

Intro to git & Github



git



GitHub

Workshop Outline

Talk:

- Git and Github

Workshop:

- Github basics
- Using RStudio for version control and collaboration

Requirements:

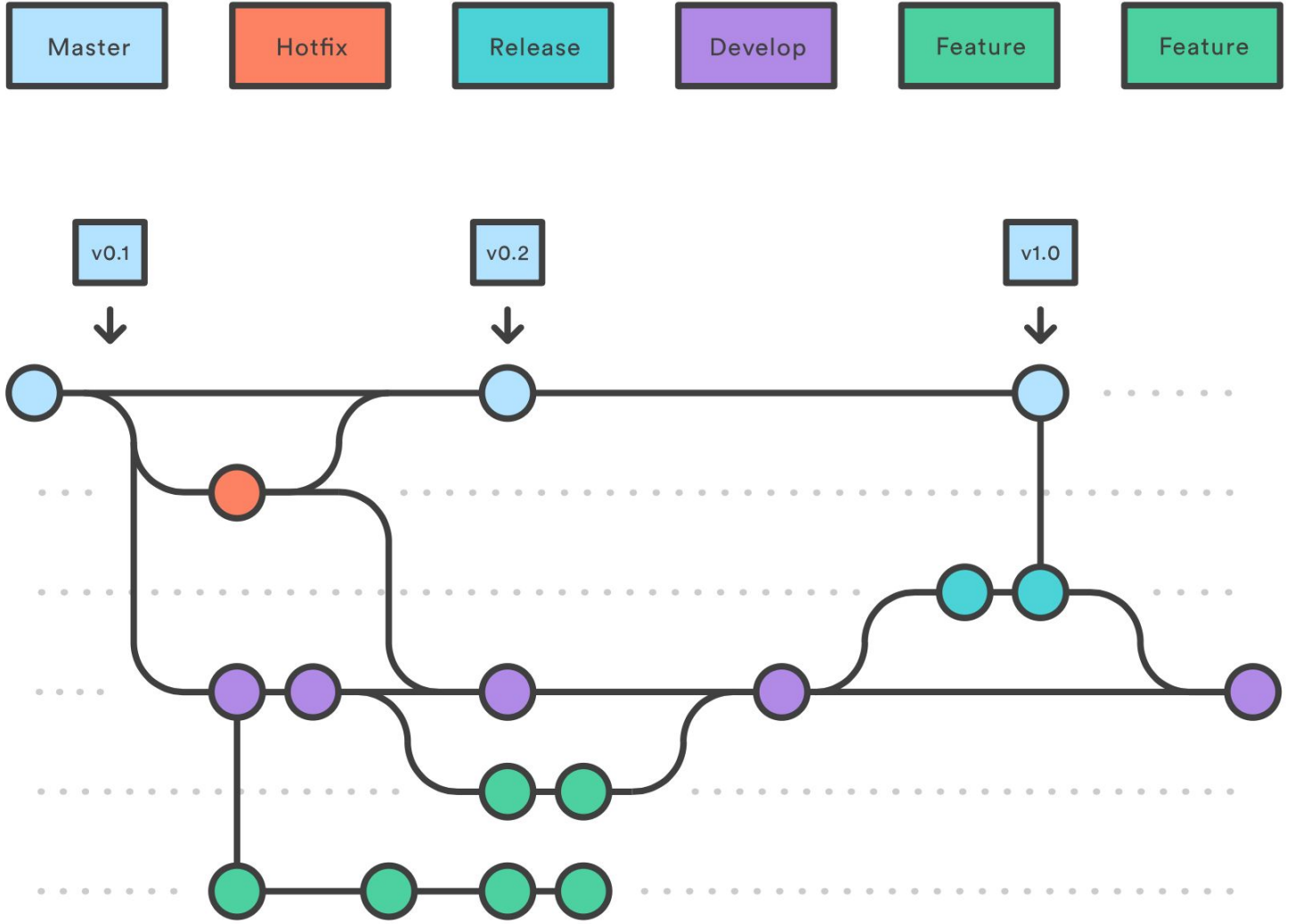
- Github account (<https://github.com>)
- RStudio (<https://rstudio.cloud>)



git

a free and open source distributed version control system

VERSION CONTROL



Why use version control?

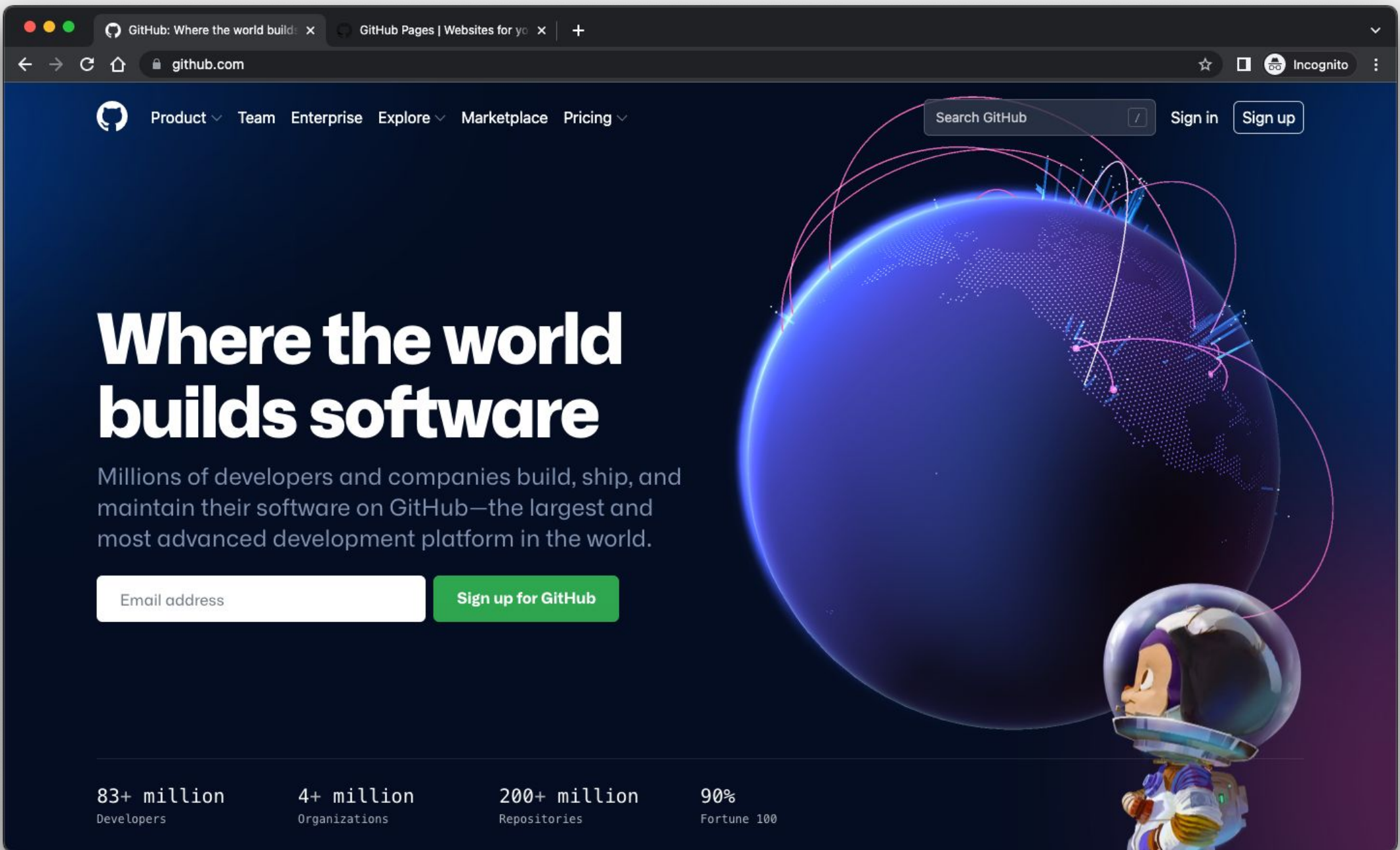
Track changes to the code, while enhancing communication and collaboration between team members.



git



GIT VS GITHUB (GITLAB, BITBUCKET)



Where the world builds software

Millions of developers and companies build, ship, and maintain their software on GitHub—the largest and most advanced development platform in the world.

[Sign up for GitHub](#)

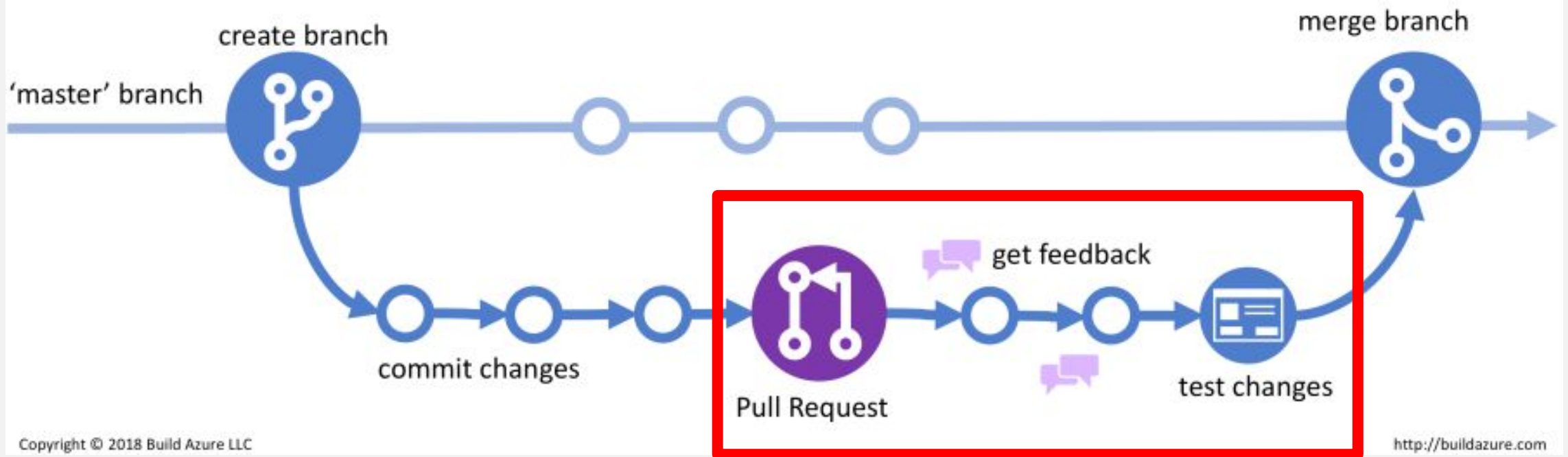
83+ million
Developers

4+ million
Organizations

200+ million
Repositories

90%
Fortune 100

GitHub Flow



Code lives on GitHub



```
# Create the linear regression
model = lm(
  height~Age,
  data = ageandheight
)
summary(model) # Review the results
```

```
# Create the linear regression
model = lm(
  height~Age,
  data = ageandheight
)
summary(model) # Review the results
```

← pull request



pull

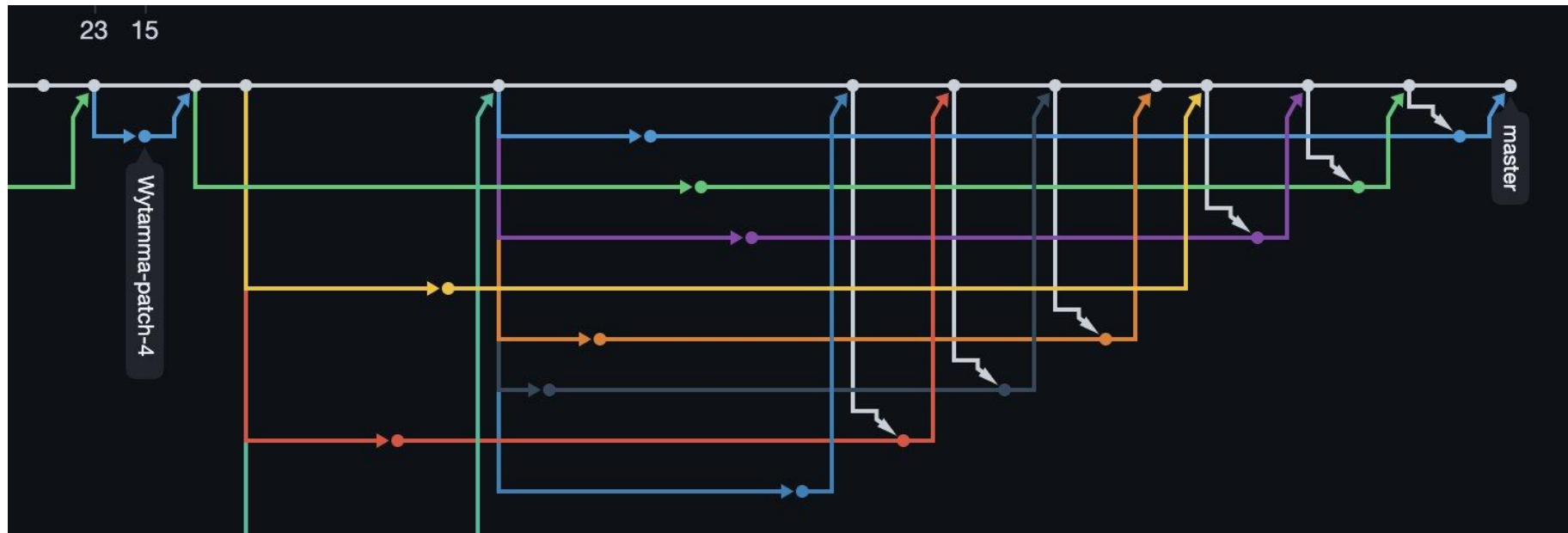
push

```
# Create the linear regression
model = lm(
  height~Age,
  data = ageandheight
)
summary(model) # Review the results
```

push



Can handle complex changes



8 things to do with git and github

Creating a new repository

Cloning a repository

Branching

Committing changes

Pull requests

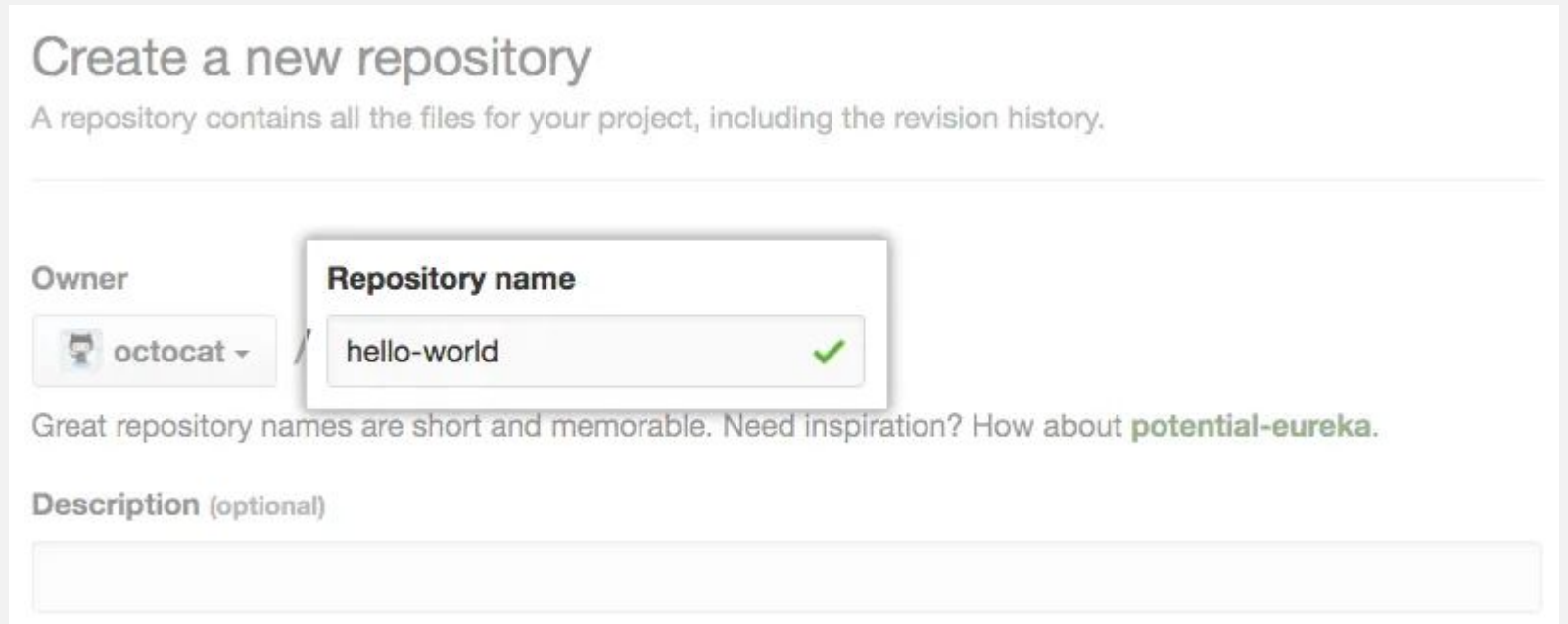
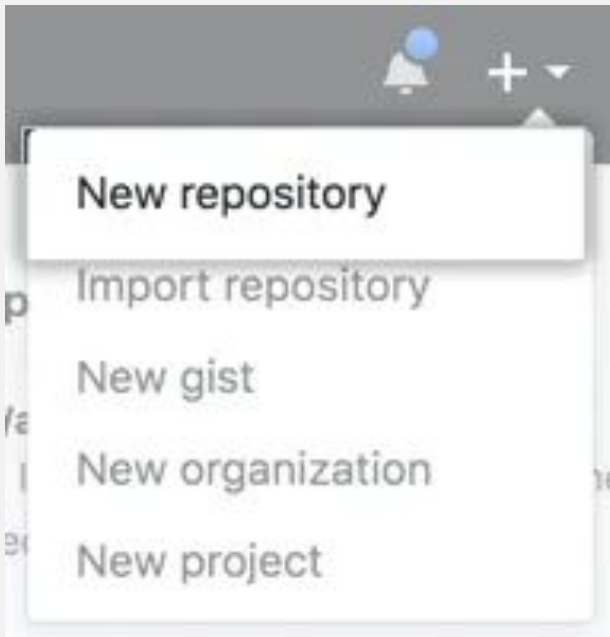
Collaboration

Issue tracking

Continuous integration

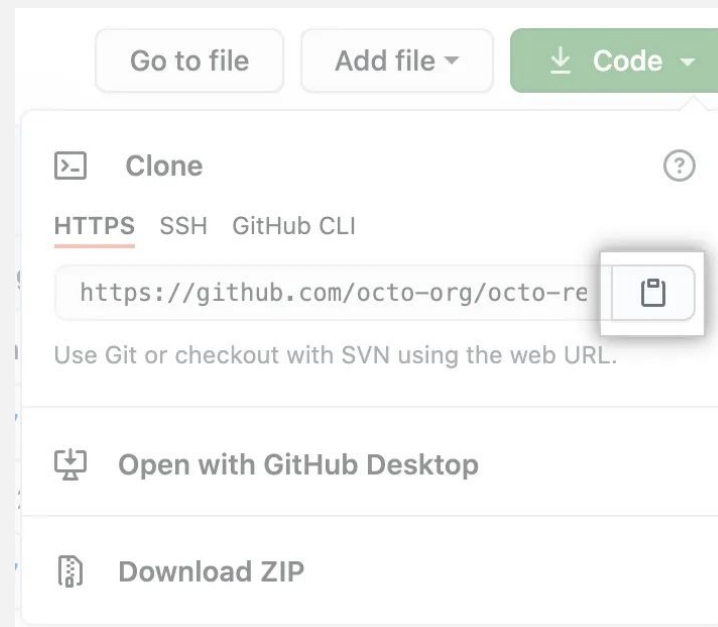
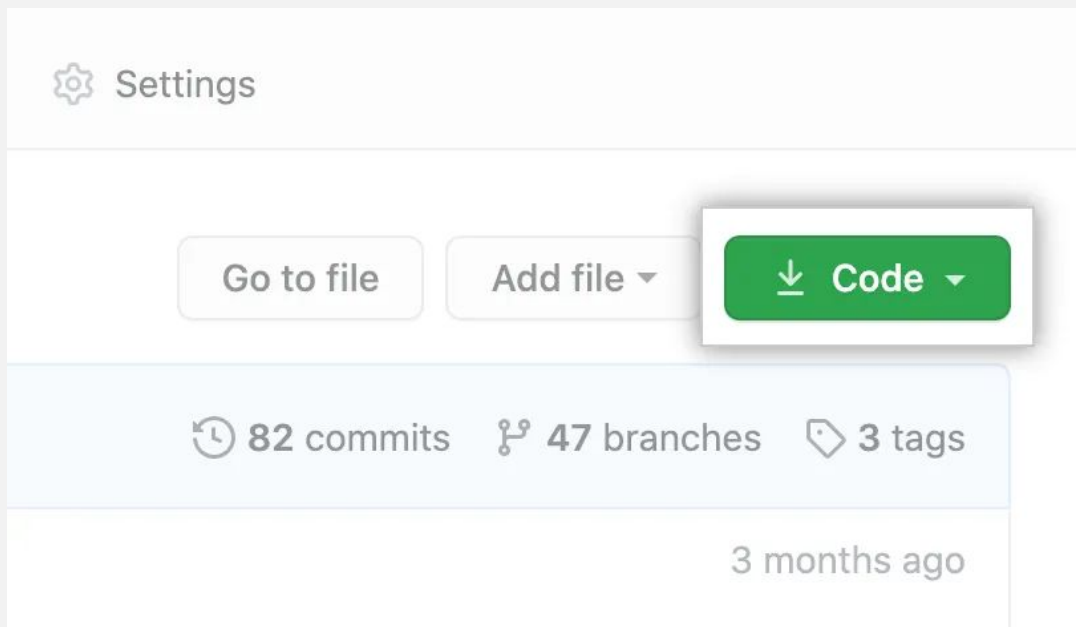
Creating a new repository

You can create a new Git repository on GitHub and start tracking your code changes by using git commands.



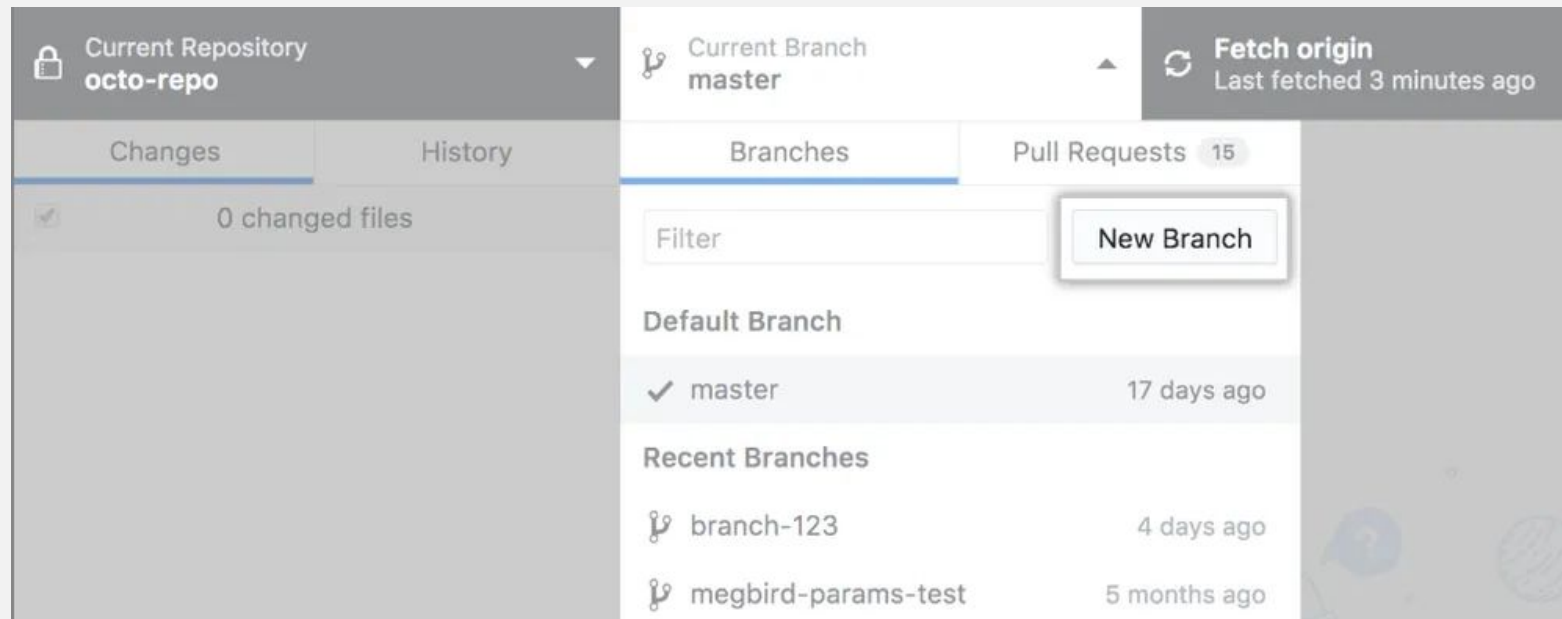
Cloning a repository

Clone an existing Git repository to your local machine to get the code and start working on it.



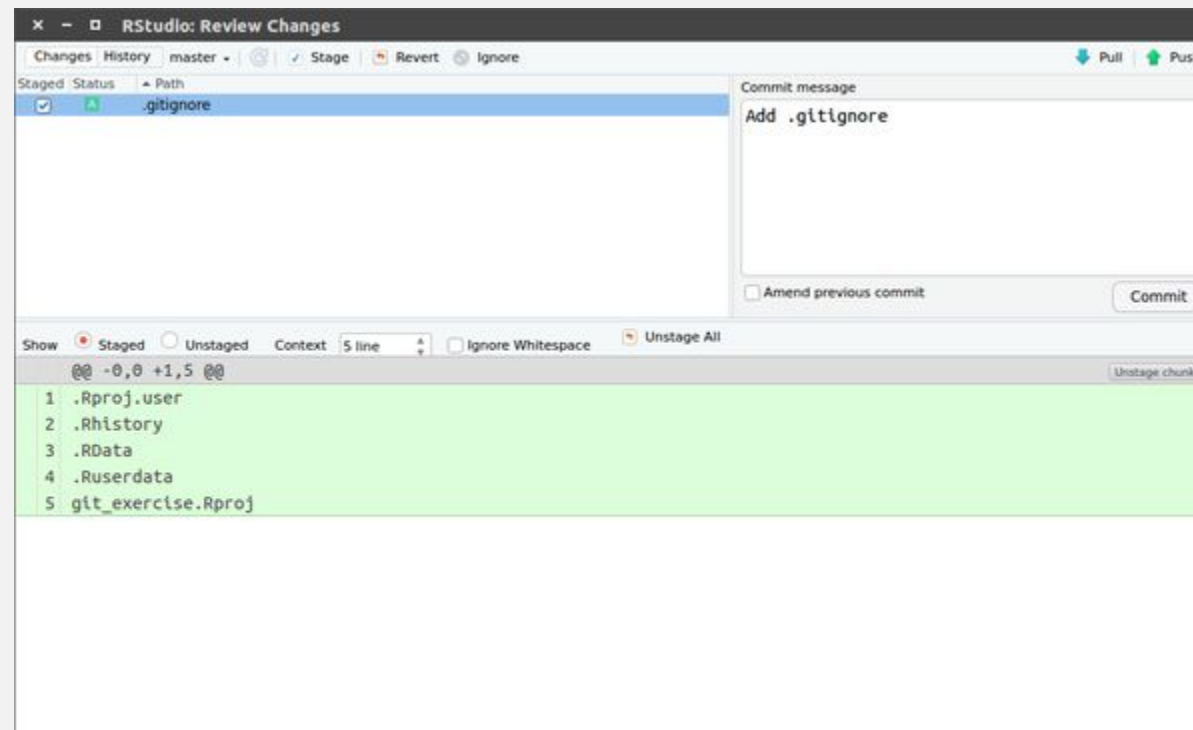
Branching

You can create a new branch in Git to work on a new feature or fix a bug, **without affecting the main codebase.**



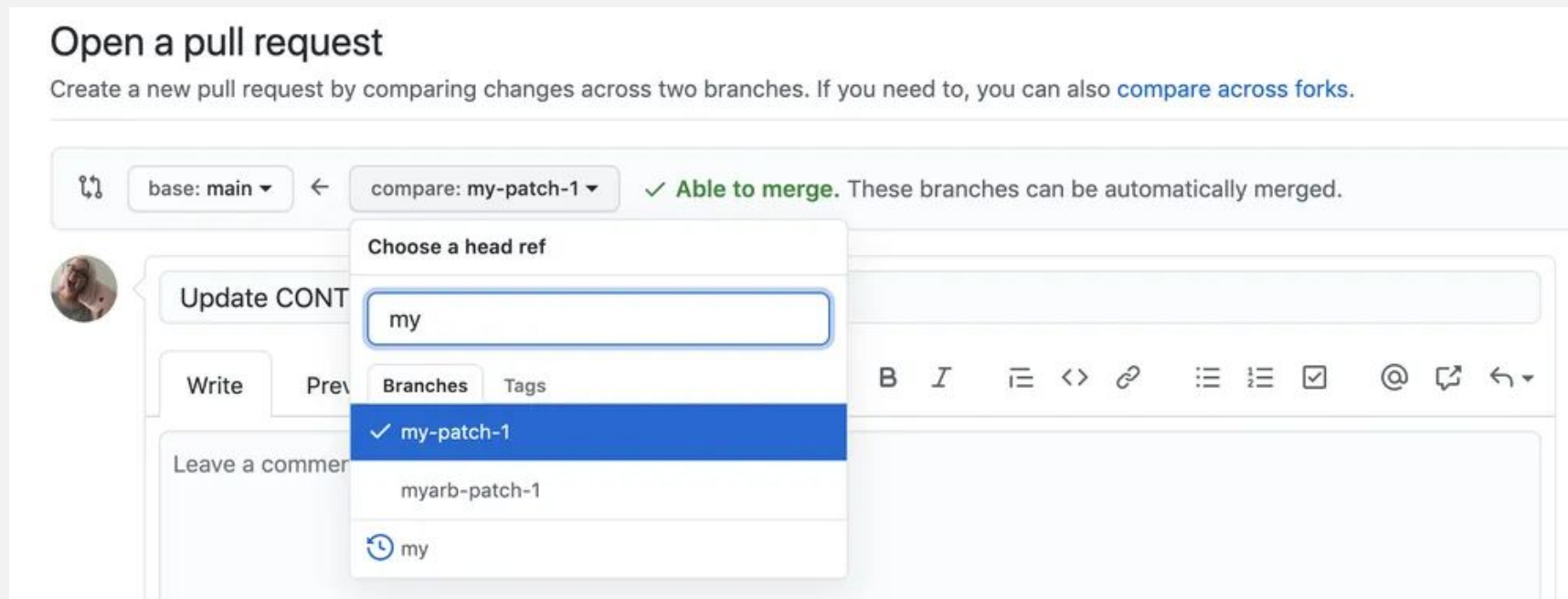
Commit changes

You can commit changes to your local repository and push them to the remote repository (/branch) on GitHub.



Pull requests

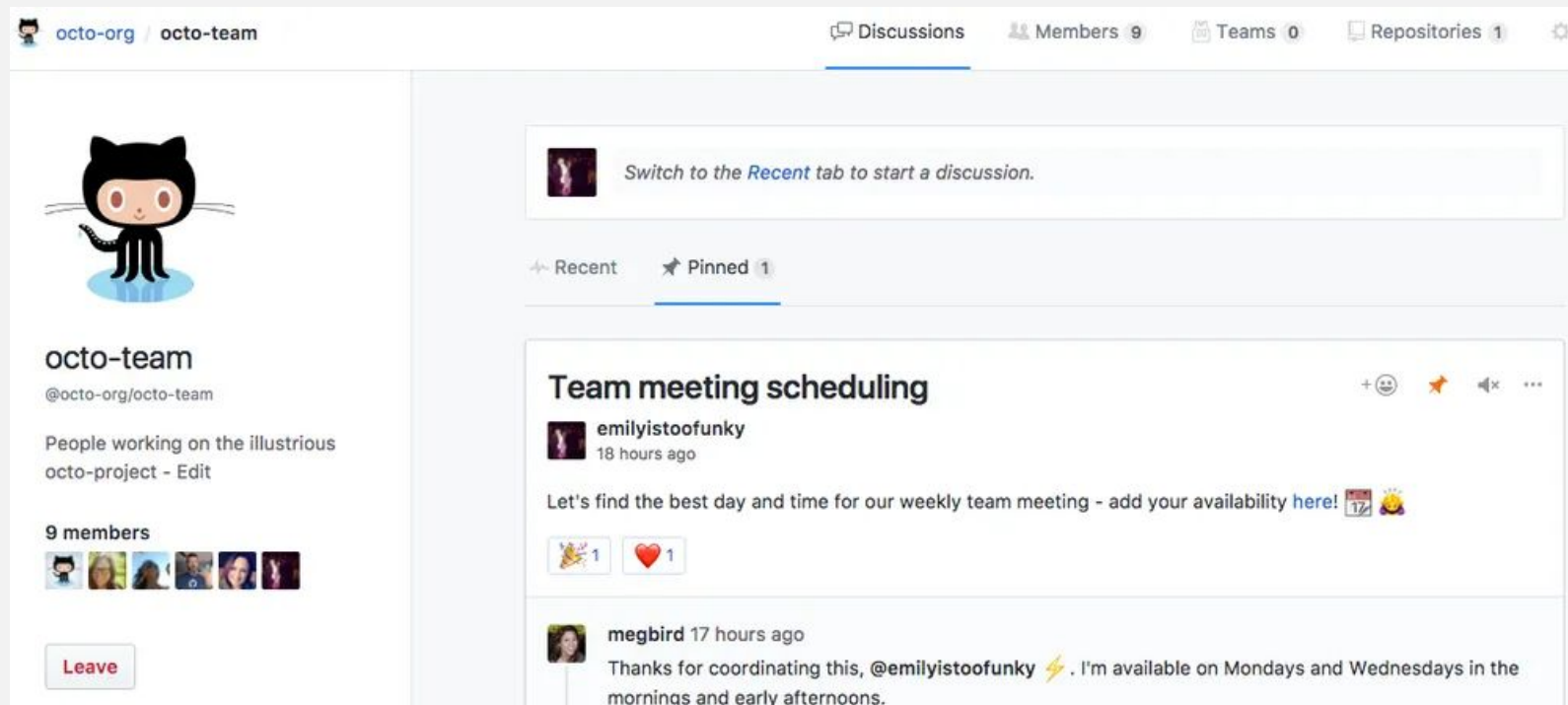
You can use pull requests to suggest changes to a repository or to merge a branch into the main branch.



The screenshot shows the GitHub interface for opening a pull request. At the top, it says "Open a pull request" and "Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#)." Below this, there are two dropdown menus: "base: main" and "compare: my-patch-1". To the right of these is a green checkmark and the text "Able to merge. These branches can be automatically merged." A dropdown menu is open over the "compare: my-patch-1" field, titled "Choose a head ref". It lists several options: "my" (selected), "my-patch-1" (with a checkmark), "myarb-patch-1", and "my" (with a refresh icon). The background shows a comment box with "Update CONT" and "Write" and "Prev" buttons, and a rich text editor with various formatting icons.

Collaboration

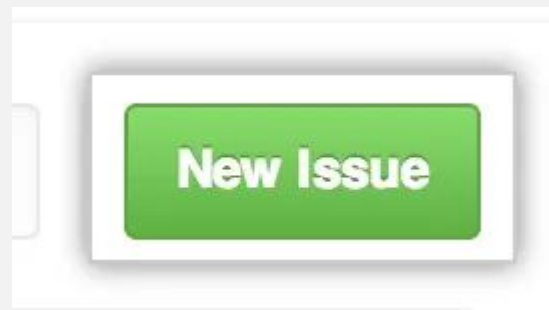
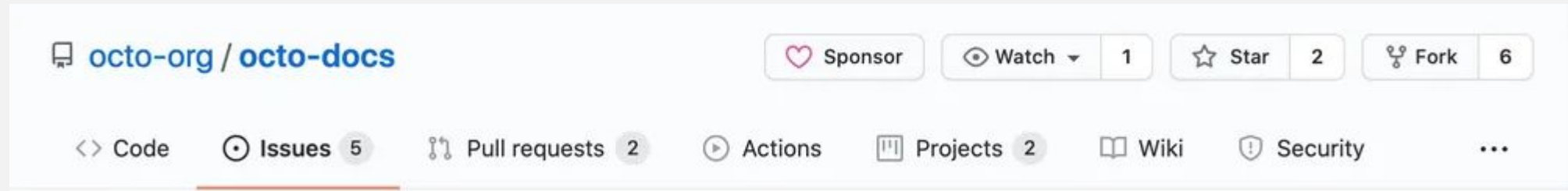
GitHub offers a social media-like experience with its commenting features, allowing developers to collaborate on issues, pull requests, and code reviews in real-time, making it easier to communicate and work together as a team.



The screenshot shows the GitHub interface for the 'octo-team' discussion page. At the top, there are navigation tabs for 'Discussions', 'Members 9', 'Teams 0', and 'Repositories 1'. The left sidebar features the GitHub logo, the team name 'octo-team', the handle '@octo-org/octo-team', a description 'People working on the illustrious octo-project - Edit', and a list of 9 team members. A 'Leave' button is visible at the bottom of the sidebar. The main content area shows a discussion titled 'Team meeting scheduling' by user 'emilyistoofunky' posted 18 hours ago. The discussion text reads: 'Let's find the best day and time for our weekly team meeting - add your availability [here!](#)'. Below the text are reaction icons for '1' and '1'. A comment from 'megbird' 17 hours ago says: 'Thanks for coordinating this, @emilyistoofunky ⚡. I'm available on Mondays and Wednesdays in the morninas and early afternoons.'

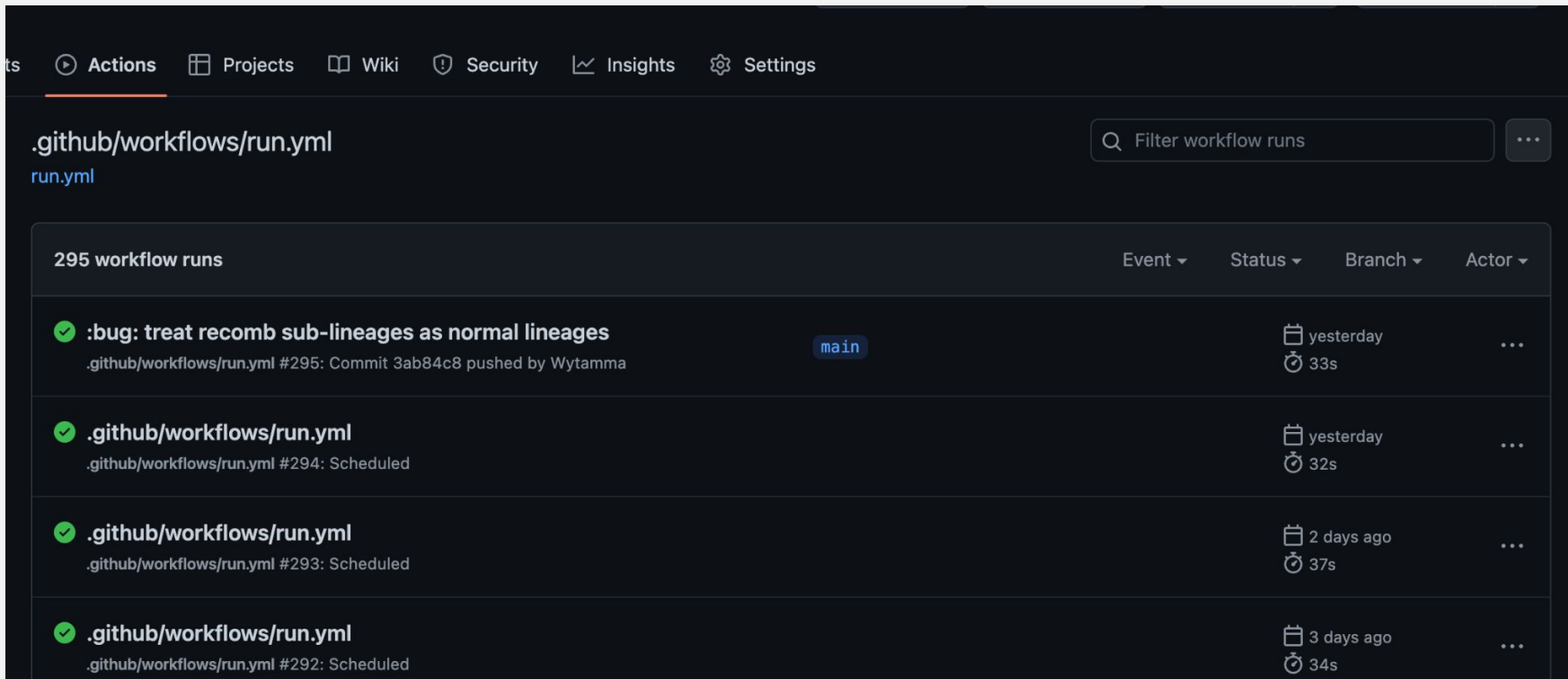
Issue tracking

You can use GitHub issues to track bugs, feature requests, and other tasks related to your project.



Continuous integration & deployment

You can use GitHub Actions to automatically build, test, and deploy your code whenever changes are made to the repository.



The screenshot shows the GitHub Actions interface for a repository. The top navigation bar includes links for Actions, Projects, Wiki, Security, Insights, and Settings. The main content area displays the workflow file `.github/workflows/run.yml` and a list of 295 workflow runs. The runs are filtered by event, status, branch, and actor. The first run is a bug report titled `:bug: treat recomb sub-lineages as normal lineages` on the `main` branch, triggered by a commit push. The subsequent runs are scheduled workflow runs for the same file.

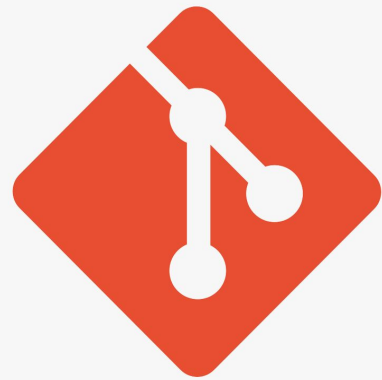
Event	Status	Branch	Actor
<code>:bug: treat recomb sub-lineages as normal lineages</code>	Success	<code>main</code>	Wytamma
<code>.github/workflows/run.yml</code>	Success	Scheduled	
<code>.github/workflows/run.yml</code>	Success	Scheduled	
<code>.github/workflows/run.yml</code>	Success	Scheduled	

What constitutes a good PR?

- It will be a complete piece of work that adds value in some way.
- It will have a title that reflects the work within, and a summary that helps to understand the context of the change.
- There will be well written commit messages, with well crafted commits that tell the story of the development of this work.
- Ideally it will be small and easy to understand. Single commit PRs are usually easy to submit, review, and merge.
- The code contained within will meet the best practises set by the team wherever possible.

<https://gist.github.com/mikepea/863f63d6e37281e329f8>

Some git commands



git

GETTING AND CREATING PROJECTS

git init [directory]

- Create an empty Git repository or reinitialize an existing one
- This command creates an empty Git repository - basically a .git directory with subdirectories for objects, refs/heads, refs/tags, and template files. An initial branch without any commits will be created (see the --initial-branch option below for its name).

```
git clone <repository>
```

- Clone a repository into a new directory
- Clones a repository into a newly created directory, creates remote-tracking branches for each branch in the cloned repository (visible using `git branch --remotes`), and creates and checks out an initial branch that is forked from the cloned repository's currently active branch.

BASIC SNAPSHOTTING

```
git add [<pathspec>...]
```

- Add file contents to the index
- This command updates the index using the current content found in the working tree, to prepare the content staged for the next commit. It typically adds the current content of existing paths as a whole, but with some options it can also be used to add content with only part of the changes made to the working tree files applied, or remove paths that do not exist in the working tree anymore.

`git status [<pathspec>...]`

- Show the working tree status
- Displays paths that have differences between the index file and the current HEAD commit, paths that have differences between the working tree and the index file, and paths in the working tree that are not tracked by Git (and are not ignored by [gitignore\[5\]](#)). The first are what you **would** commit by running `git commit`; the second and third are what you **could** commit by running `git add` before running `git commit`.

git diff [<path>...]

- Show changes between commits, commit and working tree, etc
- Show changes between the working tree and the index or a tree, changes between the index and a tree, changes between two trees, changes resulting from a merge, changes between two blob objects, or changes between two files on disk.

git commit [<pathspec>...]

- Record changes in the repository
- Create a new commit containing the current contents of the index and the given log message describing the changes. The new commit is a direct child of HEAD, usually the tip of the current branch, and the branch is updated to point to it (unless no branch is associated with the working tree, in which case HEAD is "detached" as described in [git-checkout\[1\]](#)).

SHARING AND UPDATING PROJECTS

git fetch [<refspec>...]

- Download objects and refs from another repository
- **git fetch** is the command that tells your local **git** to retrieve the latest meta-data info from the original (yet doesn't do any file transferring. It's more like just checking to see if there are any changes available).

```
git pull [<repository> [<refspec>...]]
```

- Fetch from and integrate with another repository or a local branch
- Incorporates changes from a remote repository into the current branch. In its default mode, git pull is shorthand for git fetch followed by git merge FETCH_HEAD.


```
git push [<repository> [<refspec>...]]
```

- Update remote refs along with associated objects
- Updates remote refs using local refs, while sending objects necessary to complete the given refs.

BRANCHING AND MERGING

git branch [<pattern>...]

- List, create, or delete branches
- If --list is given, or if there are no non-option arguments, existing branches are listed; the current branch will be highlighted in green and marked with an asterisk. Any branches checked out in linked worktrees will be highlighted in cyan and marked with a plus sign. Option -r causes the remote-tracking branches to be listed, and option -a shows both local and remote branches.

git checkout [<branch>]

- Switch branches or restore working tree files
- Updates files in the working tree to match the version in the index or the specified tree. If no pathspec was given, **git checkout** will also update HEAD to set the specified branch as the current branch.

git merge [<branch>]

- Join two or more development histories together
- Incorporates changes from the named commits (since the time their histories diverged from the current branch) into the current branch. This command is used by **git pull** to incorporate changes from another repository and can be used by hand to merge changes from one branch into another.

```
git tag <tagname>
```

- Create, list, delete or verify a tag object signed with GPG
- Add a tag reference in refs/tags/, unless -d/-l/-v is given to delete, list or verify tags.

```
git checkout tags/<tagname>
```

- This will checkout out the tag in a 'detached HEAD' state. In this state, "you can look around, make experimental changes and commit them, and [discard those commits] without impacting any branches by performing another checkout".

UNDO


```
git restore <pathspec>...
```

- Restore working tree files
- Restore specified paths in the working tree with some contents from a restore source. If a path is tracked but does not exist in the restore source, it will be removed to match the source.
- `git restore --source 7173808e script.R`

Markdown

a lightweight markup language for creating formatted text using a plain-text editor.

Element	Markdown Syntax
Heading	# H1 ## H2 ### H3
Bold	**bold text**
Italic	<i>*italicized text*</i>
Blockquote	> blockquote
Ordered List	1. First item 2. Second item 3. Third item
Unordered List	- First item - Second item - Third item
Code	`code`
Horizontal Rule	---
Link	[title] (https://www.example.com)
Image	![alt text] (image.jpg)



```
# monitauR <img src='images/logo.png' align="right" height="210" />
```

Easily and remotely monitor the progress of your R scripts.

```
```R  
devtools::install_github("wytamma/monitauR")
```
```

Example

Consider the script `example_scripts/square.R`

```
```R  
#< Setting up the square function
square <- function(x)
 x*x
}
#< Computing the square
square(5)
```
```

monitauR

Easily and remotely monitor the progress of your R scripts.

```
devtools::install_github("wytamma/monitauR")
```

Example

Consider the script `example_scripts/square.R`

```
#< Setting up the square function
square <- function(x) {
  x*x
}
#< Computing the square
square(5)
```



R Markdown

Combine R code and Markdown

~/Documents/rmarkdown - gh-pages - RStudio

2-chunks.Rmd

```
1 ---
2 title: "Magma Demo"
3 output: html_document
4 ---
5
6 ```{r include = FALSE}
7 knitr::opts_chunk$set(echo = FALSE)
8 ```
9
10 ```{r message = FALSE, warning = FALSE}
11 library(viridis)
12 ```
13
14 The code below demonstrates the Magma palette in the
15 [viridis](https://github.com/sjmgarnier/viridis) package. It
16 displays a contour map of the Maunga Whau volcano in Auckland, New
17 Zealand.
18
19 ## Magma colors
20
21 ```{r fig.cap = "The Maunga Whau volcano, Auckland."}
22 image(volcano, col = viridis(200, option = "A"))
23 ```
```

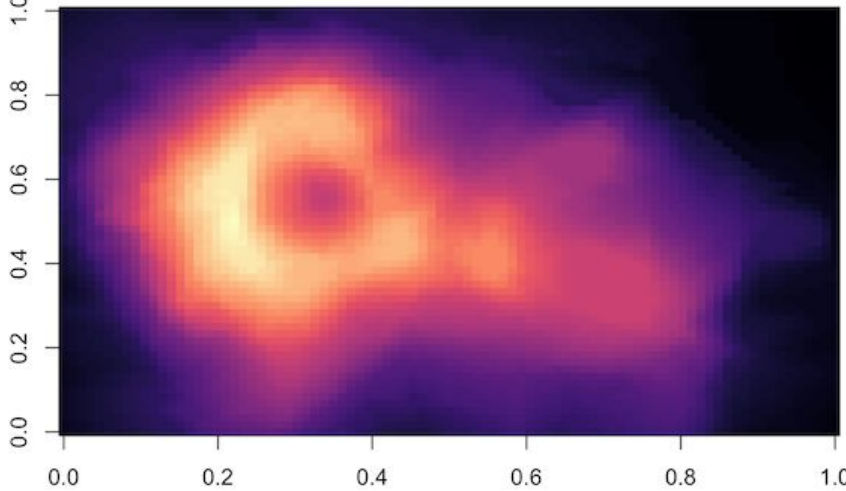
Environment History Build Git

Files Plots Packages Help Viewer

Magma Demo

The code below demonstrates the Magma palette in the [viridis](https://github.com/sjmgarnier/viridis) package. It displays a contour map of the Maunga Whau volcano in Auckland, New Zealand.

Magma colors



The Maunga Whau volcano, Auckland.

2:14 Magma Demo R Markdown

Console