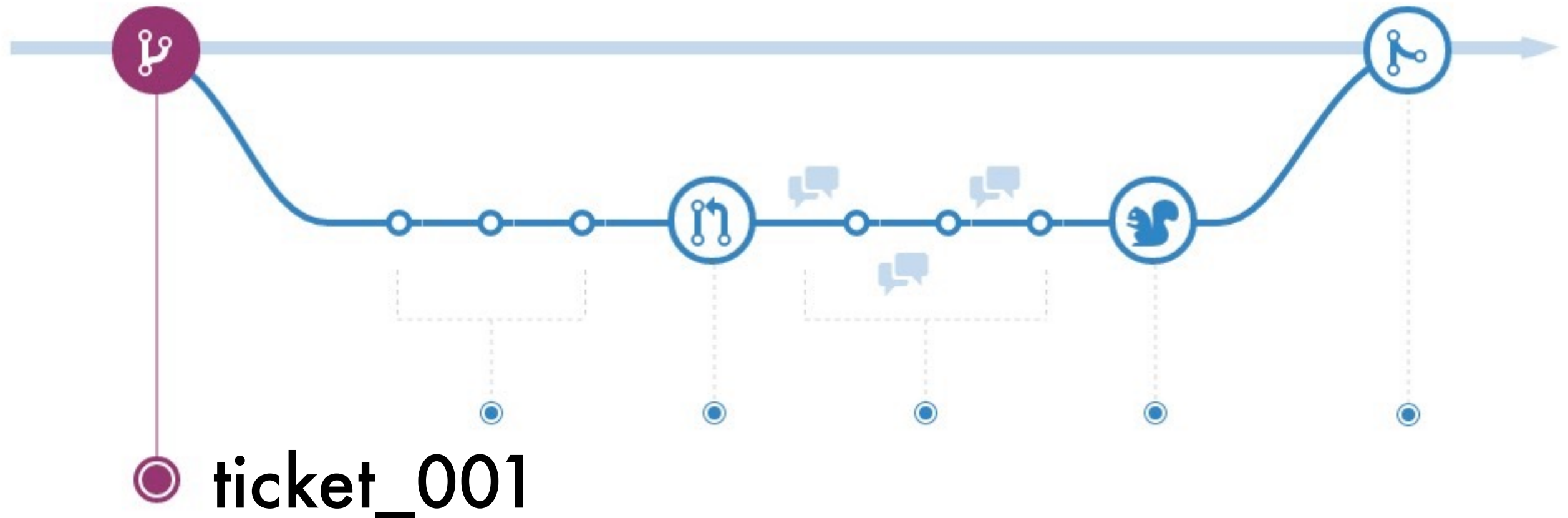
# GITHUB WORKFLOW - CREATE A BRANCH



- When you're working on a project, you're going to have a bunch of different features or ideas in progress at any given time – some of which are ready to go, and others which are not. Branching exists to help you manage this workflow.



# GITHUB WORKFLOW - CREATE A BRANCH

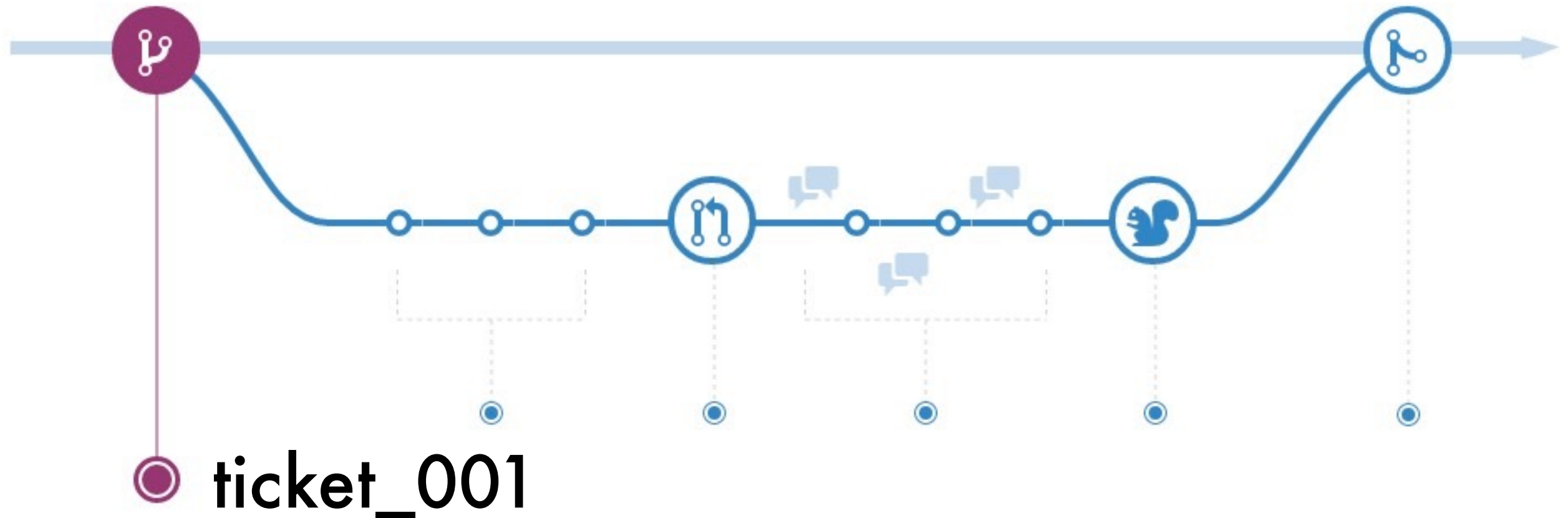


- When you create a branch in your project, you're creating an environment where you can try out new ideas. Changes you make on a branch don't affect the master branch, so you're free to experiment and commit changes, safe in the knowledge that your branch won't be merged until it's ready to be reviewed by someone you're collaborating with.

<https://guides.github.com/introduction/flow/index.html>



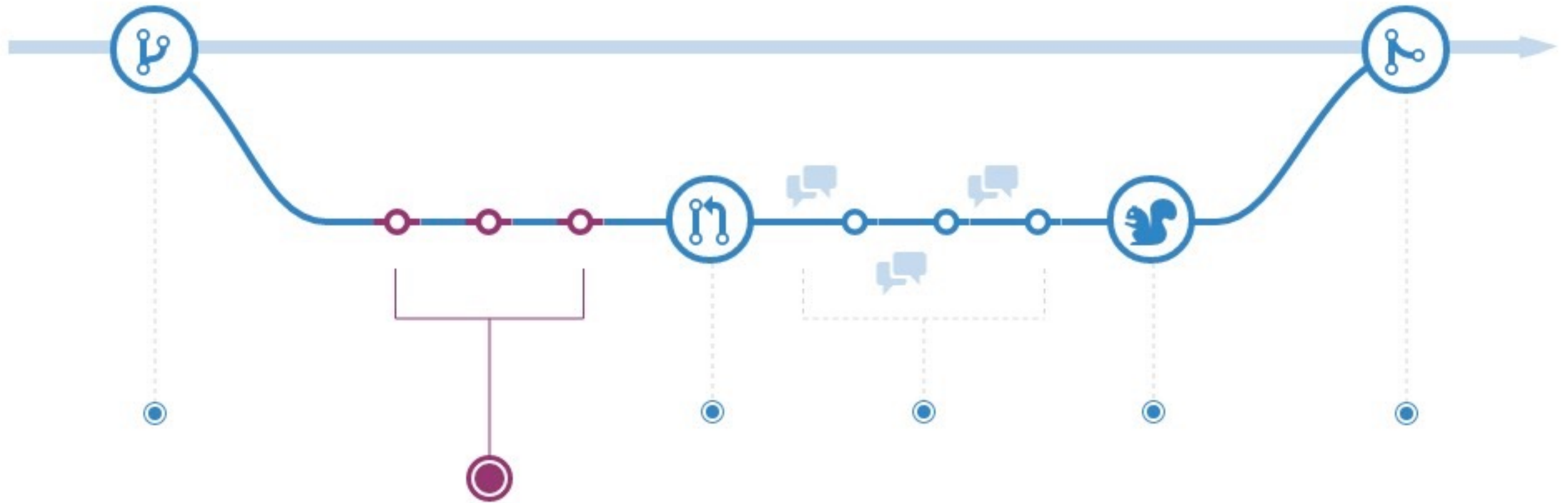
# GITHUB WORKFLOW - CREATE A BRANCH



- Branching is a core concept in Git, and the entire GitHub Flow is based upon it. There's only one rule: anything in the master branch is always deployable.
- Your branch name should be descriptive (e.g., refactor-authentication, user-content-cache-key, make-retina-avatars), so that others can see what is being worked on.

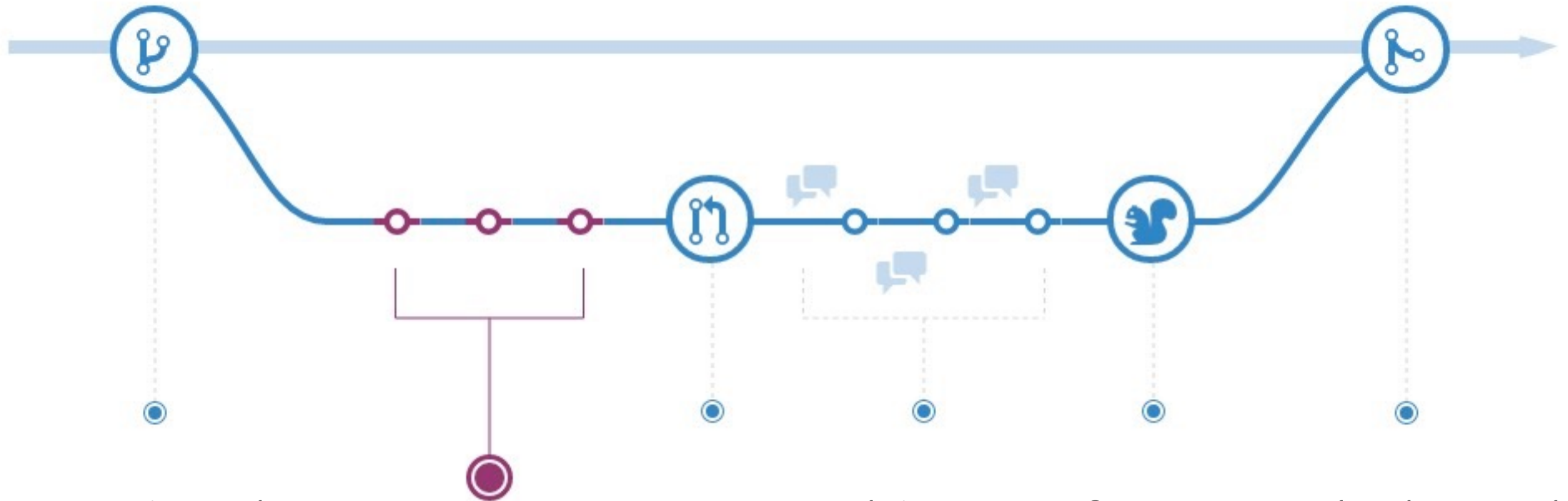
<https://guides.github.com/introduction/flow/index.html>

# GITHUB WORKFLOW - ADD COMMITS



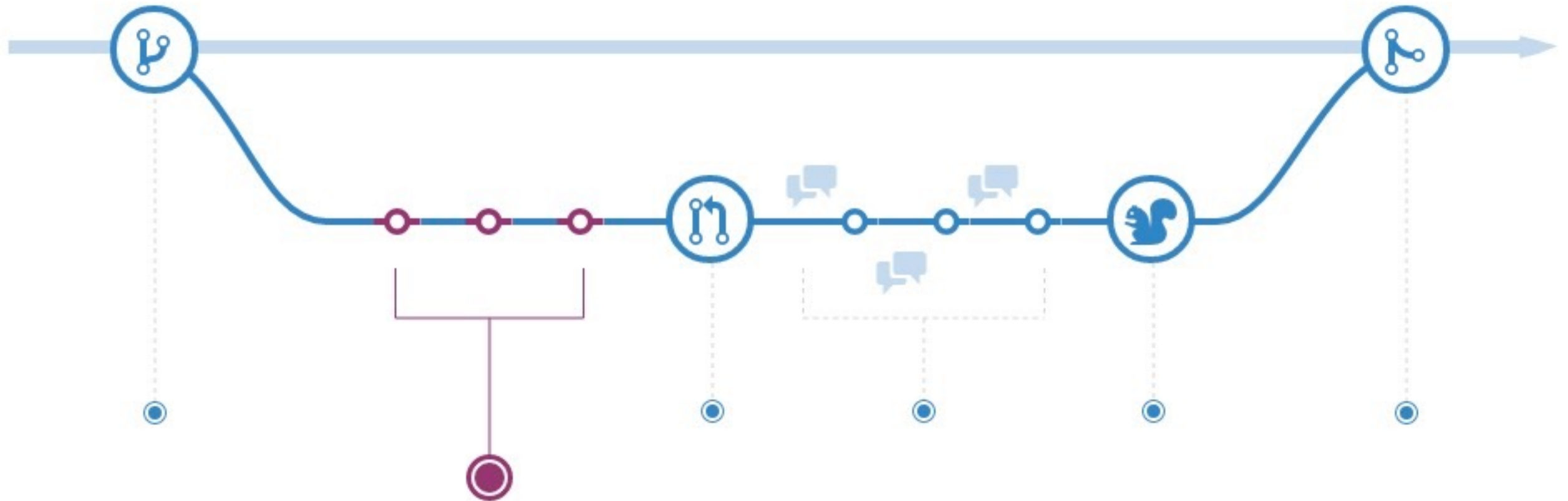
- Once your branch has been created, it's time to start making changes. Whenever you add, edit, or delete a file, you're making a commit, and adding them to your branch. This process of adding commits keeps track of your progress as you work on a feature branch.

# GITHUB WORKFLOW - ADD COMMITS



- Commits also create a transparent history of your work that others can follow to understand what you've done and why. Each commit has an associated commit message, which is a description explaining why a particular change was made. Furthermore, each commit is considered a separate unit of change. This lets you roll back changes if a bug is found, or if you decide to head in a different direction.

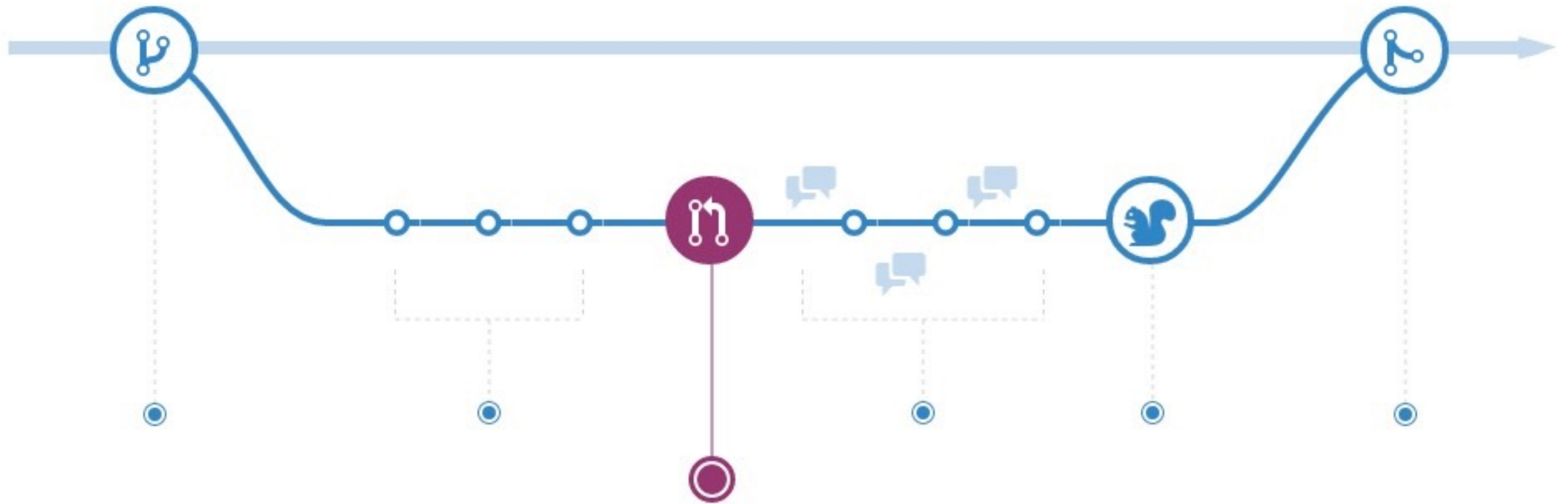
# GITHUB WORKFLOW - ADD COMMITS



- Commit messages are important, especially since Git tracks your changes and then displays them as commits once they're pushed to the server. By writing clear commit messages, you can make it easier for other people to follow along and provide feedback.



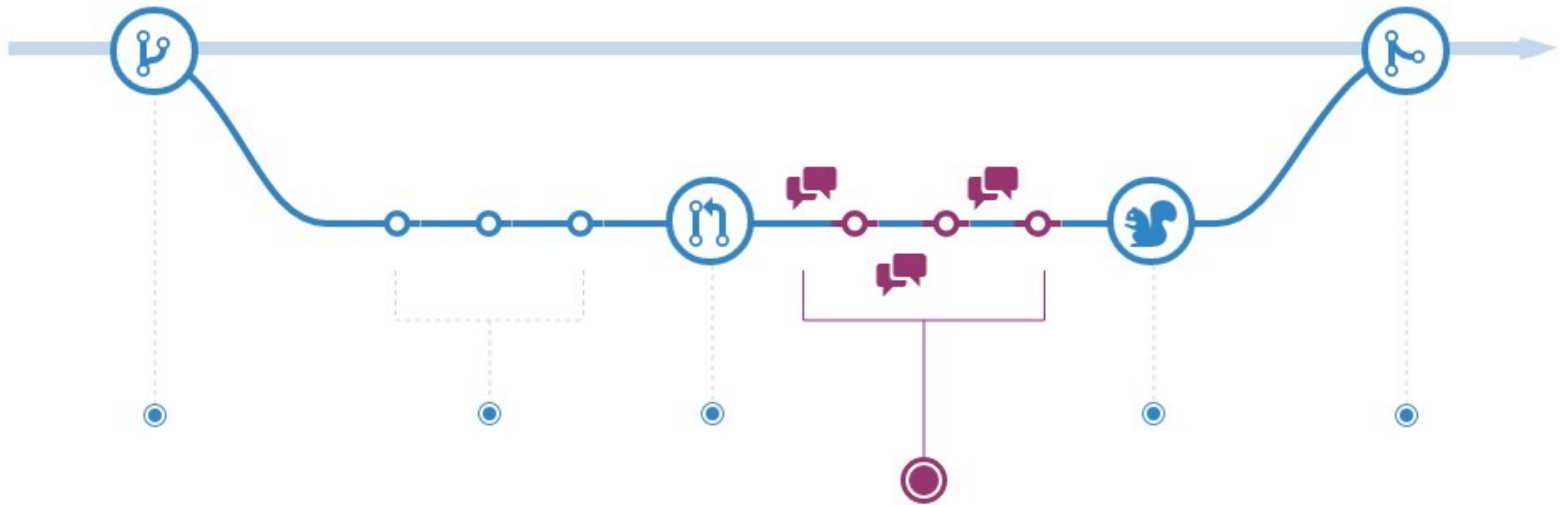
# GITHUB WORKFLOW - PULL REQUEST



- Pull Requests initiate discussion about your commits. Because they're tightly integrated with the underlying Git repository, anyone can see exactly what changes would be merged if they accept your request.



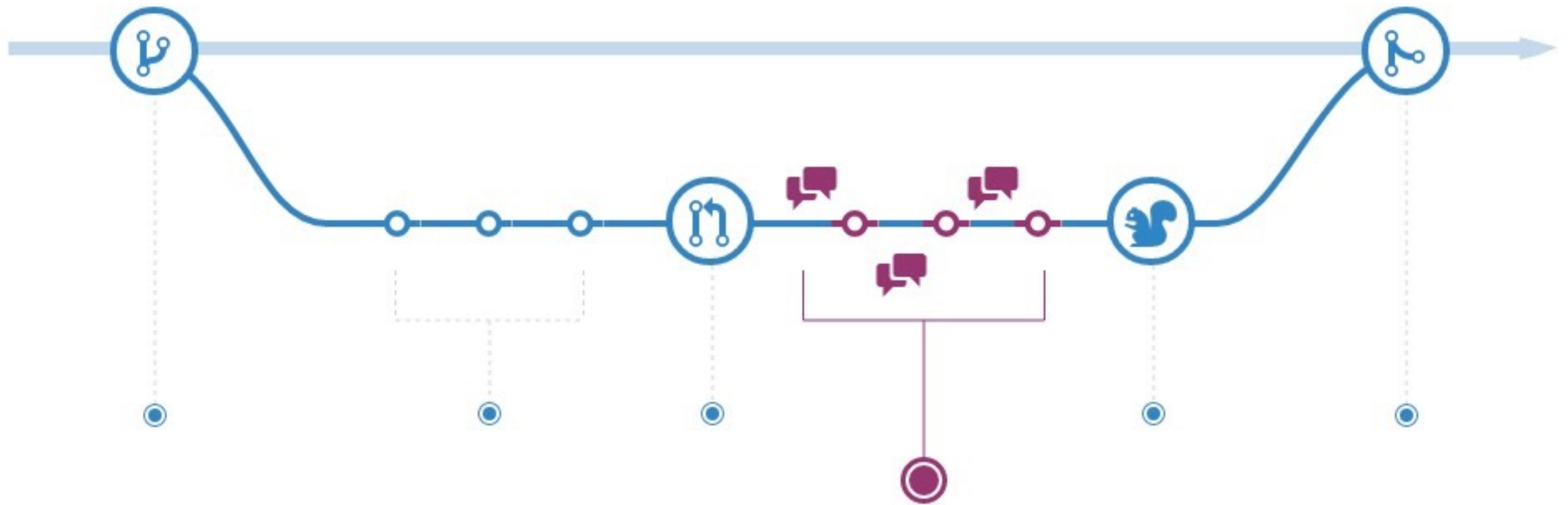
# GITHUB WORKFLOW - CODE REVIEW



- Once a Pull Request has been opened, the person or team reviewing your changes may have questions or comments. Perhaps the coding style doesn't match project guidelines, the change is missing unit tests, or maybe everything looks great and props are in order. Pull Requests are designed to encourage and capture this type of conversation.



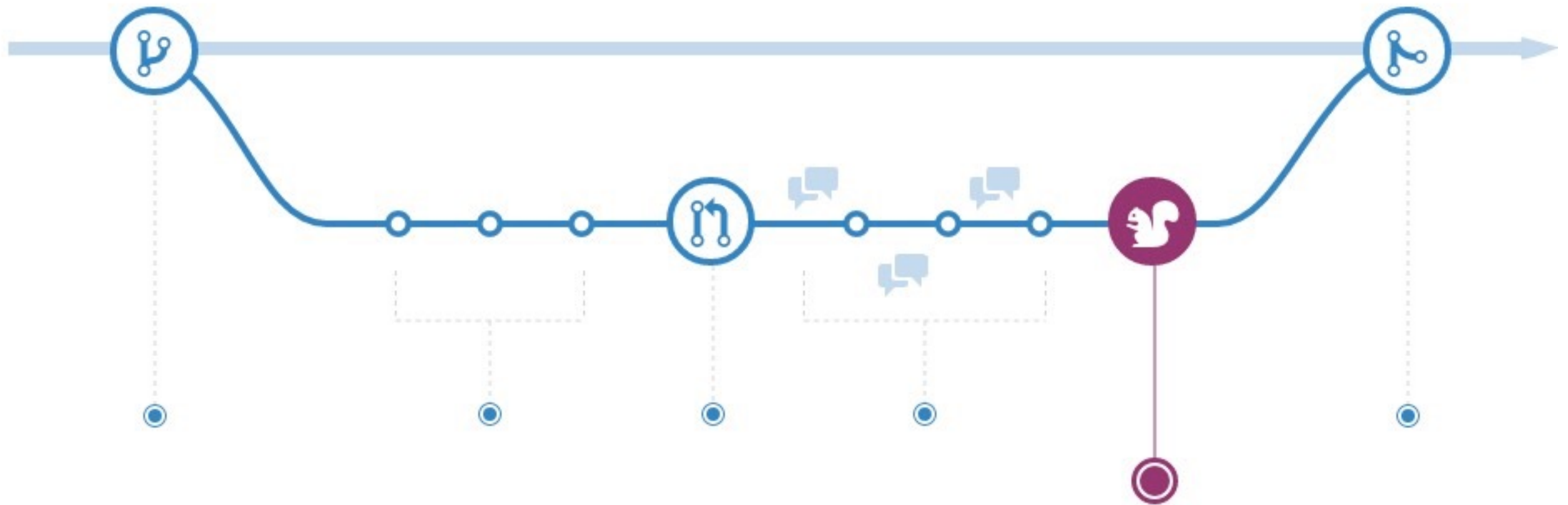
# GITHUB WORKFLOW - CODE REVIEW



- You can also continue to push to your branch in light of discussion and feedback about your commits. If someone comments that you forgot to do something or if there is a bug in the code, you can fix it in your branch and push up the change. GitHub will show your new commits and any additional feedback you may receive in the unified Pull Request view.

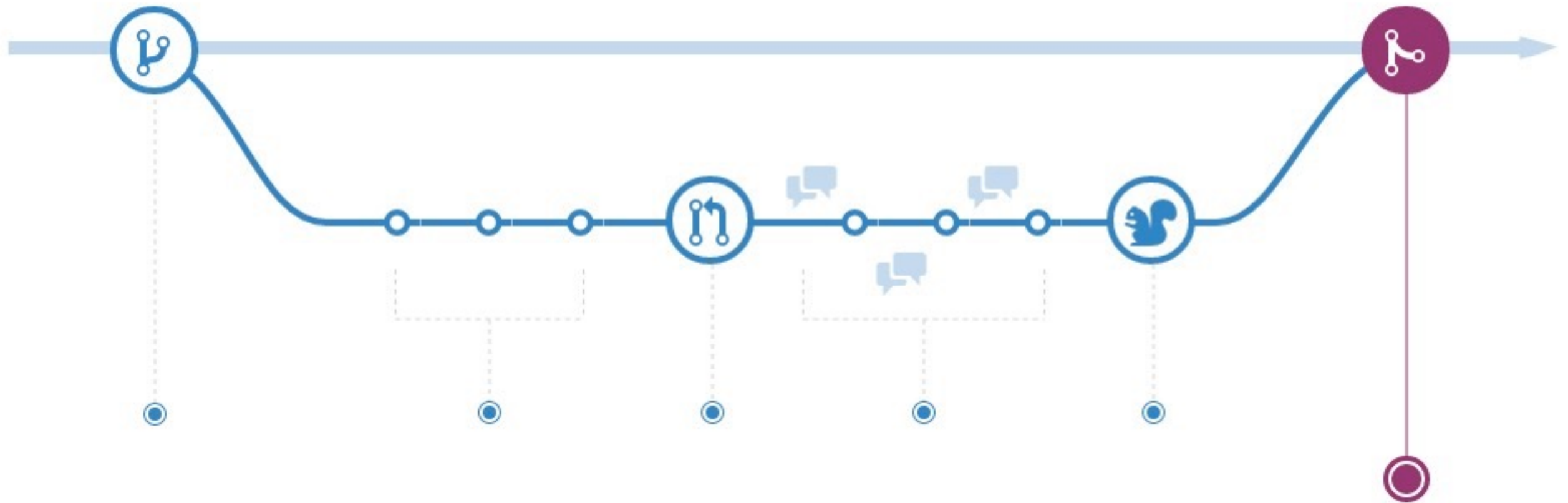
<https://guides.github.com/introduction/flow/index.html>

# GITHUB WORKFLOW - DEPLOY



- Once your pull request has been reviewed and the branch passes your tests, you can deploy your changes to verify them in production.

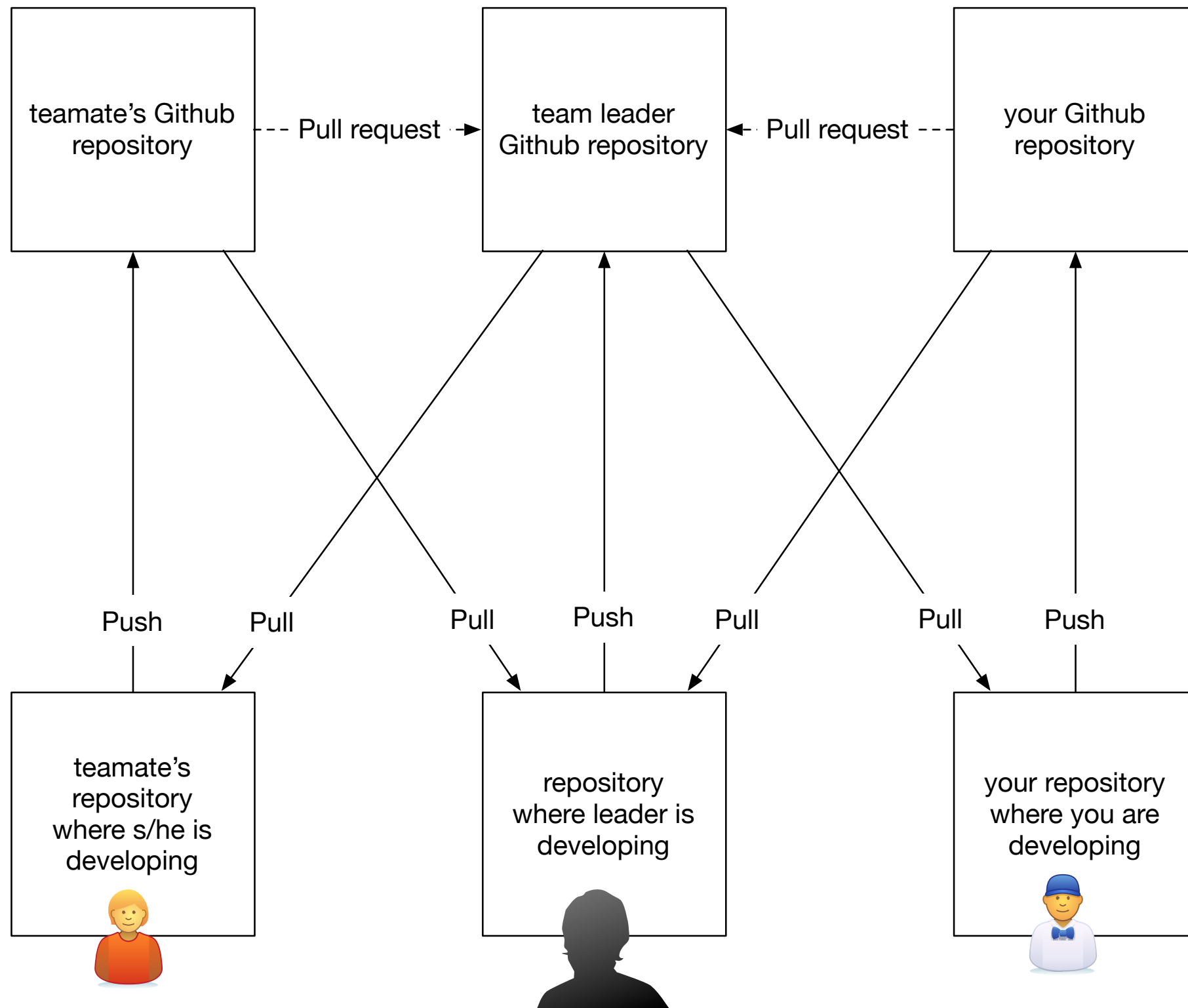
# GITHUB WORKFLOW - MERGE



- Now that your changes have been verified in production, it is time to merge your code into the master branch.
- Once merged, Pull Requests preserve a record of the historical changes to your code. Because they're searchable, they let anyone go back in time to understand why and how a decision was made.



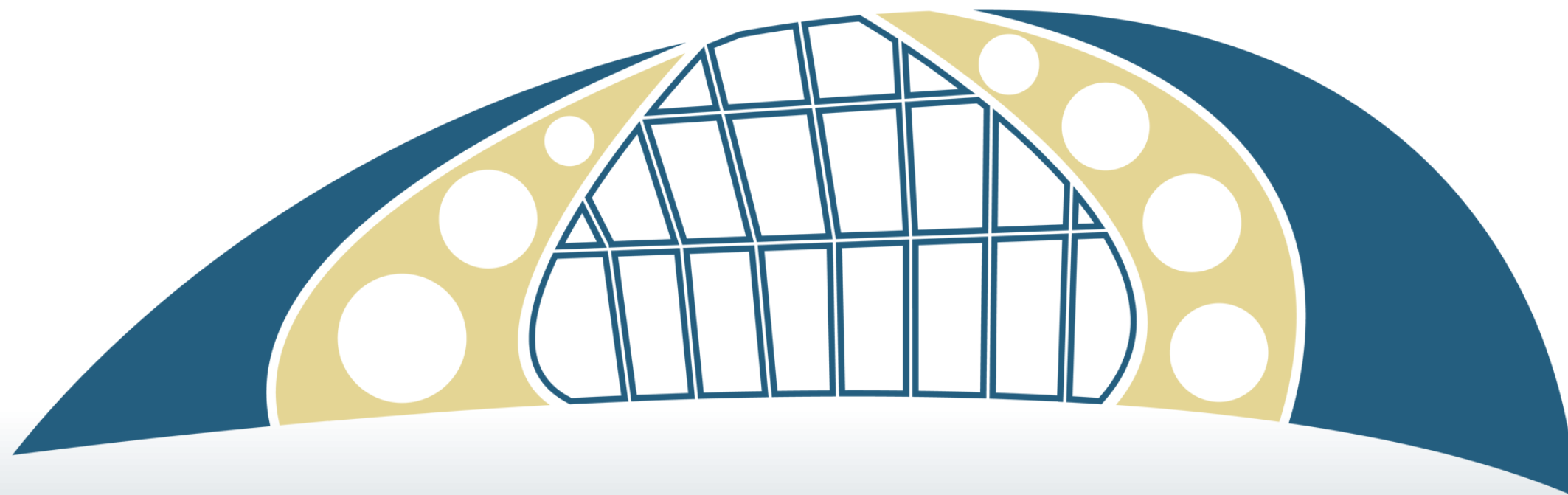
# GITHUB WORKFLOW - X PATTERN



# BASIC PROCESS

- ☐ Update your code from the official repository - pull from master
- ☐ Branch from master in order to start working on your code
- ☐ Write your code
- ☐ Write tests for your code
- ☐ Commit your changes as you are making discrete progress
- ☐ When you are done with your assignment, push to your github repo
- ☐ Initiate a Pull Request from the official repository
- ☐ Official repository merges your branch into the master branch





WESTMONT **INSPIRED**  
— COMPUTING LAB —