

# GitHub

## workshop

Bastien Brugger

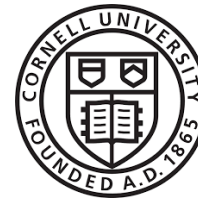
Ishan Mishra

Ryan Peterson



Day 3 – 2019/06/19

Cornell  
University



...ERROR

Merali (2010) *Nature* 467, 775-777

...why scientific programming does not  
compute

“Problems created by bad documentation are further amplified when successful codes are modified by others to fit new purposes. *The result is the bane of many graduate students or postdocs’ life: the “monster code”.* Sometimes decades old, these codes are notoriously messy and become progressively more nightmarish to handle”

...ERROR

Merali (2010) *Nature* 467, 775-777

...why scientific programming does not  
compute

> **38%** of scientists spend at  
least one fifth of their time  
developing software.

> Only **47%** of scientists  
have a good understanding of  
software testing.

> Only **34%** of scientists  
think that formal training  
in developing software is  
important.



...ERROR

Merali (2010) *Nature* 467, 775-777

...why scientific programming does not  
compute

**Five tips** to make scientific code more robust:

- *Track your material*
- *Write testable software*
- *Test the software*
- *Encourage sharing of software*
- *Use a version-control system (VCS)*

> **38%** of scientists spend at least one fifth of their time developing software.

> Only **47%** of scientists have a good understanding of software testing.

> Only **34%** of scientists think that formal training in developing software is important.

...ERROR

Merali (2010) *Nature* 467, 775-777

...why scientific programming does not compute

**Five tips** to make scientific code more robust:

- *Track your material*
- *Write testable software*
- *Test the software*
- *Encourage sharing of software*
- *Use a version-control system (VCS)*

GitHub

> **38%** of scientists spend at least one fifth of their time developing software.

> Only **47%** of scientists have a good understanding of software testing.

> Only **34%** of scientists think that formal training in developing software is important.

# Summary

1. What is **VCS** and why use it?
2. Creating and understanding your first **GitHub** repository
3. Managing an actual **test** case
4. Best practices for code **accessibility** and **documentation**
5. Creating your **website** with GitHub Pages
6. Other **resources**

# Summary

All material can be found at

[github.com/BastienBrugger/ERESV-github](https://github.com/BastienBrugger/ERESV-github)

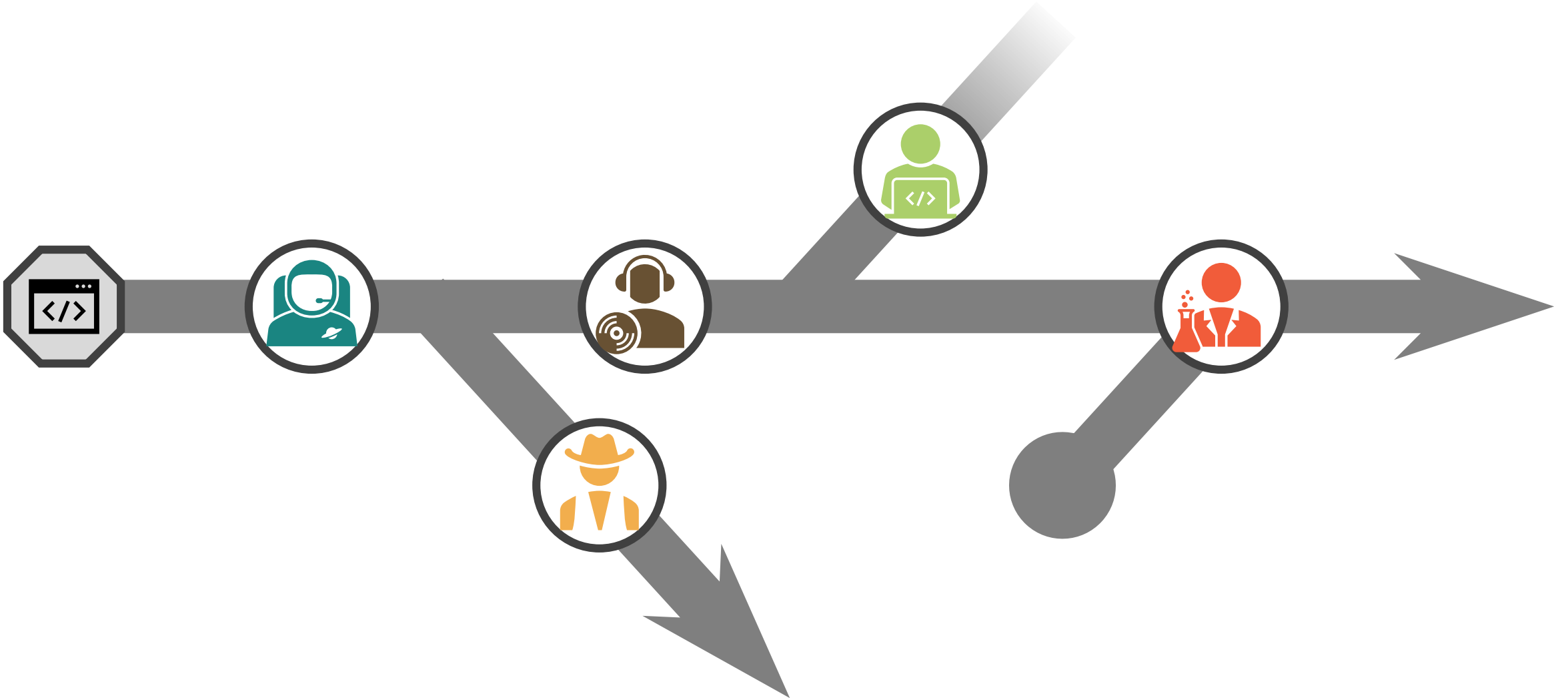
Disclaimer:

**Git** *open-source VCS, the **tool** that  
manages your code history*

**GitHub** *hosting **service** for *Git* repositories*



# How VCS works

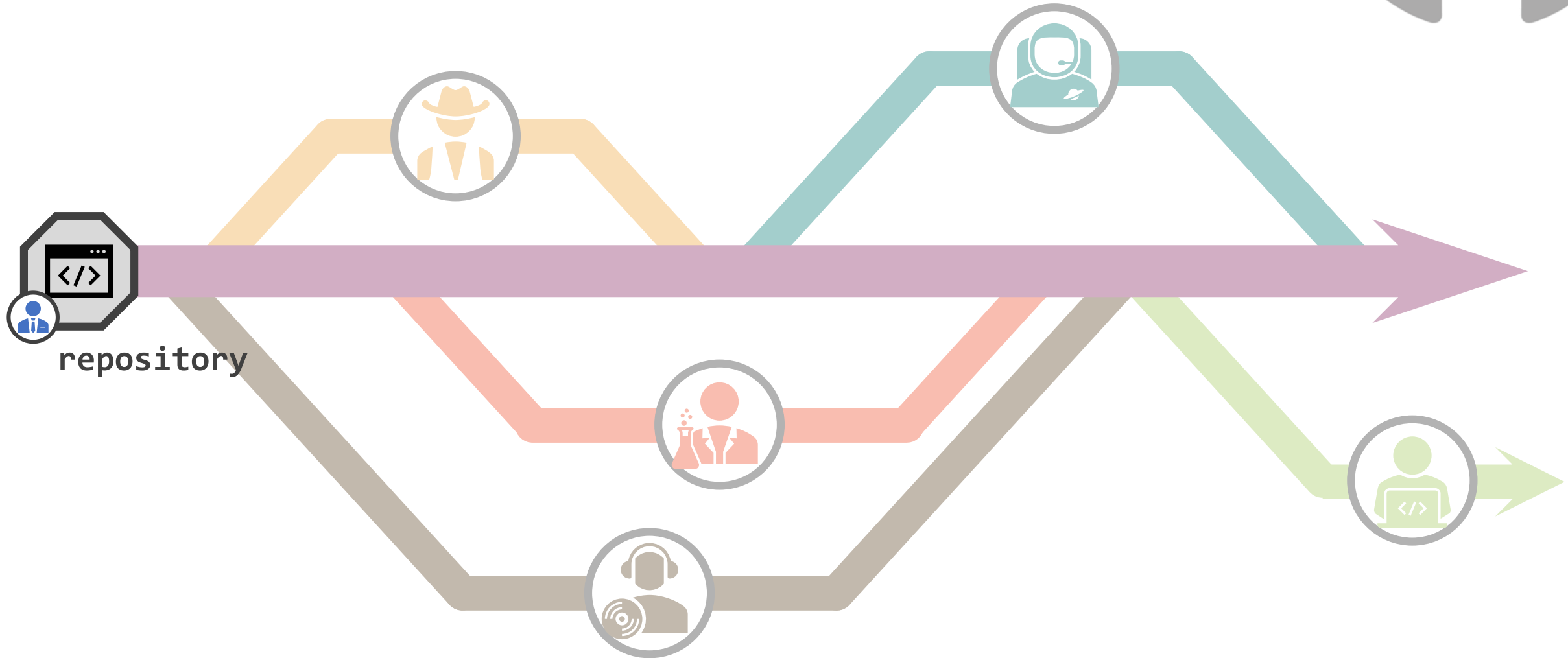
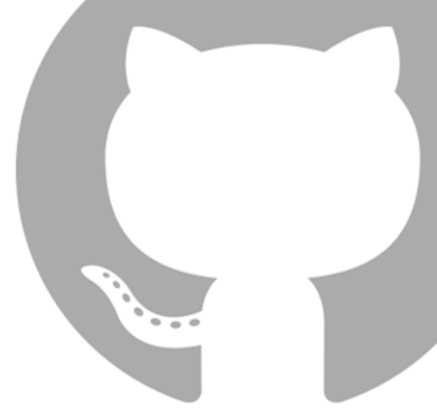




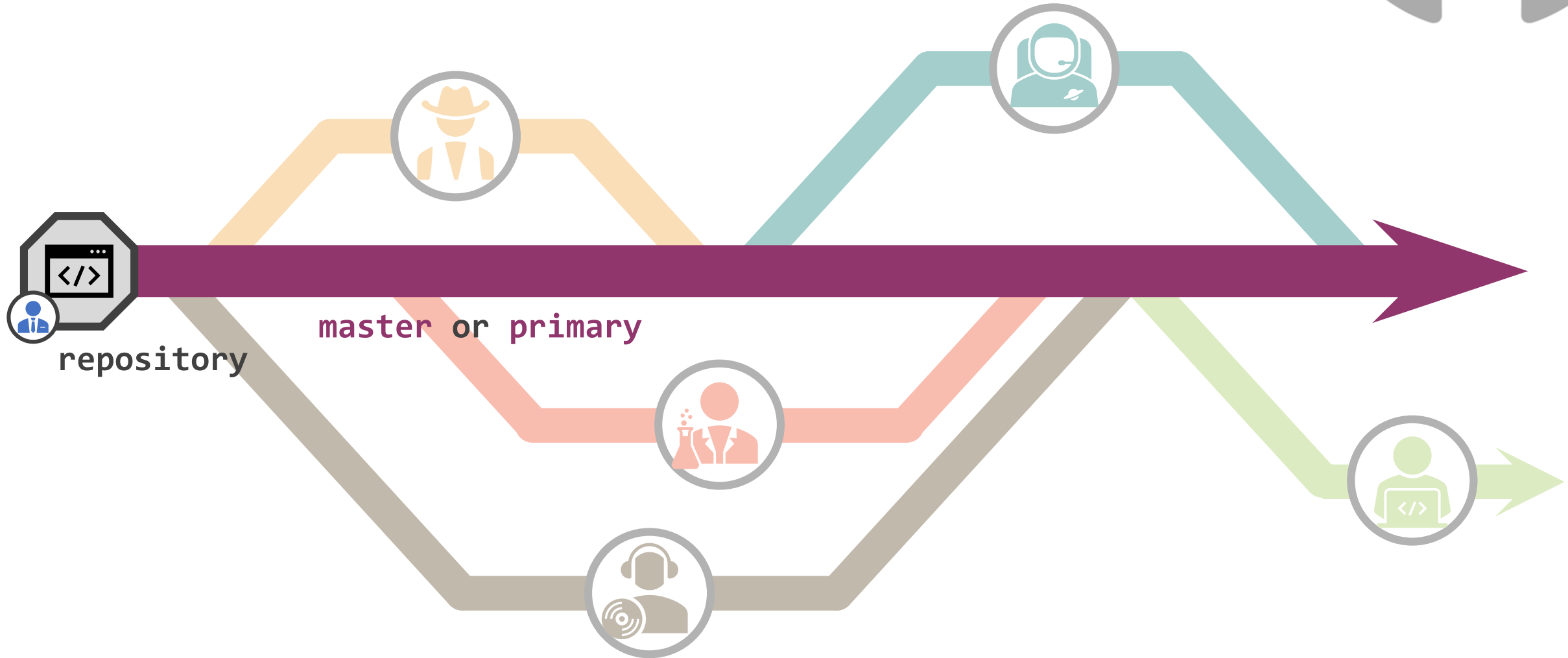
# How VCS works



# How GitHub works



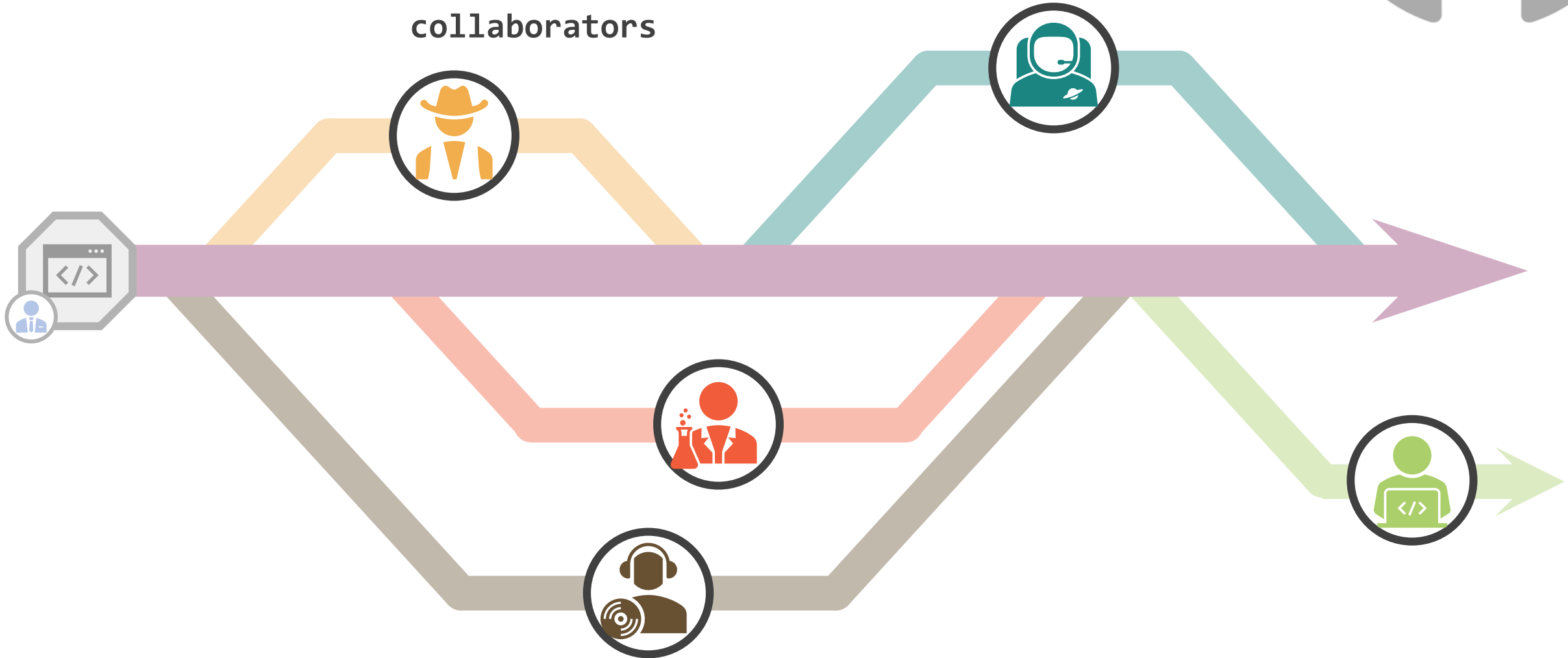
# How GitHub works



# How GitHub works

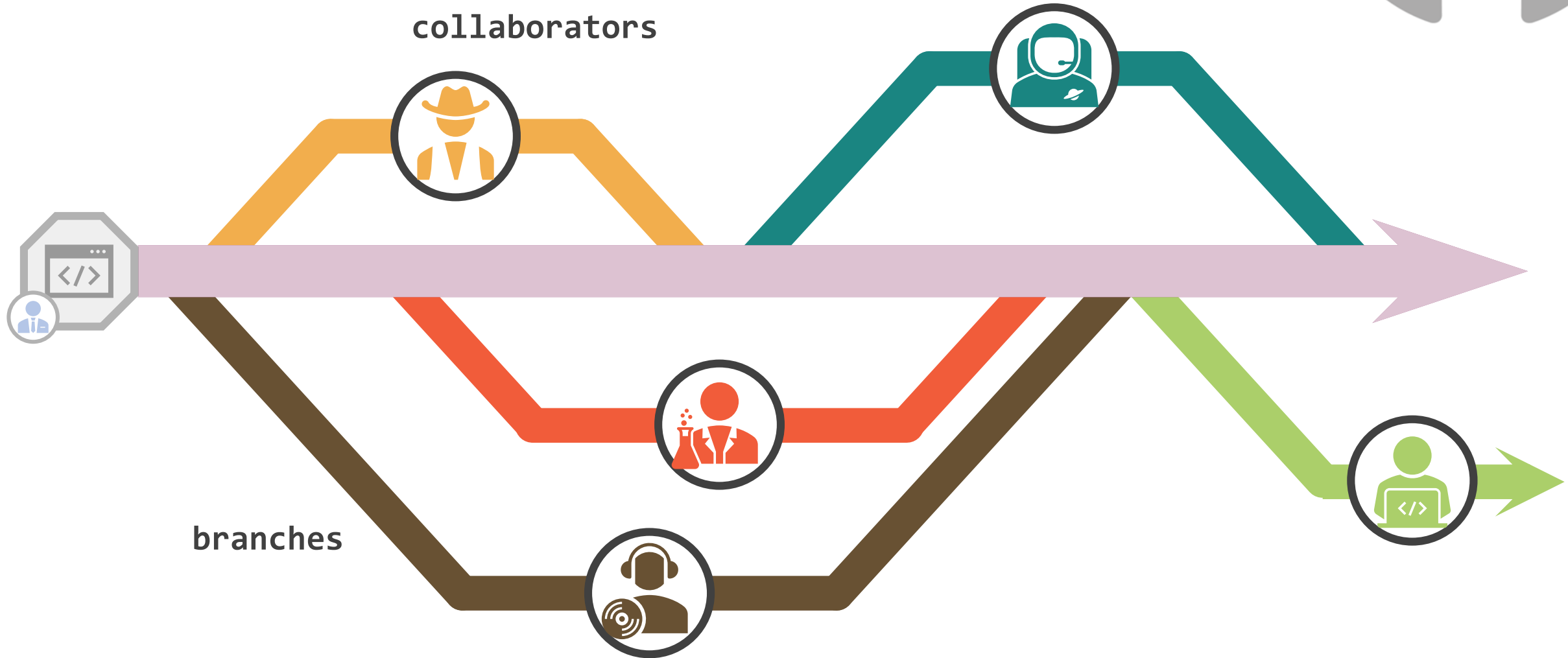


collaborators

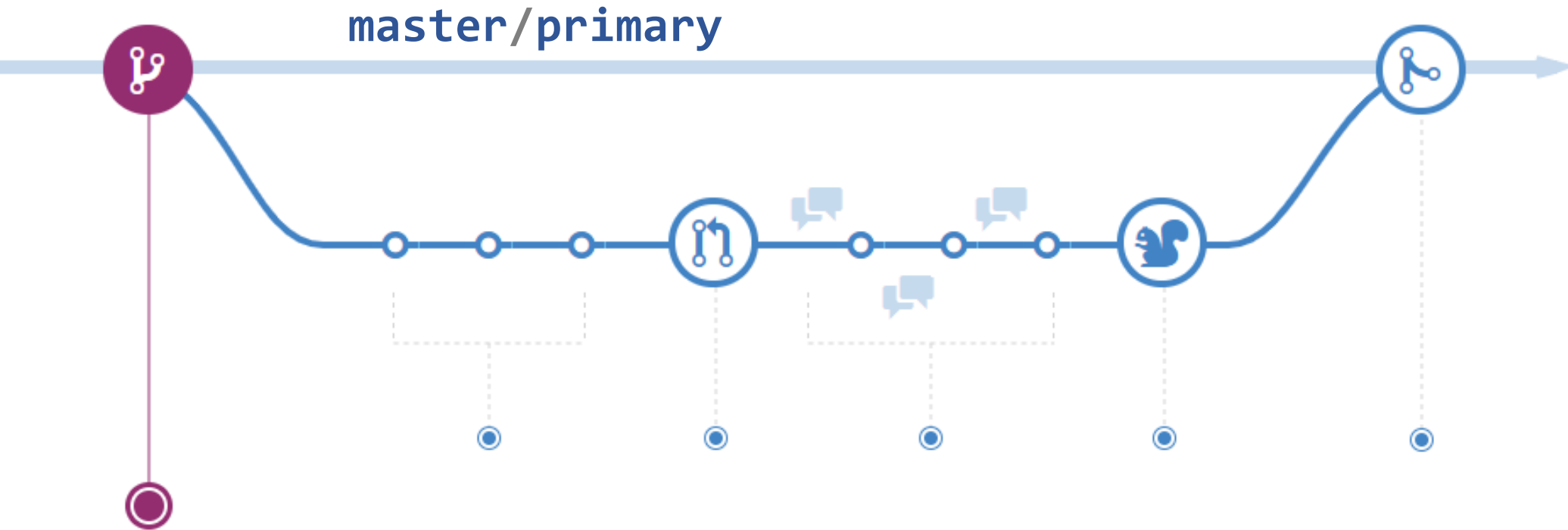




# How GitHub works



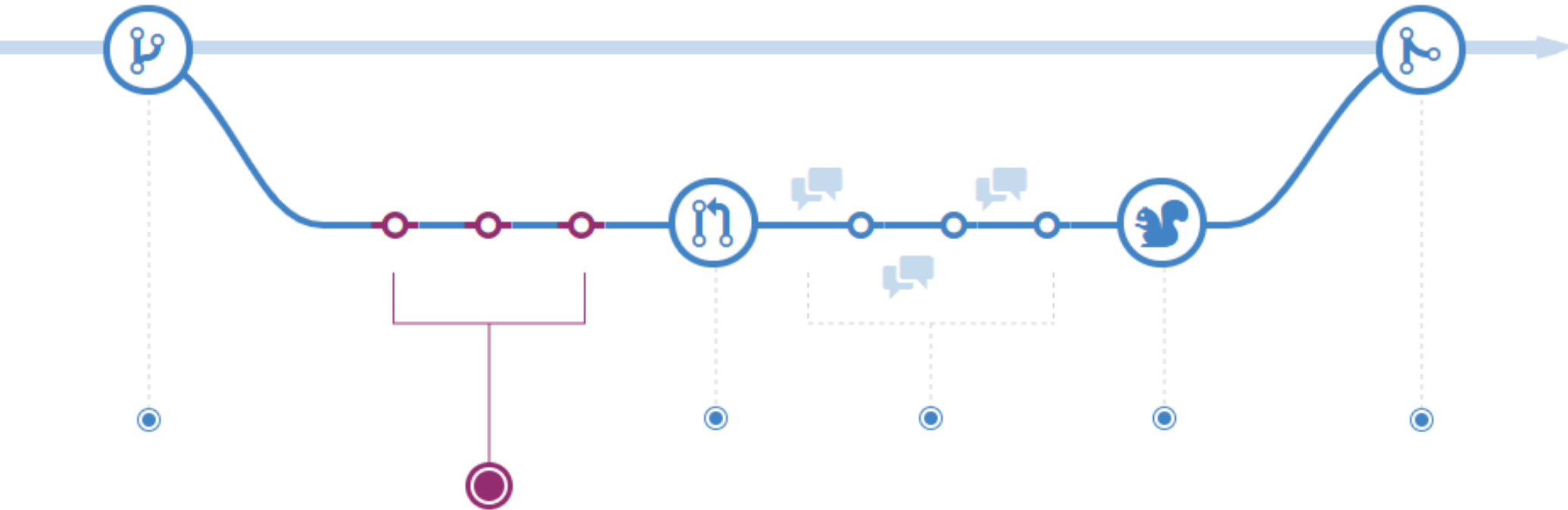
# GitHub flow



## Create a **branch**

*Copy files in an environment where you can **experiment** new ideas, without affecting the **master***

# GitHub flow

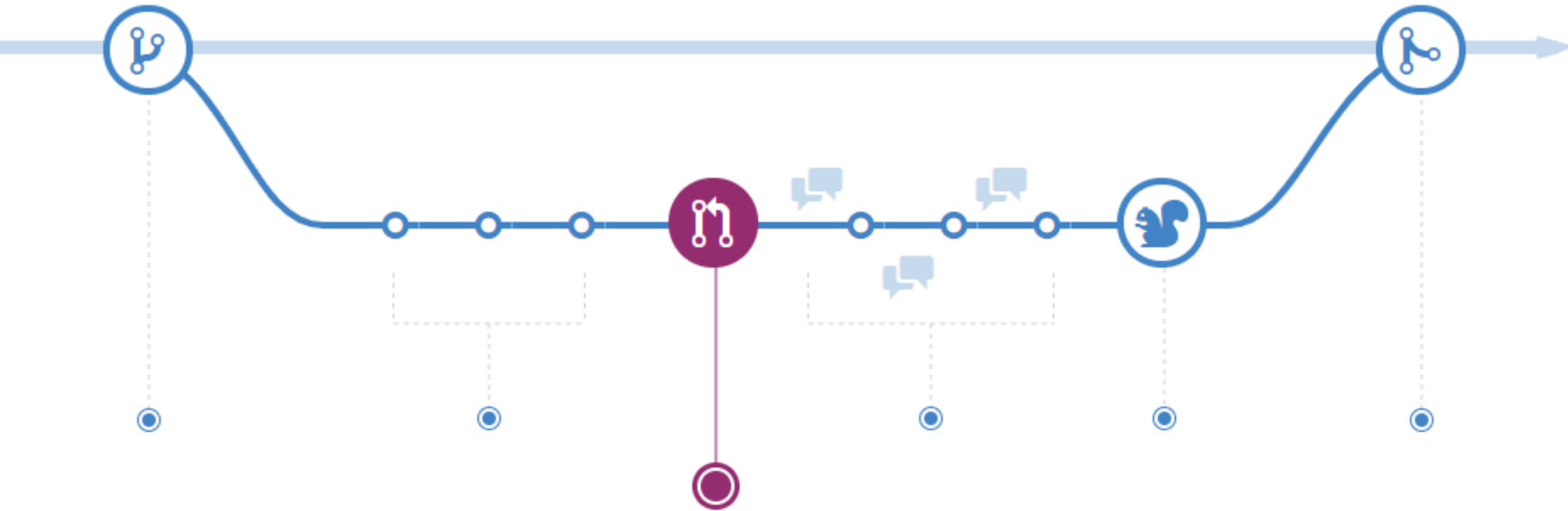
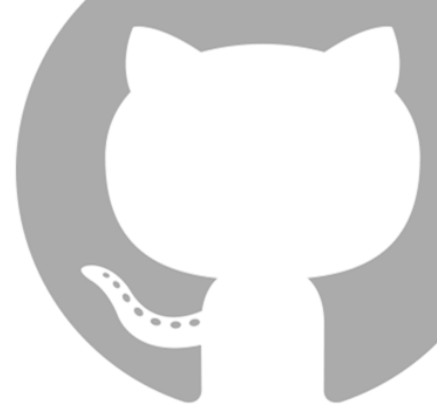


## Add **commits**

*Adding, editing or deleting files*

*Creates a **transparent history** of your work: each **commit** has an associated message*

# GitHub flow



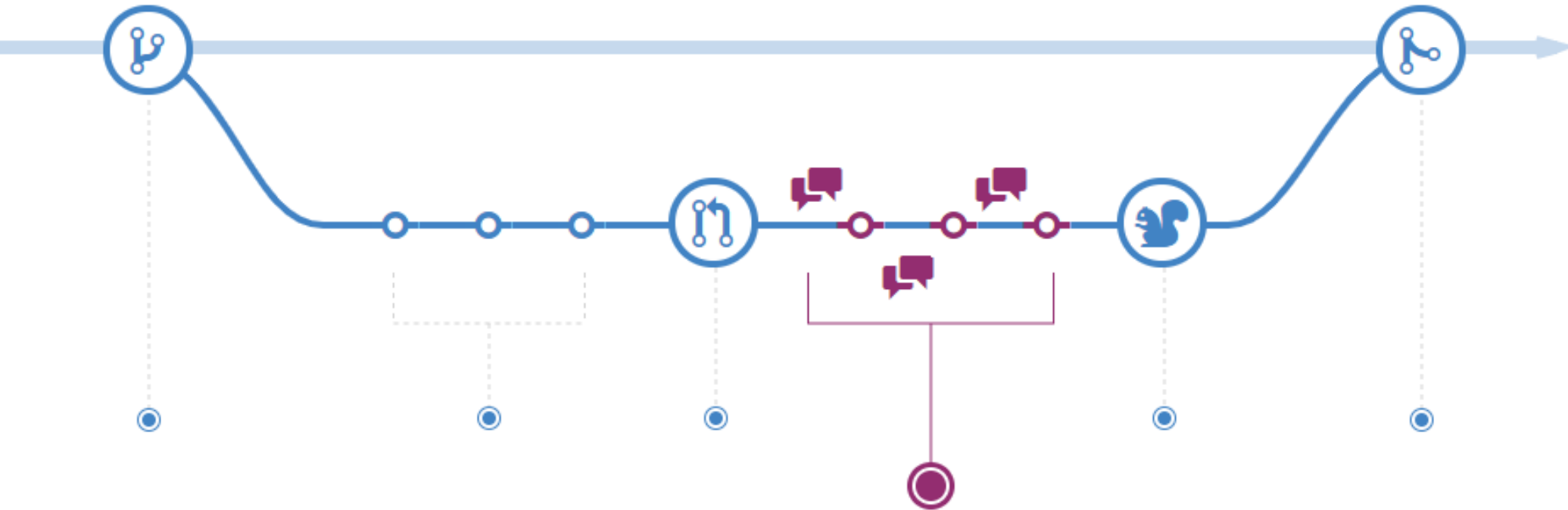
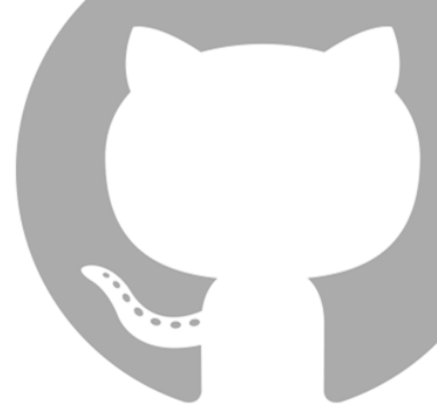
Open a **pull request**

*Show your changes to other **collaborators** and initiate discussion*

*When you're ready to **add your work**, when you want to **share ideas**,  
when you're stuck and **need help**...*



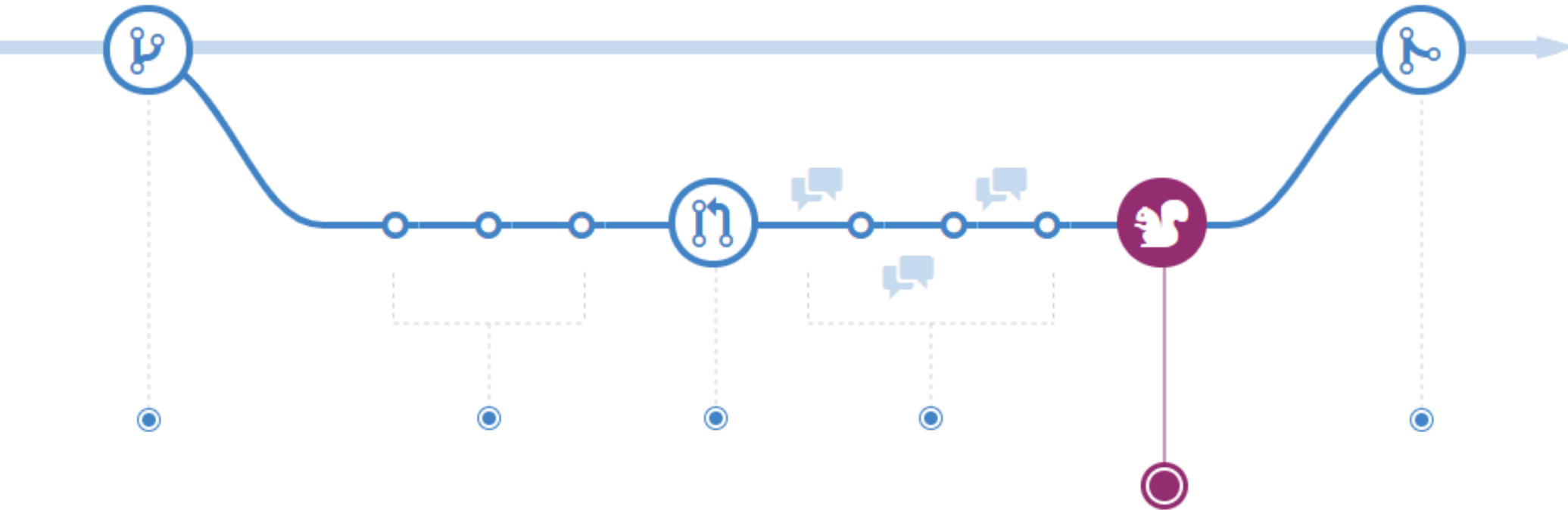
# GitHub flow



Discuss and review code

*Check if everything is fine*

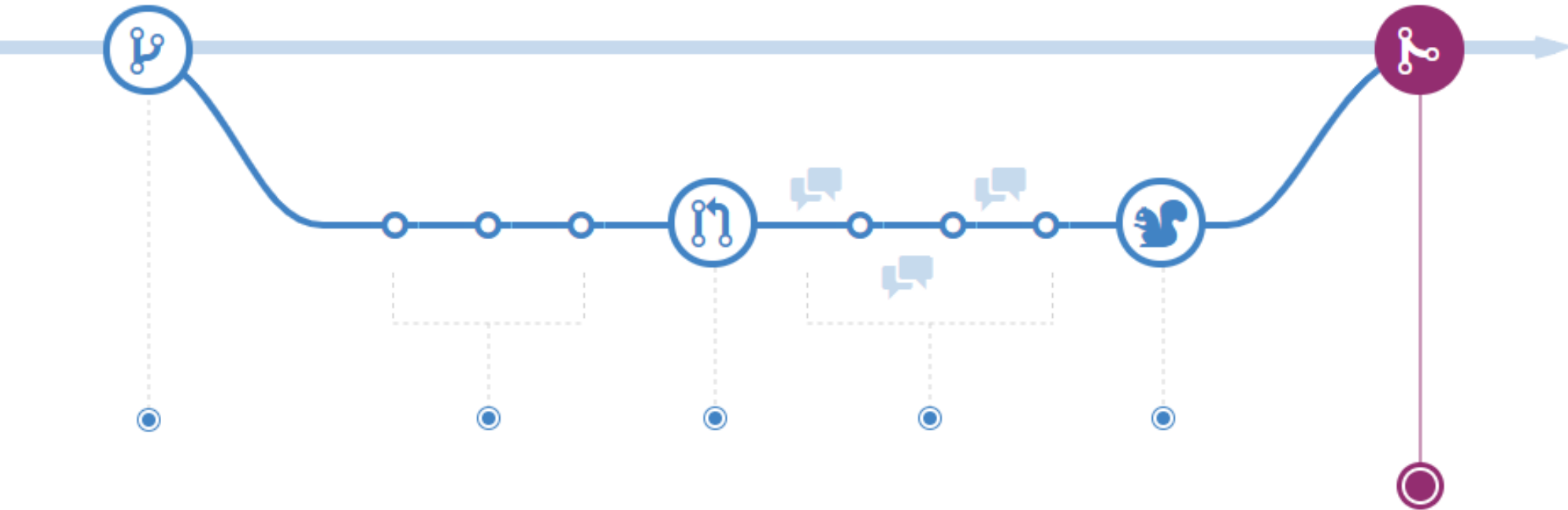
# GitHub flow



**Deploy**

*For final testing*

# GitHub flow



## Merge

*Merge your code into the  
**master** branch*

*Record is preserved*

# How GitHub works

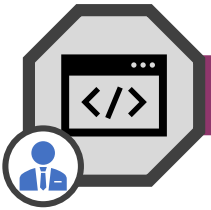


**clone** a repository

*Copy the files on your computer*

*No GitHub account required*

*Can be used offline*





# How GitHub works

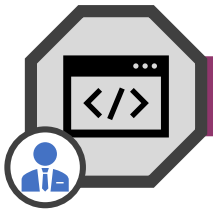


## clone a repository

*Copy the files on your computer*

*No GitHub account required*

*Can be used offline*



## fork a repository

*Copy the repo to your GitHub account*

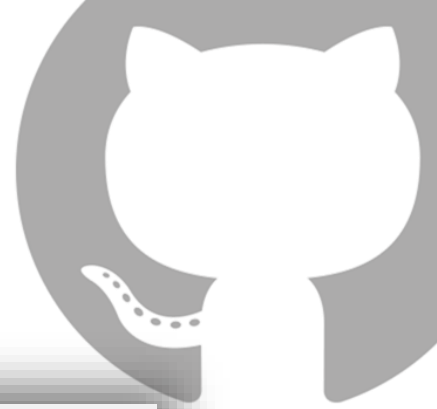
*Still attached to the original: submit pull requests to it, import updates from it*



# Setting up

A new account

Go to  
**github.com/join**



## Join GitHub

The best way to design, build, and ship software.



Step 1:  
Set up your account



Step 2:  
Choose your subscription



Step 3:  
Tailor your experience

### Create your personal account

**Username \***

This will be your username. You can add the name of your organization later.

**Email address \***

We'll occasionally send updates about your account to this inbox. We'll never share your email address with anyone.

**Password \***

Make sure it's at least 15 characters OR at least 8 characters including a number and a lowercase letter. [Learn more.](#)

By clicking "Create an account" below, you agree to our [Terms of Service](#) and [Privacy Statement](#). We'll occasionally send you account-related emails.

Create an account

### You'll love GitHub

Unlimited public repositories

Unlimited private repositories

✓ Limitless collaboration

✓ Frictionless development

✓ Open source community

# Setting up

## Your dashboard

Main hub  
for your  
activities

Public  
*profile info,  
repositories,  
contributions...*



The Octocat  
octocat

Follow

★ PRO

GitHub

San Francisco

octocat@github.com

<http://www.github.com/blog>

Block or report user

Overview

Repositories 8

Projects 0

Stars 3

Followers 2.7k

Following 9

### Popular repositories

#### Spoon-Knife

This repo is for demonstration purposes only.

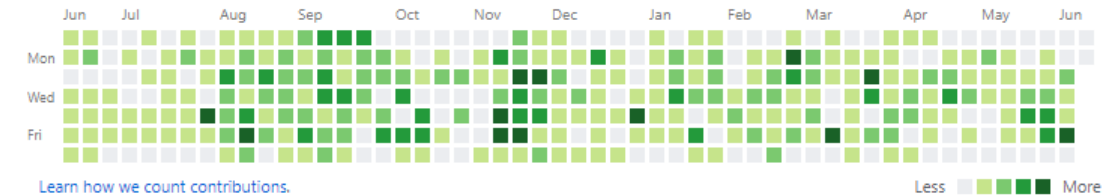
HTML ★ 10.1k 104k

#### Hello-World

My first repository on GitHub!

★ 1.5k 1.3k

### 1,587 contributions in the last year



@octobox

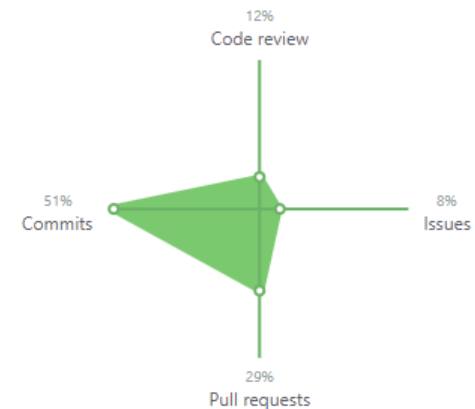
@ipfs

@ipfs-shipyard

More

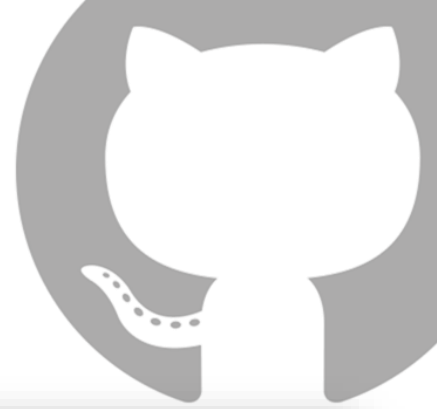
### Activity overview

Contributed to [octobox/octobox](#),  
[24pullrequests/24pullrequests](#),  
[ipfs/package-managers](#) and 5 other repositories

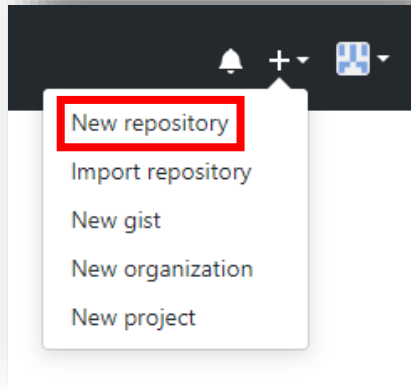


# Setting up

## Your first repository



Upper right corner:



Choose:

- name
- description
- status (*public/private*)
- README

The screenshot shows the 'Create new repository' page on GitHub. At the top, there's a 'PUBLIC' label and a 'New repository' icon. The 'Owner' is set to 'hubot' and the 'Repository name' is 'hello-world', which is marked with a green checkmark. Below this, a message says: 'Great repository names are short and memorable. Need inspiration? How about **petulant-shame**.' The 'Description (optional)' field contains the text 'Just another repository'. Under the 'Visibility' section, the 'Public' radio button is selected, with the text 'Anyone can see this repository. You choose who can commit.' The 'Private' option is also visible. The 'Initialize this repository with a README' checkbox is checked, with a note: 'This will allow you to `git clone` the repository immediately. Skip this step if you have already run `git init` locally.' At the bottom, there are two dropdown menus: 'Add .gitignore: None' and 'Add a license: None', followed by an information icon. A large green 'Create repository' button is at the very bottom.

**Owner**      **Repository name**

PUBLIC       **hubot**    /    hello-world    ✓

Great repository names are short and memorable. Need inspiration? How about **petulant-shame**.

**Description** (optional)

Just another repository

☒ **Public**  
Anyone can see this repository. You choose who can commit.

☐ **Private**  
You choose who can see and commit to this repository.

☒ **Initialize this repository with a README**  
This will allow you to `git clone` the repository immediately. Skip this step if you have already run `git init` locally.

Add .gitignore: **None**    |    Add a license: **None**    ⓘ

**Create repository**

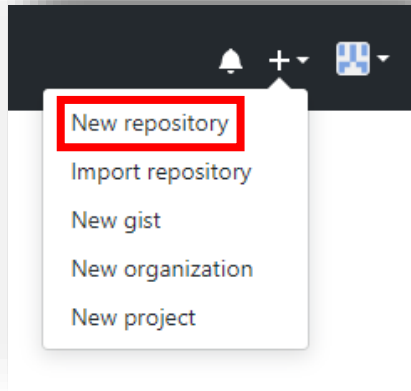


# Setting up

## Your first repository

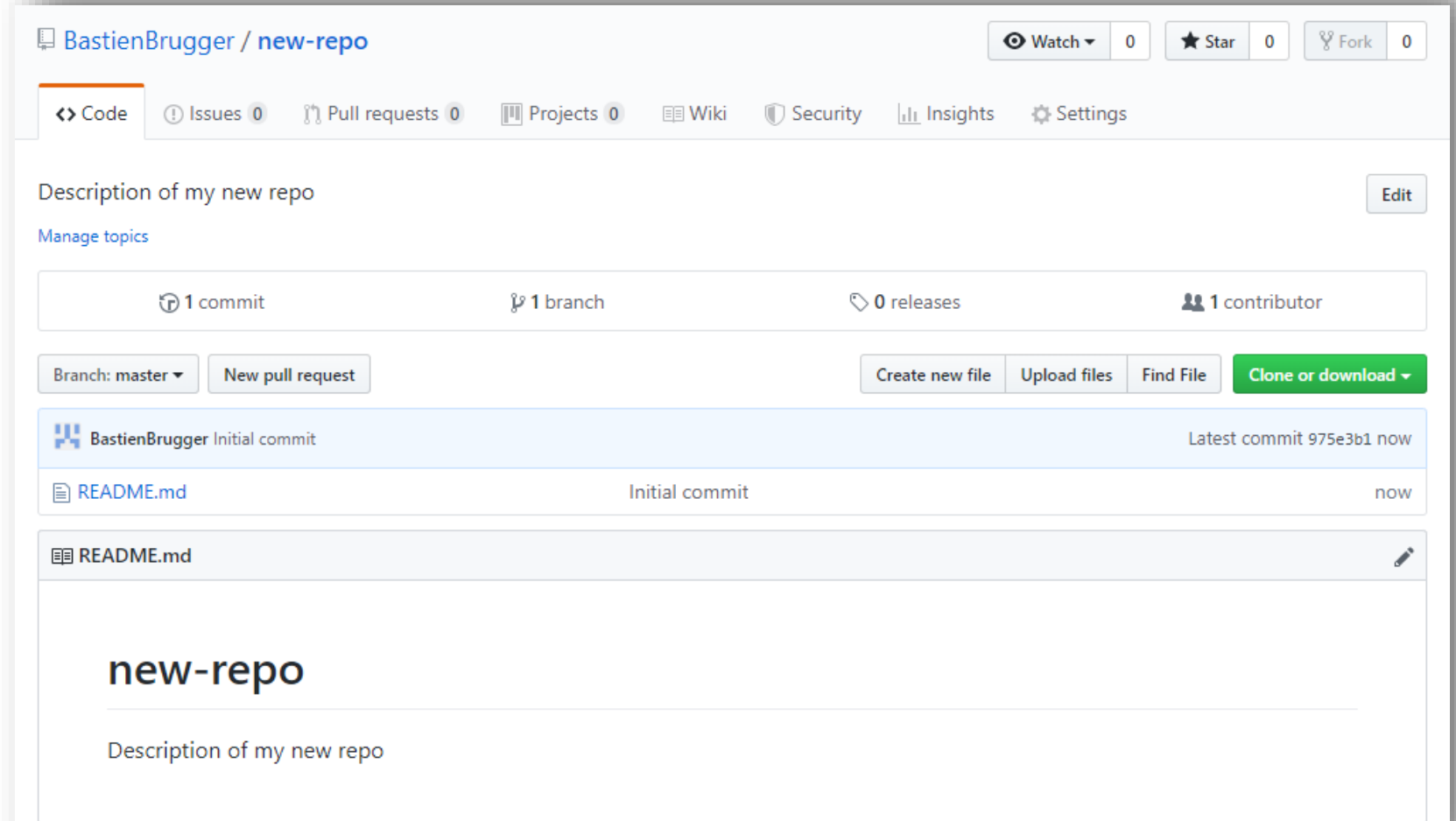


Upper right corner:

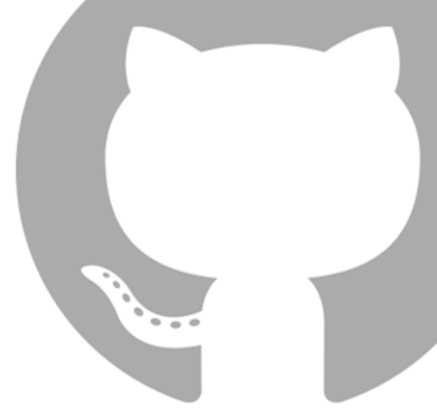


Choose:

- name
- description
- status (*public/private*)
- README



# Managing repos



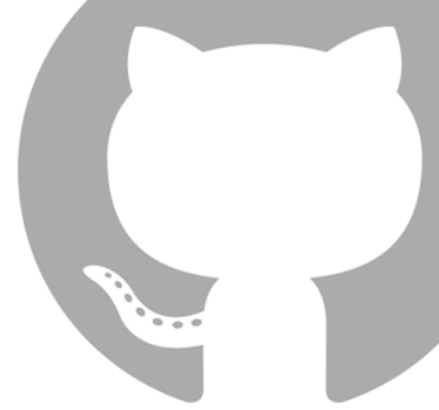
Three options:

## 1 Browser

---

- Available everywhere, no compatibility issues
- Create/fork repos, manage files
- Be social and discover existing repos
- *Impossible to run your program online*

# Managing repos

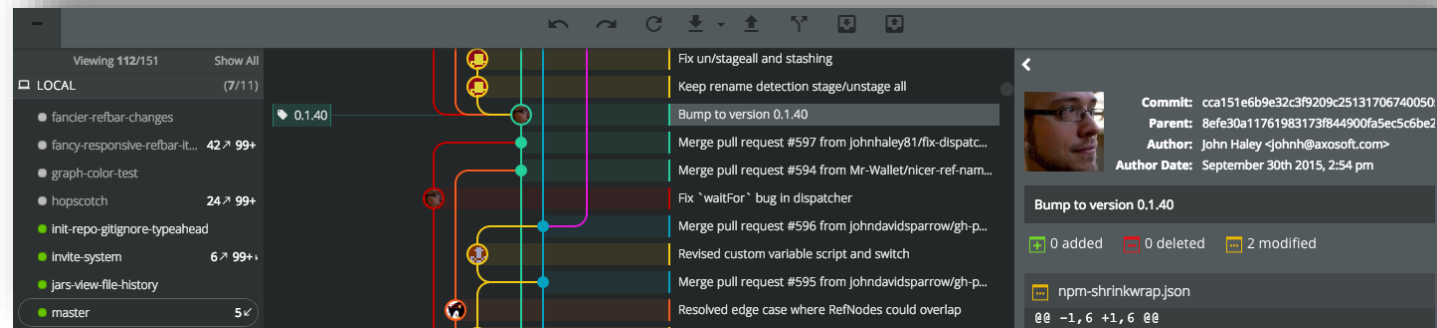


Three options:

## 2 Graphical User Interface (GUI)

---

- Multiple software options available
- Combines text editor + Git functions (*commit, pull request...*)
- User-friendly + graphic representation of collaboration



# Managing repos



Three options:

## 3 Command Line Interface

- Control over everything you do
- Automation via custom scripts
- Most documented method online

# Managing repos



Three options:

1

Browser

[github.com/\[username\]](https://github.com/[username])

2

GUI

[git-scm.com/downloads/guis](https://git-scm.com/downloads/guis)

3

Command line

[git-scm.com/download/\[linux/mac/win\]](https://git-scm.com/download/[linux/mac/win])

*Who won? Who's next? You decide!*