Introduction to Version Control

Using Git and GitHub

Module Overview

This module will cover the following:

- What is Git?
- Common Operations
- Local Repos
- GitHub Flow

Getting Started

Version Control

- Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later.

Distributed Version Control

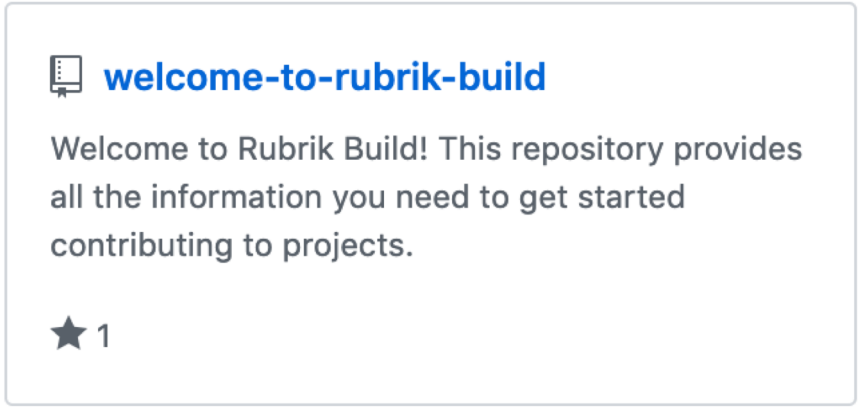
- Rather than have only one single place for the full version history of the software, every developer's working copy of the code is also a repository that can contain the full history of all changes.

What is Git?

- **Git** is the version control technology of choice for basically everybody right now, from developers to designers
  - Git is a mature, actively maintained open source project originally developed in 2005 by Linus Torvalds
- **GitHub** is the social code-hosting platform, but there are many other version control systems (BitBucket, etc.) that use git.

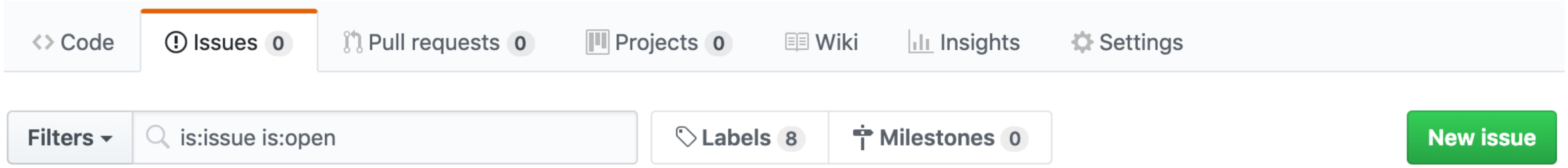
Repositories

- A **repository** is usually used to organize a single project. Repositories can contain folders and files, images, videos, spreadsheets, and data sets – anything your project needs.
- We recommend including:
  - *README*, or a file with information about your project
  - *License*
  - *Code of Conduct*
  - *Contributing*



Common Operations

Issues



- Issues are used to track ideas, enhancements, tasks, or bugs for work on GitHub.
- Rubrik Build uses issues to collect user feedback, report software bugs, and organize tasks we'd like to accomplish within a repository.

Labels for Beginners

Document TF destroy implementation on bootstrap resource #7

Edit | New issue

Open vDingus opened this issue on Jan 10 · 0 comments

vDingus commented on Jan 10 — Member

What would you like to be added: Update documentation to reflect the behavior of the terraform destroy with regards to the bootstrap resource.

Why is this needed: When destroying the bootstrap resource we essentially do nothing, this is to protect users from shooting themselves in the foot and accidentally resetting a cluster. This is the appropriate behavior (the alternative is too risky), but we need document it.

vDingus added exp-beginner labels on Jan 10

drew-russell added the first-timer label on Jan 16

Assignees

No one—assign yourself

Labels

exp-beginner
first-timer
kind-docs

Projects

None yet

Milestone

No milestone

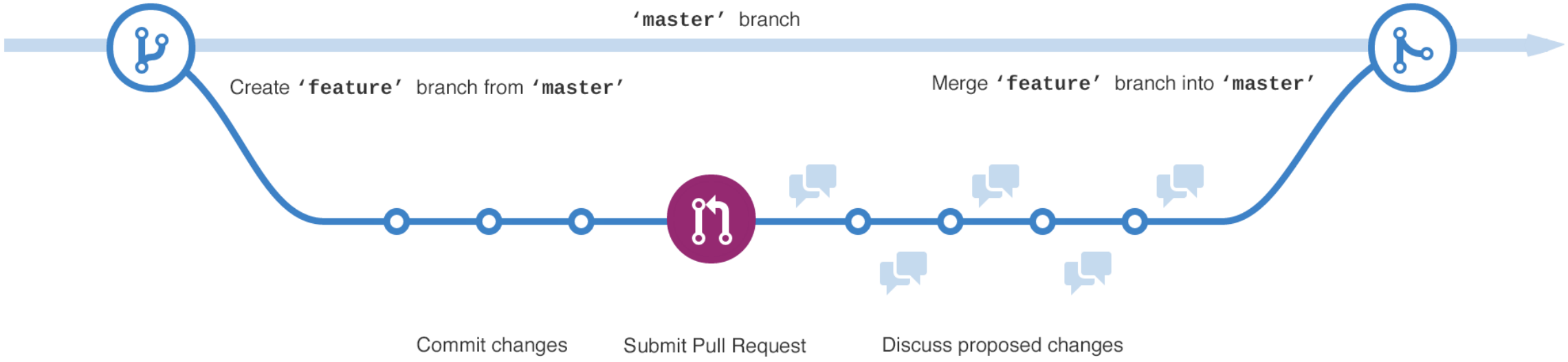
Branching

- **Branching** is the way to work on different versions of a repository at one time.
- By default your repository has one branch named `master` which is considered to be the definitive branch. We use branches to experiment and make edits before committing them to master.
- When you create a branch off the master, you're making a copy, or snapshot, of master as it was at that point in time. If someone else made changes to the master branch while you were working on your branch, you could pull in those updates.

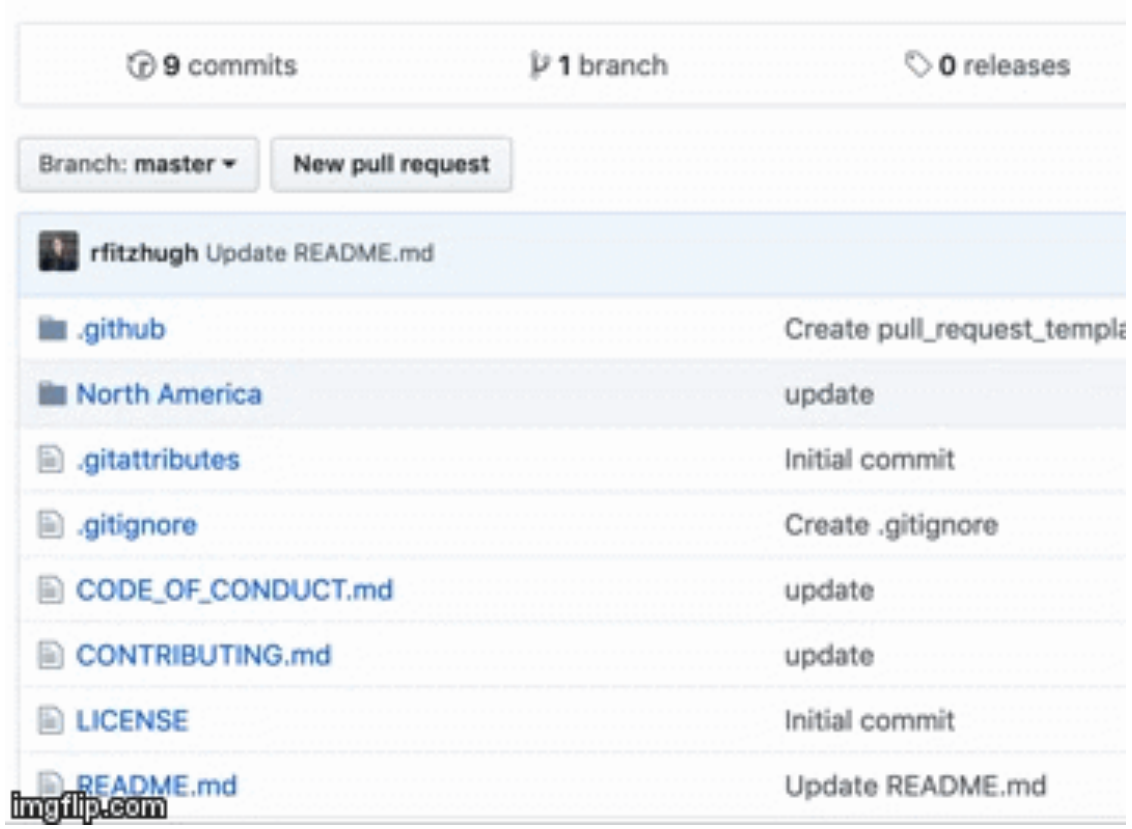
Branching (cont.)

This diagram shows:

- The `master` branch
- A new branch called `feature` (because we're doing 'feature work' on this branch)



Creating a Branch

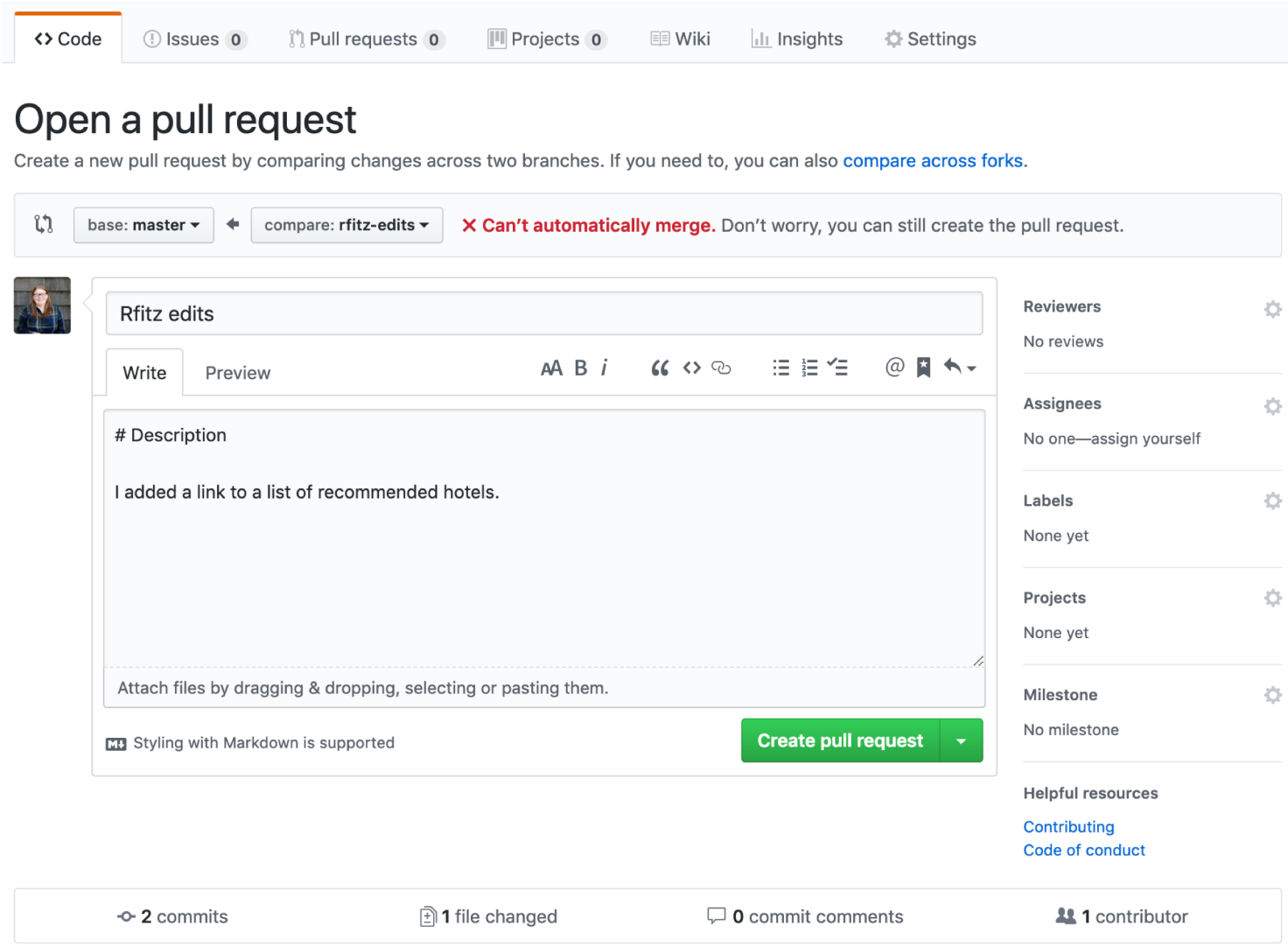


Forks

- A *fork* is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project.
- Most commonly, forks are used to either propose changes to someone else's project or to use someone else's project as a starting point for your own idea. You can:
  - Fork the repository
  - Make the change
  - Submit a *pull request* to the project owner

Pull Requests

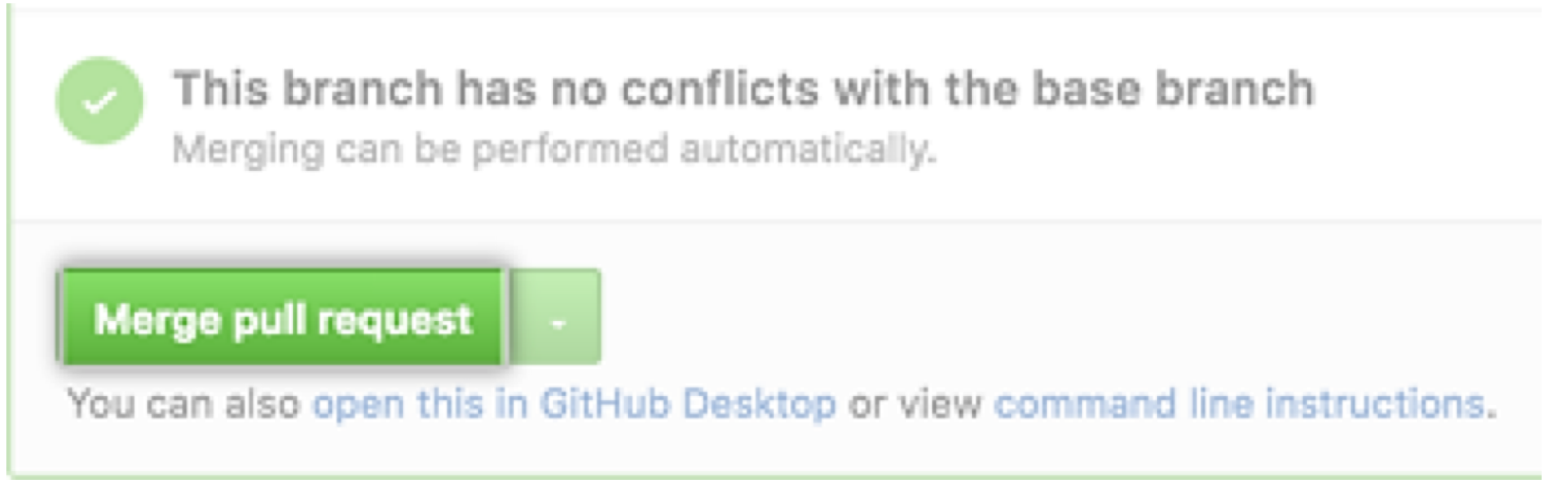
When you open a *pull request*, you're proposing your changes and requesting that someone review and pull in your contribution and merge them into their branch.



Merging

Merging commits all the branch diffs into `master` once the work is complete.

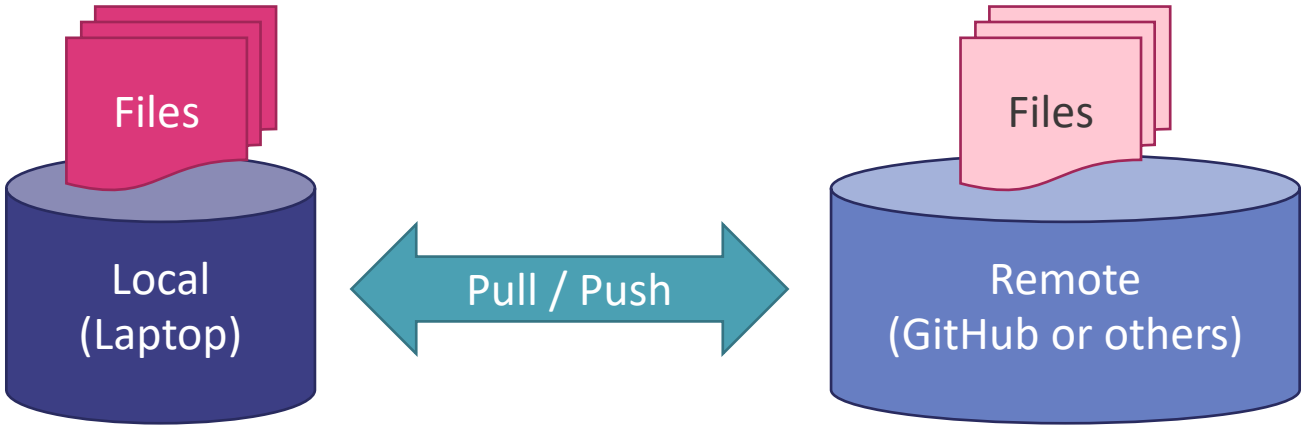
Typically there are reviews and tests before approving and merging the pull request.



Local Repositories

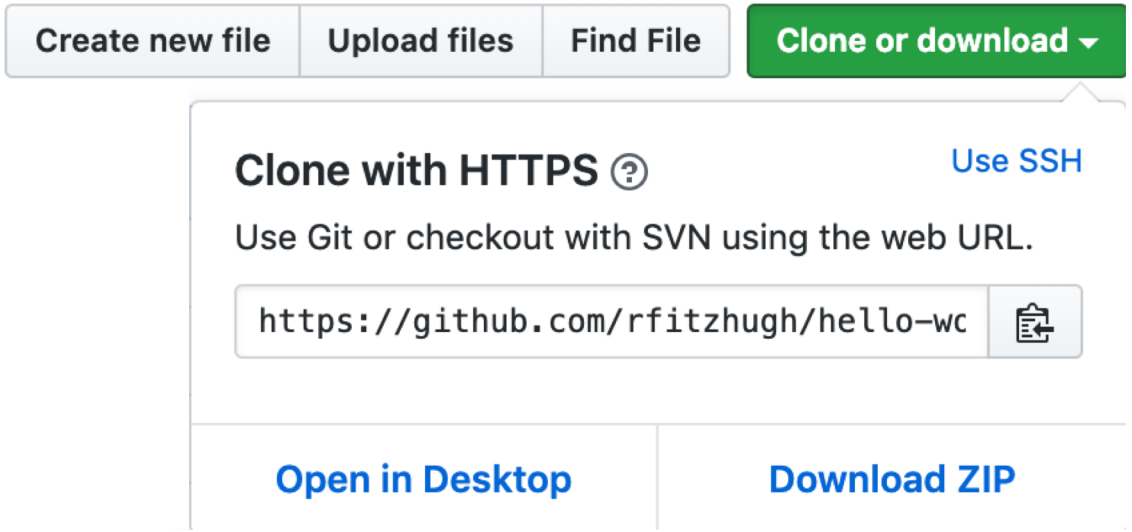
What is a Local Repository?

- A *local* repository (“repo”) is an offline copy of the repository and all of its files are stored on your computer.
- When you create a repository on GitHub, it exists as a *remote* repository.



Clone

- You can clone your repository to create a *local* copy on your computer and sync between the two locations.



Fetch

- Fetch retrieves new work done by other people. Fetching from a repository grabs all the new remote-tracking branches and tags *without* merging those changes into your own branches.
- Merges must be done manually as a next step.

Pull

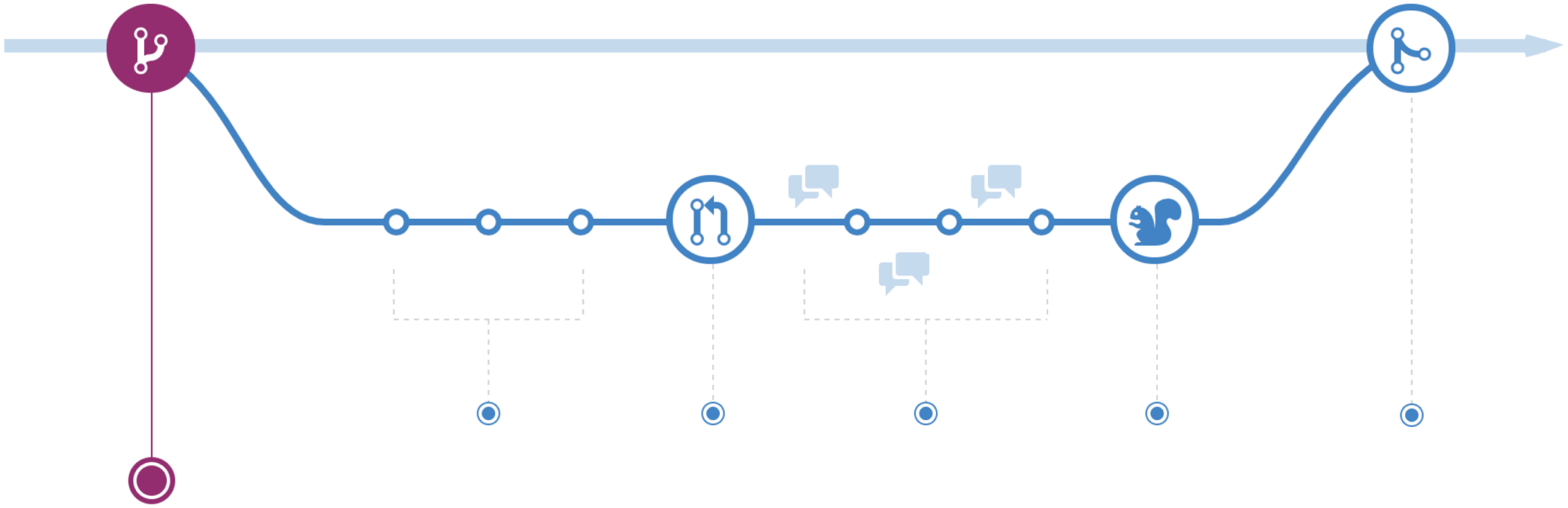
- Pull is a convenient shortcut for completing both fetch and merge operations in the same command.

Push

- Use push to push commits made on your local branch to a remote repository

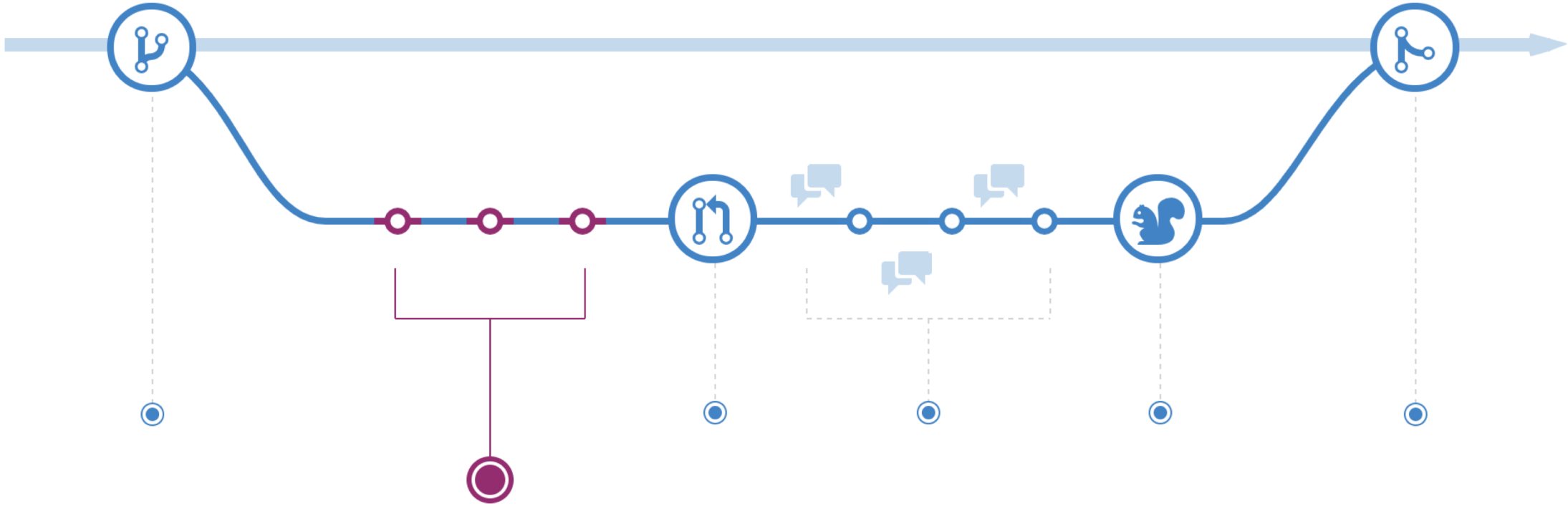
GitHub Flow

GitHub Flow

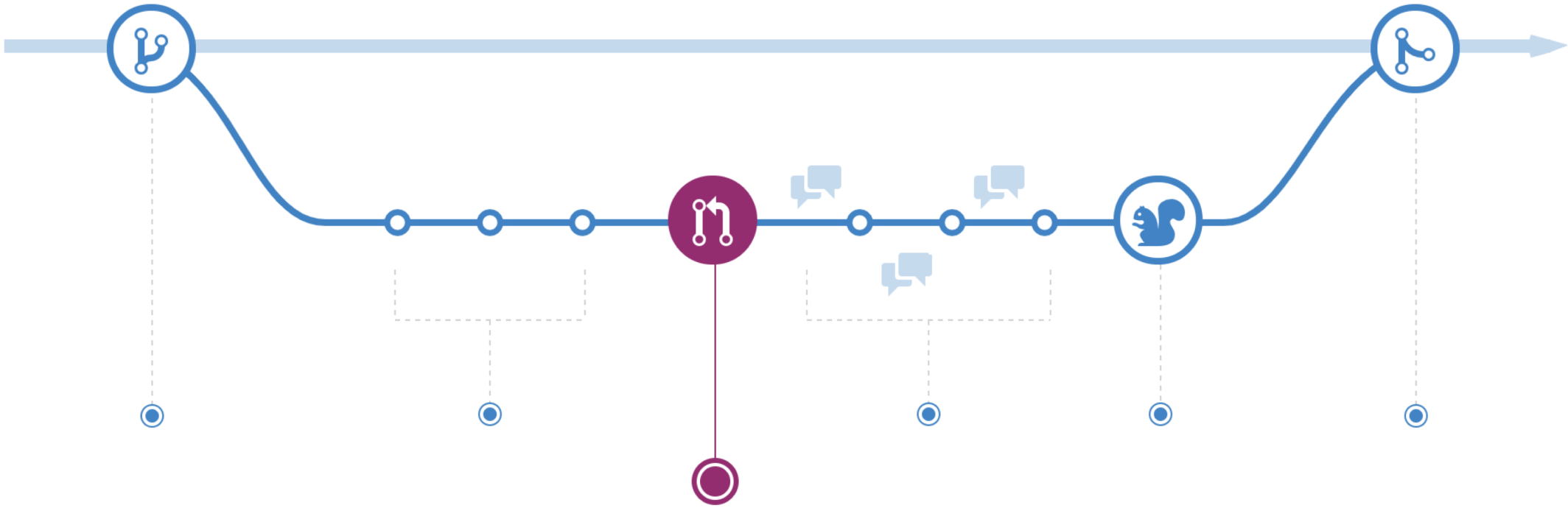


Create a branch

GitHub Flow

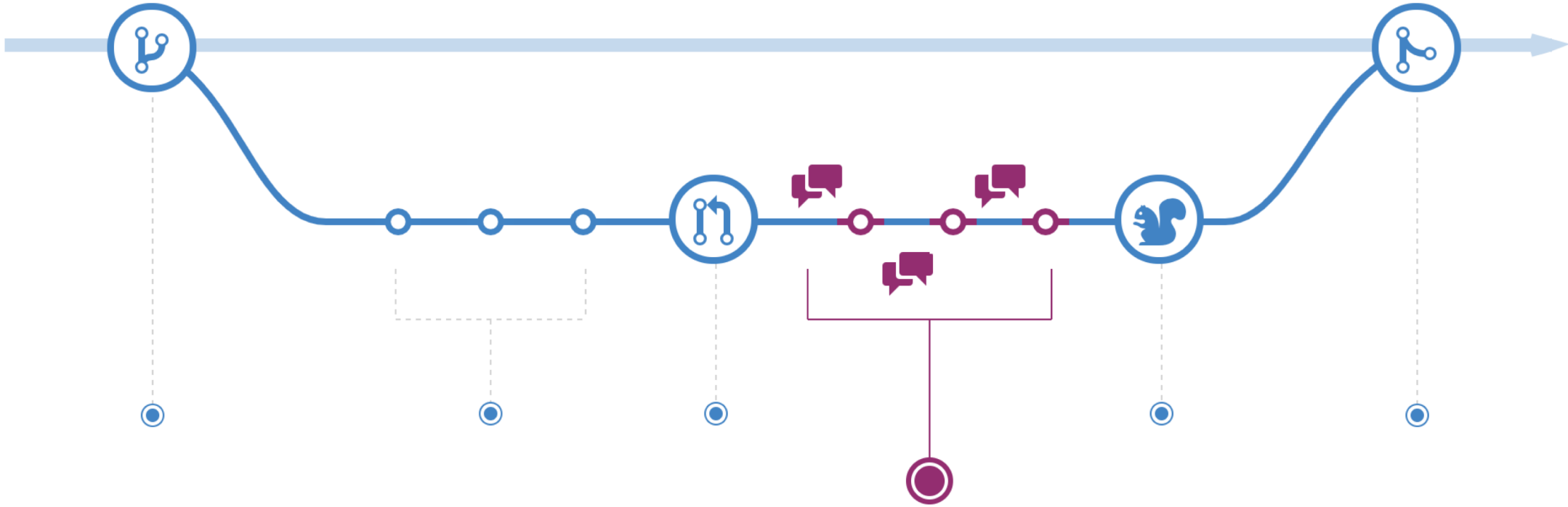


GitHub Flow

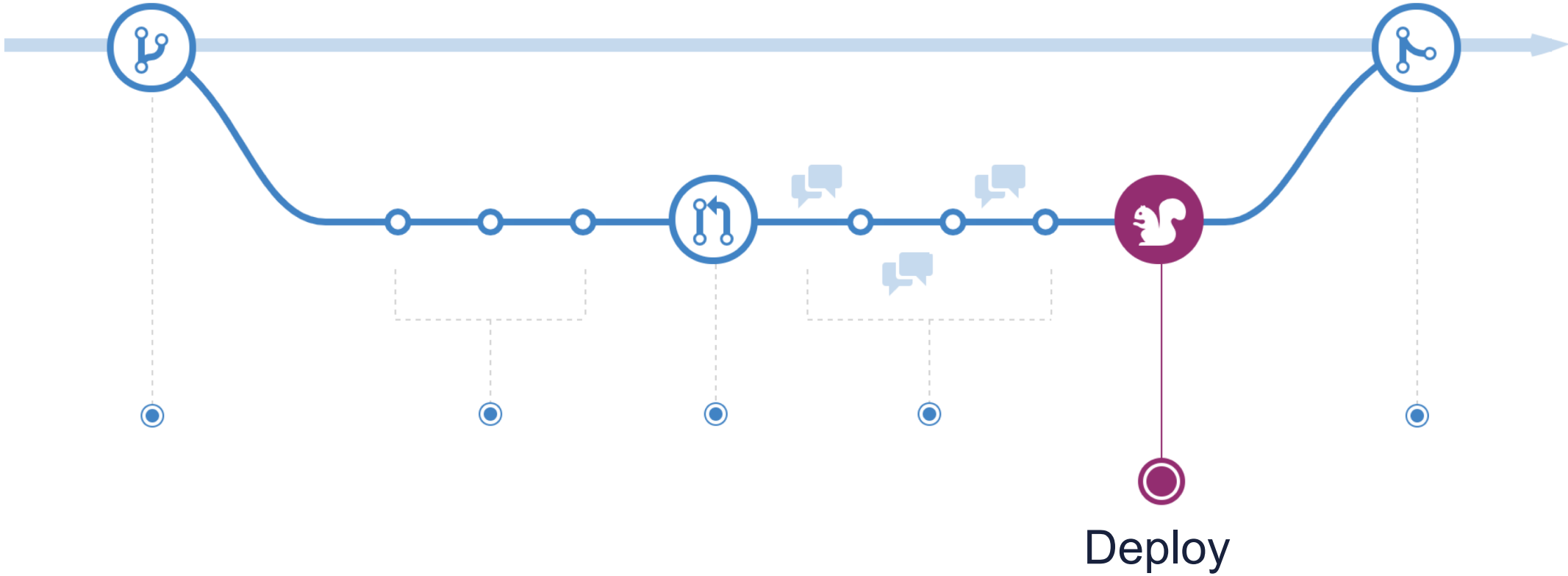


Create a pull request

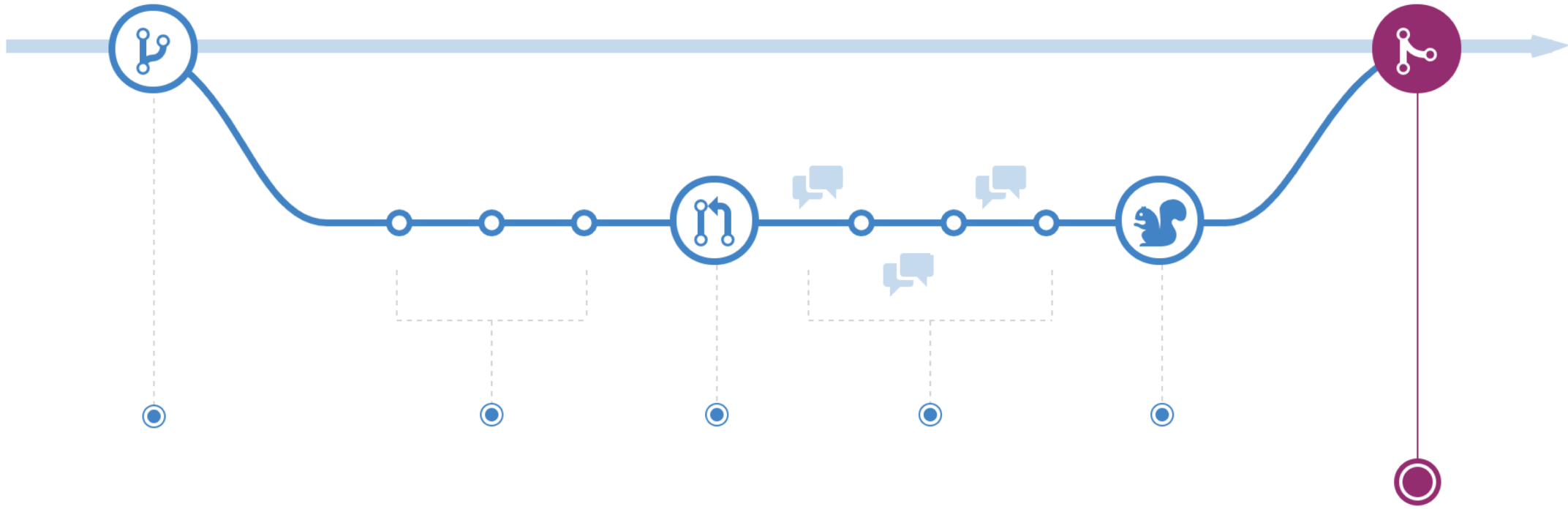
GitHub Flow



GitHub Flow



GitHub Flow



build

Building the Future of Data Management