



GitHub Foundations

Lecture Slides

These lecture slides are provided for personal and non-commercial use only

Please do not redistribute or upload these lecture slides elsewhere.

Good luck on your exam!

Version Control Systems (VCS)

Version Control Systems (VCS) are **designed to track changes or revisions to code.**



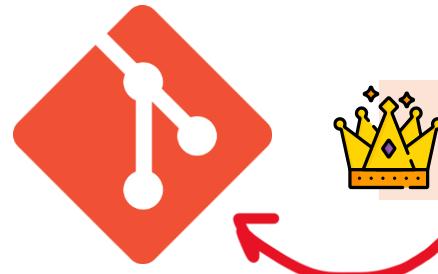
1990



2000



2005



The most popular VCS is **Git**.



Centralized VCS (DVCS)



Decentralized VCS (DVCS)

DVCS (and git specifically) became **very popular** for many reasons:

- Full local history and complete a copy of the repo locally
- Straightforward and efficient branching and merging
- Better performance, Improved fault tolerance, flexible workflows, work fully offline



Version Control normally represents revisions being represented as graph like structure. So you'll hear terms like **tree**, **trunk** and **branches**

Git



Git is a **distributed version control systems (DVCS)** created by **Linus Torvald**.



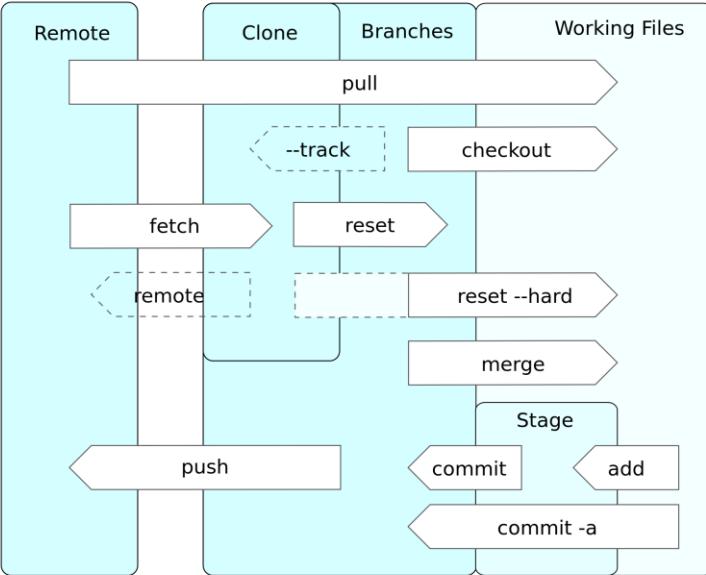
creator of the linux kernel

Branches:	Description	Date	Author	SHA
origin/65-final-category	correct the town	11 Oct 2023 12:11	Andrew Brown	16796d58
main origin origin/HEAD	commit files (...)	11 Oct 2023 11:48	Andrew Brown	abf341eb
origin/63-payday-home	commit files	7 Oct 2023 14:01	Andrew Brown	a8b839a9
origin/terraform-cloud	yes!	30 Sep 2023 13:59	Andrew Brown	1256b515
please work!!!!		30 Sep 2023 13:54	Andrew Brown	e5baa1d5
lets go		30 Sep 2023 13:49	Andrew Brown	33b8ea39
comment out more		30 Sep 2023 13:42	Andrew Brown	a8e47468
comment out code make things work		30 Sep 2023 13:40	Andrew Brown	c98c7107
try and fix our issues.		30 Sep 2023 13:31	Andrew Brown	47b32b8f
wip		30 Sep 2023 13:23	Andrew Brown	ad732e57
2.6.0	#56 fix issues with provider, setup two diff...	29 Sep 2023 12:21	Andrew Brown	02fa4eb7
origin/56-multi-home-terraform-cloud	#56 fix iss...	29 Sep 2023 12:21	Andrew Brown	8e026527
2.5.0	#52 test our terra home on terratowns test ...	29 Sep 2023 10:07	Andrew Brown	89dfc0e6
origin/52-terratown-test	#52 test our terra home ...	29 Sep 2023 10:06	Andrew Brown	895e72c3
2.4.0	47 terratowns provider (#51)	28 Sep 2023 15:31	Andrew Brown	f0ef63f1
origin/47-terratowns-provider	Merge branch 'main'...	28 Sep 2023 15:31	Andrew Brown	d23bf312
\$47	finish implementation for terraform provider	28 Sep 2023 15:29	Andrew Brown	004a68fc
wip		28 Sep 2023 12:56	Andrew Brown	46f53c2e
2.3.0	47 terratowns provider (#50)	28 Sep 2023 11:52	Andrew Brown	5f9189ee
#47	setup skeleton for resource	28 Sep 2023 11:51	Andrew Brown	103d77d1
Merge branch 'main' into 47-terratowns-provider		27 Sep 2023 16:35	Andrew Brown	257ca11b
2.2.0	47 terratowns provider (#49)	27 Sep 2023 16:33	Andrew Brown	976bd7cb
Merge branch 'main' into 47-terratowns-provider		27 Sep 2023 16:33	Andrew Brown	ad0c0347
#47	implement terraform provider block in terraform	27 Sep 2023 16:31	Andrew Brown	72c519cb

```
v 2 public/index.html
  ...
  6   6      <title>Hello Terraformers!</title>
  7   7      </head>
  8   8      <body>
  9   -      <h1>Hello Terraformers!!!!</h1>
  9   +      <h1>Hello there Terraformers!</h1>
 10  10     </body>
 11  11     </html>
```

Each change of your code (**git commit**) can be captured and tracked throughout the history of your project (**git tree**)

Common Git Terms



- **Repository:** Represents the logical container holding the codebase
- **Commit:** Represents a change of data in the local repository.
- **Tree:** Represent the entire history of a repo
- **Remote:** A version of your project hosted elsewhere, used for exchanging commits.
- **Branches:** Divergent paths of development, allowing isolated changes.
 - **Main (previously known as master)** the most common name for the default branch
- **Clone:** Creates a complete local copy of a repository, including its history.
- **Checkout:** Switches between different branches or commits in your repo.
- **Pull:** Downloads changes from a remote repository and merges them into your branch.
- **Push:** Uploads your local repository changes to a remote repository.
- **Fetch:** Downloads data from a remote repo without integrating it into your work.
- **Reset:** Undoes local changes, with options to unstage or revert commits.
- **Merge:** Combines multiple commit histories into one.
- **Staging files:** Prepares and organizes changes for a commit.
 - **Commit:** Saves your changes as a snapshot in the local repository.
 - **Add:** Adds changes to the staging area for the next commit.

Version Control Services (VCS)

Version Control Services (VCS) are **fully managed cloud services that hosts your version controlled repositories**.

These services often have additional functionality going beyond just being a remote host for your repos.

Git is the most popular and often the only choice for VCS. Often, we call these services "**git providers**"



GitHub

Owned by Microsoft. The most popular VCS offering due to ease of use and being around the longest. GitHub is primarily where open-source projects are hosted and offer rich functionality such as Issues Tracking, Automation pipelines and a host of other features.



GitLab

GitLab was an emerging competitor to GitHub and at the time had unique features such as a CI/CD pipeline and improved security measures.



BitBucket

Owned by Atlassian. Originally hosted Mercurial, later adding Git and then sunsetting Mercurial. Tight integration with Atlassian's industry standard agile project management software **JIRA**. Larger organizations commonly use BitBucket because of JIRA adoption.



SourceForge

One of the oldest places to host your source code and was the first to be free of charge to open-source projects

GitHub



GitHub is a **Version Control Service** that initially offered hosting managed remote git repositories and has **expanded to provide other offerings** around hosted codebases.

- Git Repository Hosting
- Project Management Tools
- Issue Tracking
- Pull Requests and Code Review
- GitHub Pages and Wikis
- GitHub Actions
- GitHub Copilot
- GitHub Codespaces
- GitHub Marketplace
- GitHub Gists
- GitHub Discussions
- Collaboration Features (Organizations and Teams)
- API Access and Development Tools (Github Desktop, Github CLI)
- Security Features (eg. Autodetecting credentials in repos)
- Educational Resources and Course Automation (Github Classroom)



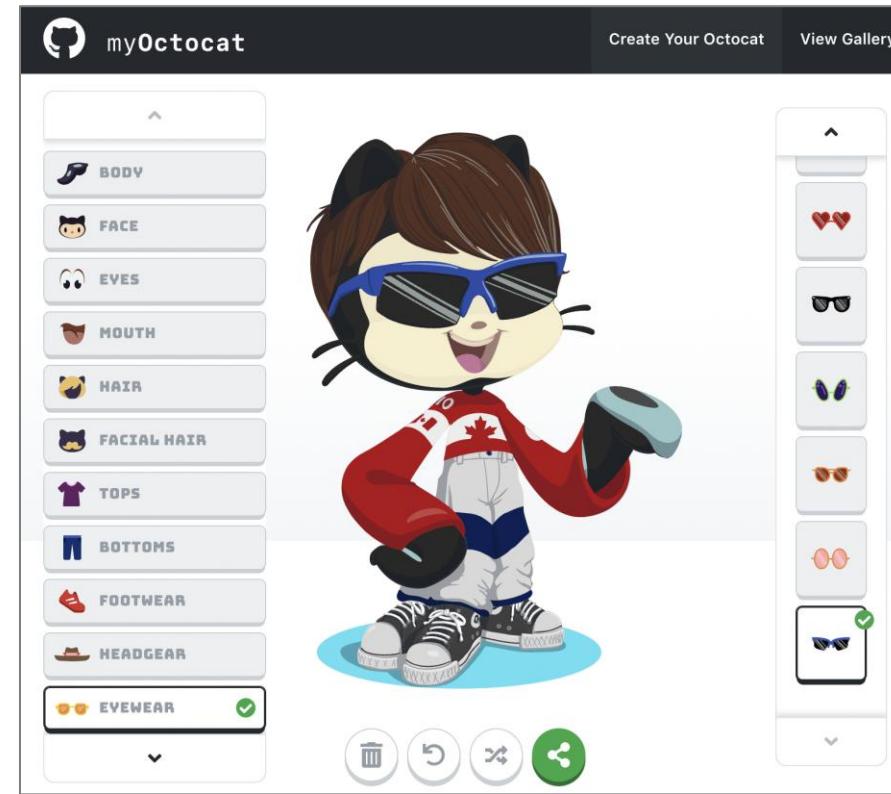
Fun Fact! GitHub was originally built using **Ruby on Rails**

GitHub Octocat



You can **create your own** Octocat

<https://myoctocat.com>



Git vs GitHub

Git	GitHub
Nature	Distributed Version control system (DVCS)
Functionality	Manages source code history
Access	Local system installation
Scope	Local repository management
Collaboration	Local changes, requires manual sharing
Usage	Command-line interface
	Version Control as a Service (VCaaS)
	Provides cloud storage for Git repositories
	Accessed via web interface
	Online collaboration and repository hosting
	Integrated tools for collaboration (issues, PRs)
	Graphical interface and additional features

GitHub Repo

A GitHub Repo is **your git repo that you push upstream to GitHub**

GitHub allows you access and manage your git repo with several functionality

From your GitHub repo page you can:

- View different branches
- View tags
- View commit history
- Explore repo's files
- View releases
- See codebase language breakdown
- View the top-level markdown files
 - Readme.md
 - Licence.md

You can perform actions such as:

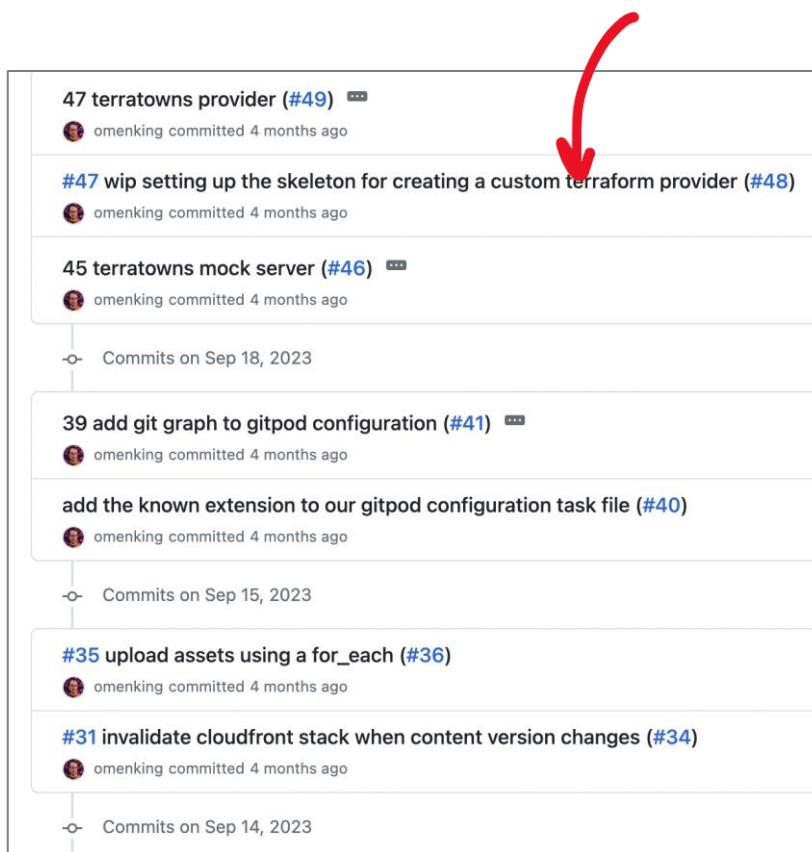
- Pinning
- Watching
- Forking
- Starring
- Cloning (downloading)

The screenshot shows a GitHub repository page for 'omenking / terraform-beginner-bootcamp-2023'. The page includes a navigation bar with links for Code, Issues (4), Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. Below the navigation is a search bar and a user profile icon. The main content area displays the repository name 'terraform-beginner-bootcamp-2023' (Public) and a note that it was generated from 'ExamProCo/terraform-beginner-bootcamp-2023'. It shows 27 branches and 29 tags. A list of recent commits by 'omenking' is shown, along with a commit file list. On the right side, there are sections for About (no description, website, or topics provided), Releases (29 tags, Create a new release), Packages (No packages published, Publish your first package), and a sidebar with options like Readme, MIT license, Activity, 59 stars, 7 watching, and 18 forks.

Commit	Author	Date	Message
commit files (#64)	omenking	3 months ago	abf341e · 30 Commits
bin		4 months ago	47 terratowns provider (#51)
journal		4 months ago	47 terratowns provider (#50)
modules/terrahome_aws		4 months ago	#56 fix issues with provider, setup two different town...
public		3 months ago	commit files (#64)
terraform-provider-terratowns		4 months ago	#56 fix issues with provider, setup two different town...
terratowns_mock_server		4 months ago	45 terratowns mock server (#46)
.env.example		4 months ago	#15 generate workaround for terraform login with gitp...
.gitignore		4 months ago	#47 wip setting up the skeleton for creating a custom...
.gitpod.yml		4 months ago	#56 fix issues with provider, setup two different town...
LICENSE		4 months ago	Initial commit

Git Commit

A Git Commit represent incremental changes to a codebase represented with a git tree (graph) at a specific time.



A screenshot of a GitHub commit history. A red arrow points from the text above to the commit list. The commits are:

- 47 terratowns provider (#49) - omenking committed 4 months ago
- #47 wip setting up the skeleton for creating a custom terraform provider (#48) - omenking committed 4 months ago
- 45 terratowns mock server (#46) - omenking committed 4 months ago
 - o- Commits on Sep 18, 2023
- 39 add git graph to gitpod configuration (#41) - omenking committed 4 months ago
- add the known extension to our gitpod configuration task file (#40) - omenking committed 4 months ago
 - o- Commits on Sep 15, 2023
- #35 upload assets using a for_each (#36) - omenking committed 4 months ago
- #31 invalidate cloudfront stack when content version changes (#34) - omenking committed 4 months ago
 - o- Commits on Sep 14, 2023

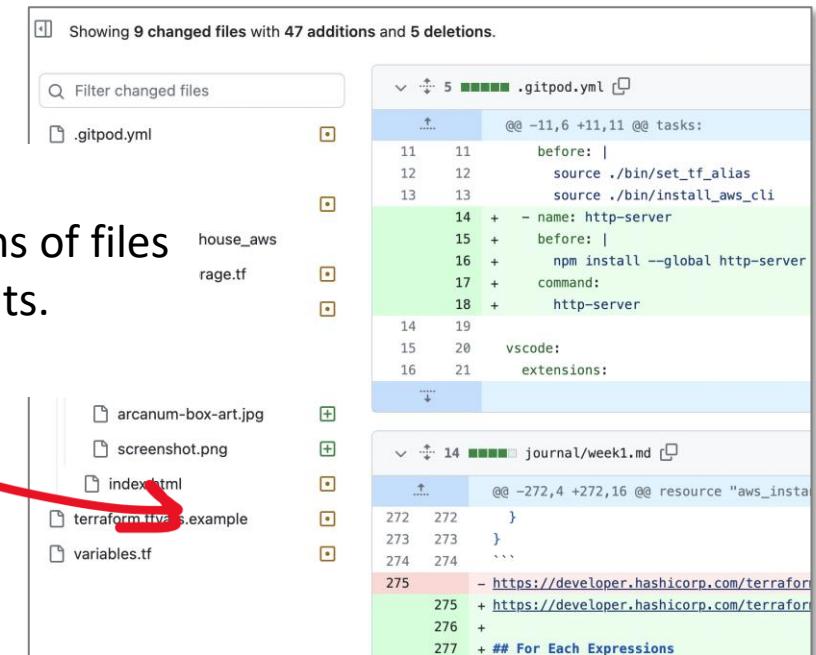
A git commit contains:

- Additions, **modifications** and deletions of files
- Additions and deletions of file contents.
- Not the whole files themselves.

Each Commit has a SHA hash that acts as an ID

- Eg. 661a91ad3b66926c4591f9d3c73c087906945f3b

You can use this **Commit SHA** to *checkout* specific commits.



A screenshot of a git diff interface. A red arrow points from the text above to the diff view. The interface shows 9 changed files with 47 additions and 5 deletions. The files include .gitpod.yml, house_aws.rage.tf, arcanum-box-art.jpg, screenshot.png, index.html, variables.tf, and journal/week1.md. The diff highlights changes in .gitpod.yml and journal/week1.md.



Git does not store the whole files in each commit but rather the state of changes. This greatly reduces the file size. To the developer the files will appear whole

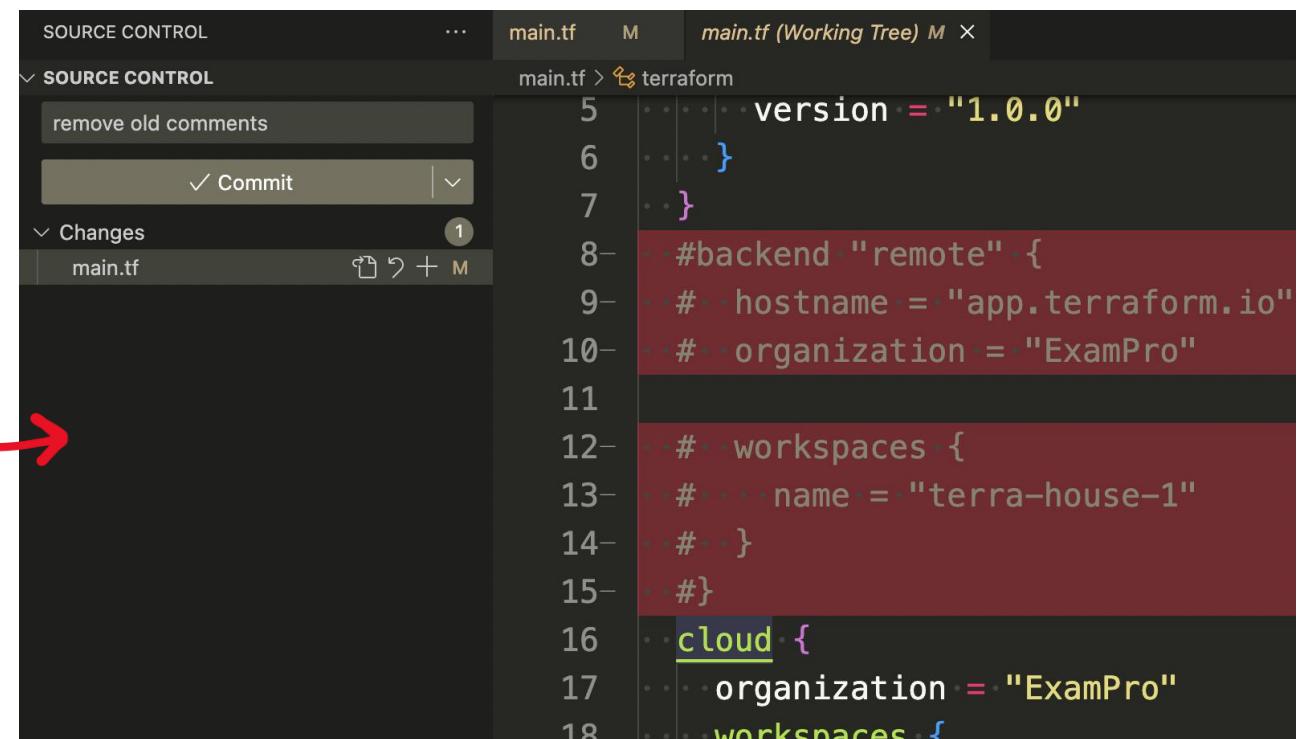
Git Commit

Components of Git Commit:

- **Commit Hash:** A unique SHA-1 hash identifier for the commit.
- **Author Information:** The name and email of the person who made the commit.
- **Commit Message:** A description of what changes the commit contains.
- **Timestamp:** The date and time when the commit was made.
- **Parent Commit Hash(es):** The SHA-1 hash of the commit(s) this commit is based on.
- **Snapshot of Content:** A snapshot of the project at the time of the commit (not the actual files, but references to them).

Commit messages are often written in a tool as its more convenient to quickly add, remote files, audit changes

VS Code has that ability via the **Source Control** window



The screenshot shows the VS Code interface with the Source Control window open. The main.tf file is selected, and the commit message "remove old comments" is entered in the commit input field. The Changes section shows one change to main.tf. The code editor on the right displays Terraform configuration code:

```
version = "1.0.0"
...
#backend "remote" {
#  hostname = "app.terraform.io"
#  organization = "ExamPro"
#
#  workspaces {
#    name = "terra-house-1"
#  }
#
#}
cloud {
  organization = "ExamPro"
  workspaces {
  }
}
```

Git Commit

Common **Git Commands** you should know:

You need add or remove files to be stage changes

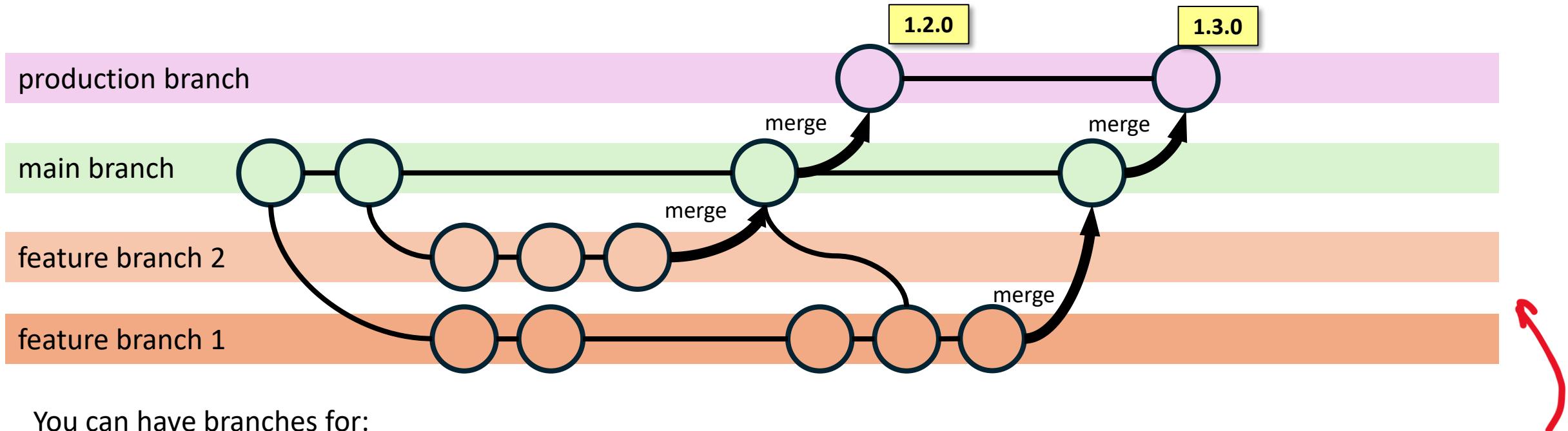


```
# Files to be staged  
git add my-file  
git add . # add all files  
git rm my-other file # remove a specific file  
  
# Commits staged changes with a message  
git commit -m "Commit message"  
  
# Automatically stages all tracked, modified files before the commit  
git commit -a -m "Commit message"  
  
#Modifies the most recent commit  
git commit --amend  
  
#Creates an empty commit, useful as a placeholder  
git commit -m "Initial commit" --allow-empty  
  
# Commits with a specified author.  
git commit -m "Message" --author="Author Name <email@example.com>"  
  
# checkout to a specific commit based on SHA hash  
git checkout 661a91ad3b66926c4591f9d3c73c087906945f3b
```

Git Branch

A **git branch** is a divergence of the state of the repo.

You can *think* of branches as being copies of a point in time that have been modified to be different.



You can have branches for:

- Specific environments: eg. Staging, development, production
- Specific to developers: eg. andrew, bayko, cindy
- branches per feature: eg. payment-gateway-feature
- Branches per bug eg. hotfix-blank-homepage

This type of workflow is very close to **GitHub Flow**

Git Branch

Common **git branch commands** you should know

```
#Lists all local branches  
git branch  
  
#Creates a new branch  
git branch [branch-name]  
  
# Checkout (switch to) a branch  
git checkout [branch-name]  
  
# Create and checkout a branch  
git checkout -b [branch name]  
  
#Deletes a branch  
git branch -d [branch-name]:  
  
# Renames a branch  
git branch -m [old-name] [new-name]  
  
# Lists both remote and local branches  
git branch -a
```



A **common workflow** for developers is to create a branch or a feature, and they need to push their branch upstream to the remote name origin.

```
git checkout -b my-new-branch  
# ... makes changes to files  
git add .  
git commit -m "my changes"  
git push -u origin my-new-branch
```



Git Remote

A git **remote** represents **the reference to remote location** where a copy of the repository is hosted.

You can have multiple remote entries for your git repo.

- "origin" as a remote name almost always seen for a repo.
 - It indicates the central or golden repo everyone is working from and represents the source of truth.

The remote entries are stored in **.git/config**

```
[remote "origin"]
url = https://github.com/exaproco/exapro.git
fetch = +refs/heads/*:refs/remotes/origin/*
[branch "production"]
remote = origin
merge = refs/heads/production
[branch "main"]
remote = origin
merge = refs/heads/main
```

Notice remote names can be **referenced**

```
# Lists all remote repositories along with their URLs.
git remote -v

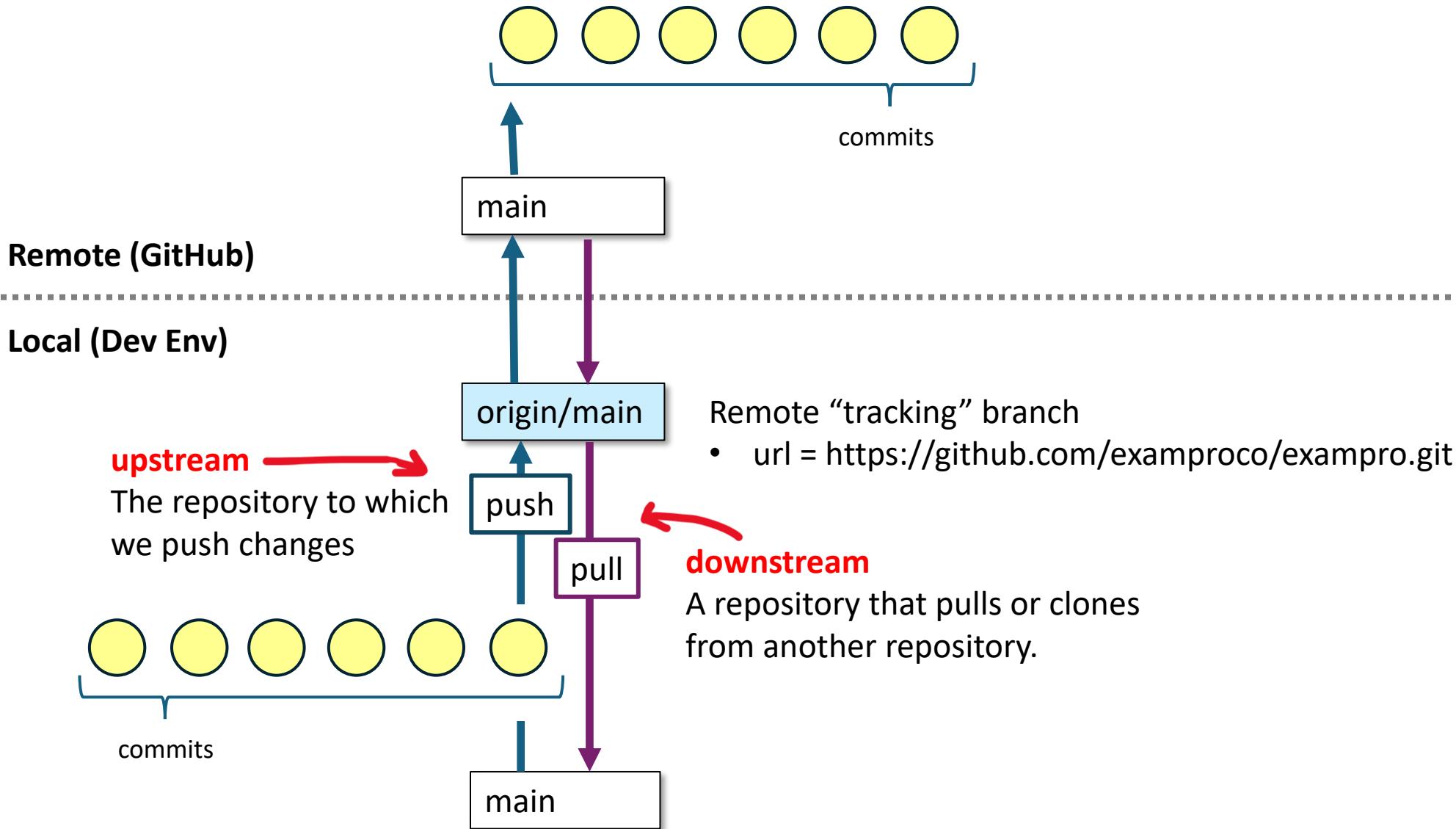
git remote add [name] [URL]
git remote remove [name]
git remote rename [old-name] [new-name]

#Pushes a branch and its commits to the specific remote.
git push [remote-name] [branch]

#Pulls updates from a remote branch.
git pull [remote-name] [branch]

# Fetch updates without pulling.
git fetch [remote-name]
```

Git Remote – Upstream and Downstream



GitHub Flow

GitHub Flow is a **light-weight workflow** for multiple developers working on a single repository

Create a Branch: For each new task or feature, create a new branch off the main branch.

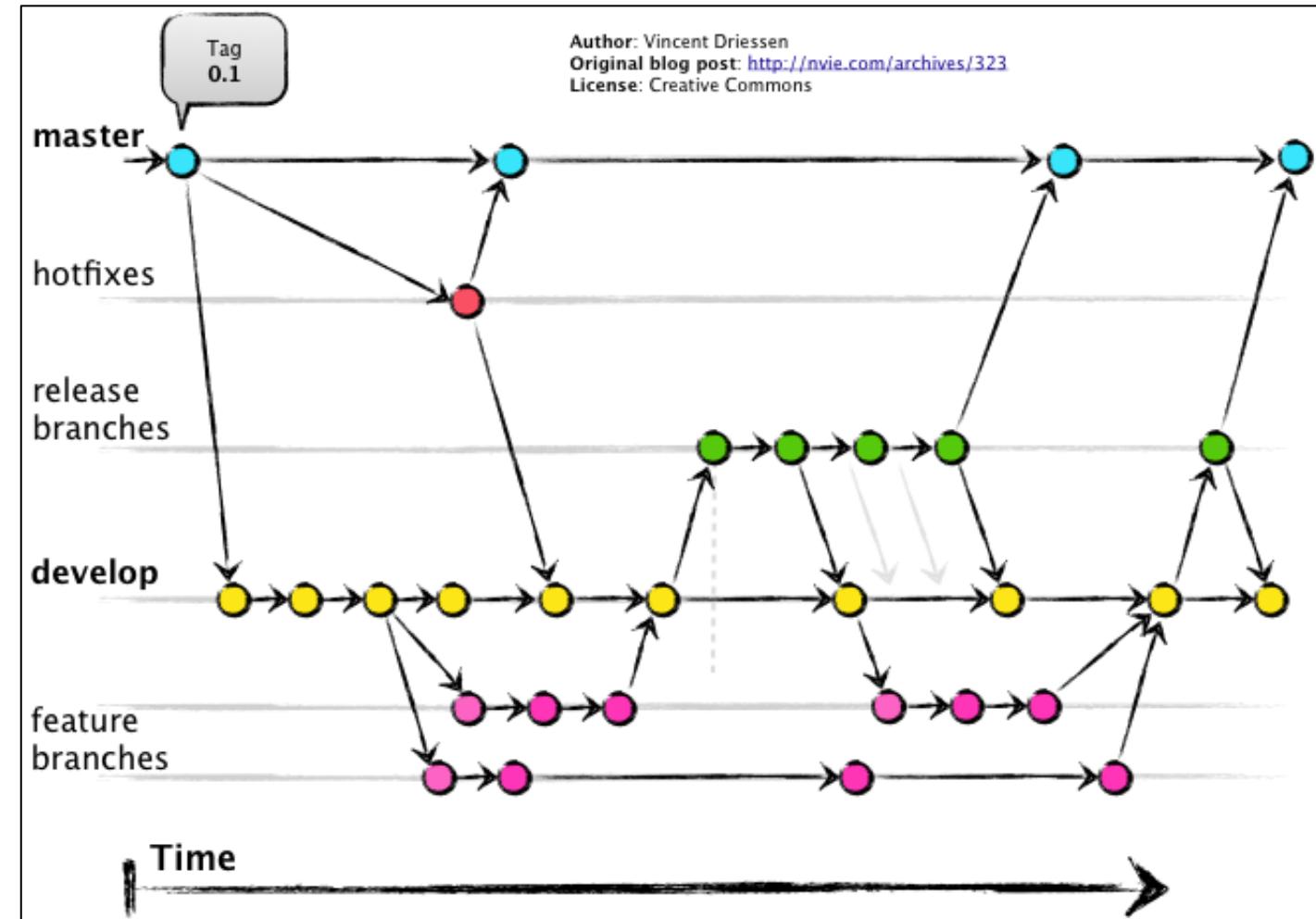
Add Commits: Make changes and commit them to your branch.

Open a Pull Request: Start a discussion about your commits, reviewing code in the pull request.

Discuss and Review: Share your pull request with teammates for feedback.

Deploy: Test your changes in a production environment.

Merge: Once your changes are verified, merge them into the main branch.



GitHub CLI

The GitHub CLI is a **command line interface to interact with your GitHub Account.**

You can quickly perform common GitHub actions without leaving your developer environment

```
gh auth login  
gh repo create github-examples --public  
gh issue create --title "Issue title" --body "Issue body"  
gh pr review --comment -b "interesting"
```

The GitHub CLI can be installed on
Windows, Linux, and macOS

```
brew install gh
```

You can quickly install the CLI for GitHub Codespaces via Features:

```
"features": {  
  "ghcr.io/devcontainers/features/github-cli:1": {}  
}
```

Core commands

- gh auth
- gh browse
- gh codespace
- gh gist
- gh issue
- gh org
- gh pr
- gh project
- gh release
- gh repo

GitHub Actions commands

- gh cache
- gh run
- gh workflow

Additional commands

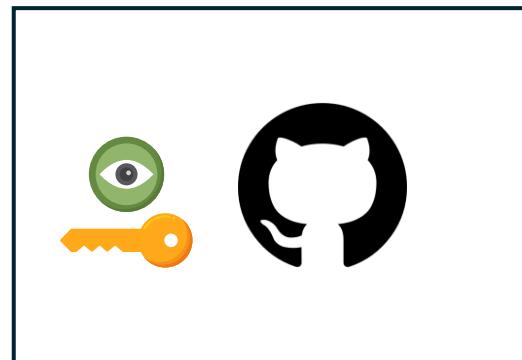
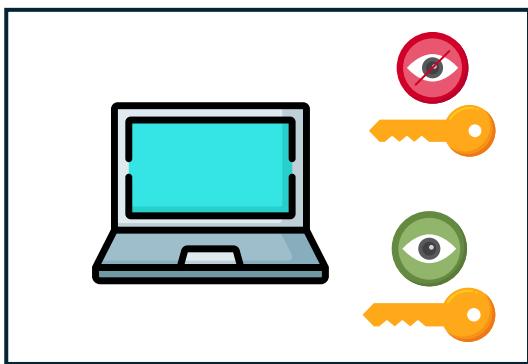
- gh alias
- gh api
- gh completion
- gh config
- gh extension
- gh gpg-key
- gh label
- gh ruleset
- gh search
- gh secret
- gh ssh-key
- gh status
- gh variable

SSH Keys

A common strategy for **authenticating to perform git operations** on your remote GitHub repo is using SSH keys

```
ssh-keygen -t rsa
```

SSH uses a public and private key to authorize remote access to servers.
ssh-keygen is a **well known command** to generate a public and private key



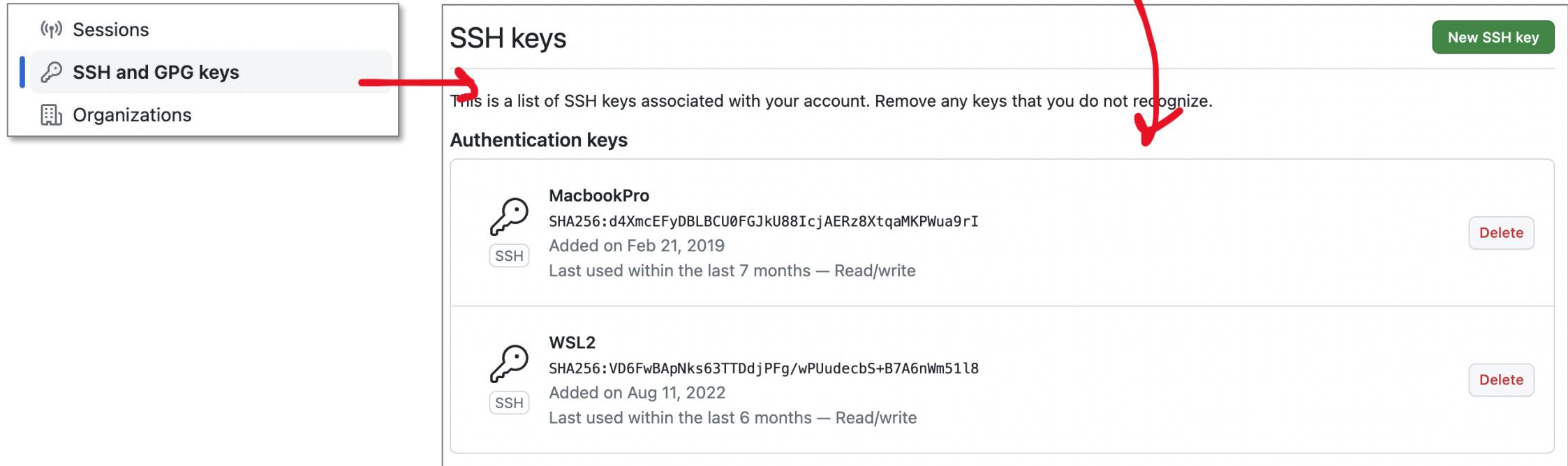
A copy of the public and private key reside on your local computer.

A copy of the public key is stored on GitHub.

- the server checks if you have the same public key
- If you do a challenge message is sent
 - Its an encrypted message with your public key
- Only a private key can decrypt the message
- Once decrypted the client signs the message also using the private key
- The server using the public key will verify the signature
- A connection is established.

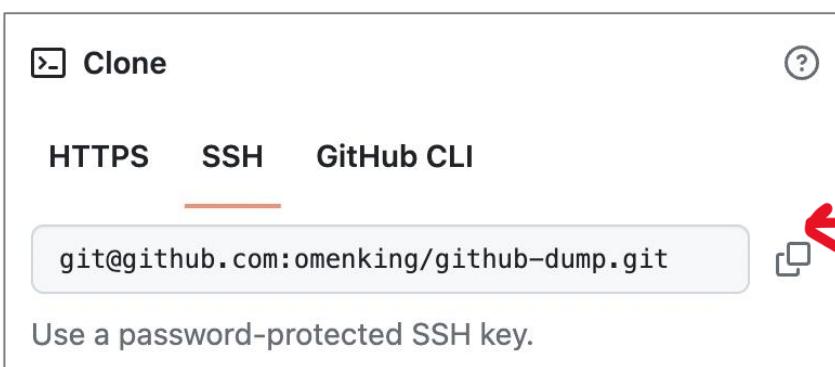
SSH Keys

Under your Account Settings of **SSH and GPG keys** you can **add your Public Keys** to your GitHub account



The screenshot shows the GitHub 'SSH keys' settings page. On the left, a sidebar menu has 'SSH and GPG keys' selected, indicated by a red arrow. The main area displays a list of SSH keys associated with the account. A red arrow points from the text 'This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.' to the 'Delete' button for the 'MacbookPro' key. The 'MacbookPro' key entry includes its SHA256 fingerprint, addition date (Feb 21, 2019), and last usage details. The 'WSL2' key entry follows a similar format. A green 'New SSH key' button is visible in the top right corner.

Key Name	SHA256 Fingerprint	Added On	Last Used	Action
MacbookPro	SHA256:d4XmcEFyDBLBCU0FGJKU88IcjAERz8XtqaMKPWua9rI	Feb 21, 2019	Within last 7 months — Read/write	Delete
WSL2	SHA256:VD6FwBApNks63TTDdjPFg/wPUudecbS+B7A6nWm51l8	Aug 11, 2022	Within last 6 months — Read/write	Delete

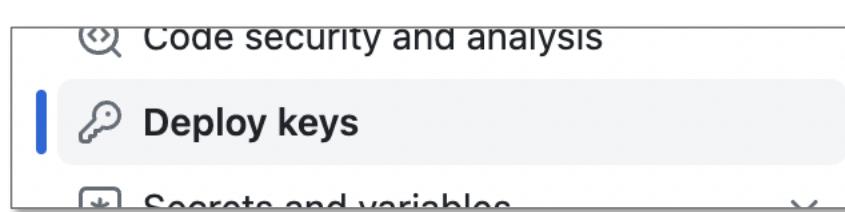


The screenshot shows the GitHub 'Clone' interface for a repository named 'omenking/github-dump'. The 'SSH' tab is selected, indicated by a red arrow. The URL 'git@github.com:omenking/github-dump.git' is displayed in a text input field, with a red arrow pointing to it. Below the input field, a note says 'Use a password-protected SSH key.'

Then you would clone your repo using the **ssh style address**.

Deploy Keys

Deploy Keys allows you **attach public keys directly to a Git Repo**



The use-case for deploy keys are for:

- **Build servers or CI/CD third-party** services that need to clone the repo so they perform a build or a deploy
- **Single Repo Access** so instead of using a shared key pair for multiple repos you have a single key pair for a single Git Repo
- Avoid Using **Personal Account Access Tokens**

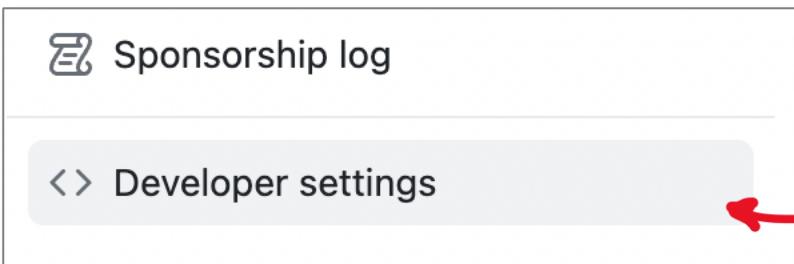
Personal Account Access Tokens

Personal Access Tokens (PAT) are an **alternative to using passwords** for authentication to GitHub when using the GitHub API or the command line

GitHub no longer lets you use passwords when interacting with the API

GitHub has two types of PATS:

- **Classic tokens**
 - They are less secure and no longer recommended for use
 - Customers with legacy systems may be still using classic token
- **Fine-grained personal access tokens**
 - granted specific permissions
 - Must have an expiry date
 - can only access specific repositories
 - can only access resources owned by a single user or organization



Personal Access Tokens is found under **Developer Settings**

Personal Account Access Tokens

When you **clone a git repo using HTTPS** you would use your personal access token to authorize the clone.

```
$ git clone https://github.com/USERNAME/REPO.git  
Username: YOUR_USERNAME  
Password: YOUR_PERSONAL_ACCESS_TOKEN
```

Instead of using gh login you could set the **GH_TOKEN**

```
export GH_TOKEN="MY_PERSONAL_ACCESS_TOKEN"  
gh pr create --title "The bug is fixed" --body "Everything works again"
```

When using **GitHub SDKs**, you need to pass your PAT to authorize API calls

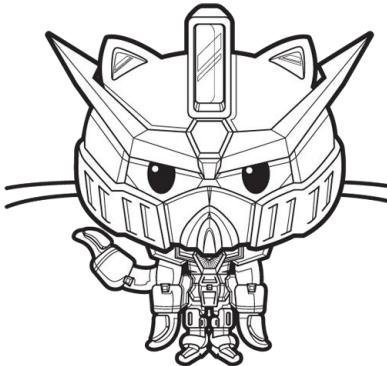
```
require 'octokit'  
octokit = Octokit::Client.new(access_token: 'YOUR-TOKEN')
```

GitHub APIs

GitHub has two APIs **REST API** and **GraphQL API**

	REST API	GraphQL API
Design Philosophy	Resource-based. Different endpoints per resource	Single endpoint, query for precise data requests
HTTP Methods	Uses GET, POST, PUT, DELETE for operations	Mainly uses POST for all operations
Data Fetching	Multiple requests for related data	Single request for complex, related data
Over-fetching/Under-fetching	Can have over-fetching or under-fetching	Precise data fetching, no over/under-fetching
Performance	Less efficient for complex systems	More efficient for complex queries
Caching	Easier due to HTTP methods	More challenging due to POST method
Learning Curve	Generally easier to learn	Steeper but offers powerful features
Flexibility	Server-driven structure	Client-driven, highly flexible
Versioning	Often requires API versioning	Less need for versioning
Ecosystem	Mature, with extensive tools	Growing, with unique tools for queries

GitHub SDKs



Octokit are the official SDKs to programmatically interact with the Git REST API

Terraform managed a git repo as Infrastructure as Code

```
resource "github_repository" "example" {
    name          = "example"
    description   = "My awesome codebase"

    visibility = "public"

    template {
        owner           = "github"
        repository     = "terraform-template-module"
        include_all_branches = true
    }
}
```

GitHub official maintains SDKs for these languages

- JavaScript / TypeScript
- C# / .NET
- Ruby
- Terraform provider

Creating a GitHub repo using **Ruby**

```
require 'octokit'

client = Octokit::Client.new(
  access_token: ENV['GH_TOKEN']
)

repo_name = "my_new_repo"
options = {
  description: "This is my new repository",
  private: false,
  has_issues: true,
  has_projects: true,
  has_wiki: true
}
client.create_repository(repo_name, options)
```

GitHub Desktop

GitHub Desktop is stand-alone application to interact with GitHub repos without the browser or via code.

Common Git and GitHub operations can be performed via the GUI for an easy-to-use experience.

GitHub desktop supports:

- MacOS
- Windows
- Linux

The screenshot shows the GitHub Desktop application interface. At the top, there are three tabs: 'Current Repository' (set to 'AWS-Examples'), 'Current Branch' (set to 'main'), and 'Fetch origin' (status: 'Never fetched'). Below these are two tabs: 'Changes' (selected) and 'History'. The main area displays a pull request titled 'refactor bash scripts, and provide etag example' by Andrew Brown. The pull request has 7 changed files. The right side shows the detailed diff for one of the files, '.gitpod.yml'. The diff highlights changes in the file, with lines 4 through 20 shown. The changes include setting up AWS-CDK, Terraform, and GPG keys, as well as updating apt-get and installing terraform. The commit message at the bottom of the diff reads: '# https://docs.aws.amazon.com/powershell/latest/reference/'.

```
refactor bash scripts, and provide etag example
Andrew Brown -O be3a698 +76 -21

7 changed files
.gitpod.yml
... @@ -4,12 +4,7 @@ tasks:
4 4 npm i -g aws-cdk
5 5 - name: terraform
6 6 before: |
7 - sudo apt-get update && sudo apt-get install -y
gnupg software-properties-common
8 - wget -O https://apt.releases.hashicorp.com/gpg
| gpg --dearmor | sudo tee /usr/share/keyrings/hashicorp-archive-keyring.gpg
9 - gpg --no-default-keyring --keyring /usr/share/keyrings/hashicorp-archive-keyring.gpg --fingerprint
10 - echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] https://apt.releases.hashicorp.com $(lsb_release -cs) main" | sudo tee /etc/apt/sources.list.d/hashicorp.list
11 - sudo apt update
12 - sudo apt-get install terraform
13 7 + ./bin/terraform_cli_install.sh
14 8 # random utilities
15 9 - name: utils
16 10 before: |
... @@ -19,23 +14,10 @@ tasks:
19 14 # https://docs.aws.amazon.com/powershell/latest/reference/
20 15 - name: powershell
```

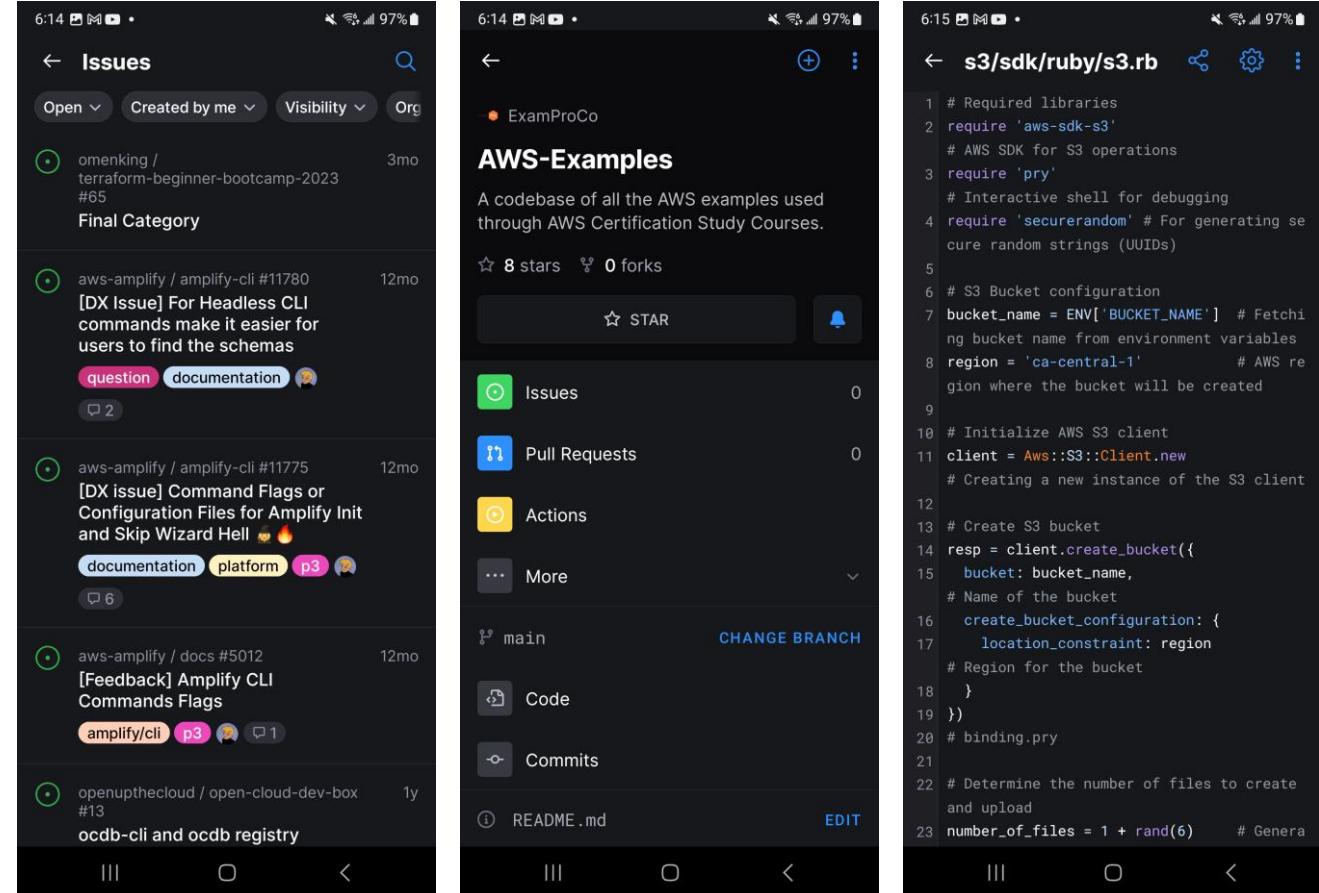
GitHub Desktop vs GitHub

Github.com		GitHub Desktop
Interface	Web-based interface accessed via browser	Standalone desktop application
Accessibility	Accessible anywhere with internet	Accessible only on installed computer
Repo Management	Directly manage repositories online	Manage local copies of repositories
Basic Git Ops.	Perform operations like fork, star, watch	Simplified interface for commit, pull, push
Collaboration	Direct collaboration tools like issues, PRs	Focuses on local repo management
Advanced Git Features	Supports advanced features via commands	Simplified experience for basic features
Integration with Tools	Integrates with various online CI/CD tools	Limited to local Git tools

GitHub Mobile

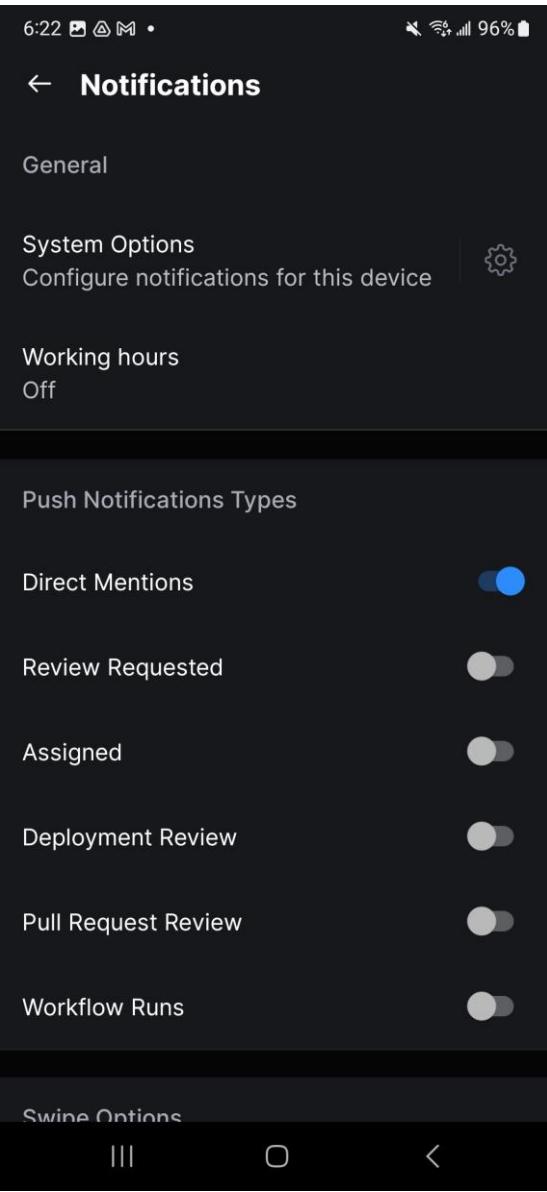
GitHub Mobile is a mobile application you can install on iOS or Android phones to perform read-only or basic GitHub repo management tasks

- Manage, triage, and clear notifications
- Read, review, and collaborate on issues and pull requests
- Edit files in pull requests
- Search for, browse, and interact with users, repositories, and organizations
- Receive a push notification when someone mentions your username
- Search through code in a specific repository
- **Secure your GitHub.com account with two-factor authentication**
- Verify your sign in attempts on unrecognized devices



GitHub Mobile can be used for UFA, which is a convenient way over other methods

GitHub Mobile Notifications

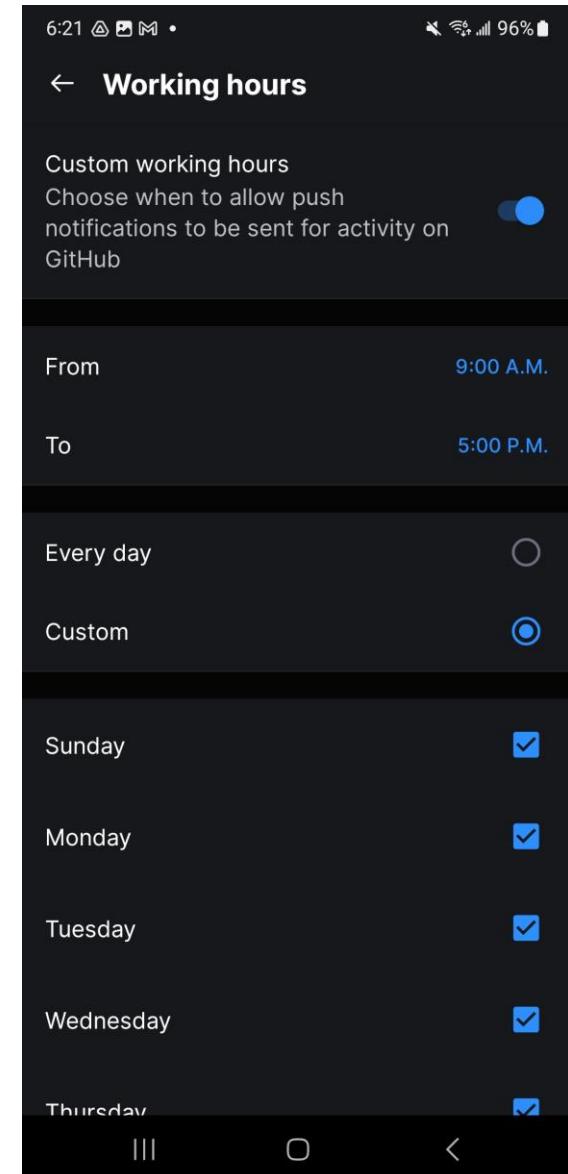


You set what **Push Notifications** you want to receive:

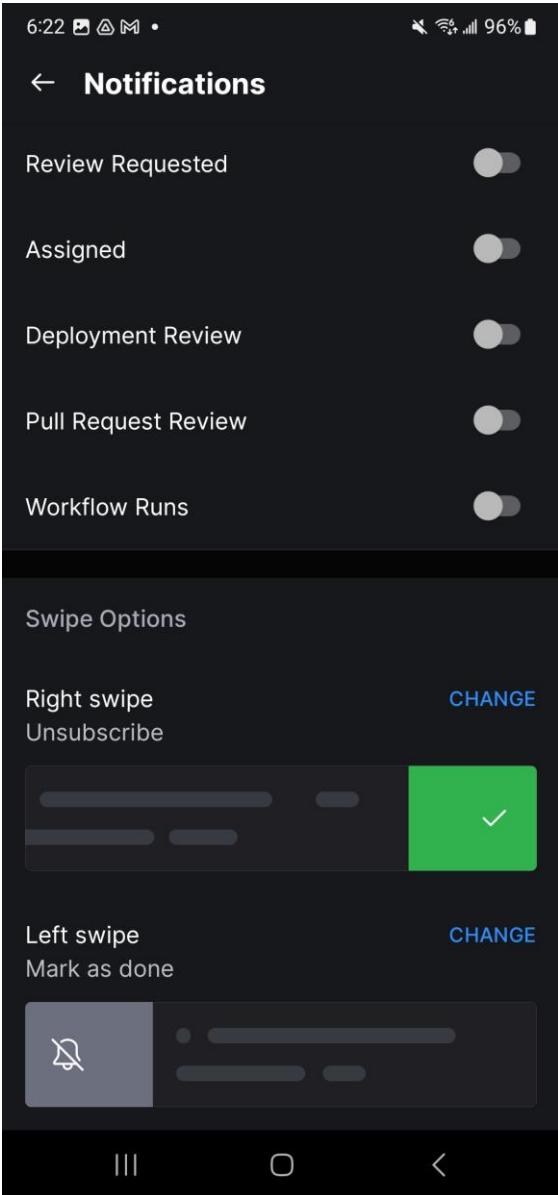
- Direct Mentions
- Review Requested
- Assigned
- Deployment Review
- Pull Request Review
- Workflow Runs



You set **Working Hours** to only receive push notifications for specific periods of time.



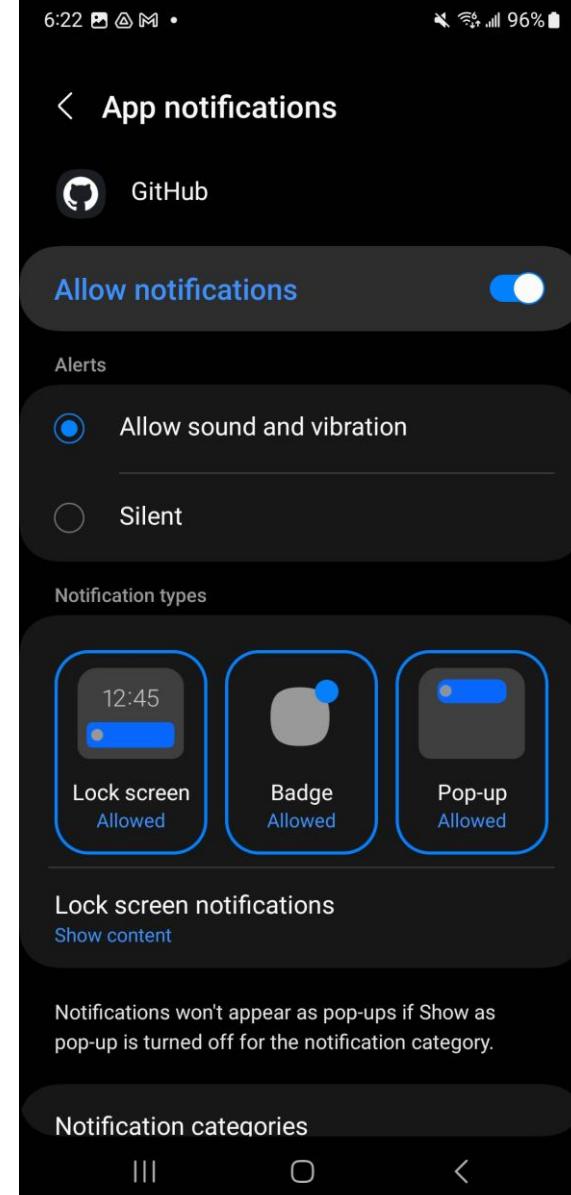
GitHub Mobile Notifications



You can set **swipe options** on notifications to perform:

- Save
- Mark as read
- Mark as done
- Unsubscribe

You can change app notifications style



Types of GitHub Accounts

There are three types of GitHub Accounts: **Personal, Organizational, Enterprise**



Personal Accounts:

These are individual accounts with a username and profile.

They can own resources like repositories and projects, and actions taken are attributed to the personal account.

They can use either GitHub Free or GitHub Pro.



Organization Accounts:

Shared accounts where multiple people collaborate on projects.

They can also own resources like repositories but are managed through individual personal accounts.

Organizations offer different roles with varying levels of access and come with security features.



Enterprise Accounts

Part of GitHub Enterprise Cloud and Server, these accounts allow for central management of multiple organizations.

They're geared towards larger setups needing centralized policy and billing management.

GitHub Personal

With a GitHub Personal account, you can have a **public profile** and also host your own public website.

Personal accounts have their own usernames:
Eg. omenking

Repo URLs for personal accounts look like:

<https://github.com/omenking/hello>



The screenshot shows a GitHub profile page for the user 'omenking'. The profile picture is a circular portrait of a man with glasses and a beard, set against a background of colorful, glowing particles. Below the picture, the name 'Andrew Brown' and the GitHub handle 'omenking' are displayed. A bio states: 'I teach people Cloud, DevOps, Data, ML, Security and Serverless.' There is a 'Edit profile' button. At the bottom, it shows '1.1k followers · 69 following'. To the right, there's a sidebar with the title 'About Me' containing the bio text. Below that is a section titled 'Free Cloud Certifications Video Courses' listing various certifications taught by the user. At the very bottom, there's a note about supporting more free cloud certification courses through a video course at ExamPro.

github.com/omenking

omenking

Overview Repositories 133 Projects Packages Stars 105

omenking / README.md

About Me

I'm Andrew Brown, I'm Canadian, dyslexic and a workaholic.

Free Cloud Certifications Video Courses

I create free cloud certification courses and I publish them to

- GCP-CDL (Google Cloud Digital Leader)
- AZ-900 (Azure Fundamentals)
- AZ-104 (Azure Administrator)
- SC-900 (Microsoft Security, Compliance, and Identity Fundamentals)
- DP-900 (Azure Data Fundamentals)
- AWS CFL-C01 (Certified Cloud Practitioner)
- AWS SAA-C01 (Solutions Architect Associate)
- AWS DVA-C01 (Developer Associate)
- AWS SOA-C01 (SysOps Administrator Associate)
- Hashicorp Terraform Associate
- Kubernetes and Cloud Native Associate
- and more!

The best way to support more free cloud certification course watch video course at [ExamPro](#)

GitHub Organization



With a GitHub Organization account can have its own **public profile** page.

It can show who belongs within the organization.

You can manage people into teams with different levels of access

Organizations have their own usernames.
Eg. ExamPro

To access a Organization repo would look like:

<https://github.com/ExamPro/AWS-Examples>

The screenshot shows the GitHub organization profile for 'ExamProCo'. The URL in the browser bar is 'github.com/ExamProCo'. The main header shows the organization's name 'ExamProCo' with a GitHub icon. Below the header, there are navigation links for Overview, Repositories (128), Projects, Packages, Teams (1), People (3), and Settings. The 'Overview' tab is selected. On the left, there is a logo for 'Exam Pro' (orange hexagon with white text). To the right of the logo, the organization's name 'ExamPro' is displayed, followed by the description 'AWS Obsessed training and certification courses'. It shows 270 followers, a location in Toronto, and contact information (website https://exampro.co and email andrew@exampro.co). Below this, there is a section for 'Popular repositories' featuring repos like 'aws-bootcamp-cruddur-2023' and 'Terraform-Associate-Labs'.

GitHub Free vs Pro

GitHub Free

- GitHub Community Support
- Dependabot alerts
- Deployment protection rules for public repositories
- Two-factor authentication enforcement
- 500 MB GitHub Packages storage
- 120 GitHub Codespaces core hours per month
- 15 GB GitHub Codespaces storage per month
- GitHub Actions features:
 - 2,000 minutes per month
 - Deployment protection rules for public repositories

GitHub Pro

- **Everything from GitHub FREE** and....
- GitHub Support via email
- 3,000 GitHub Actions minutes per month
- 2 GB GitHub Packages storage
- 180 GitHub Codespaces core hours per month
- 20 GB GitHub Codespaces storage per month
- Advanced tools and insights in private repositories:
 - Required pull request reviewers
 - Multiple pull request reviewers
 - Protected branches
 - Code owners
 - Auto-linked references
 - GitHub Pages
- Wikis
- Repository insights graphs

GitHub Organizations Plans

GitHub Organizations has three plans: **Free, Teams and *Enterprise**

FREE

- **Everything from GitHub FREE** and....
- Team access controls for managing groups
- 2,000 GitHub Actions minutes per month
- 500 MB GitHub Packages storage

Teams

- **Everything from GitHub Organizations FREE** and....
- GitHub Support via email
- 3,000 GitHub Actions minutes per month
- 2 GB GitHub Packages storage
- Advanced tools and insights in private repositories
- Required pull request reviewers
- Multiple pull request reviewers

Teams (continued...)

- Draft pull requests
- Team pull request reviewers
- Protected branches
- Code owners
- Scheduled reminders
- GitHub Pages
- Wikis
- Repository insights graphs
- The option to enable or disable GitHub Codespaces

GitHub Enterprises Deployment Options

GitHub Enterprise includes two deployment options

- GitHub Enterprise Cloud (hosted on GitHub.com)
- GitHub Enterprise Server (self-hosted)

GitHub Enterprise (both Cloud and Server)

- **Everything from GitHub Organizations Team**
- GitHub Enterprise Support
- Additional security, compliance, and deployment controls
- Authentication with SAML single sign-on
- Access provisioning with SAML or SCIM
- Deployment protection rules with GitHub Actions for private or internal repositories
- GitHub Connect
- option to purchase GitHub Advanced Security

GitHub Enterprise Cloud (specific features)

- 50,000 GitHub Actions minutes per month
- 50 GB GitHub Packages storage
- A service level agreement for 99.9% monthly uptime
- option to centrally manage policy and billing for multiple GitHub.com organizations with an enterprise account
- option to provision and manage the user accounts for your developers, by using Enterprise Managed Users

GitHub Enterprises Deployment Options

GitHub Enterprise includes two deployment options

- GitHub Enterprise Cloud (hosted on GitHub.com)
- GitHub Enterprise Server (self-hosted)

GitHub Enterprise (both Cloud and Server)

- **Everything from GitHub Organizations Team**
- GitHub Enterprise Support
- Additional security, compliance, and deployment controls
- Authentication with SAML single sign-on
- Access provisioning with SAML or SCIM
- Deployment protection rules with GitHub Actions for private or internal repositories
- GitHub Connect
- option to purchase GitHub Advanced Security

GitHub Enterprise Cloud (specific features)

- 50,000 GitHub Actions minutes per month
- 50 GB GitHub Packages storage
- A service level agreement for 99.9% monthly uptime
- option to centrally manage policy and billing for multiple GitHub.com organizations with an enterprise account
- option to provision and manage the user accounts for your developers, by using Enterprise Managed Users

Markdown

What is a Markup language?

A Markup language is a way of formatting and presenting text data in a different format.

*A common use-case for a markup language is to present data in **HTML**.*



Markdown is markup language that provides a shorthand syntax to format information into HTML.
Markdown is popular due to its easy syntax, and being readable in its raw format.

Markdown files extensions are **.md** and **.markdown**

```
# Sample Markdown

This is some basic sample markdown.

## Second Heading

* Unordered lists, and:
  1. One
  1. Two
  1. Three
* More

> Blockquote

And **bold**, *italics*, and even *italics a
later **bold***. Even ~~strikethrough~~. [A
(https://markdowntohtml.com) to somewhere.
```



HTML

```
<h1 id="sample-markdown">Sample Markdown</h1>
<p>This is some basic sample markdown.</p>
<h2 id="second-heading">Second Heading</h2>
<ul>
<li>Unordered lists, and:<ol>
<li>One</li>
<li>Two</li>
<li>Three</li>
</ol>
</li>
<li>More</li>
</ul>
<blockquote>
<p>Blockquote</p>
</blockquote>
<p>And <strong>bold</strong>, <em>italics</em>
<em>italics and later <strong>bold</strong></em>. Even
```

Sample Markdown

This is some basic sample markdown.

Second Heading

- Unordered lists, and:
 1. One
 2. Two
 3. Three
- More

Blockquote

And **bold**, *italics*, and even *italics and later bold*. Even ~~strikethrough~~. [A link](#) to somewhere.

Raw Markdown

HTML Preview

Markdown Basic Syntax

All markdown parsers supports the following markdown syntax

Headings

H1

H2

H3

Text Formatting

italicized text

bold text

Blockquote

> blockquote

Unordered List

- First item
- Second item
- Third item

Ordered List

1. First item
1. Second item
1. Third item

the numbers will change to be the correct numbers

Link

[title](<https://www.example.com>)

Image

![alt text](image.jpg)

Code Inline

`puts "hello world"'

*codeblocks use **backticks** and not single quotes.*

Backticks key is above the tab key on the keyboard

Horizontal Rule



Markdown Extended Syntax

Markdown parsers may support the following extended markdown syntax

Table

Syntax	Description
----- -----	
Header Title	
Paragraph Text	

Fenced Code Blocks

```
```rb
def hello_world
 puts "hello world"
end
````
```

You can sometimes have syntax highlighting for codeblocks by telling the codeblock the language being used.

Footnote

Here's a sentence with a footnote. [^1]

[^1]: This is the footnote.

Task List

- [x] Write the press release
- [] Update the website
- [] Contact the media

Emoji

That is so funny! :joy:

Subscript

H²O

Superscript

X²

Heading ID

My Great Heading {#custom-id}

Definition List

term
: definition

Strikethrough

~~The world is flat.~~

Highlight

I need to highlight these ==very important words==.

Text Formatting Toolbar

Every comment field on GitHub contains a **text formatting toolbar**, which allows you to format your text without learning GitHub-flavoured Markdown syntax



Add a comment

Write Preview

Add your comment here...

Markdown is supported | Paste, drop, or click to add files

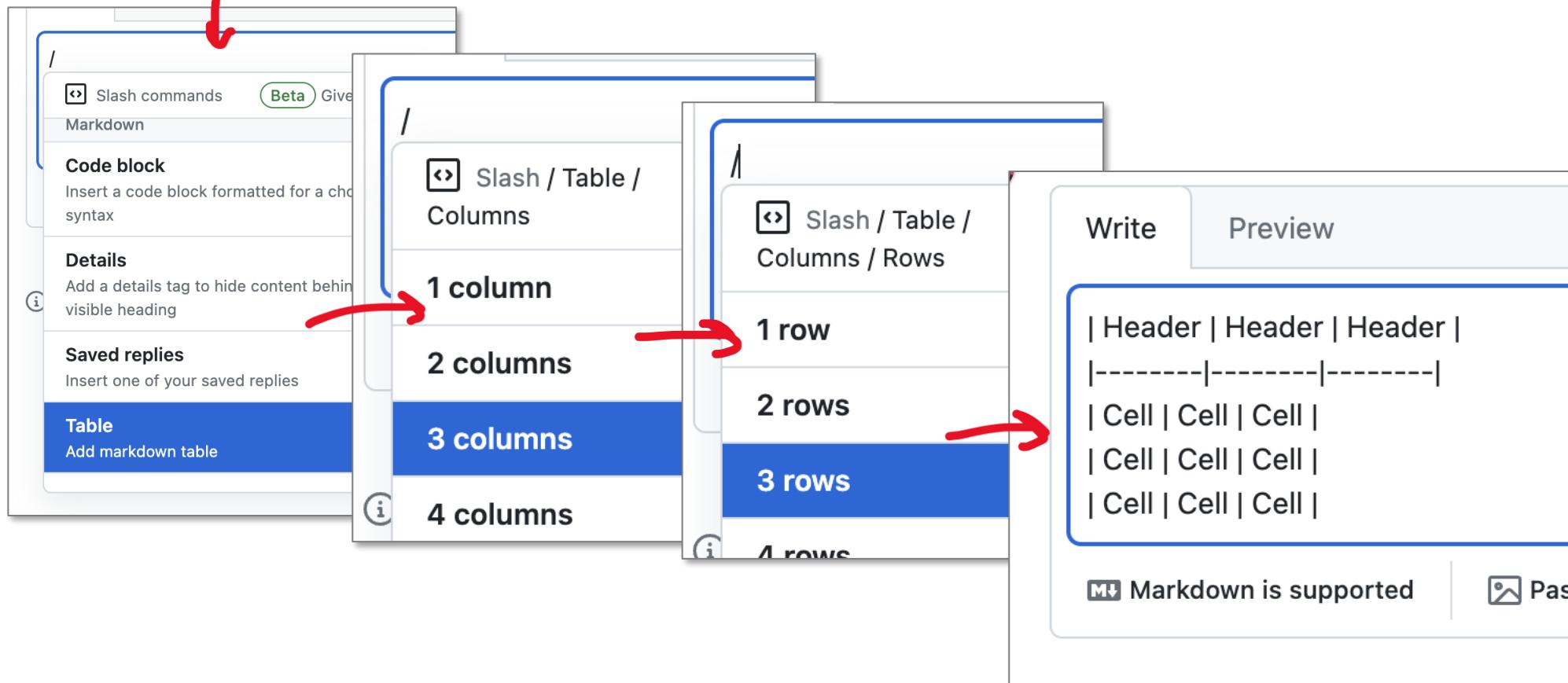
Close issue ▾ Comment

A screenshot of a GitHub comment input field. At the top left are "Write" and "Preview" buttons. Below is a text area placeholder "Add your comment here...". At the bottom are "Markdown is supported" and "Paste, drop, or click to add files" buttons. At the very bottom are "Close issue" and "Comment" buttons. A large red rectangle highlights the toolbar above the text area. The toolbar contains various icons: H, B, I, bold italic, code, horizontal line, list, link, @, reply, and a square icon.

Slash Commands

Slash Commands (within comment boxes) on GitHub.com provides convenience features such as formatting markdown.

Example of creating a markdown table using the slash command



GitHub Flavoured Markdown (gfm)

GitHub Flavoured Markdown (GFM) is the **dialect of Markdown** that is currently **supported for user content on GitHub.com and GitHub Enterprise**

GFM provides powerful functionality:

- Collapsible sections
- Embed mathematical expressions
- Embed diagrams
- Relative paths or linking to files in the same repo
- Tasks lists can be converted into Issues
- Extended formatting for tables
- Shorthands to autolink to issues, pull requests and repos
- Render code snippets from other codesbase via linking
- And more...

The detailed **GFM spec** is located here:

<https://github.github.com/gfm/>



GitHub Flavoured Markdown (gfm)

Supports embedded syntax within tables

| Command | Description |
|-------------------------|---|
| <code>git status</code> | List all <i>*new or modified*</i> files |
| <code>git diff</code> | Show file differences that haven't been staged |

Supports advanced cell alignment

| Left-aligned | Center-aligned | Right-aligned |
|-------------------------|-------------------------|-------------------------|
| <code>git status</code> | <code>git status</code> | <code>git status</code> |
| <code>git diff</code> | <code>git diff</code> | <code>git diff</code> |

| Command | Description |
|-------------------------|---|
| <code>git status</code> | List all <i>new or modified</i> files |
| <code>git diff</code> | Show file differences that haven't been staged |

| Left-aligned | Center-aligned | Right-aligned |
|-------------------------|-------------------------|-------------------------|
| <code>git status</code> | <code>git status</code> | <code>git status</code> |
| <code>git diff</code> | <code>git diff</code> | <code>git diff</code> |

GitHub Flavoured Markdown (gfm)

The **<details>** tag can be used to create collapsible sections

```
<details>  
  <summary>Tips for collapsed sections</summary>  
  
  ### You can add a header  
  
  You can add text within a collapsed section.  
  
  You can add an image or a code block, too.  
  
  ```ruby  
 puts "Hello World"
  ```  
  
</details>
```

► Tips for collapsed sections



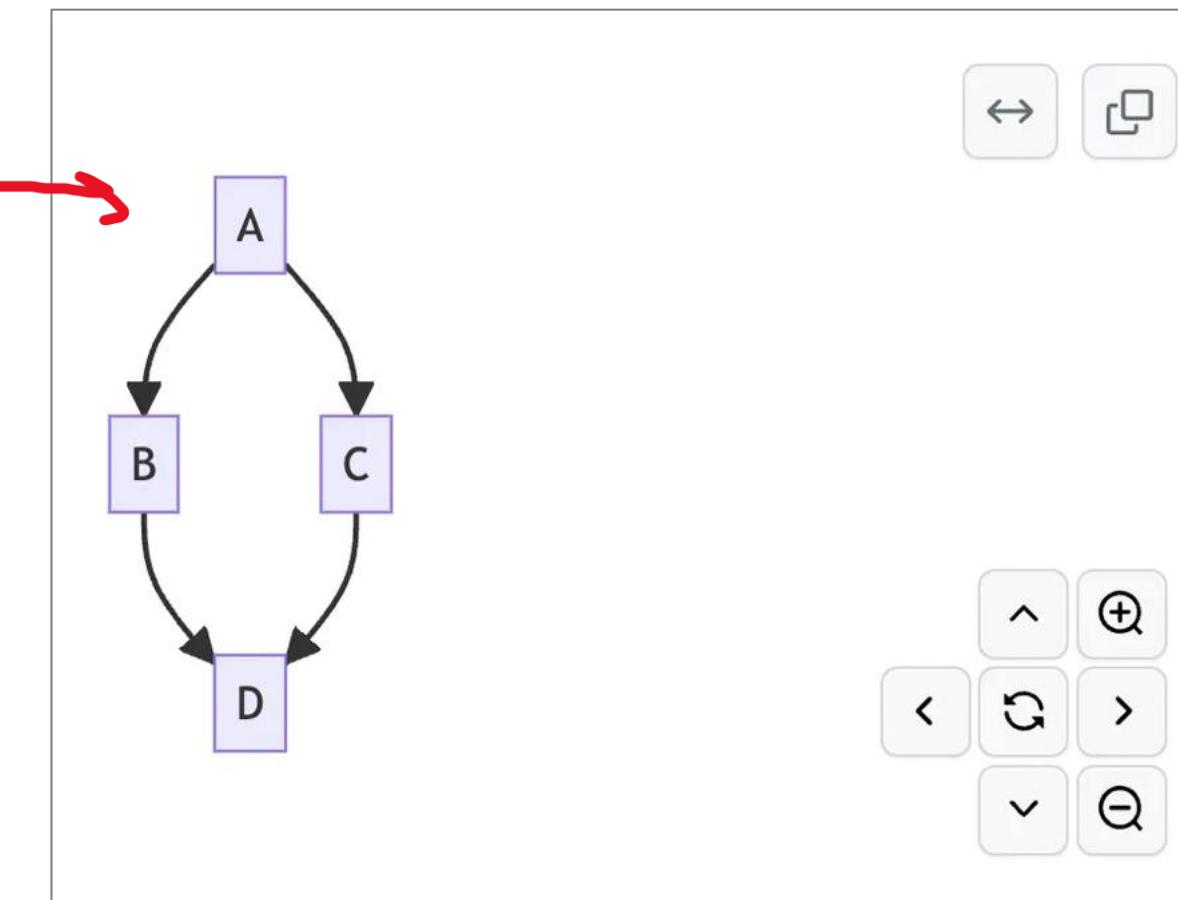
GitHub Flavoured Markdown (gfm)



Mermaid is a Markdown-inspired tool that renders text into diagrams

Here is a simple flow chart:

```
```mermaid
graph TD;
 A-->B;
 A-->C;
 B-->D;
 C-->D;
```
```

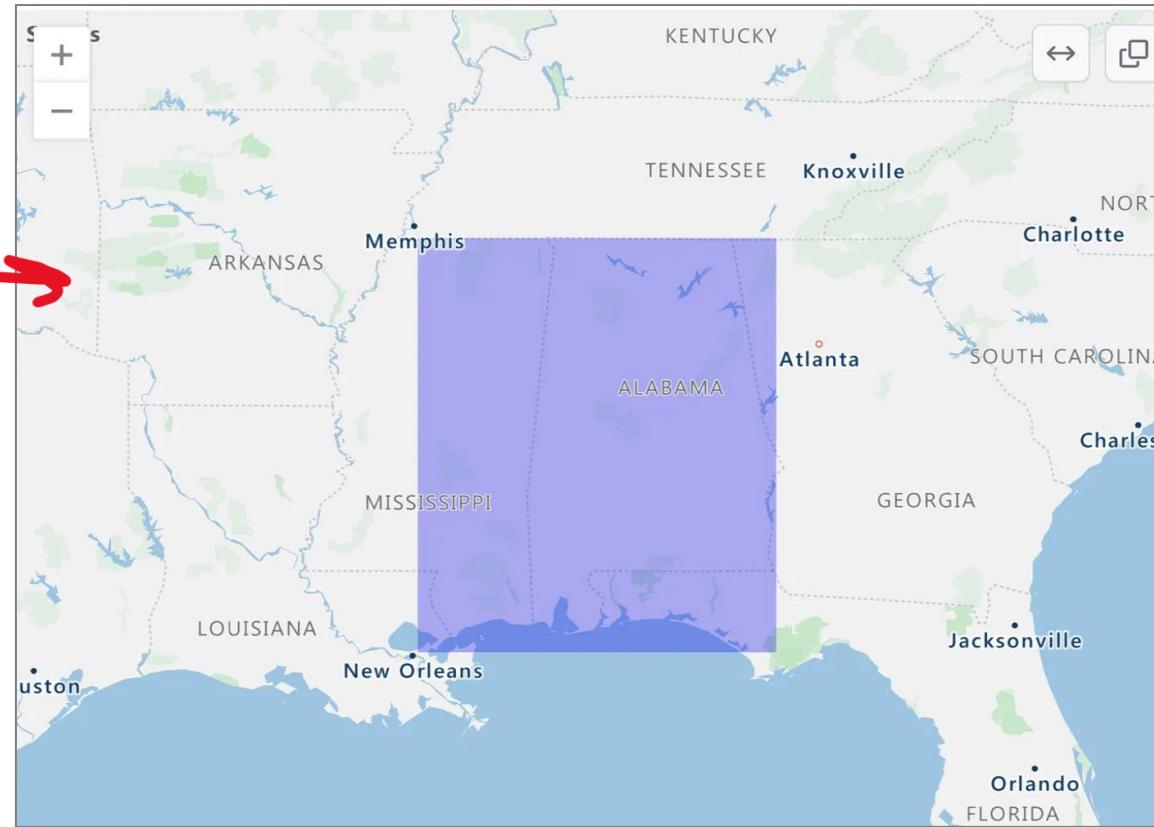


GitHub Flavoured Markdown (gfm)



GeoJSON or TopoJSON syntax to create interactive maps.

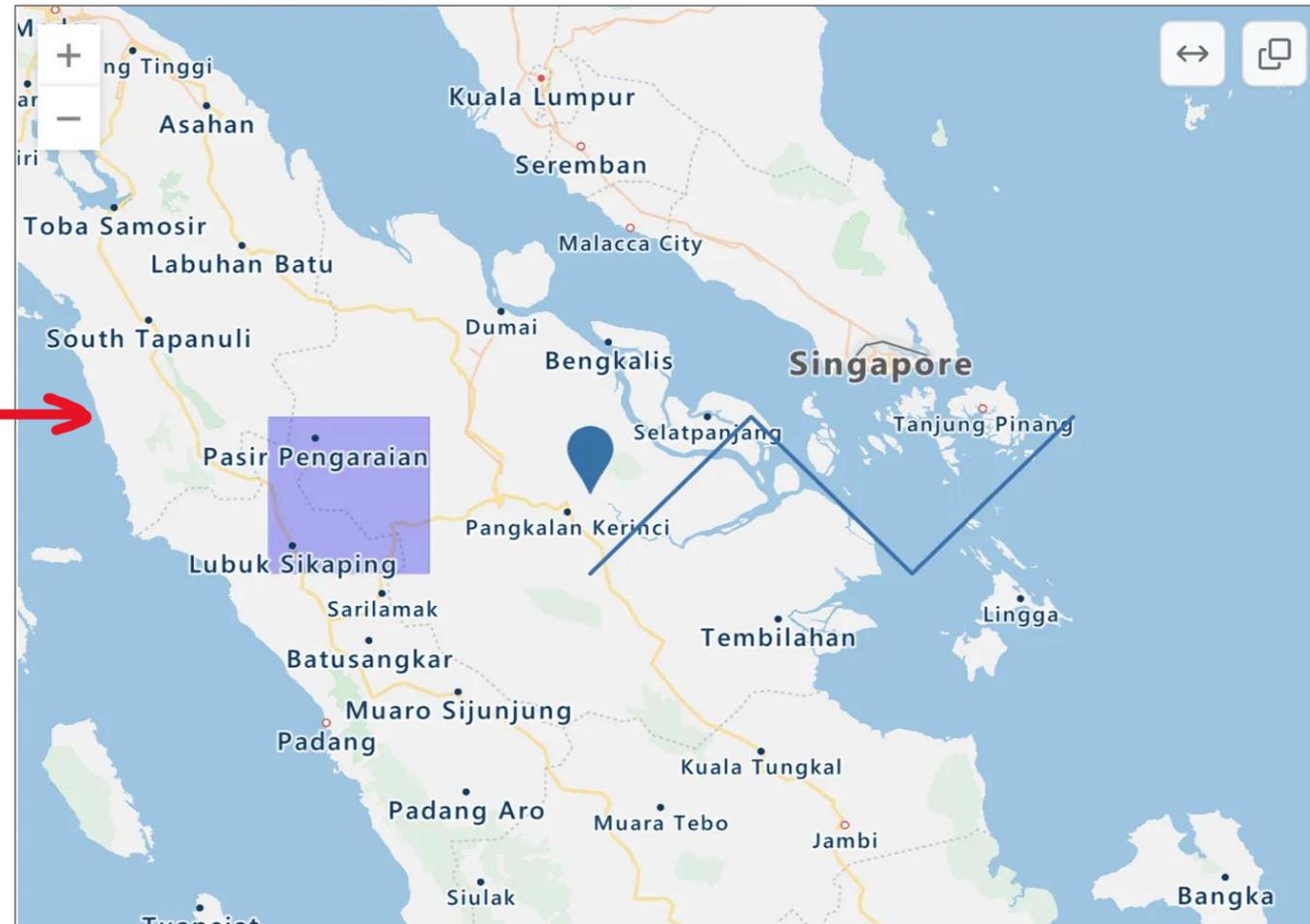
```
```geojson
{
 "type": "FeatureCollection",
 "features": [
 {
 "type": "Feature",
 "id": 1,
 "properties": {
 "ID": 0
 },
 "geometry": {
 "type": "Polygon",
 "coordinates": [
 [
 [
 [-90,35],
 [-90,30],
 [-85,30],
 [-85,35],
 [-90,35]
]
]
]
 }
 }
]
}
````
```



Geojson example

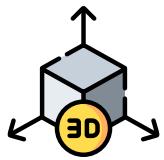
GitHub Flavoured Markdown (gfm)

```
```topojson
{
 "type": "Topology",
 "transform": {
 "scale": [0.0005000500050005, 0.0001000100010001000],
 "translate": [100, 0]
 },
 "objects": {
 "example": {
 "type": "GeometryCollection",
 "geometries": [
 {
 "type": "Point",
 "properties": {"prop0": "value0"}, "coordinates": [4000, 5000]
 },
 {
 "type": "LineString",
 "properties": {"prop0": "value0", "prop1": 0}, "arcs": [0]
 },
 {
 "type": "Polygon",
 "properties": {"prop0": "value0", "prop1": {"this": "that"}}, "arcs": [[1]]
 }
]
 }
 },
 "arcs": [[[4000, 0], [1999, 9999], [2000, -9999], [2001, 0]]]
}
````
```



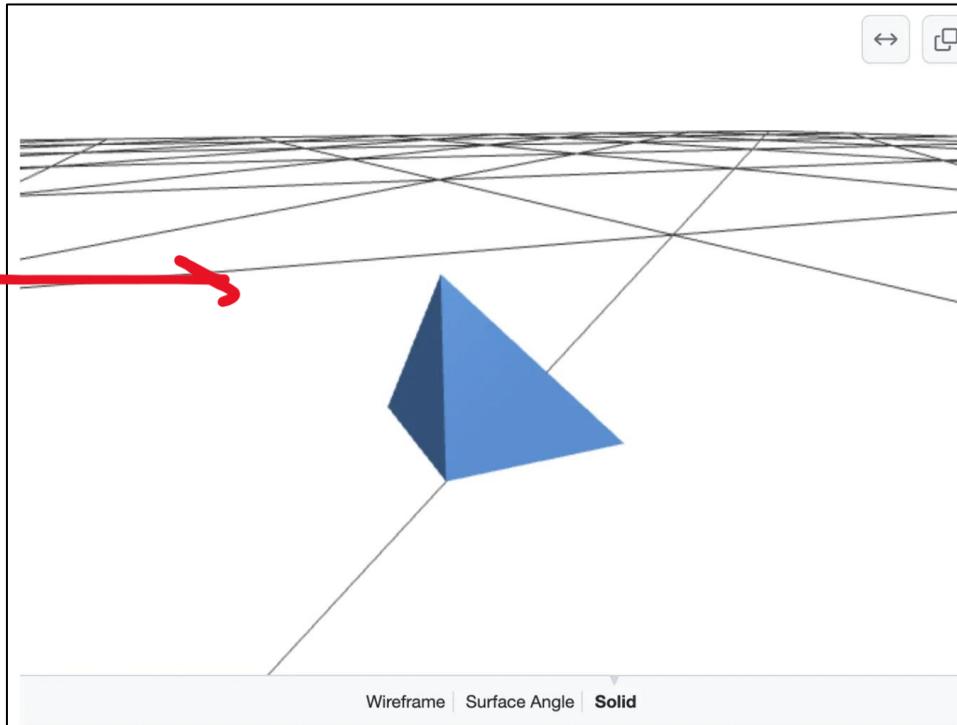
Topojson example

GitHub Flavoured Markdown (gfm)



ASCII STL syntax directly in markdown to create interactive 3D models

```
```stl
solid cube_corner
facet normal 0.0 -1.0 0.0
 outer loop
 vertex 0.0 0.0 0.0
 vertex 1.0 0.0 0.0
 vertex 0.0 0.0 1.0
 endloop
endfacet
facet normal 0.0 0.0 -1.0
 outer loop
 vertex 0.0 0.0 0.0
 vertex 0.0 1.0 0.0
 vertex 1.0 0.0 0.0
 endloop
endfacet
facet normal -1.0 0.0 0.0
 outer loop
 vertex 0.0 0.0 0.0
 vertex 0.0 0.0 1.0
 vertex 0.0 1.0 0.0
 endloop
endfacet
facet normal 0.577 0.577 0.577
 outer loop
 vertex 1.0 0.0 0.0
 vertex 0.0 1.0 0.0
 vertex 0.0 0.0 1.0
 endloop
endfacet
endsolid
````
```



GitHub Flavoured Markdown (gfm)

GFM can render mathematical expressions via Latex formatted math.

GFM uses MathJax underneath as the tool to render

Use the **\$\$** to denote start and end of latex

```
**The Cauchy-Schwarz Inequality**
$$\left( \sum_{k=1}^n a_k b_k \right)^2 \leq \left( \sum_{k=1}^n a_k^2 \right) \left( \sum_{k=1}^n b_k^2 \right)$$
```

Or use the ```match Fencing block

```
**The Cauchy-Schwarz Inequality**

```math
\left(\sum_{k=1}^n a_k b_k \right)^2 \leq \left(\sum_{k=1}^n a_k^2 \right) \left(\sum_{k=1}^n b_k^2 \right)
```
```
```
````
```

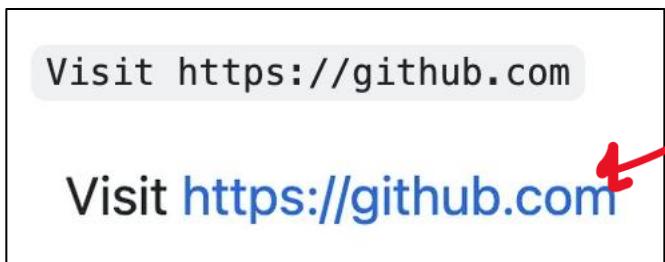
The Cauchy-Schwarz Inequality

$$\left( \sum_{k=1}^n a_k b_k \right)^2 \leq \left( \sum_{k=1}^n a_k^2 \right) \left( \sum_{k=1}^n b_k^2 \right)$$



# GitHub Flavoured Markdown (gfm)

GFM will turn raw link automatically into a link

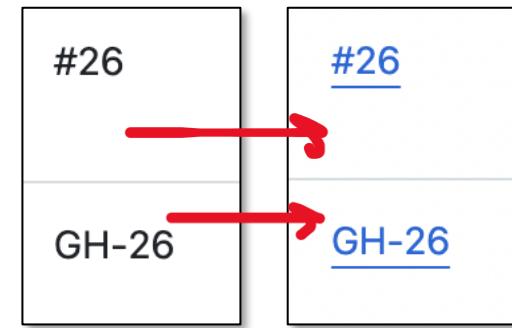


Within comments you can quickly link to repos:



Commit SHAs can be autolinked as well:

Within comments you can quickly link to issues and pull requests



<https://github.com/jlord/sheetsee.js/issues/26>

| Reference type              | Raw reference                                                                                                                                                                                | Short link                                     |
|-----------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------|
| Commit URL                  | <a href="https://github.com/jlord/sheetsee.js/commit/a5c3785ed8d6a35868bc169f07e40e889087fd2e">https://github.com/jlord/sheetsee.js/commit/<br/>a5c3785ed8d6a35868bc169f07e40e889087fd2e</a> | <a href="#">a5c3785</a>                        |
| SHA                         | a5c3785ed8d6a35868bc169f07e40e889087fd2e                                                                                                                                                     | <a href="#">a5c3785</a>                        |
| User@SHA                    | jlord@a5c3785ed8d6a35868bc169f07e40e889087fd2e                                                                                                                                               | <a href="#">jlord@a5c3785</a>                  |
| Username/<br>Repository@SHA | jlord/<br>sheetsee.js@a5c3785ed8d6a35868bc169f07e40e889087fd2e                                                                                                                               | <a href="#">jlord/<br/>sheetsee.js@a5c3785</a> |

# GitHub Flavoured Markdown (gfm)

Tasks lists can within an open issue can be created.

- You can associate tasks to specific issues (creating subtasks)
- GitHub can keep track of open or closed state of linked issue
- You can onclick turn a task and associate with an issue

- [x] #739
- [ ] <https://github.com/octo-org/octo-repo/issues/740>
- [ ] Add delight to the experience when all tasks are complete :tada:

Convert text into issues #739

Keep issue state and checkboxes in sync #740

Add delight to the experience when all tasks are complete 🎉



You can also easily reorder tasks

Convert text int

⋮  Add delight to the e



# Uploading Files

You can directly upload files by dragging them into the markdown supported textareas in GitHub

The screenshot shows a GitHub comment input interface. At the top left is a user profile picture. Next to it is the text "Add a comment". Below this is a toolbar with "Write" and "Preview" tabs, and various rich text editing icons (bold, italic, etc.). The main text area contains the following Markdown code:

```
![Uploading github-logo.png...]()
![github-logo](https://github.com/omenking/terraform-beginner-bootcamp-2023/assets/7776/345412c5-c60c-43df-b055-a94d5bb9b1d6)
```

To the left of the first line of code, the text "While its upload it will show a **placeholder**" is written in black, with a red arrow pointing from the word "placeholder" to the placeholder text "[Uploading ...]().". To the left of the second line of code, another red arrow points from the word "placeholder" to the placeholder text "[github-logo](...)".

At the bottom of the text area, there are two status indicators: "Markdown is supported" and "Paste, drop, or click to add files". On the right side of the text area, there are two small circular icons (a lightbulb and a 'G') and a "Close with comment" button with a dropdown arrow. A green "Comment" button is located at the bottom right.

GitHub will **host the file** with a directory similar to your Git Repo

- Assume files you upload for public repos are public.

# Uploading Files

The maximum file size is:

- **10MB** for images and gifs
- **10MB** for videos uploaded to a repository owned by a user or organization on a free GitHub plan
- **100MB** for videos uploaded to a repository owned by a user or organization on a paid GitHub plan
- **25MB** for all other files

Supported files:

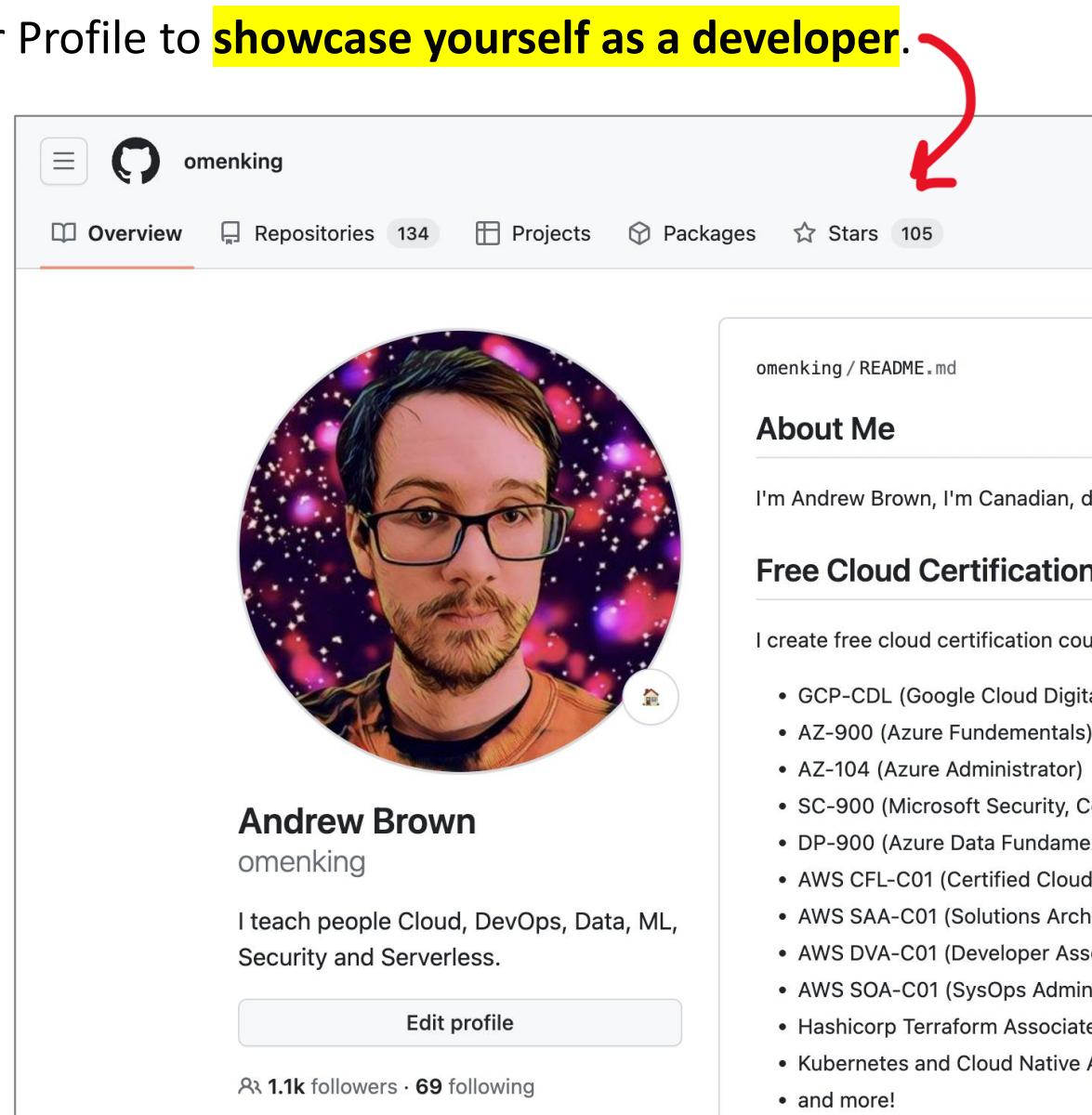
- PNG (*.png*)
- GIF (*.gif*)
- JPEG (*.jpg, .jpeg*)
- SVG (*.svg*)
- Log files (*.log*)
- Markdown files (*.md*)
- Microsoft Word (*.docx*)
- PowerPoint (*.pptx*)
- Excel (*.x/sx*) documents
- Text files (*.txt*)
- PDFs (*.pdf*)
- ZIP (*.zip, .gz, .tgz*)
- Video (*.mp4, .mov, .webm*)

# GitHub User Profile Features

For Personal Accounts you can have a public GitHub User Profile to **showcase yourself as a developer.**

Features of your GitHub user profile include:

- **Profile Picture and Bio:** You can upload a profile picture and write a short bio about yourself.
- **User Profile Readme:** a readme file rendered on your profile.
- **Repositories:** Showcases your repositories, including options to pin your favorite ones.
- **Contributions Graph:** A graph displaying your contributions over the past year
  - (commits, pull requests, issues, and code reviews).
- **Followers and Following:** Displays the number of followers you have and the number of users you are following.
- **Stars:** Shows repositories you have starred.
- **Organizations:** Lists the GitHub organizations you are a part of.
- **Contribution Activity:** Detailed list of recent activity on GitHub.

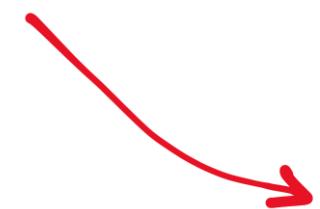


# GitHub User Profile Features

The user profile readme allows you place whatever you like on your profile.

In order to create a user profile readme you need to:

- Create a public repo with with your username.
  - eg. Omenking/omenking
- Create a README.md in the repo



omenking / omenking

<> Code    Issues    Pull requests    Actions    Projects

omenking/omenking is a special repository: its README.md will appear on your profile.

omenking / README.md in main

Edit    Preview

```
1 ## About Me
2
3 I'm Andrew Brown, I'm Canadian, dyslexic and a workaholic.
4
5
6
7 ## Free Cloud Certifications Video Courses
8
9 I create free cloud certification courses and I publish them to
```

# GitHub User Profile Features

In your account settings under Public Profile you can change:

- Display name
- Bio
- URL
- Social Account URLs

You can also specify if you are looking for work

**Jobs profile**

Available for hire

**Save jobs profile**

**Public profile**

**Name**  
Andrew Brown  
Your name may appear around GitHub where you contribute or are mentioned. You can remove it at any time.

**Public email**  
andrew@monsterboxpro.com X Remove  
You can manage verified email addresses in your [email settings](#)

**Bio**  
I teach people Cloud, DevOps, Data, ML, Security and Serverless.  
You can @mention other users and organizations to link to them.

**Pronouns**  
Don't specify

**URL**  
<https://www.exampro.co>

**Social accounts**

 <https://twitter.com/andrewbrown>

 <https://www.linkedin.com/in/andrew-wc-brown/>

 Link to social profile

  
Profile picture  
Edit

# GitHub User Profile Features

You can pin specific GitHub repos that own to best **showcase your projects**

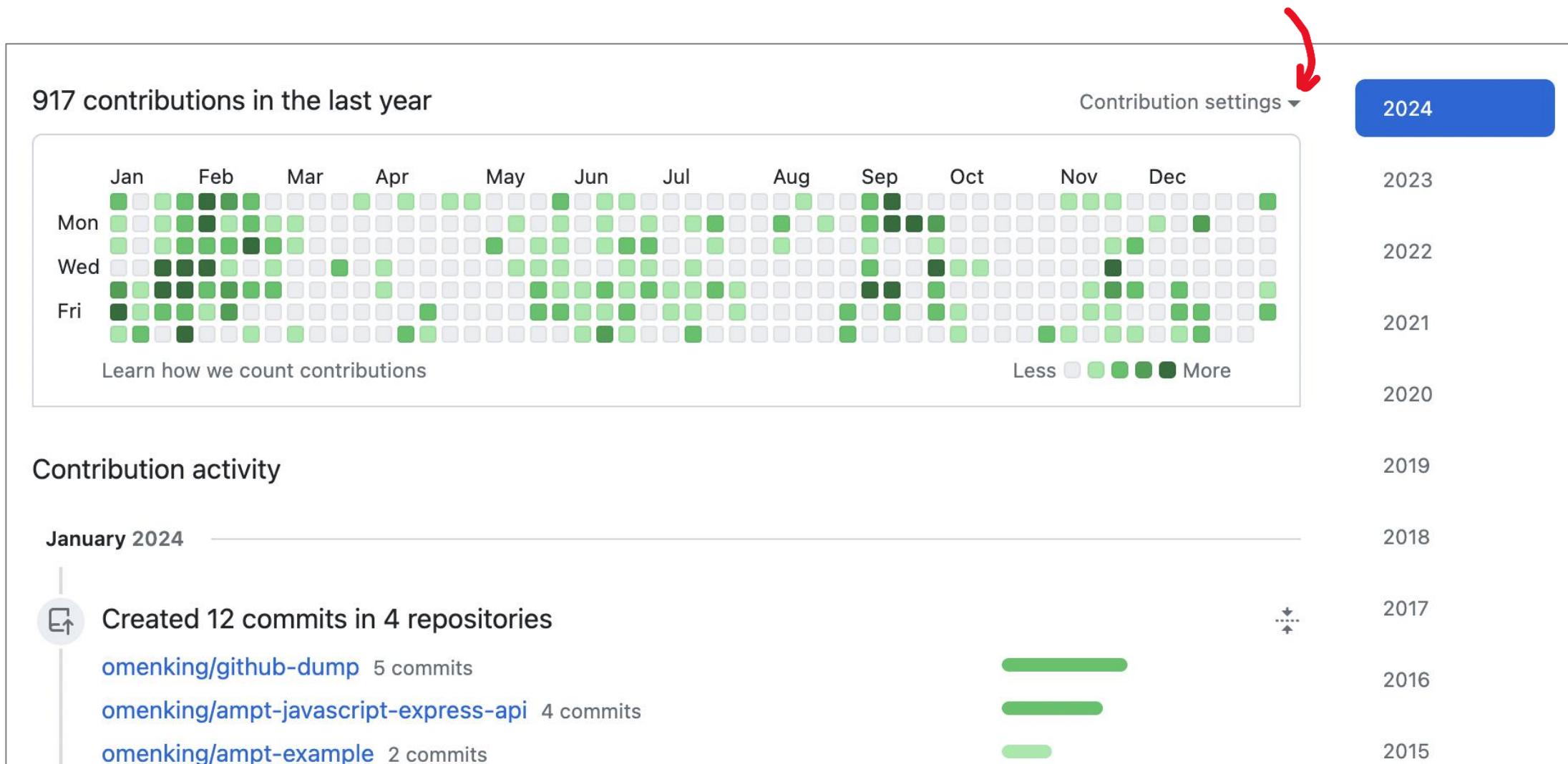
A screenshot of a GitHub user profile page. On the left, there's a section titled "Popular repositories" displaying five repository cards. From top to bottom: 1. "aws-bootcamp-cruddur-2023" (Public, 365 stars, 562 forks). 2. "terraform-beginner-bootcamp-2023" (Public, HCL, 59 stars, 18 forks). 3. "swap-n-pop" (Public, 46 stars, 7 forks). 4. "pipezania" (Public, 20 stars, 13 forks). 5. "100DaysOfCloud" (Public, Forked from sjose5/100DaysOfCloud, 15 stars, 5 forks). A red arrow points from the text "You can pin specific GitHub repos that own to best showcase your projects" towards the "Customize your pins" button at the top right of the repository list.

A screenshot of a modal window titled "Edit pinned items". The window contains the following text: "Select up to six public repositories or gists you'd like to show to anyone." Below this is a search bar labeled "Filter repositories and gists". Under "Show:", there are two checked checkboxes: "Repositories" and "Gists". It also says "6 remaining". A list of repositories follows, each with a pinning checkbox and a "Save pins" button at the bottom right. The list includes:

| Repository                       | Stars | Forks |
|----------------------------------|-------|-------|
| aws-bootcamp-cruddur-2023        | 365   | 562   |
| terraform-beginner-bootcamp-2023 | 59    | 18    |
| swap-n-pop                       | 46    | 7     |
| pipezania                        | 20    | 13    |
| omenking                         | 16    | 1     |
| 100DaysOfCloud                   | 15    | 1     |
| fyi                              | 9     | 1     |
| localstack-gitpod-template       | 9     | 1     |
| weakling                         | 8     | 1     |
| aws-cli-labs                     | 6     | 1     |
| github-docs-example              | 6     | 1     |

# GitHub User Profile Features

Contributor Activity in your profile will show **how active (for public projects) you are on GitHub.**



# README Files

Readme files are markdown files that provided documentation/instructional information.

A repo that has a **Readme.md or README file in project's root** will be rendered on the GitHub Repo page for each access.

The screenshot shows a GitHub repository page with the following details:

- Header:** README, Code of conduct, MIT license, Security, edit, three-dot menu.
- Content:**
  - Welcome to Rails**
  - What's Rails?**

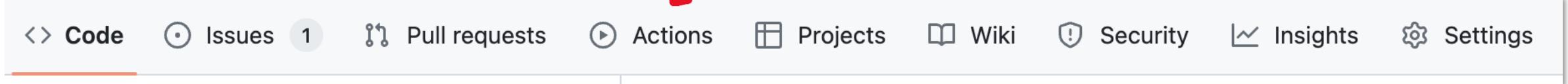
Rails is a web-application framework that includes everything needed to create data according to the [Model-View-Controller \(MVC\)](#) pattern.

Understanding the MVC pattern is key to understanding Rails. MVC divides your application into three layers: Model, View, and Controller, each with a specific responsibility.
  - Model layer**

The **Model layer** represents the domain model (such as Account, Product, Person, Post, etc.) and encapsulates the business logic specific to your application. In Rails, database-backed model classes are derived from `ActiveRecord::Base`. [Active Record](#) allows you to present the data from database rows as objects and embellish these data objects with business logic methods. Although most Rails models are backed by a database, models can also be ordinary Ruby classes, or Ruby classes that implement a set of interfaces as provided by the [Active](#)
- Sidebar:** Filter headings, Welcome to Rails, What's Rails?, Model layer, View layer, Controller layer, Frameworks and libraries, Getting Started, Contributing, License.

# Basic Repo Navigation

Within a GitHub repo you'll have a **navigation bar** to the various feature of your GitHub repo:



- **Code:** The main tab where the repository's source code, files, and folders are located.
- **Issues:** Tracks problems or ideas for the project, allowing collaboration and discussion.
- **Pull Requests:** Used for managing contributions from other users, enabling code review and discussion before merging changes.
- **Actions:** Manages continuous integration and continuous deployment (CI/CD) workflows.
- **Projects:** A board for organizing and prioritizing work, similar to kanban or task management boards.
- **Wiki:** A space for the project's documentation.
- **Security:** Features security-related resources, including security policies and vulnerability reports.
- **Insights:** Provides statistics and insights on the project's activity and contributions.
- **Settings:** Contains repository settings, including access controls, webhooks, and other configurations

# Basic Repo Navigation

Within a GitHub Repo when you click into a file you can then easily navigate all repo files

You can jump to **specific files** using the search



You can see preview of the **file content**

The screenshot shows a GitHub repository interface. On the left, there's a sidebar titled 'Files' with a dropdown set to 'main'. Below it is a search bar labeled 'Go to file'. A red box highlights the file 'cpbvt.rb' under the 'lib/cpbvt' directory, which is also selected in the sidebar. To the right, the main content area displays the file 'cpbvt.rb' from the repository 'cloud-project-bootcamp-validation-tool'. The file was last updated by 'bayko' with the commit message 'puller wip'. The code editor shows 50 lines of code, with line 1 being 'require\_relative 'cpbvt/module\_defs''. A red arrow points from the text 'You can see preview of the file content' to the code editor area.

```
1 require_relative 'cpbvt/module_defs'
2 require_relative 'cpbvt/version'
3 require_relative 'cpbvt/uploader'
4 require_relative 'cpbvt/downloader'
5 require_relative 'cpbvt/manifest'
6
7 # --- require cpbvt/payloads/azure/commands/*
8 aws_commands_path = File.join(File.dirname(__FILE__), 'cpbvt', 'payloads', 'aws')
9 Dir.glob(aws_commands_path, &method(:require))
10 # ---
11 require_relative 'cpbvt/payloads/azure/general_params'
12 require_relative 'cpbvt/payloads/azure/runner'
13 require_relative 'cpbvt/payloads/azure/command'
14
15 # --- require cpbvt/payloads/gcp/commands/*
16 aws_commands_path = File.join(File.dirname(__FILE__), 'cpbvt', 'payloads', 'gcp')
17 Dir.glob(aws_commands_path, &method(:require))
18 # ---
19 require relative 'cpbvt/payloads/gcp/general_params'
```

# Creating a new repo

When you create a repo, you choose an owner:

- Personal account
- An organization that you belong to

You need to choose an available github repo name

- Names are unique based on the scope of the owner

Your repo can either be public or private

You can quickly add a:

- Readme file
- .gitignore file
- License file

GitHub CLI can be used to create a GitHub repo

```
gh repo create my-project --public --clone
```

The screenshot shows the GitHub 'Create repository' form. It includes fields for 'Owner \*' (set to 'omenking'), 'Repository name \*' (set to 'my-new-repo' with a note that it's available), a description field (empty), and visibility options ('Public' selected). Below these are sections for initializing the repository with a README file and adding a .gitignore template. A note at the bottom indicates the repository is being created in a personal account.

Owner \* Repository name \*

omenking / my-new-repo my-new-repo is available.

Great repository names are short and memorable. Need inspiration? How about [glowing-telegram](#) ?

Description (optional)

Public Anyone on the internet can see this repository. You choose who can commit.

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

Initialize this repository with:

Add a README file This is where you can write a long description for your project. [Learn more about READMEs](#).

Add .gitignore

.gitignore template: None

Choose which files not to track from a list of templates. [Learn more about ignoring files](#).

Choose a license

License: None

A license tells others what they can and can't do with your code. [Learn more about licenses](#).

(i) You are creating a public repository in your personal account.

Create repository

# Maintaining a repo

## Repository name

github-dump

Rename

Repo names are scoped based on the personal or organizational account.



You can **change your repo name**, (but try not to because it can break external links and documentation pointing to a public repo)

## Default branch



The default branch is considered the “base” branch in your repository. Pull requests and merge commits are automatically made, unless you specify a different base branch.

main



You can **change the base branch** (default branch).  
You can rename the default branch.  
“main” is the unspoken best practice for naming your base branch

You can **opt-in-and-out of some features for your GitHub Repo**  
Features will appear in the GitHub repo navigation bar



**Wikis**  
Wikis host documentation for your repository.

**Issues**  
Issues integrate lightweight task tracking into your repository. Keep track of bugs, feature requests, and other issues by referencing them in commit messages.

[Upgrade](#) [Learn more about wikis](#)

# Maintaining a repo



The **Danger Zone** contains actions you need to think twice about because they cannot be undone.

## Change repo visibility

When you make a private repo private to public:

- The code will be visible to everyone who can visit <https://github.com>
- Anyone can fork your repository.
- All push rulesets will be disabled.
- Your changes will be published as activity.

## Disable Branch Protection Rules

Branch protect rules are strict workflow rules like disallowing anyone pushing to main.

You can disable all protections temporarily eg. For quick fixes

## Transfer ownership

When you want someone else to be the owner of the repo.

## Archive this repository

When the repo to be read-only.

## Danger Zone

### Change repository visibility

This repository is currently public.

[Change visibility](#)

### Disable branch protection rules

Disable branch protection rules enforcement and APIs

[Disable branch protection rules](#)

### Transfer ownership

Transfer this repository to another user or to an organization where you have the ability to create repositories.

[Transfer](#)

### Archive this repository

Mark this repository as archived and read-only.

[Archive this repository](#)

### Delete this repository

Once you delete a repository, there is no going back. Please be certain.

[Delete this repository](#)

## Delete this repository

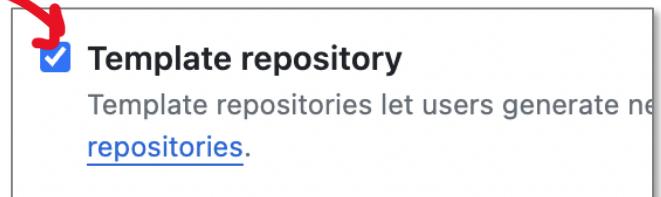
When you want to delete your repo.

Once it's gone its gone, there is not trash bin you can pull it out of.

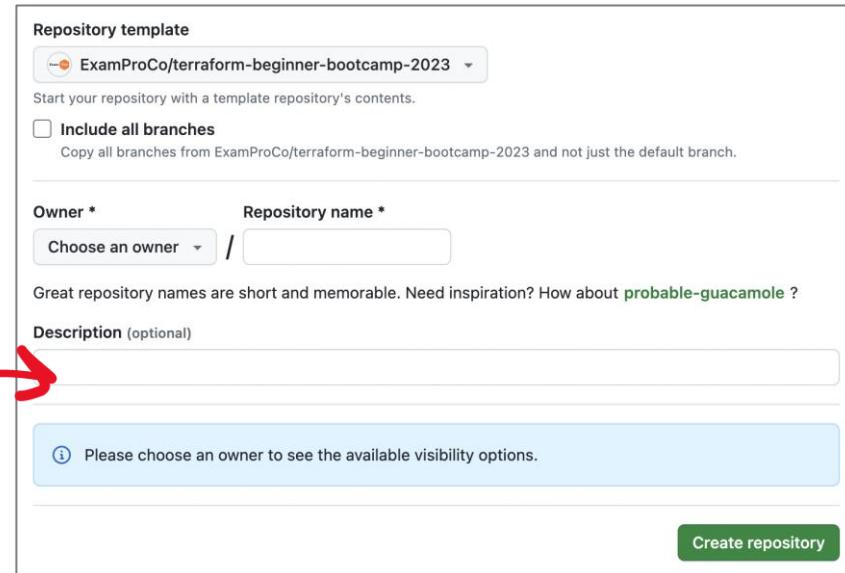
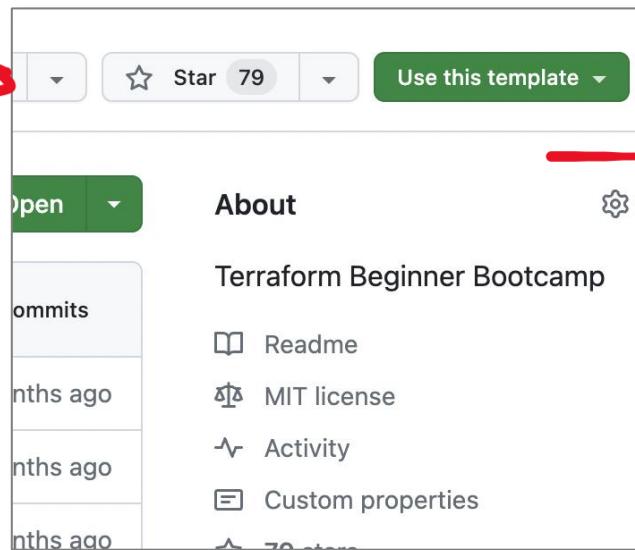
# Repo Templates

**GitHub Repo Templates** is a feature for public repos that **allow other GitHub users to make a copy of the contents of the template repo** to use as starting point of their own repo.

We set the repo to be a template

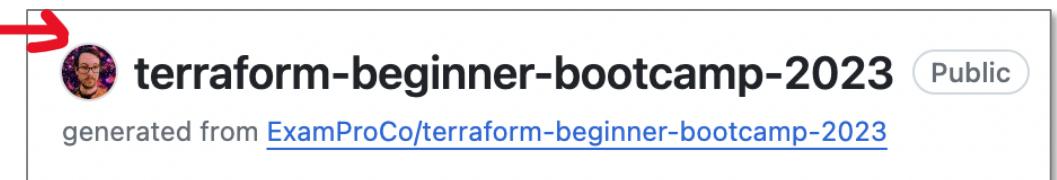


When users go to the repo, they'll see a **Use this template** button to create a new repo from the template.

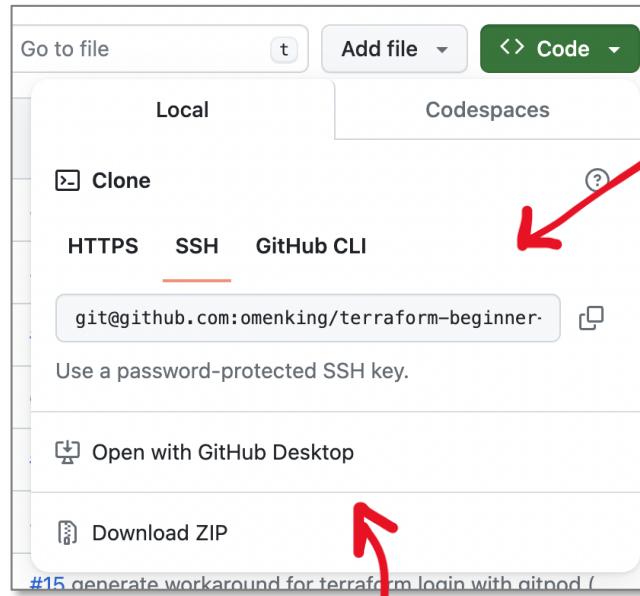


Templates are different from cloning because it's a clean repo with no previous history.

Their repo will indicate they **generated** their repo from the repo template



# Cloning a repo



You can also:

- Open in GitHub Desktop which will perform the clone for you
- Download a ZIP containing the contents of the repo

You can **clone a repo programmatically** three different ways:

## 1. HTTPS

```
git clone https://github.com/omenking/terraform-beginner-bootcamp-2023.git
```

*You supply GitHub username and password on clone, you need to set git to cache credentials*

## 2. SSH

```
git clone git@github.com:omenking/terraform-beginner-bootcamp-2023.git
```

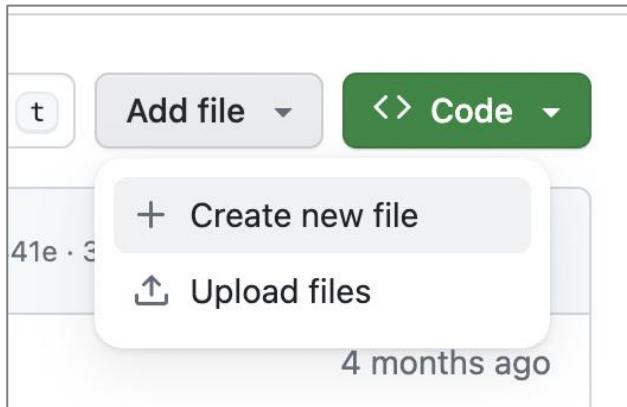
*Authorization is based on SSH keypair  
you have to upload the Public key to your GitHub account*

## 3. GitHub CLI

```
gh repo clone omenking/terraform-beginner-bootcamp-2023
```

*Authorization is based on the credentials when did a gh login*

# Adding files to a repo



Within the GitHub UI:

- Multiple Text and Binary files can be uploaded.
- New single text file can be **created**

When creating files in place, you have a basic text editor.

You can also **create nested folders by typing a forward slash in the filename.**

A screenshot of the GitHub text editor. The file path is 'terraform-beginner-bootcamp-2023 / mysub / new.rb'. The code in the editor is: 

```
1 puts "hello World"
```

 The editor has standard text editor features like 'Edit', 'Preview', 'Commit changes...', and code styling options for 'Spaces', '2', and 'No wrap'.

When adding multiple files that you also need to edit, you can also quickly use **Github.dev** at no cost

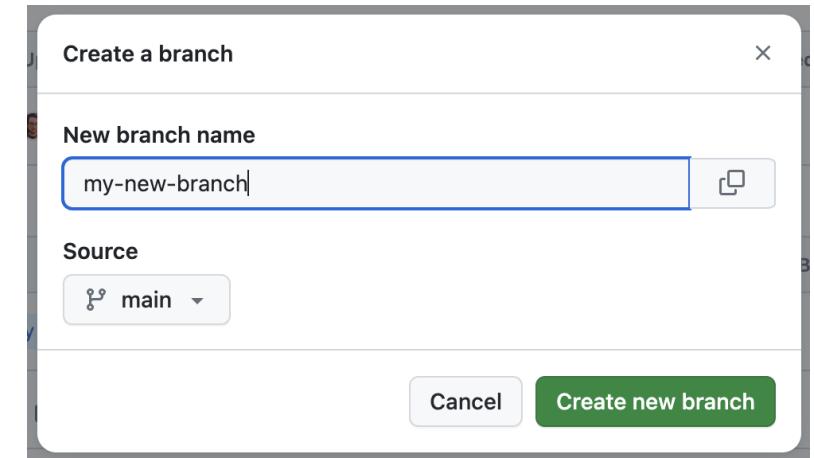
# Creating branches

Using git we can create a new branch from an existing branch.  
We then want to push upstream our branch to GitHub

```
git checkout -b staging
change files, create commits
git push -u origin staging
```

*-u is the short hand flag for –set-upstream.*

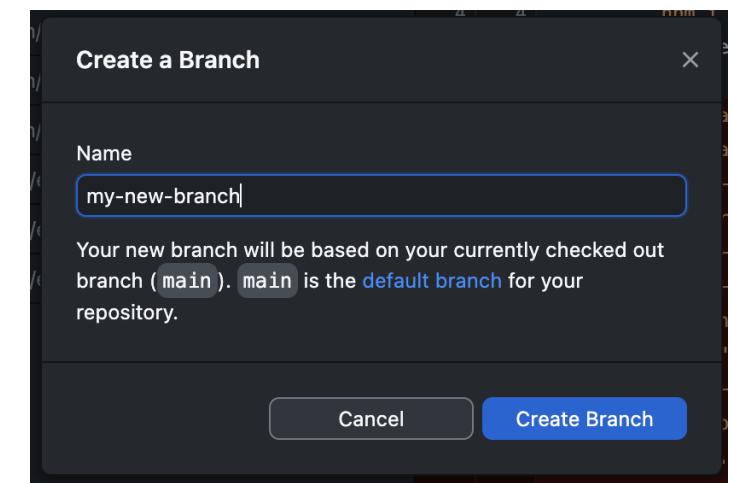
Branches can be directly created within GitHub UI.



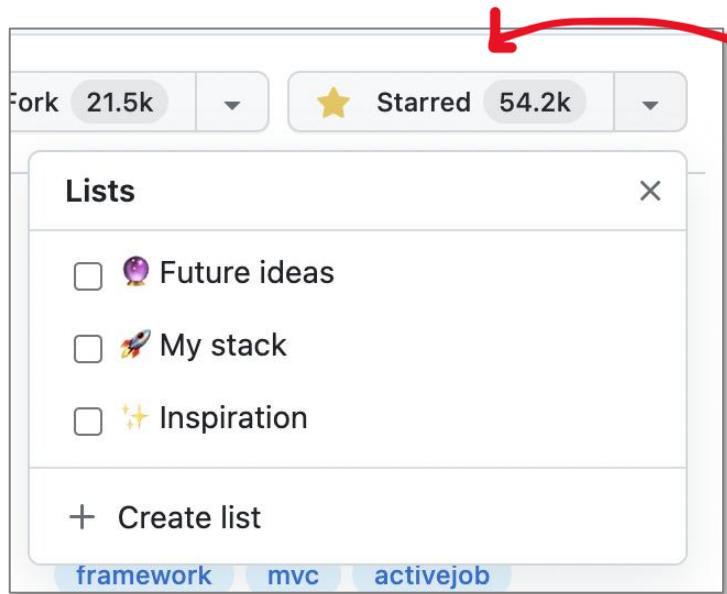
Branches can be created from Issues.  
The branch and issues will be associated/link



Branches can be created  
in GitHub Desktop



# Starring repos



Similar to bookmarking, it's a way to mark a repository as interesting or to keep track of it for reference.

Stars are public and can be seen by everyone, often used as a **measure of a project's popularity**.

You easily find pages you starred at: <https://github.com/stars>

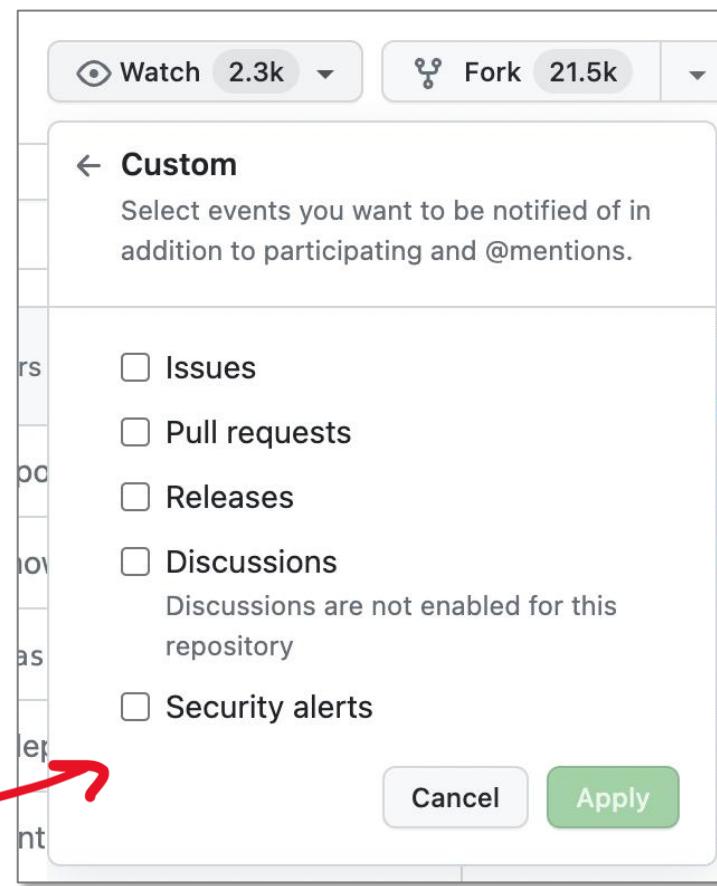
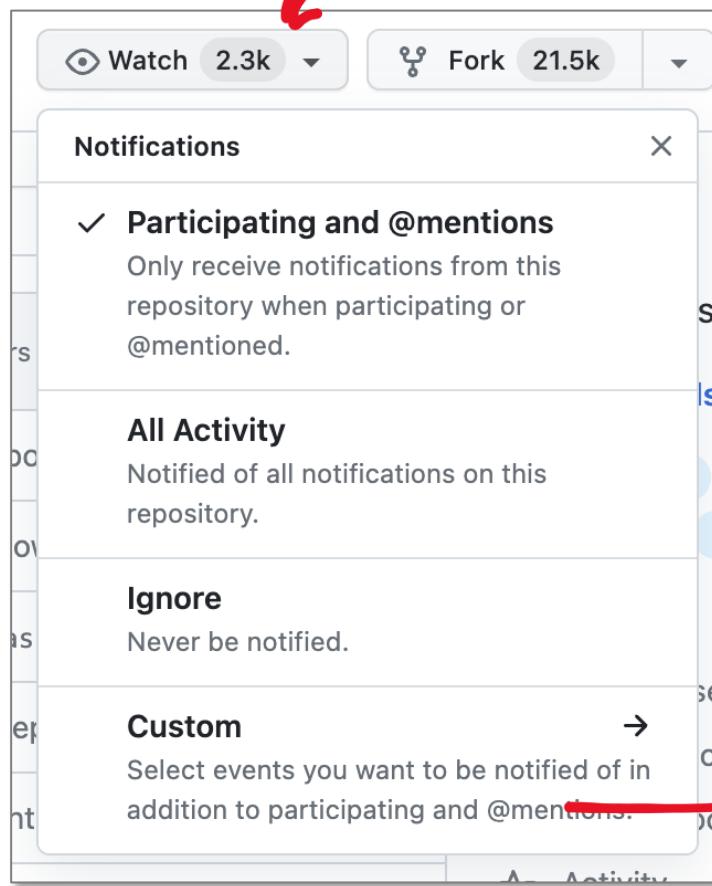
A screenshot of the GitHub 'stars' page. The URL 'github.com/stars' is in the address bar. A red arrow points to the 'Sort: Recently starred' button. The page lists 'STARRED REPOSITORIES' with two items:

- hightechu / hightechu-discord-bot**  
HighTechU - Discord Bot (Summer 2021)  
JavaScript ⭐ 1 ⚡ 3  
Starred 3 weeks ago
- Marconiusiii / OhCraps**  
Python-based Craps game for Terminal.  
Python ⭐ 6 ⚡ 1  
Starred on Jul 14, 2022

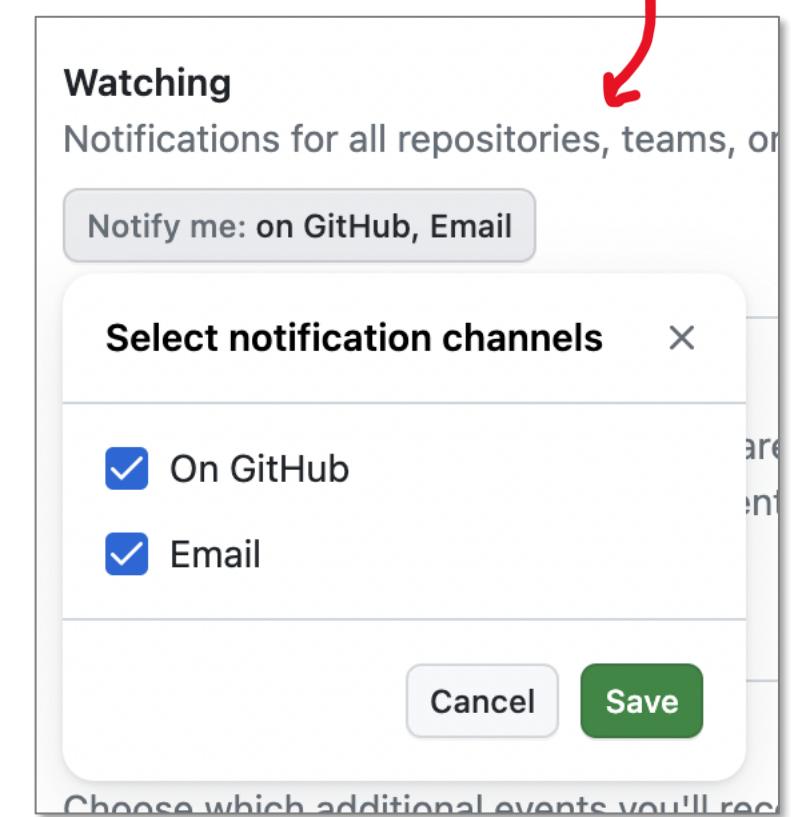
On the right, there are filters for 'All stars', 'All repositories', 'Your repositories', 'Others' repositories', 'Topics', and a 'Filter by languages' dropdown.

# Watching Repos

Watching a repo allows you to stay informed about activities occurring within a repo.



In your GitHub Account, you can specify how you want to be notified



# Feature previews

Feature previews allows you to **enable or disable features that are in beta** in your personal account

The image shows three screenshots illustrating the Feature preview interface. The first screenshot on the left shows a sidebar with options: Upgrade, Copilot, Feature preview (which has a red arrow pointing to it), and Settings. The second screenshot in the middle shows a 'Feature preview' dialog with a blue header bar containing the text 'Colorblind themes'. Below the header is a list of features: Command Palette, Rich Jupyter Notebook Diffs, Project Migration, Deployments Dashboard View, New organization repositories view, and Slash Commands. The third screenshot on the right is a detailed view of the 'Colorblind themes' section, titled 'Colorblind themes'. It displays two side-by-side interface mockups. The left mockup is for 'Light Protanopia & Deutanopia' and the right is for 'Dark Protanopia & Deutanopia'. Both show color calibration sliders for 'Light Tritanopia' and 'Dark Tritanopia'. A 'Disable' button is located in the top right corner of this panel. At the bottom of the slide, there is descriptive text about color blindness and a note about purchasing study material.

Varying by gender and geography, color blindness can affect on average 5-6% of men and up to 0.4% of women. Our previous colorblind theme has been split into two new themes:

- Light/dark Protanopia & Deutanopia for red/green color blindness.

The best way to support more free cloud certification courses is to purchase additional study material layers.

# Tags

**Tagging** is used to capture a point in history to mark a version release of your codebase

GitHub makes it **easy to explore tagged versions** of Git repo.

The image shows two screenshots of a GitHub repository page for 'rails'. The left screenshot shows the main repository page with a 'main' branch selected, 89 branches, and 516 tags. A red arrow points from the text 'GitHub makes it easy to explore tagged versions of Git repo.' down to the tags count. The right screenshot shows a dropdown menu titled 'Switch branches/tags' where the 'Tags' tab is selected. It lists several tags: v7.1.2, v7.1.1 (which is highlighted), v7.1.0, v7.1.0.rc2, and v7.1.0.rc1.

```
git tag v1.0.0 # tag a commit
git push --tags # push tag to remote
git checkout v1.0.0 # checkout codebase at tag
git tag -d v1.0.0 # delete local tag
git push --delete origin tagname # delete remote tag
```

CI/CD pipelines may be configured to deploy on the presence of a new tag on production branch

# GitHub Releases

GitHub Releases allows you to **create releases with release notes and linked assets** such as zip source or binaries for specific platforms

The screenshot illustrates the GitHub Releases interface. On the left, a sidebar shows 'Releases 109' with a 'Latest' release labeled '7.1.2' (tagged on Nov 10, 2023) and '+ 108 releases'. A red arrow points from this sidebar to the main release card. The main card displays a release titled '7.1.0' made on Oct 5, 2023, by user 'rafaelfranca'. It includes a 'Compare' button. A red arrow points from the main release card to the 'Assets' section on the right. The 'Assets' section lists two items: 'Source code (zip)' and 'Source code (tar.gz)'. Below these are reaction counts: 11 likes, 87 comments, 51 hearts, and 30 rocket boosts, with a total of 118 people reacted.

Releases 109

7.1.2 Latest  
on Nov 10, 2023

+ 108 releases

Oct 5, 2023

rafaelfranca

v7.1.0 d39db5d

Compare

7.1.0

Active Support

- Fix AS::MessagePack with ENV["RAILS...

Jonathan Hefner

- Add a new public API for broadcasting lo...

This feature existed for a while but was  
Broadcasting log allows to send log mes...  
is used by default in the development en...  
"development.log" file.

Assets 2

Source code (zip)

Source code (tar.gz)

11 87 51 30 118 people reacted

# GitHub Packages

GitHub Packages is a platform for **hosting and managing packages, including containers** and other dependencies.

Pushing Docker container to **GitHub Packages**

```
GH_USERNAME="andrew-wc-brown"
IMAGE_NAME="hello-world"
VER="1.0.0"

Sign-in to docker
echo $GH_TOKEN | docker login ghcr.io -u $GH_USERNAME --password-stdin
Tag docker container
docker tag $IMAGE_NAME:$VER ghcr.io/$GH_USERNAME/$IMAGE_NAME:$VER
Push to Docker Packages
docker push ghcr.io/$GH_USERNAME/$IMAGE_NAME:$VER
```

```
FROM alpine:latest
CMD echo "Hello World!"
```

Dockerfile

```
docker build -t hello-world .
docker run hello-world
```

Supported package registry:

- JavaScript (npm package)
- Ruby (gems)
- Java (Maven and Gradle packages)
- .NET (NuGet packages)
- Docker images

Build and run Docker image



GitHub Actions could be used to build and then publish packages to GitHub Packages

# Repo Insights



For a GitHub Repo under the Insights tab, you can gain lots of statistical graphs about the repo.

Insights contains:

- **Pulse:** Overview of recent activity (issues, pull requests).
- **Contributors:** Lists contributors and their activity stats.
- **Community Standards:** Checks for essential community health files.
- **Commits:** History of all commits in the repo.
- **Code Frequency:** Graph of code additions and deletions over time.
- **Dependency Graph:** Visualizes code dependencies.
- **Network:** Shows fork relationships and variations.
- **Forks:** Number and links to repository forks.

Repository insights is not listed as an available feature in the GitHub free tier?

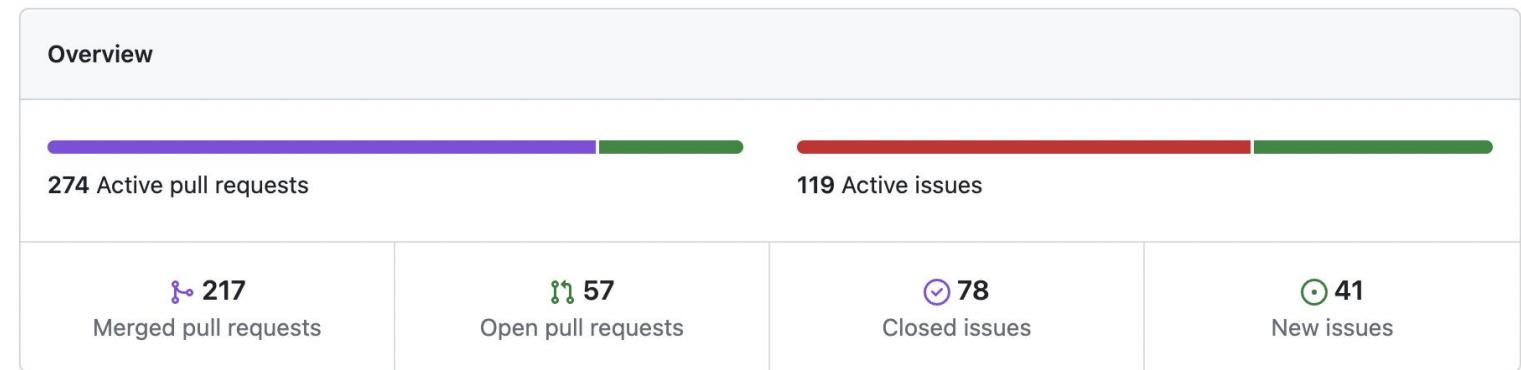
# Repo Insights

## Pulse contains:

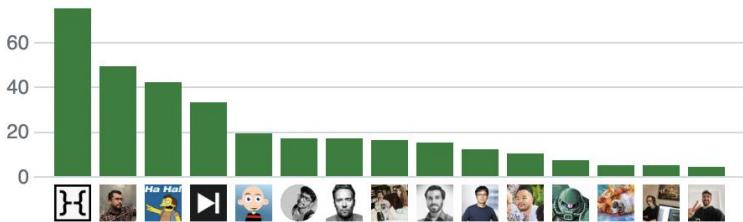
- Pull requests open/close ratio
- Issues open/close ratio
- Summary of activity
- Graph of top contributors
- List of merged pull requests
- List of issues closed
- List of issues opened
- List of unresolved conversations

December 11, 2023 – January 11, 2024

Period: 1 month ▾



Excluding merges, **80 authors** have pushed **250 commits** to main and **389 commits** to all branches. On main, **482 files** have changed and there have been **5,415 additions** and **7,844 deletions**.



217 Pull requests merged by 73 people

↳ [Fix #50713] Do not trigger loading of ActiveJob::Base in ActiveJob::TestHelper  
#50715 merged 6 hours ago

↳ Make sure attribute\_aliases is consistent

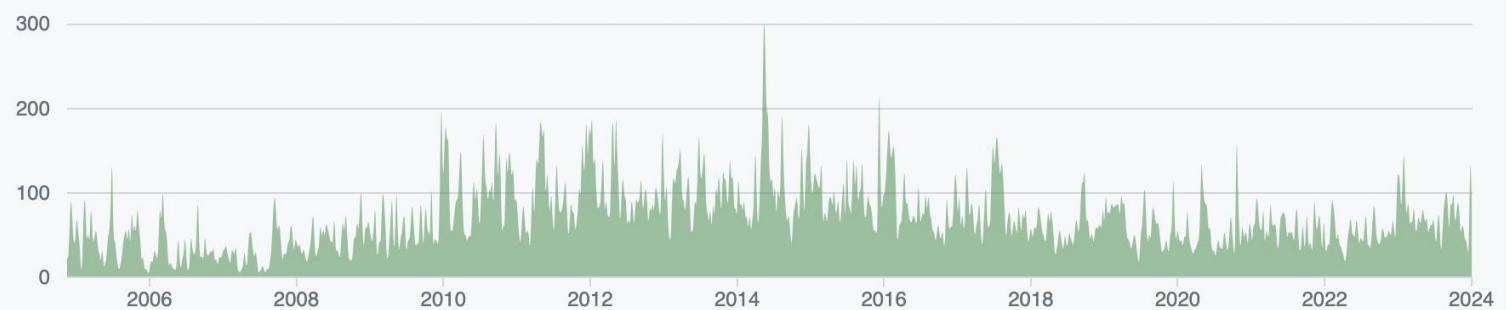
# Repo Insights

## Contributors contains:

- Graph of all commits over a time period
- Graph of specific contributors commits over a time period

Nov 21, 2004 – Jan 11, 2024

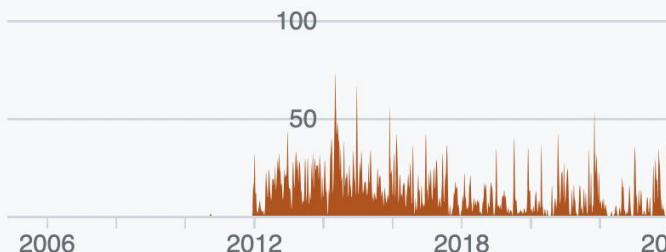
Contributions to main



rafaelfranca

8,093 commits

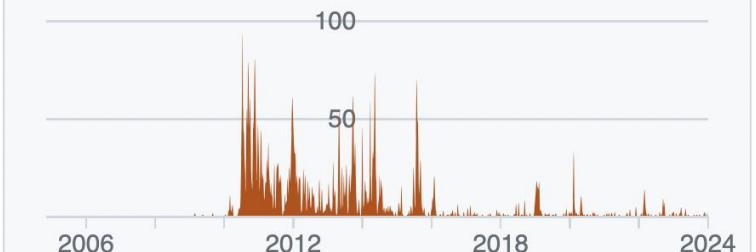
#1



tenderlove

5,298 commits

#2



# Repo Insights

## Community Standards

- A checklist of recommended community standards and how much a community profile has completed

This graph will only show for repos that have community profiles.

Community profiles are for open-source projects on GitHub.

### Community Standards

Here's how this project compares to [recommended community standards](#).

#### Checklist

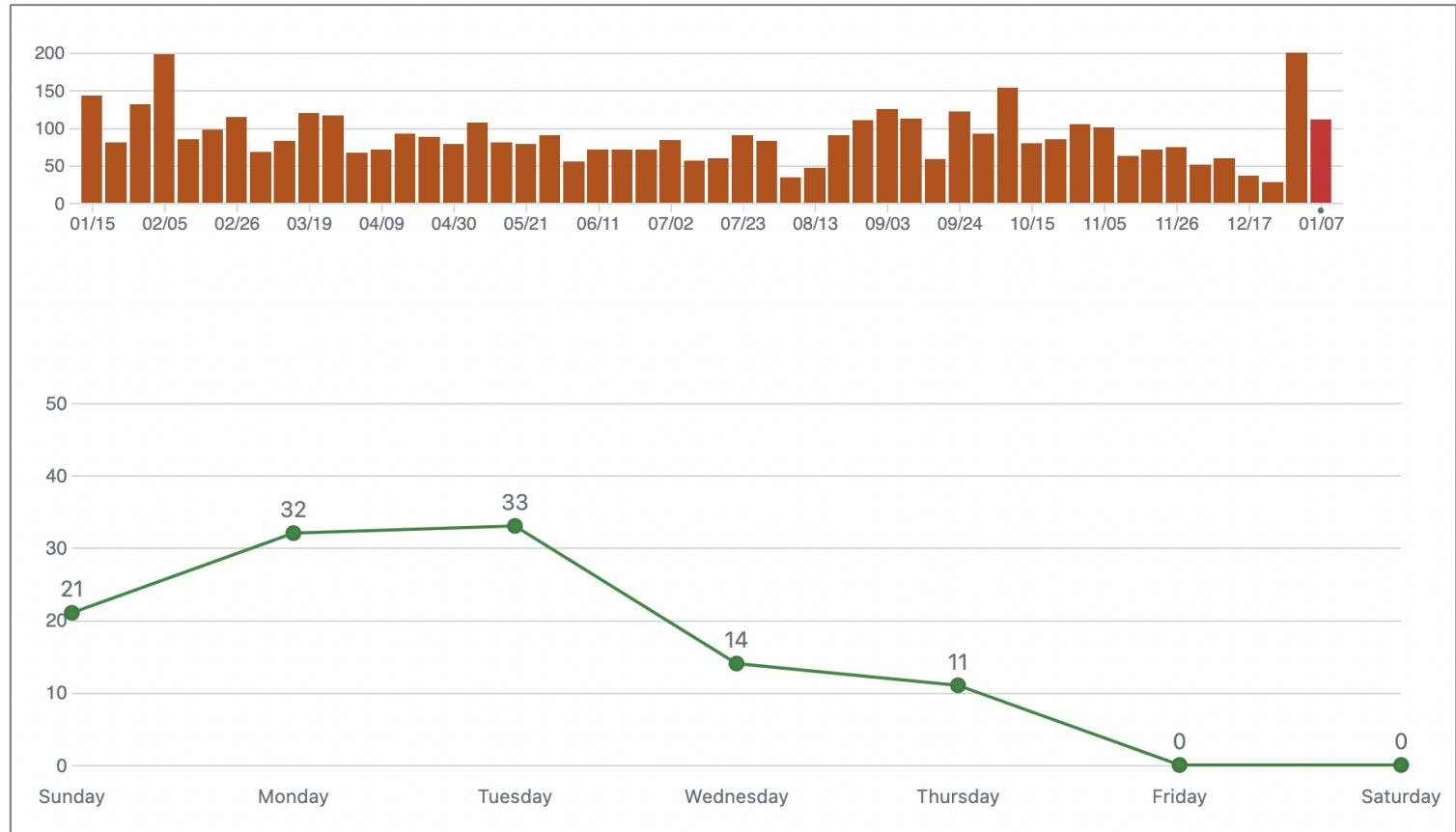
- | ✓ Description                              |
|--------------------------------------------|
| ✓ README                                   |
| ✓ Code of conduct                          |
| ✓ Contributing                             |
| ✓ License                                  |
| ✓ Security policy                          |
| ● Issue templates                          |
| ✓ Pull request template                    |
| ● Repository admins accept content reports |

What is [the community profile](#)?

# Repo Insights

## Commits

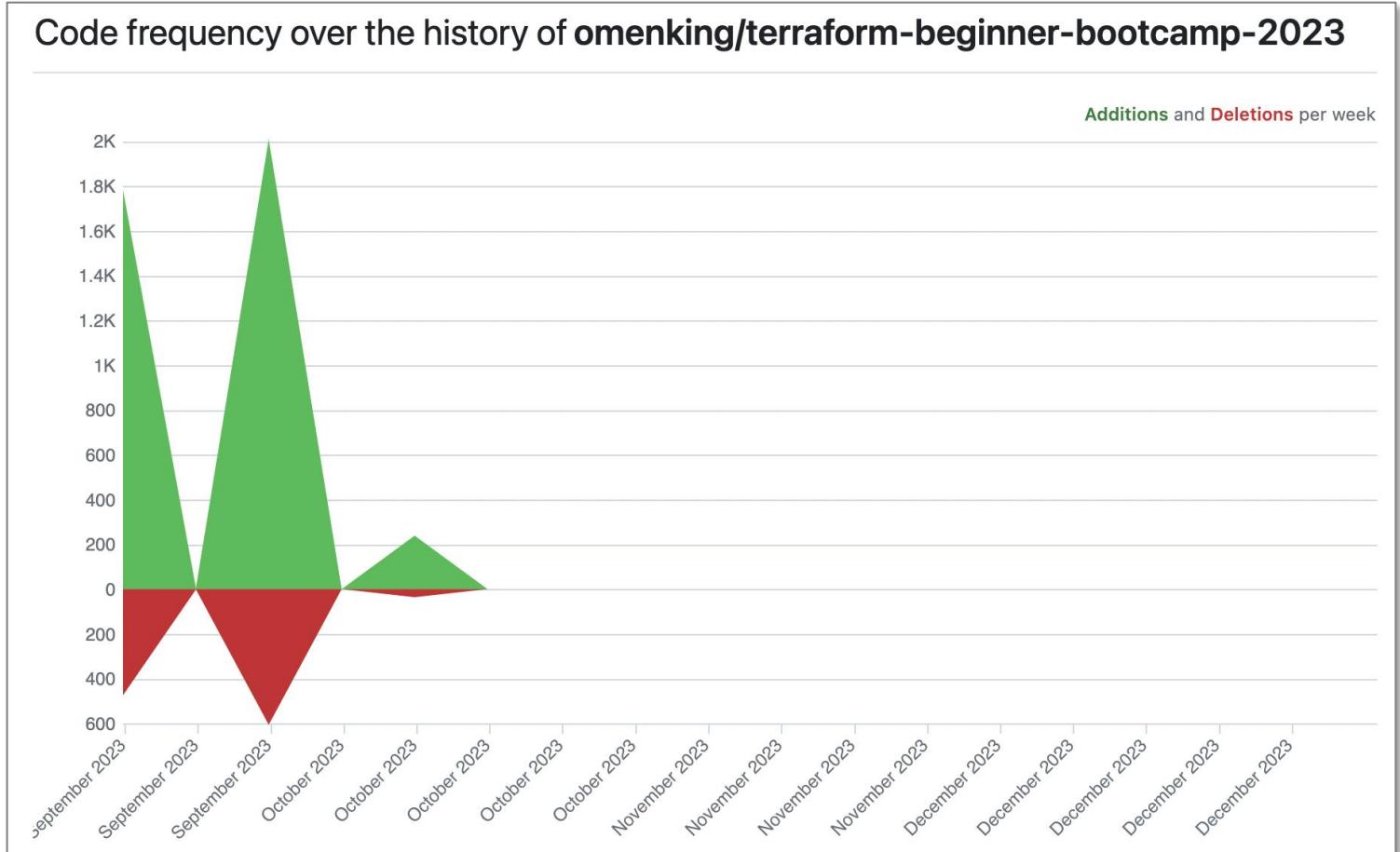
- Number of commits per week for the last 52 weeks. (for a year)
- The average number of commits by day of the week for the selected week



# Repo Insights

**Code frequency** contains:

- The number of additions and deletions of code to a report per month.



If the additions and deletions is too large the information will be able to graph.

# Repo Insights

## Dependency graph contains:

- List of dependencies
- List of dependents
- Export Software Bills of Materials (SBOMs)

Dependency graph

Dependencies   Dependents   [Export SBOM](#)

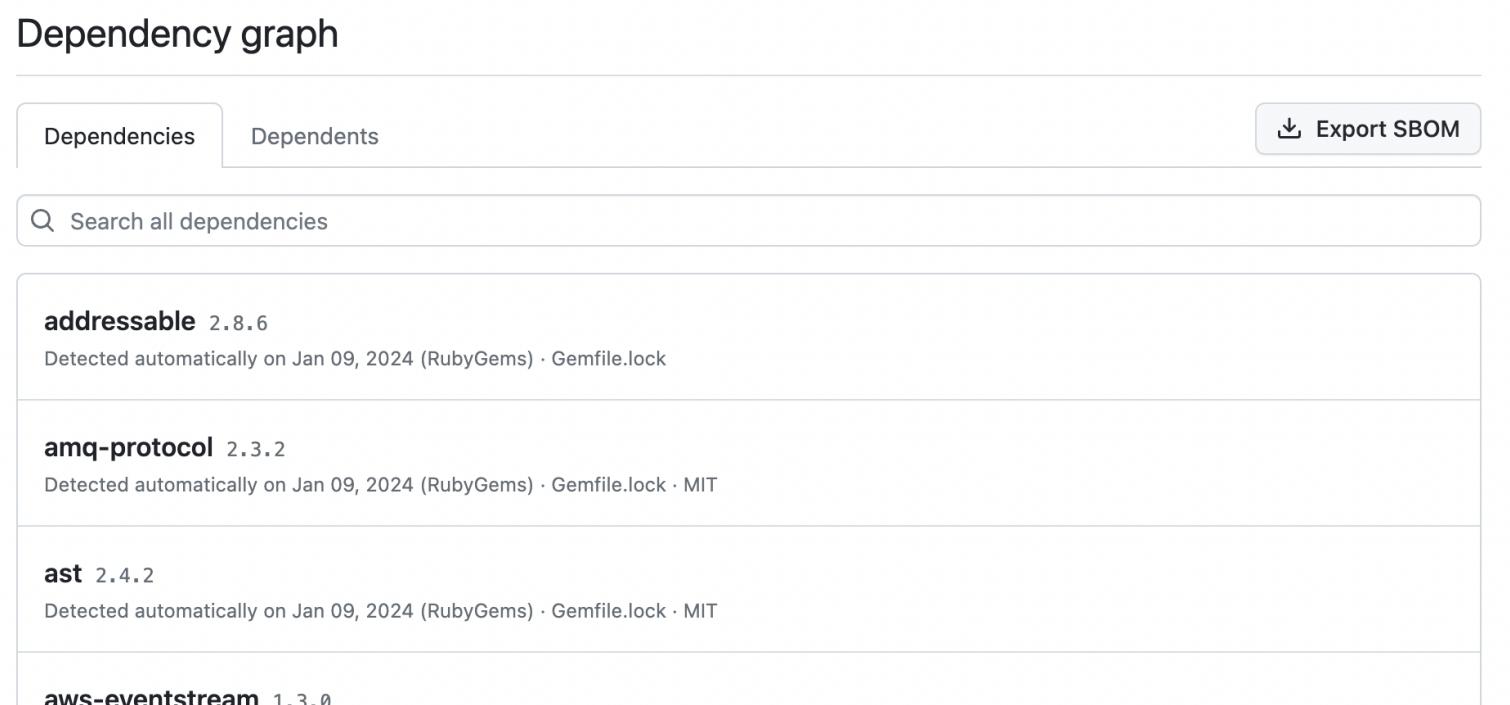
Search all dependencies

**addressable** 2.8.6  
Detected automatically on Jan 09, 2024 (RubyGems) · Gemfile.lock

**amq-protocol** 2.3.2  
Detected automatically on Jan 09, 2024 (RubyGems) · Gemfile.lock · MIT

**ast** 2.4.2  
Detected automatically on Jan 09, 2024 (RubyGems) · Gemfile.lock · MIT

**aws-eventstream** 1.3.0



Security and compliance teams increasingly request software bills of materials (SBOMs) to identify the open-source components of their software projects, assess their vulnerability to emerging threats, and verify alignment with license policies.

# Repo Insights

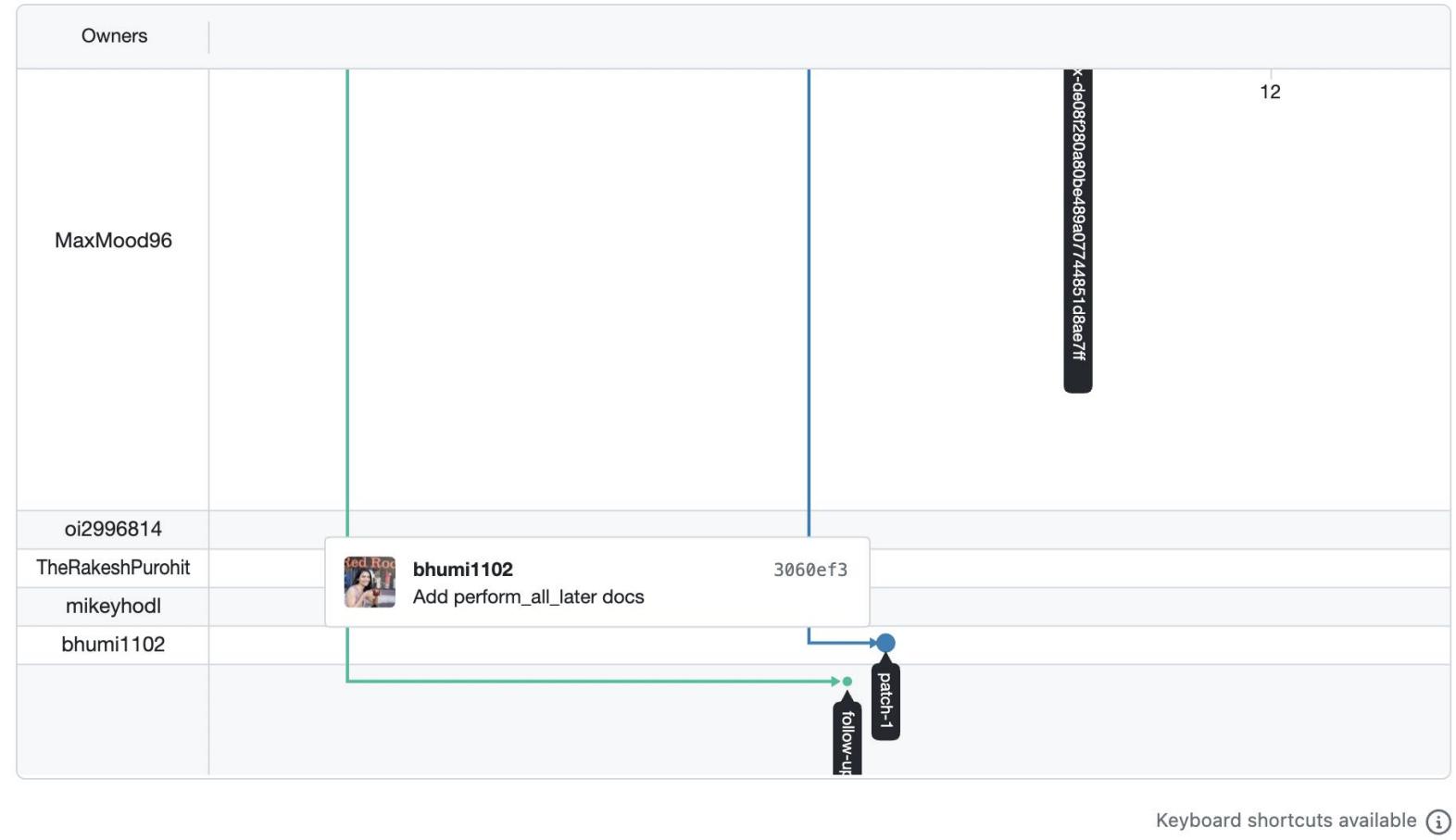
## Network contains:

- 100 most recently pushed forks
  - You can read the commits to determine the difference of these forks.

### Network graph

Timeline of the most recent commits to this repository and its network ordered by most recently pushed to.

The repository network shows the 100 most recently pushed forks.



# Repo Insights

## Forks contains:

- Contains a list of filterable forks

Switch to tree view

### Forks

Period: 2 years Repository type: Active Sort: Most starred Defaults Saved

|                                                                                                                          |      |   |   |   |                      |                      |
|--------------------------------------------------------------------------------------------------------------------------|------|---|---|---|----------------------|----------------------|
|  <a href="#">makandra / rails</a>       | ☆ 62 | 0 | 0 | 0 | Created 12 years ago | Updated 3 months ago |
|  <a href="#">rorlakr / rails-guides</a> | ☆ 37 | 0 | 4 | 0 | Created 10 years ago | Updated 7 months ago |
|  <a href="#">basecamp / rails</a>       | ☆ 29 | 0 | 0 | 1 | Created 16 years ago | Updated 6 months ago |
|  <a href="#">Shopify / rails</a>        | ☆ 14 | 0 | 0 | 1 | Created 8 years ago  | Updated 9 hours ago  |
|  <a href="#">sununa97 / rails</a>     |      |   |   |   |                      |                      |

# Creating Issues

Opening an Issue is straight forward.  
You enter a **title** and a markdown-supported  
**description**



The screenshot shows the GitHub web interface for creating a new issue. At the top, there are navigation links: Code, Issues (1), Pull requests (1), Discussions, Actions, Projects (1). Below these, a user profile picture is shown next to the text "Add a title". A text input field contains the text "I found a bug". Below this, another section titled "Add a description" is visible, featuring a rich text editor toolbar with options like Write, Preview, H, B, I, etc. A large text area for the description is empty. At the bottom right of the interface, there is a green "Submit new issue" button.

You can use the **GitHub CLI** to create Issues.



```
Create an issue interactively
$ gh issue create
Creating issue in owner/repo
? Title My new issue
? Body [(e) to launch nano, enter to skip]
http://github.com/owner/repo/issues/1
$
```

# Issues vs Discussions vs Pull Requests

## Issues

Tracking tasks, bugs, enhancements, and other actionable items.

- Often linked to code changes
- Can be linked to **PRs**

## Discussions

Facilitating conversations and Q&As about a wide range of topics related to the project.

- Categorized by topics
- Can be “converted” to issue
  - Really just creates an issue with a soft link to the issue
- **Not directly linked to code changes**

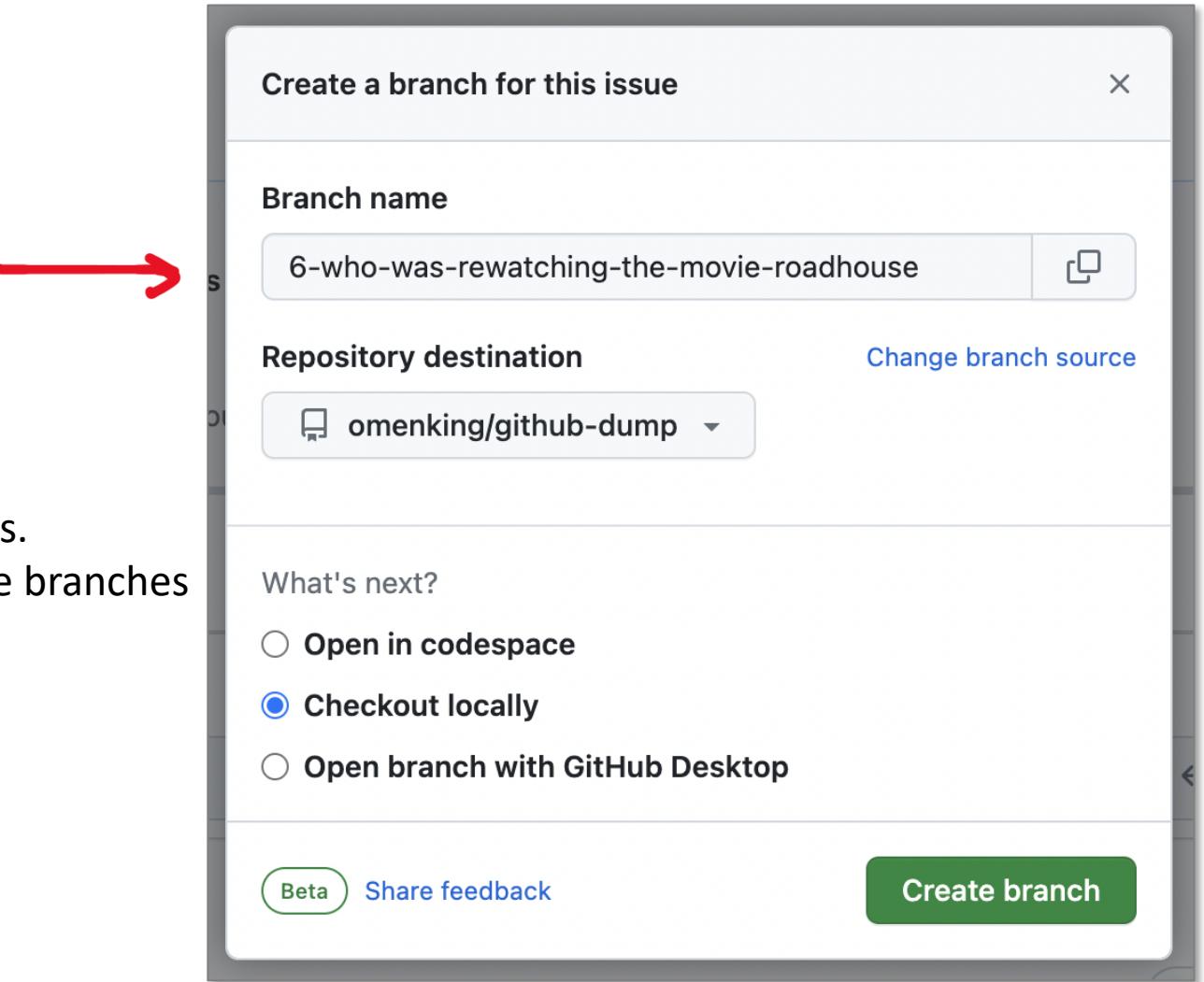
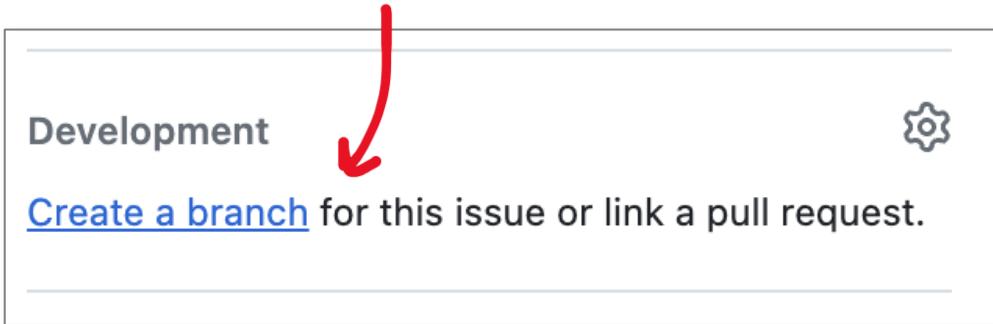
## Pull Requests

Proposing, reviewing, and merging code changes into the codebase.

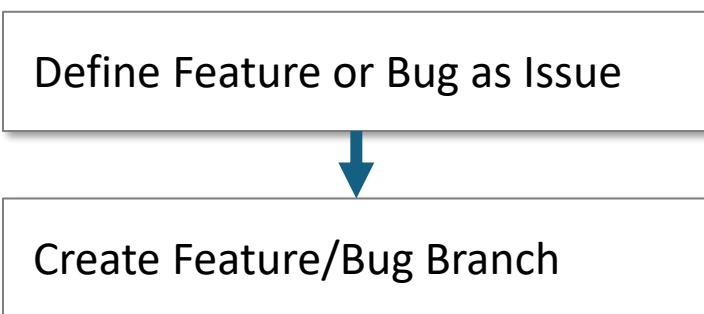
- Directly linked to code changes
- Can be linked to **Issues**

# Create a Branch from an Issue

From an Issue you can create an **associated branch**.

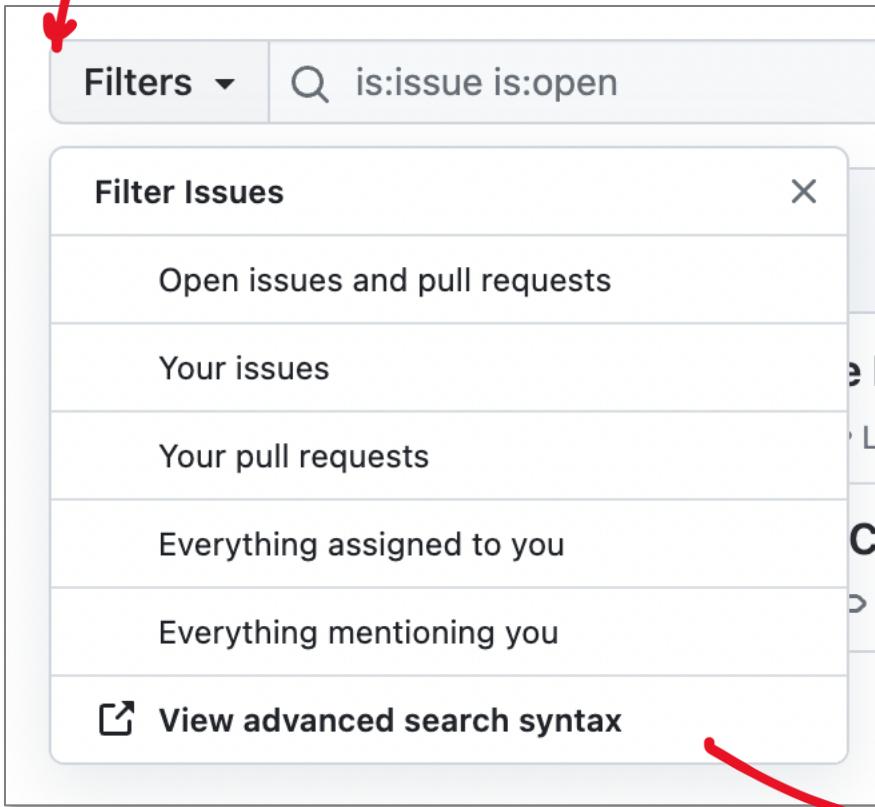


A common workflow is creating Feature or Bug branches.  
By defining your Issues upfront, you quickly create these branches



# Search and Filter Issues

You can apply filter rules to quickly sort though lots of Issues



```
is:issue author:username label:bug
is:issue label:bug label:urgent state:open
is:issue assignee:username milestone:"v1.0"
is:issue no:assignee repo:username/repository comments:>=10
is:issue created:>=2022-01-01 created:<=2022-12-31
is:issue involves:username comments:0
is:issue language:python updated:>2022-01-01
```

You can use **advanced search syntax**.

# Issues Templates

Issues Templates are **markdown templates that are preloaded for new Issues**.

They help ensure users creating issues provide all the relevant and expected information



```
Write Preview H B

Steps to reproduce
<!-- (Guidelines for creating a bug report are [available here](https://edgeguides.rubyonrails.org/contributing_to_ruby_on_rails.html#reproducing-a-bug) -->
<!-- Paste your executable test case created from one of the scripts found at (https://edgeguides.rubyonrails.org/contributing_to_ruby_on_rails.html#reproducing-a-bug) -->
```ruby
# Your reproduction script goes here
```

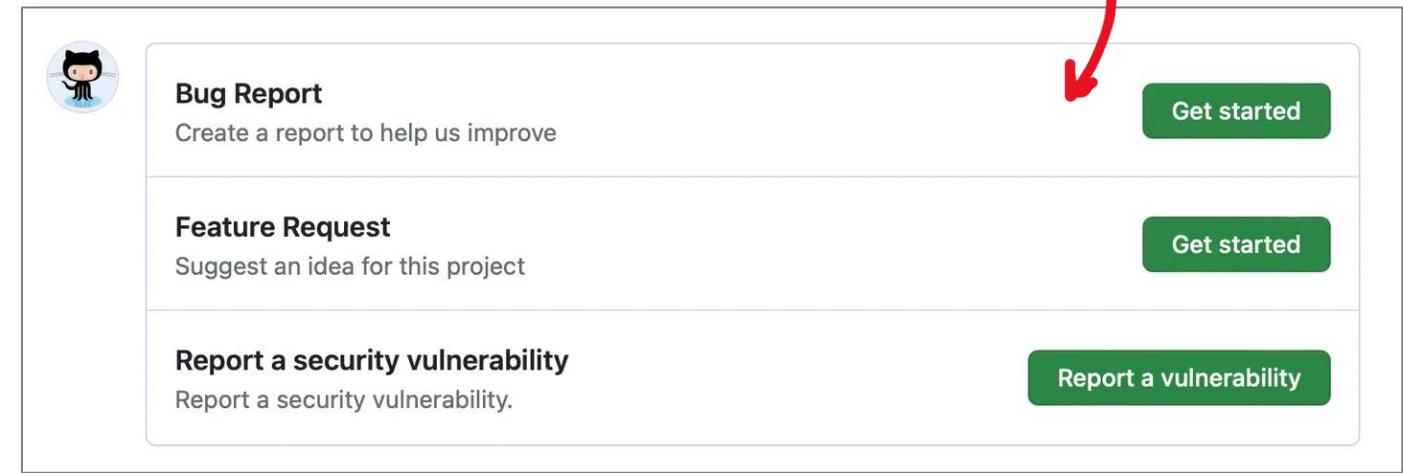
Expected behavior
<!-- Tell us what should happen -->

Actual behavior
<!-- Tell us what happens instead -->

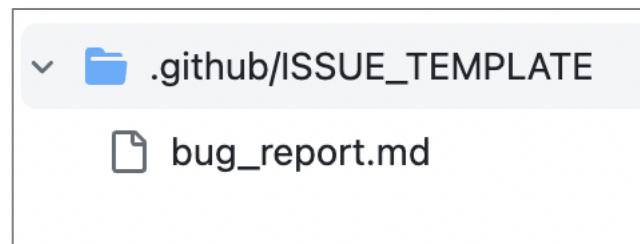
System configuration
Rails version:

Ruby version:
```

You can **create multiple issues templates** to better improve the context of issues:



Issue Templates are stored as markdown files in **.github/ISSUE\_TEMPLATES** folder in your repo



GitHub has a wizard GUI to easily create Issue Templates

# Issues Forms

**Issues Forms** is the evolution of Issues Templates.

You use a **YAML formatted file** to create Issue forms  
for stricter entry of issue information



```
name: Bug Report
description: File a bug report
title: "[Bug]: "
labels: ["bug", "triage"]
projects: ["octo-org/1", "octo-org/44"]
assignees:
 - octocat
body:
 - type: markdown
 attributes:
 value: |
 Thanks for taking the time to fill out
 - type: input
 id: contact
 attributes:
 label: Contact Details
 description: How can we get in touch with
 placeholder: ex. email@example.com
 validations:
```

Issue: Bug Report

File a bug report. If this doesn't look right, [choose a different type](#).

[Bug]:

Thanks for taking the time to fill out this bug report!

Contact Details

How can we get in touch with you if we need more info?

ex. email@example.com

What happened?

Also tell us, what did you expect to happen?

A bug happened!

Version

What version of our software are you running?

# Pinning Issues

Up to 3 issues can be pinned so they appear at the top of the Issue page.

The screenshot shows a user interface for managing issues. On the left, a sidebar contains options: Lock conversation, Pin issue (with an info icon), Transfer issue, Convert to discussion, and Delete issue. A red arrow points from the 'Pin issue' option to the pinned issue card on the right. The main area has tabs for Issues (2), Pull requests (1), Discussions, Actions, and Projects. Below the tabs, a pinned issue is displayed in a box:

**Who was rewatching the movie Roadhouse?** ×

#6 opened 3 hours ago by omenking

● Open

Filters ▾

Below the pinned issue, there are two filter options:

- 2 Open ✓ 0 Closed
- Who was rewatching the movie Roadhouse?  
#6 opened 3 hours ago by omenking ↳ Launch this Co...

# Pull Requests

A Pull Request (PR) is a **formal process to put forth changes**, that can be manually or automatically reviewed before its accepted into your base (main) branch.

The screenshot shows a list of pull requests. At the top, there are filters for 'Open' (1) and 'Closed' (0) pull requests, and dropdown menus for 'Author', 'Label', 'Projects', 'Milestones', 'Reviews', 'Assignee', and 'Sort'. A red arrow points to the 'Assignee' dropdown. Below the filters, a single pull request is listed: '#2 Implement Payment Gateway System' by omenking, opened 1 hour ago, with 1 review and 2 comments. It also mentions 13 tasks.

## Benefits of a Pull Request

- **Collaborative Review:** Enhances code quality through team discussions and peer feedback.
- **Change Tracking:** Provides a record of code changes and related discussions.
- **Automated Testing:** Enables integration with tools for automated checks and tests.
- **Controlled Integration:** Manages safe and reviewed merging of code changes.
- **Open Source Friendly:** Simplifies contributions and collaboration in open-source projects.

A pull request is not a feature of git, but a workflow. Services like GitHub can automate the Pull Request workflow

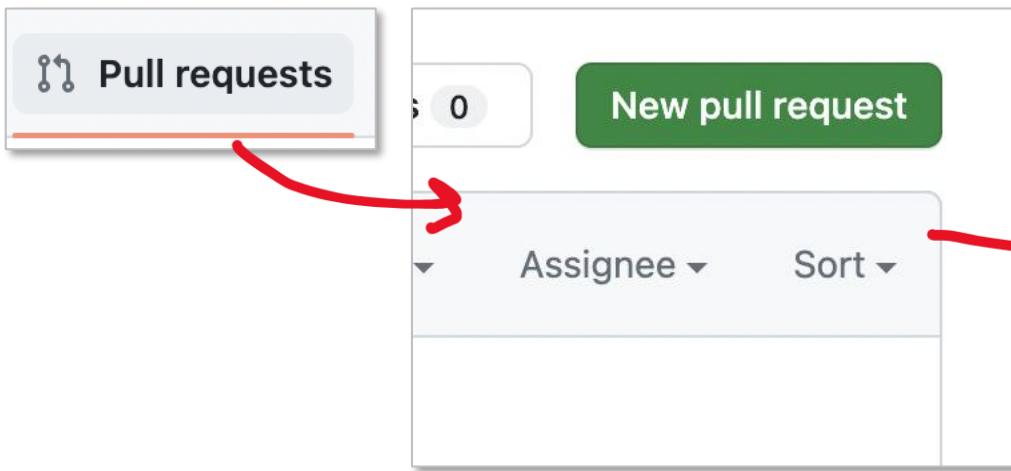
# Creating Pull Requests

We set the:

**Base** – who we are going to merge into

**Head** – the changes to pull in

In the GitHub repo we  
navigation to **Pull Requests**



We create a **new pull requests**

We can use the **Github CLI** to create a pull request from a branch.

```
git checkout -b cool-feature-branch
git commit -am "Your commit message"
git push -u origin cool-feature-branch
gh pr create --base main --head cool-feature-branch
```

A screenshot of the GitHub Pull Request creation interface. At the top, it shows 'base: main' and 'compare: my-feature'. A green checkmark indicates 'Able to merge'. The 'Create pull request' button is visible. Below, it shows '1 commit', '1 file changed', and '1 contributor'. It lists a commit from 'omenking' on Jan 12, 2024, with the message 'hello!'. The diff view shows a single addition to 'hello.txt' with the content '+ Hello World!'. Red arrows point from the GitHub CLI command block towards the 'Create pull request' button and the commit details.

# Pull Requests – Base and Compare

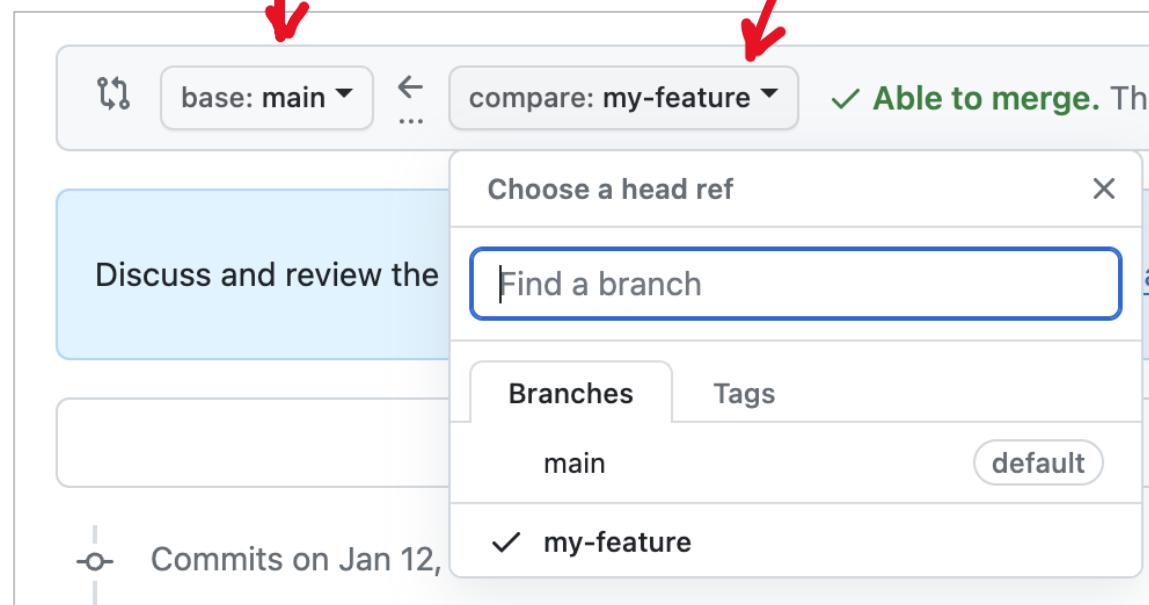
**Base and Compare** determines **the direction of the merge** for a pull request

**Base** is who you want to merge into

- This is usually main branch or an environment specific branch

**Compare** is what will be merged into Base

- Compare is choosing a Head ref
- This usually a bug or feature branch



# Pull Requests – Base and Compare

You can also **compare across forks** this is how a fork stays up to date, or how forks can suggest you to accept their changes.

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also [compare across forks](#) or [learn more](#).

base repository: omenking/github-dump ▾ base: main ↵ ... head repository: omenking/github-dump ▾ compare: my-feature ▾

✓ Able to merge. These branches can be automatically merged.

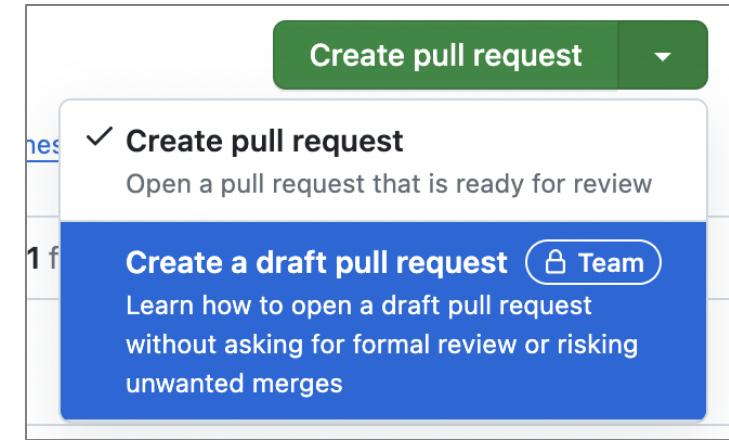
You'll now be able to choose repo's **from forks**

# Draft Pull Requests

A **Draft Pull Request** on GitHub is a feature that **allows you to open a pull request but mark it as a work-in-progress (WIP)**.



- Draft pull requests cannot be merged
- code owners are not automatically requested to review draft pull requests

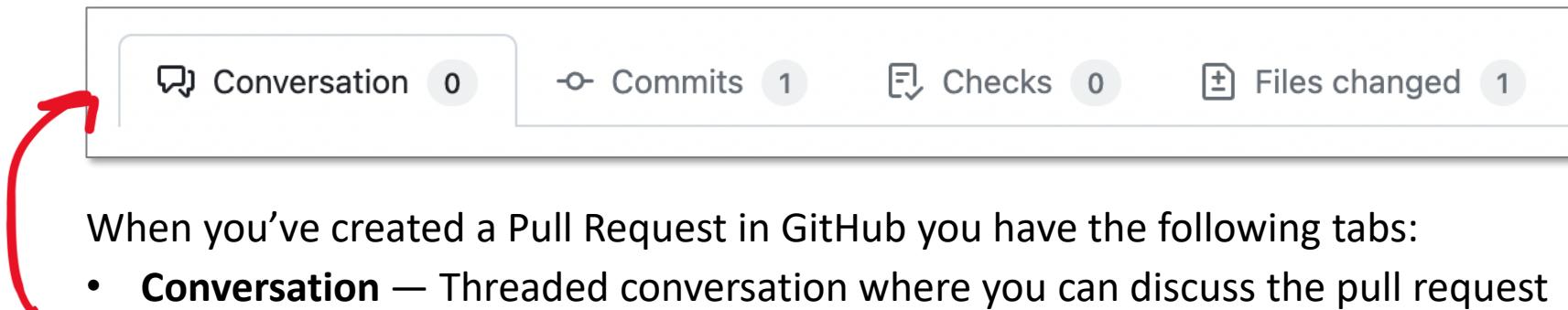


## Use case for Draft Pull Requests

- **Indicating Work-in-Progress:** Communicates that the pull request is not ready for review or merging.
- **Preventing Premature Merging:** Ensures incomplete work is not accidentally merged.
- **Facilitating Early Feedback and Collaboration:** Allows for early sharing and discussion of code changes.
- **Continuous Integration Testing:** Enables CI tests during the development process.
- **Transitioning to a Ready State:** Easy switch from draft to ready for final review and merging.
- **Organizing Work and Priorities:** Helps in managing and tracking ongoing work in large projects.

Draft Pull Requests is a feature only for **GitHub Organizations Teams**

# Pull Requests Tabs

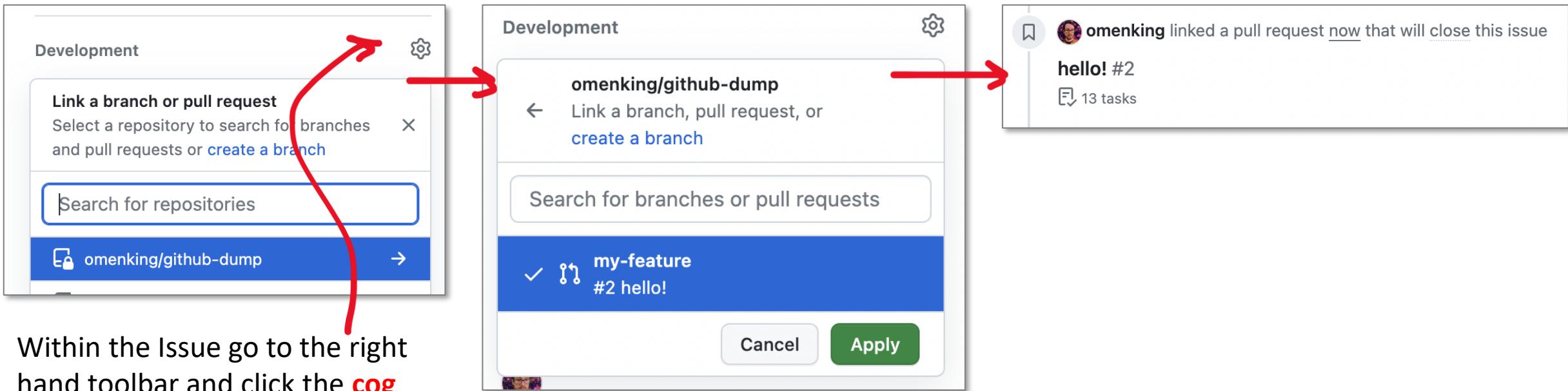


When you've created a Pull Request in GitHub you have the following tabs:

- **Conversation** — Threaded conversation where you can discuss the pull request
- **Commits** — Commits that are being merged into base
- **Checks** — Checks that need to be passed to merge PR.
  - could be apps installed or GitHub PR rules eg. Past test code suites
- **Files changes** — A list of files to be changed showing additions and deletions

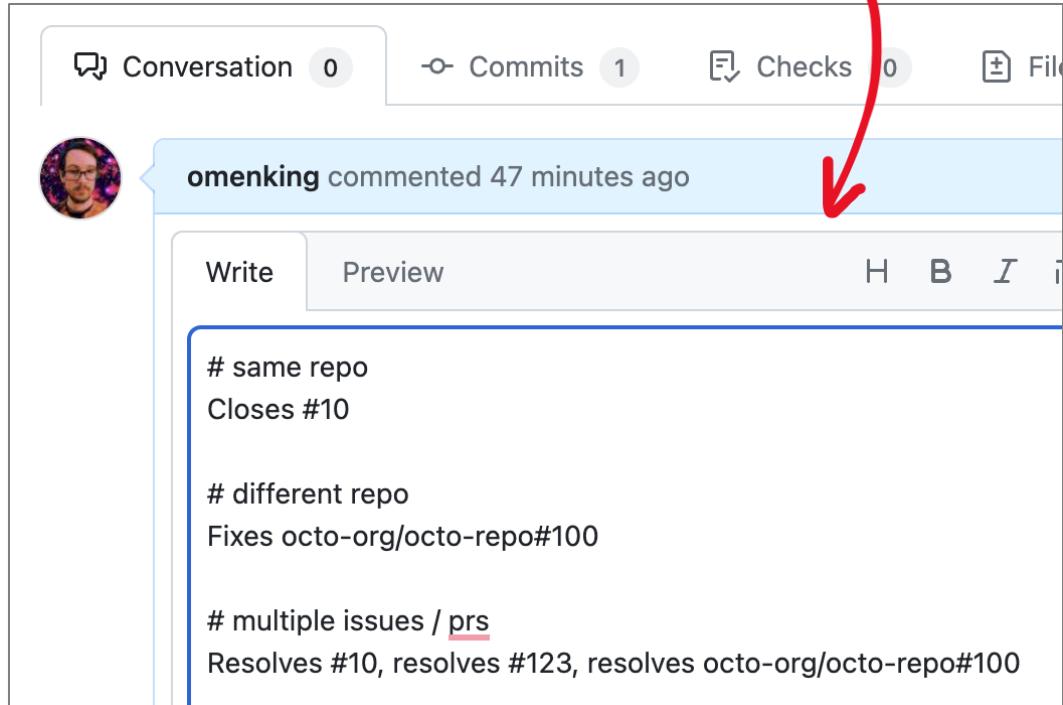
# Linked Activity within a Pull Request

You can Link Issues to Pull Requests so that the state of the pull request will automatically close the Issue



# Linked Activity within a Pull Request

You can link a pull request to an issue by using a supported keyword in the **pull request's description** or in a commit message.



## Supported Keywords

- close
- closes
- closed
- fix
- fixes
- fixed
- resolve
- resolves
- resolved



The pull request **must be** on the default branch.

# Pull Request Statuses

Open

The default status when a pull request is created. It's open for discussion and review.

Draft

Indicates the pull request is a work-in-progress and not yet ready for review.

Closed

The pull request has been closed without being merged. This status is used when the proposed changes are no longer needed or if the branch has been rejected.

Merged

The pull request's changes have been merged into the target branch.  
This status indicates a successful conclusion of the pull request process.

Changes Requested

This status is used during the review process when a reviewer requests changes before the pull request can be merged.

Review Required

Indicates that the pull request requires a review before it can be merged.  
This status is common in repositories where reviews are a mandatory part of the workflow.

Approved

The pull request has been reviewed and approved for merging by the required number of reviewers.

Conflict

Indicates that there are conflicts between the pull request's branch and the target branch that need to be resolved before merging.

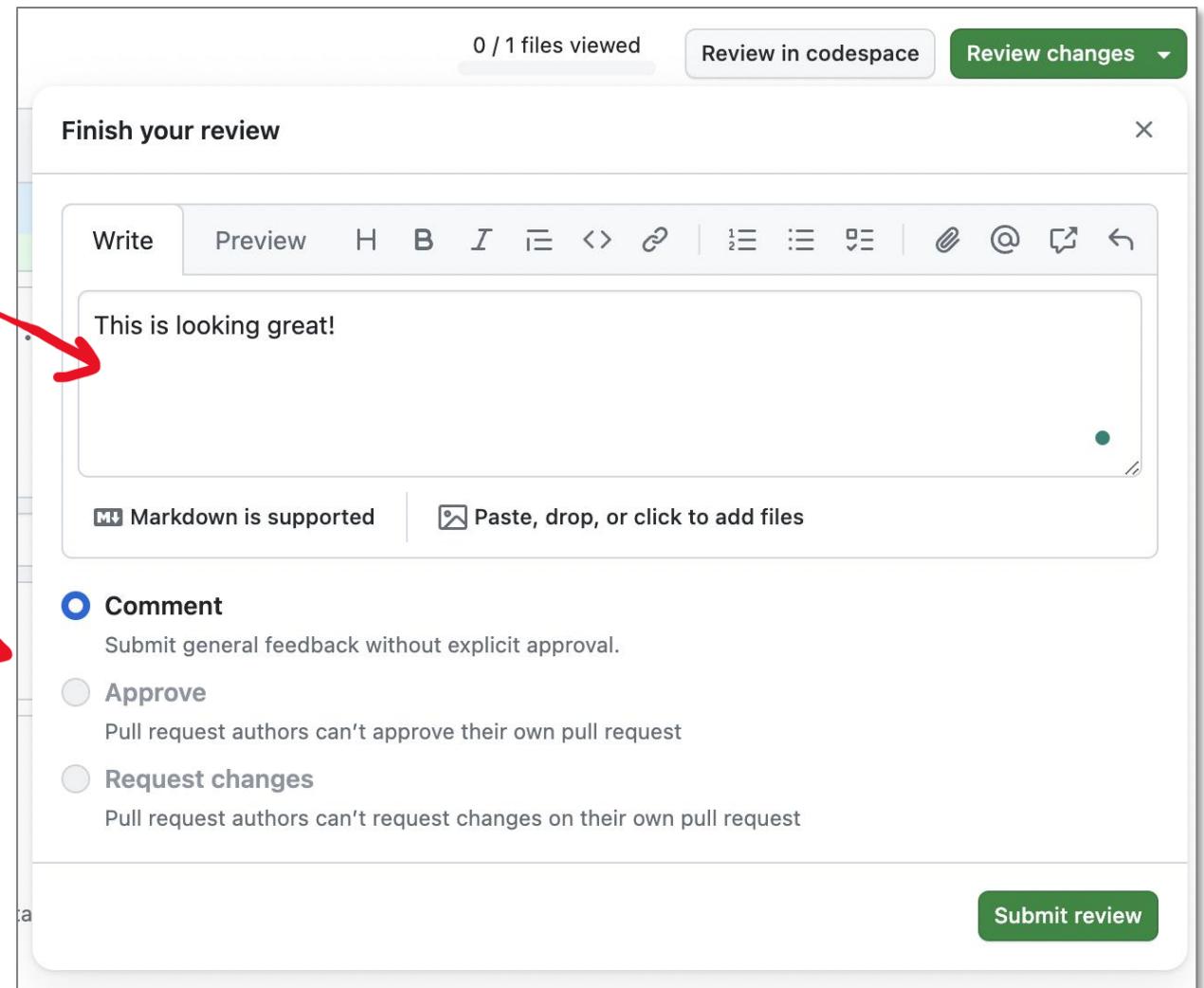
Ready for Review

A pull request initially marked as draft can be changed to this status once it's ready for review.

# Commenting in Pull Requests

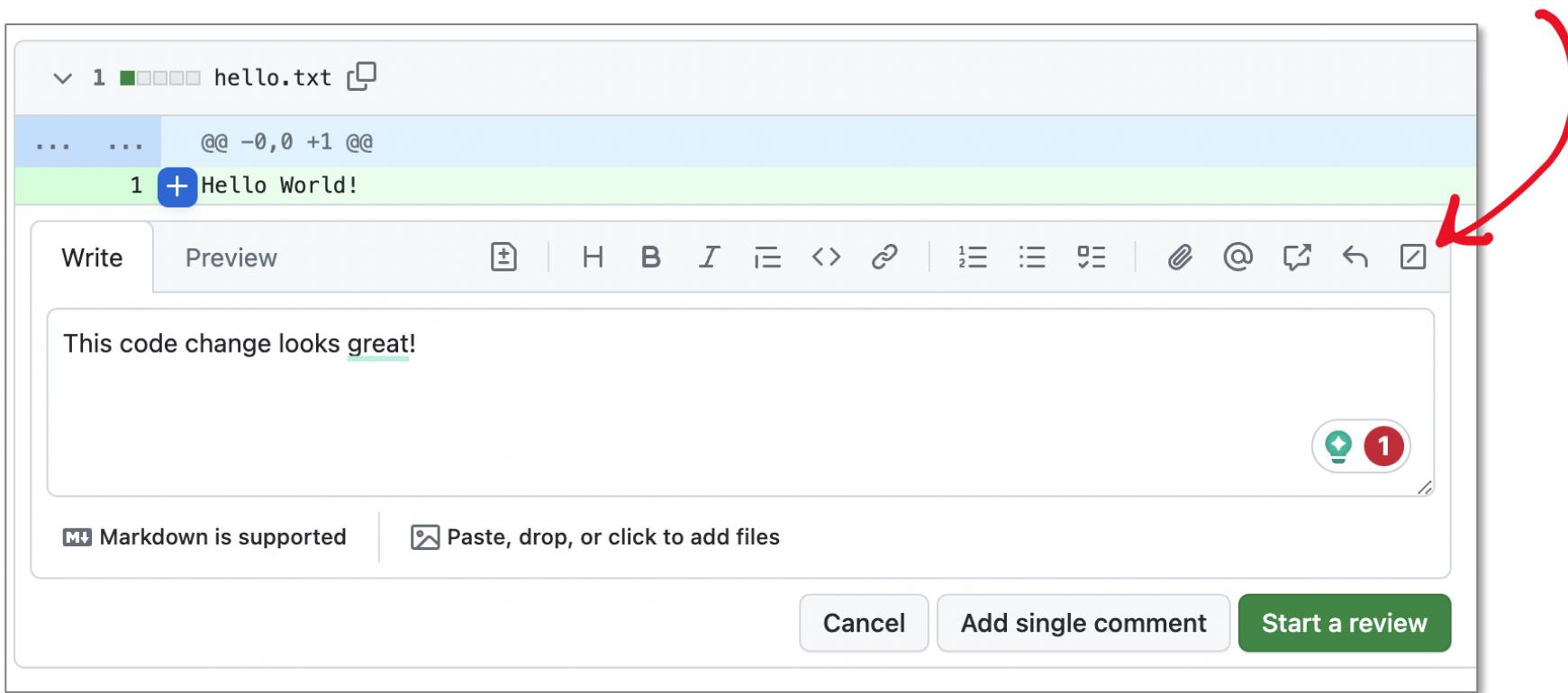
When you comment you are submitting a review to be considered before a Pull Request is being accepted.

If you are not the author, but also a designated reviewer you can **request changes** or mark the Pull Request as **Approved** so it can be considered for merging.



# Commenting in Pull Requests

Under **Files Changed** you can also apply a review on **a very specific line of code**.



# Code Reviews with CODEOWNERS file

**CODEOWNERS** GitHub repo specific file to define individuals or teams that are responsible for **specific code** in a repository

```
* @global-owner1 @global-owner2
*.js @js-owner
*.go docs@example.com
*.txt @octo-org/octocats
/build/logs/ @doctocat
docs/* docs@example.com
apps/ @octocat
/docs/ @doctocat
/scripts/ @doctocat @octocat
**/logs @octocat
/apps/ @octocat
/apps/github
/apps/ @octocat
/apps/github @doctocat
```



CODEOWNER files uses a **similar syntax to .gitignore**.

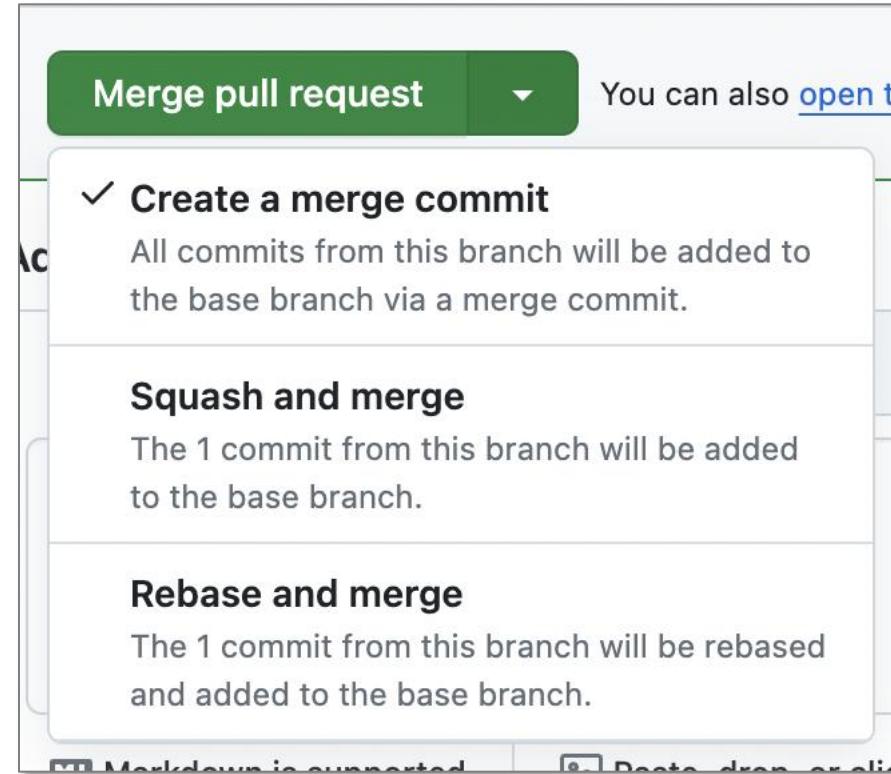
When a pull request is opened that modifies any files matching a pattern in the CODEOWNERS file, GitHub **automatically requests a review from the specified code owners.**

The CODEOWNERS files goes in either the project root, .github or docs directory.

# Pull Request Options

When merging a pull request there are a few options

- 1. Create a merge commit**
  - All commits will be added
- 2. Squash and merge**
  - 1 commit will be added
- 3. Rebase and merge**
  - 1 commit will be added and rebased



The use-case depends on your team's workflow. They may prefer only a single commit is added to keep the base branch git tree clean and readable.

# Required Reviewers

The screenshot shows a GitHub pull request review interface. On the left, a sidebar titled "Reviewers" allows selecting up to 15 reviewers, with a search bar and a list of suggestions including "bayko" and "Everyone else". The main area shows a review comment box with a rich text editor, a "Leave a comment" input field, and a "Submit review" button. Below the comment box are three review options: "Comment", "Approve", and "Request changes". The "Approve" option is selected. Red annotations include a red arrow pointing to the "Approve" radio button and another red arrow pointing to the "Review changes" button at the top right.

Pull Request can have **multiple required reviewers** who must write a review that approves the changes.

The screenshot shows a pull request status card with a green checkmark icon and the text "Review requested". It indicates "1 pending reviewer" and lists "omenking" as the user requested for review. A red annotation with a curved arrow points to the "pending reviewer" section, with the text "The review is expected".

The review with **Approve** is provided

The changes are approved **passing the check** for the repo

The screenshot shows a pull request status card with a green checkmark icon and the text "Changes approved". It indicates "1 approving review" and "1 approval". It lists "omenking" as the user who approved the changes. A red annotation with a curved arrow points to the "Changes approved" section, with the text "The review is expected".

# Pull Request Templates

**Pull Request Templates** are similar to Issue Templates  
They will populate the pull request textarea with the specified template

You create a file in the  
**.github/pull\_request\_template.md**



Add a description

Write Preview H B I

```

name: Pull Request Checklist
about: pull request checklist

Pull Request Template

Description
Please include a summary of the change and which issue is fixed. Include dependencies that are required for this change.

Fixes # (issue number)
```

Markdown is supported Paste, drop, or click to add files

Technically you can create multiple Pull Request Templates in a folder called `.github/PULL_REQUEST_TEMPLATE`



GitHub kind-of supports multiple Pull Requests Templates but you have to assemble your own URL with a query strings, so **its not as convenient to use as multiple Issue templates.**

# GitHub Discussions

GitHub Discussions is **community communication tool** for your public or private GitHub Repos.

- Threaded Conversations
- Categorization
- Community Interaction
- Markdown Support
- Pin Discussions
- Conversation Polls
- Discussion Voting
- Conversion to Issues
- Notifications
- Searchable and Linkable
- GitHub Integration

The screenshot shows the GitHub Discussions interface. At the top, there are three main discussion cards:

- GitHub Copilot Chat now generally available for organizations and individuals** by Copilot · Akash1134
- GitHub Community: Here's to an incredible 2023** by Announcements · grandmas-favorite
- Community-in-a-box** by Discover · ossanna16

Below these cards is a search bar with the query "is:open". To the right of the search bar are buttons for sorting ("Sort by: Latest activity"), filtering ("Label", "Filter: Open"), and creating a new discussion ("New discussion").

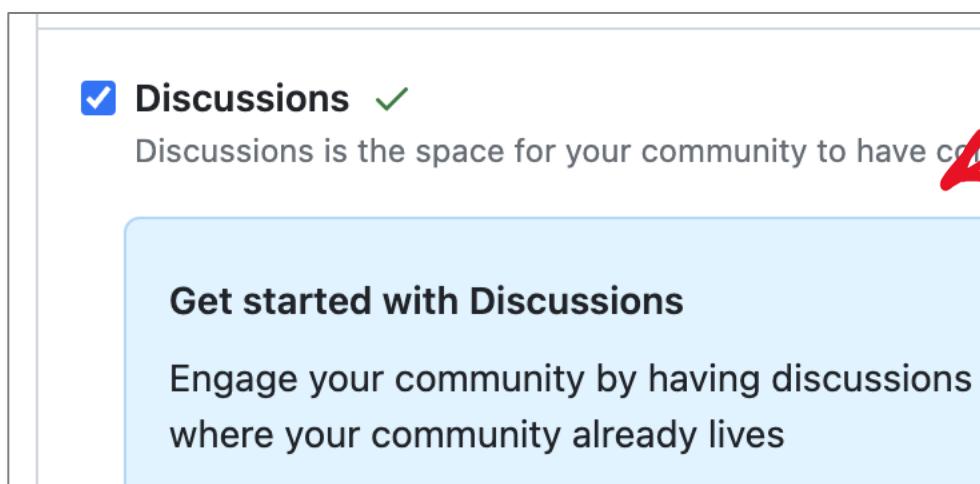
The main area is divided into two columns: "Categories" on the left and "Discussions" on the right. The "Categories" column includes links for "View all discussions" and various topic categories: Announcements, Discover, General, GitHub Education, New to GitHub, Programming Help, GitHub Product Categories, Accessibility, Actions, and API and Webhooks. The "Discussions" column lists several open discussions, each with a title, a user icon, a timestamp, and a "Most helpful" section on the right showing upvotes and users who have voted.

A red arrow points from the text "Community-in-a-box" in the first card to the "Community-in-a-box" discussion listed in the "Discussions" column.

| Discussion Title                                                                                        | User     | Category         | Created        | Upvotes | Most Helpful        |
|---------------------------------------------------------------------------------------------------------|----------|------------------|----------------|---------|---------------------|
| Why my github student developer pack is not being verified ? its about to be a month still not verified | nabinyd  | GitHub Education | 3 minutes ago  | 0       | faridnec (13)       |
| test                                                                                                    | HyatMyat | Mobile           | 4 hours ago    | 5       | queenofcorgis (12)  |
| demo                                                                                                    | HyatMyat | Lists            | 4 hours ago    | 5       | ananuness (8)       |
| Can't See Code In Repo on Android Using Browser Interface on GitHub                                     | chtfjfi  | General          | 31 minutes ago | 0       | ghostinhershell (3) |

# Discussions

Turn on Discussions under your **GitHub Repo Features**



A **Discussions tab** will appear in your GitHub Repo. You'll have some default categories available.

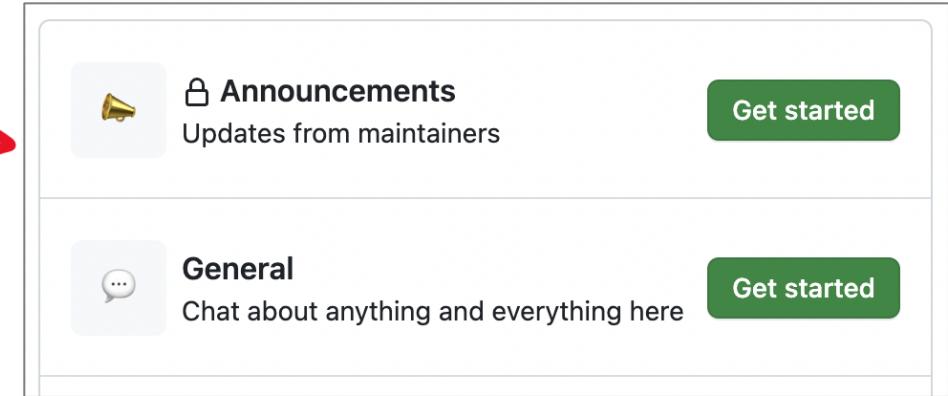
A screenshot of the GitHub 'Discussions' tab. A red arrow points from the text above to the 'Discussions' tab itself. The tab has several sections: 'Categories' (Announcements, General, Ideas, Polls, Q&A, Show and tell), 'Discussions' (a list of discussions), and a sidebar with 'Start a new discussion' and 'Welcome to discussions!'.

# Discussion Options

When someone creates a Discussion, they  
**need to choose a Category.**



GitHub provides some Categories, but  
GitHub repo owners **can create their own.**



**Discussion Format**

- Open-ended discussion**  
Enable your community to have conversations about tips and tricks or just chatting.
- Question / Answer**  
Enable your community to ask questions and vote on answers.
- Announcement**  
Share updates and news with your community across all categories, but anyone can comment.
- Poll**  
Gauge interest, vote, and interact with your community.

When you create a new category, you will choose a **Discussion Format**:

- **Open-ended discussion** — no voting of best answer, just open discussions
- **Question / Answer** — vote on the best suggested answer.
- **Announcement** — admins or maintainers can only post, anyone can reply
- **Poll** — Create Poll and let members vote



Categories can be grouped into **sections**

# Marking Answers in a Discussion

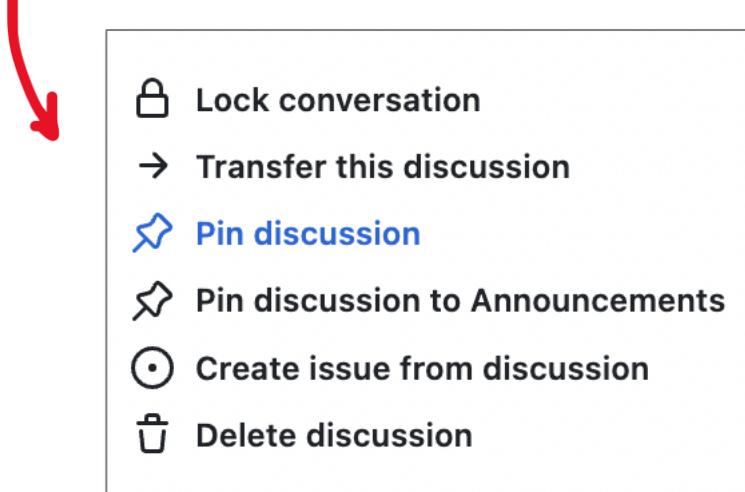
For Q&A Discussion Formats you can **mark an answer** as the recommended answer regardless of upvotes to help direct which question you think is the most credible.

The screenshot shows a Q&A discussion interface. On the left, a comment by user 'omenking' is displayed with the text 'Smash it'. Below it are buttons for 'Mark as answer' (with a checkmark icon), upvote ('↑ 1'), and smiley face. A red arrow points from the 'Mark as answer' button to a tooltip. The tooltip has a title 'Marking answers' and a description: 'Marking a comment as the answer helps others find the solution and gives credit to the comment author.' It also contains a 'Got it!' button and ends with 'Add your answer here...'. On the right, another comment by 'omenking' is shown with the text 'I don't know how'. This comment has an upvote count of '↑ 1'. A red arrow points from the 'Smash it' button of the first comment to the 'Smash it' button of the second comment. The second comment is highlighted with a green background and includes the text 'Answered by omenking now', a 'Smash it' button, and a 'View full answer ↓' link.

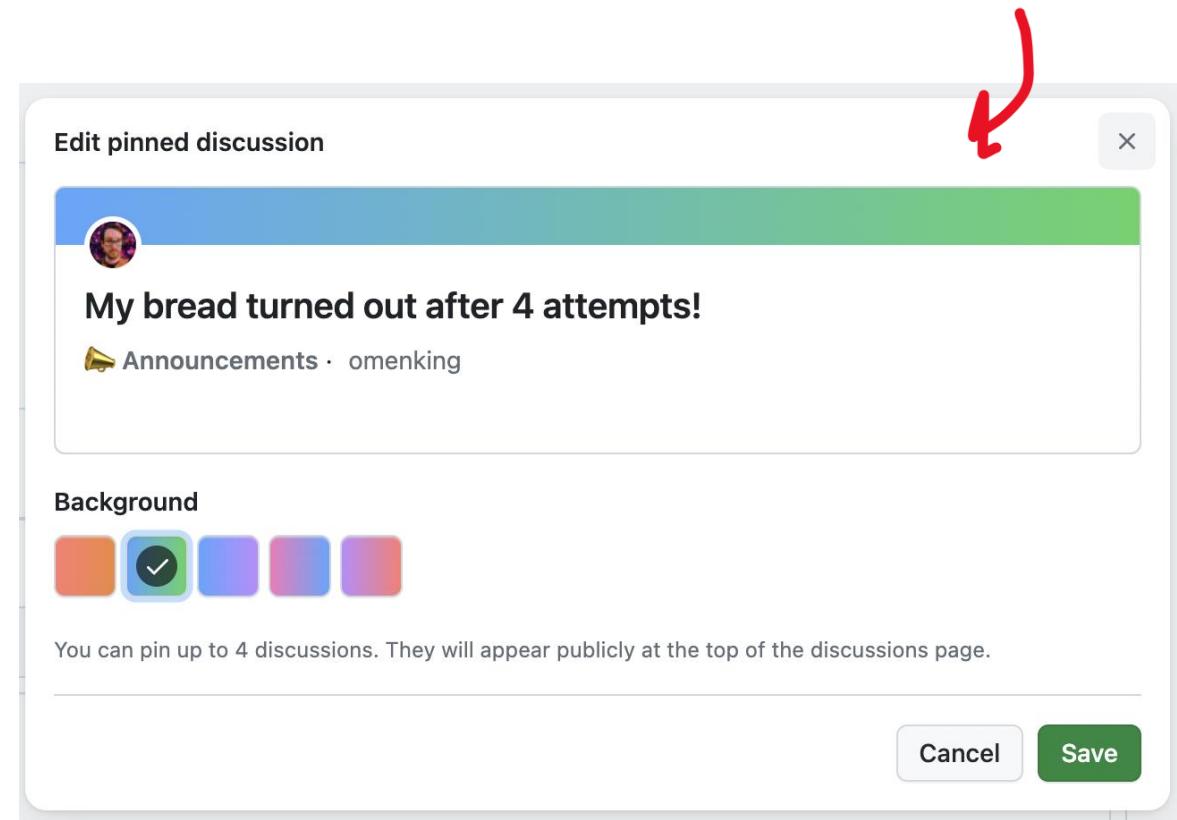
# Pinning Discussions

**Pinning Discussions** puts discussions fixed at the top of the community page.

You can **pin** a discussion the Discussions home page or for a specific category



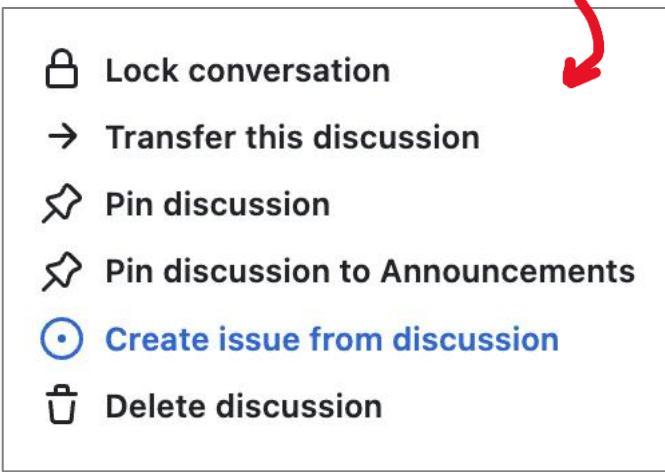
You can **customize** the look of a pinned discussion.



# Convert Discussion to an Issue

You can **"convert"** a Discussion to an Issue

In the discussion choose to **create issue**.



It will **prepopulate an Issue** and place a Discussion link in the description

A screenshot of a GitHub issue creation interface. The title field contains "Who was rewatching the movie Roadhouse?". The description field contains the following text:  
### Discussed in <https://github.com/omenking/github-dump/discussions/5>  
<div type='discussions-op-text'>  
<sup>Originally posted by \*\*omenking\*\* January 13, 2024</sup>  
I thought you'd be taller.</div>

Under the **Discussion's Events**  
it will have a link to the issue

A screenshot of the discussion events section. It shows a new event entry: "omenking Created issue #6 1m".

# Notification Configuration Options

In your account settings you have a Notifications tab to configure **notifications options**.

The screenshot shows the GitHub account settings page for user Andrew Brown (omenking). The left sidebar lists various settings categories, and the 'Notifications' tab is highlighted with a blue border. The main content area is titled 'Notifications' and contains several configuration sections. A red arrow points from the text above to the 'Notifications' tab in the sidebar.

**Default notifications email**  
Choose where you'd like emails to be sent. You can add more email addresses. Use custom routing to send different email addresses to be used for individual organizations.  
andrew@monsterboxpro.com ▾ Custom routing

**Automatically watch repositories**  
When you're given push access to a repository, automatically receive notifications for it.  
 On

**Automatically watch teams**  
Anytime you join a new team or are subscribed to updates for a team @mentioned.  
 On

**Subscriptions**

**Watching**  
Notifications for all repositories, teams, or conversations you're watching. [View watched repositories](#)

# Notification Filters Options

You can filter notifications using a **filter syntax**:

The screenshot shows a sidebar titled 'Filter notifications' with a blue header bar. Below it, under 'Available filters', are several items listed with their descriptions:

- repo:** filter by repository
- is:** filter by status or discussion type
- reason:** filter by notification reason
- author:** filter by notification author
- org:** filter by organization

There are **predefined saved filters**

The screenshot shows a list of predefined filters with their counts:

| Filter Name      | Count |
|------------------|-------|
| Assigned         | 8     |
| Participating    | 8     |
| Mentioned        | 2     |
| Team mentioned   | 0     |
| Review requested | 0     |

You can **create more saved filters** using the filter syntax

The screenshot shows a table of predefined filters with their corresponding filter syntax:

| Name             | Filter inbox by...      |
|------------------|-------------------------|
| Assigned         | reason:assign           |
| Participating    | reason:participating    |
| Mentioned        | reason:mention          |
| Team mentioned   | reason:team-mention     |
| Review requested | reason:review-requested |

The screenshot shows a dropdown menu with the following options:

- Group by: Date ▾
- Repository
- ✓ Date
- subscribed

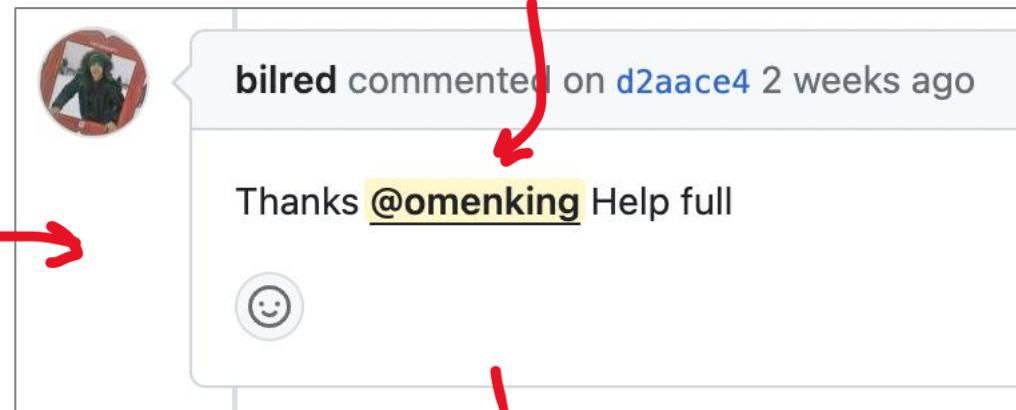
A red arrow points from the text below to the 'Date' option in the dropdown.

You can **group** Notification Threads by Repo or by Date.

# Mentions in Threads

If someone mentions you in a thread than you can find these mention under the **Mentioned Filters**.

A mention is when someone using a **@** followed by your username



A screenshot of the GitHub inbox. On the left, there are filters: "Inbox" (750 notifications), "Saved", "Done", "Filters" (with a gear icon), "Assigned", "Participating" (8 notifications), and a blue button "Mentioned" (2 notifications). In the center, there's a search bar with "reason:mention". Below it is a filter panel with "Select all" and two items: "omenking/aws-bootcamp-cruddur-2023 minor fixes to notifications page" and "aws/aws-cli #7702 Error parsing parameter '--budget': Unable to load paramfile". To the right, there are two notifications: one from "omenking" mentioning the user about minor fixes, and another from "aws/aws-cli" mentioning the user about a parsing error. Red arrows point from the "Mentioned" filter in the sidebar to the "omenking" notification and from the "Mentioned Filters" text above to the "omenking" mention in the notification.

# Notifications Inbox

The Notifications Inbox provides a chronological list of notifications threads. This allows you to keep on top of all subscribed activities with associated GitHub repos.

<https://github.com/notifications>

The screenshot shows the GitHub Notifications inbox interface. On the left, there's a sidebar with filters for Saved (1), Done (✓), Assigned (8), Participating (8), Mentioned (2), Team mentioned, and Review requested. The main area shows a list of notifications with columns for the repository, issue number, title, changes, subscribers, user icon, and timestamp. The first notification is from RhinoSecurityLabs/clougoat #32, titled 'Feature Request: Minimum AWS Policy Template'. The second is from winglang/docsite, and the third is from winglang/docsite again. The fourth is from WCD-LMS/WeCloudDataLMS #1094. The fifth and sixth are from winglang/docsite. The last one is from winglang/docsite again. All notifications are marked as 'subscribed'.

| Repository                 | Issue # | Title                                        | Changes | Subscribers | User | Timestamp    |
|----------------------------|---------|----------------------------------------------|---------|-------------|------|--------------|
| RhinoSecurityLabs/clougoat | #32     | Feature Request: Minimum AWS Policy Template | +2      | subscribed  |      | 12 hours ago |
| winglang/docsite           | #759    | feat(docs): update docs                      | +1      | subscribed  |      | yesterday    |
| winglang/docsite           | #759    | feat(docs): update docs                      | +2      | subscribed  |      | yesterday    |
| WCD-LMS/WeCloudDataLMS     | #1094   | fix question cloner not working              | +1      | subscribed  |      | yesterday    |
| winglang/docsite           | #758    | feat(docs): update docs                      | +1      | subscribed  |      | 2 days ago   |
| winglang/docsite           | #758    | feat(docs): update docs                      | +2      | subscribed  |      | 2 days ago   |

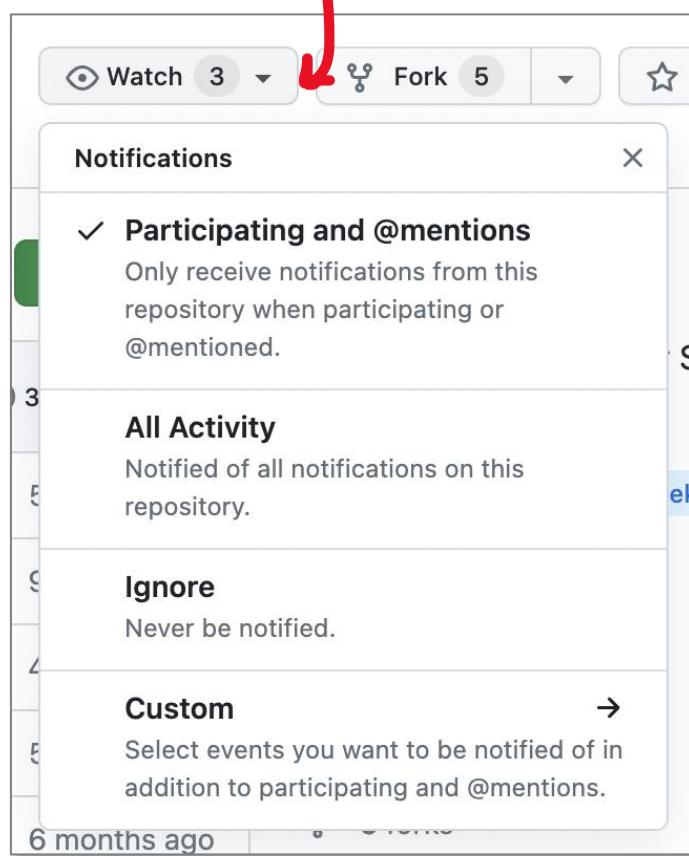
In the far bottom right, we can access other **Notifications views:**

A dropdown menu titled 'Manage notifications' is shown. It includes sections for 'Notification settings', 'Watched repositories', 'Subscriptions', and a 'Manage notifications' button.

- ExamProCo/te... 2
- Notification settings
- Watched repositories
- Subscriptions
- Manage notifications ▾

# Notification Watching

You can “watch” repos on GitHub to get notified of activities.  
*Your own repos you already set as watching.*



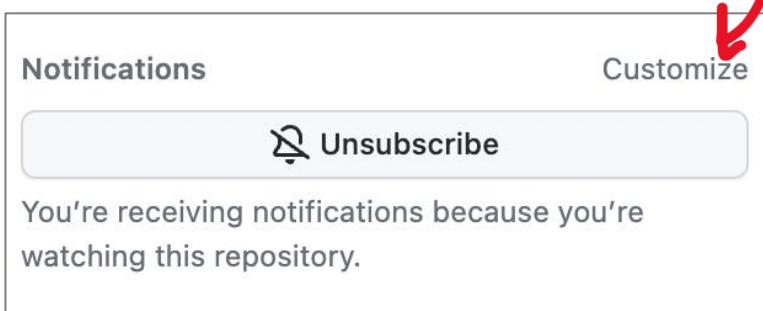
You can see a list of watching under notifications

<https://github.com/watching>

A screenshot of the GitHub 'Watching' list page at <https://github.com/watching>. The URL is highlighted with a pink box and a red arrow. The page shows a list of repositories being watched, such as 'omenking/github-dump' and 'ExamProCo/AWS-Examples'. On the right side, there is a sidebar with 'Notification settings' for 'Participating and @mentions', 'All Activity', 'Ignore', and 'Custom'.

# Notification Subscriptions

Subscriptions allow you to follow activities for a **specific Pull Request or Issue.**



You can see all your subscriptions under Notifications.

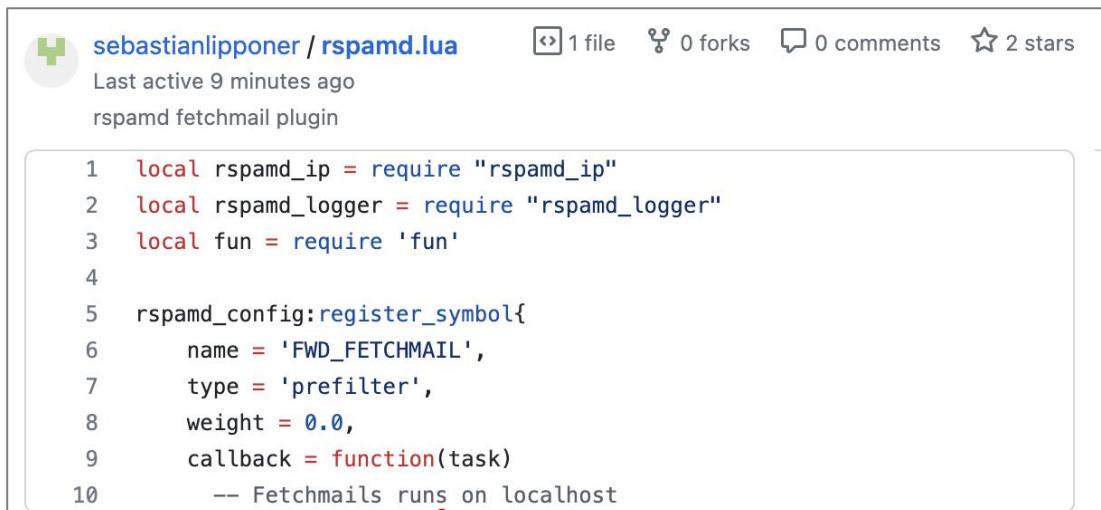
A screenshot of the GitHub Notifications / Subscriptions page. The URL <https://github.com/notifications/subscriptions> is highlighted with a pink box and a red arrow. The page shows a list of subscriptions with details like the repository name, issue title, and activity history. The "Subscriptions" tab is selected. The entire screenshot is enclosed in a light gray border.

| Subscription Details                                                    | Reason                                        | Repository                                | Sort Order               |
|-------------------------------------------------------------------------|-----------------------------------------------|-------------------------------------------|--------------------------|
| omenking/github-dump Finish the Github Foundations Certification course | #1 opened 2 hours ago by omenking             | omenking/github-dump                      | Most recently subscribed |
| omenking/terraform-beginner-bootcamp-2023 My new ticket                 | #68 opened yesterday by omenking              | omenking/terraform-beginner-bootcamp-2023 |                          |
| grpc/grpc Request: Prebuilt musllinux arm artifacts                     | #34998 opened on Nov 16, 2023 by MattDynamite | grpc/grpc                                 |                          |
| omenking/terraform-beginner-bootcamp-2023 Final Category                | #65 opened on Oct 11, 2023 by omenking        | omenking/terraform-beginner-bootcamp-2023 |                          |
| omenking/terraform-beginner-bootcamp-2023 commit files                  | #66 opened on Oct 11, 2023 by omenking        | omenking/terraform-beginner-bootcamp-2023 |                          |

# GitHub Gists

Gists provide a **simple way to share code snippets with others**  
**Every gist is a Git repository**, which means that it can be **forked** and **cloned**.

<https://gist.github.com>



sebastianlippner / **rspamd.lua** 1 file 0 forks 0 comments 2 stars  
Last active 9 minutes ago  
rspamd fetchmail plugin

```
1 local rspamd_ip = require "rspamd_ip"
2 local rspamd_logger = require "rspamd_logger"
3 local fun = require 'fun'
4
5 rspamd_config:register_symbol{
6 name = 'FWD_FETCHMAIL',
7 type = 'prefilter',
8 weight = 0.0,
9 callback = function(task)
10 -- Fetchmails runs on localhost
```

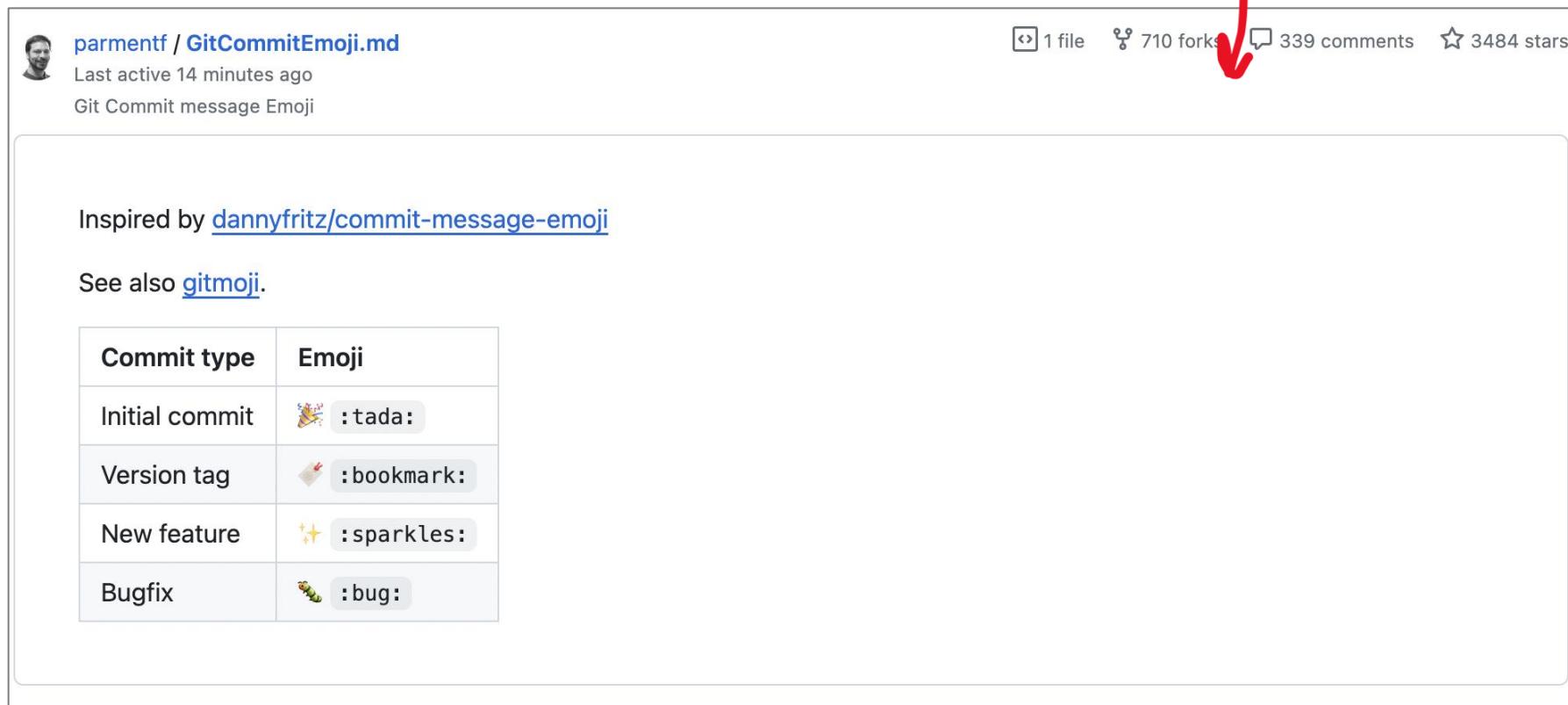
A **public gist** on [gist.github.com](https://gist.github.com)

Gists can be **public** or **secret**.

- Public Gists are searchable
- Public Gists show up in Discover
  - <https://gist.github.com/discover>
- Secret gists are not searchable and do not show up in discovery
- Secrets gists are not private
  - You can share the URL with friends
- Secret Gists can be changed to be public
- Public Gists cannot be changed to be private
- You can pin Gists to your profile for other to easily find
  - Gists allow other users to comment
  - Gists allow you to easily navigation revisions

# Creating GitHub Gists

If a Gist is a **markdown file** It will render the HTML.  
This is a great way to share one-off documentation



parmentf / [GitCommitEmoji.md](#)

Last active 14 minutes ago

Git Commit message Emoji

Inspired by [dannyfritz/commit-message-emoji](#)

See also [gitmoji](#).

| Commit type    | Emoji        |
|----------------|--------------|
| Initial commit | 🎉 :tada:     |
| Version tag    | 🔖 :bookmark: |
| New feature    | ✨ :sparkles: |
| Bugfix         | 🐛 :bug:      |

1 file 710 forks 339 comments 3484 stars

# GitHub Gists

Create a Gist is a straightforward.

Name it:

Hello World

Provide the **filename**

with extension.

- The file extension will allow the file to apply syntax highlighting

Provide your **code**

hello.rb

```
1 puts "Hello World"
```

Spaces

2

No wrap

Use `Control+Shift+m` to toggle the `tab` key moving focus.

Add file

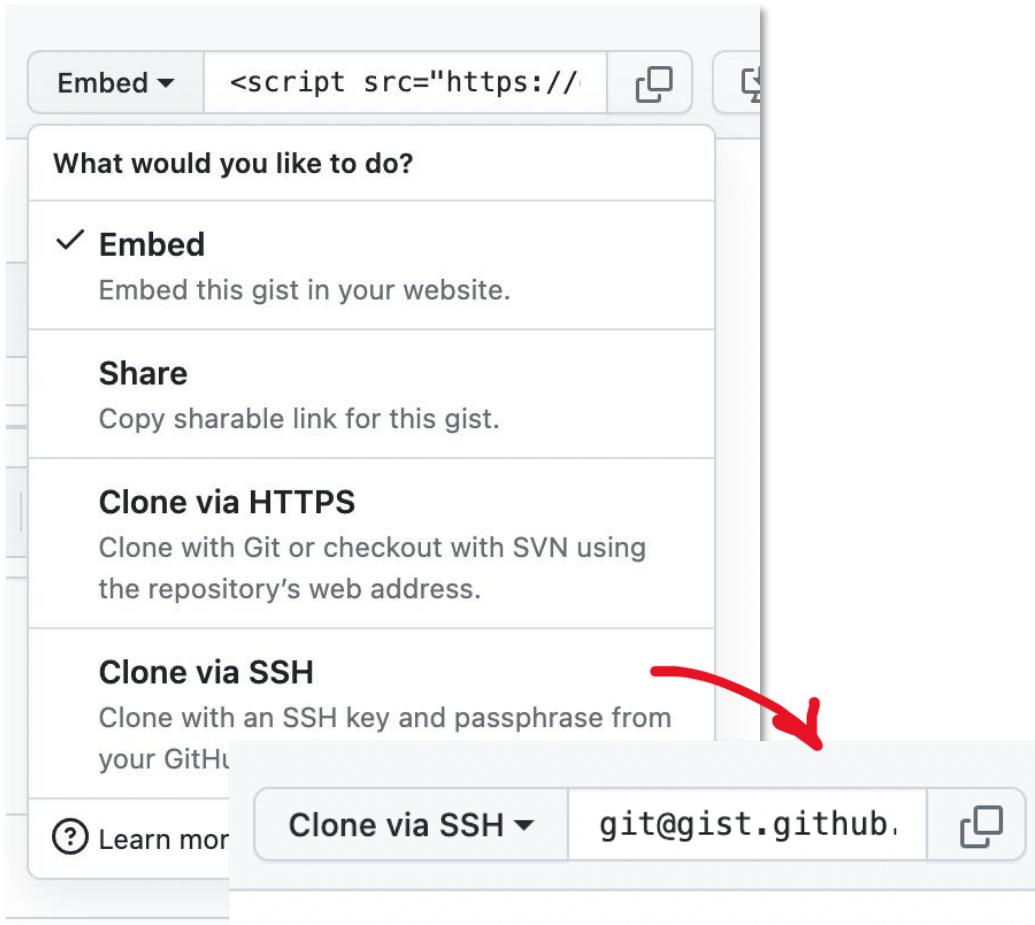
Attach associate files to the gist

Create secret gist

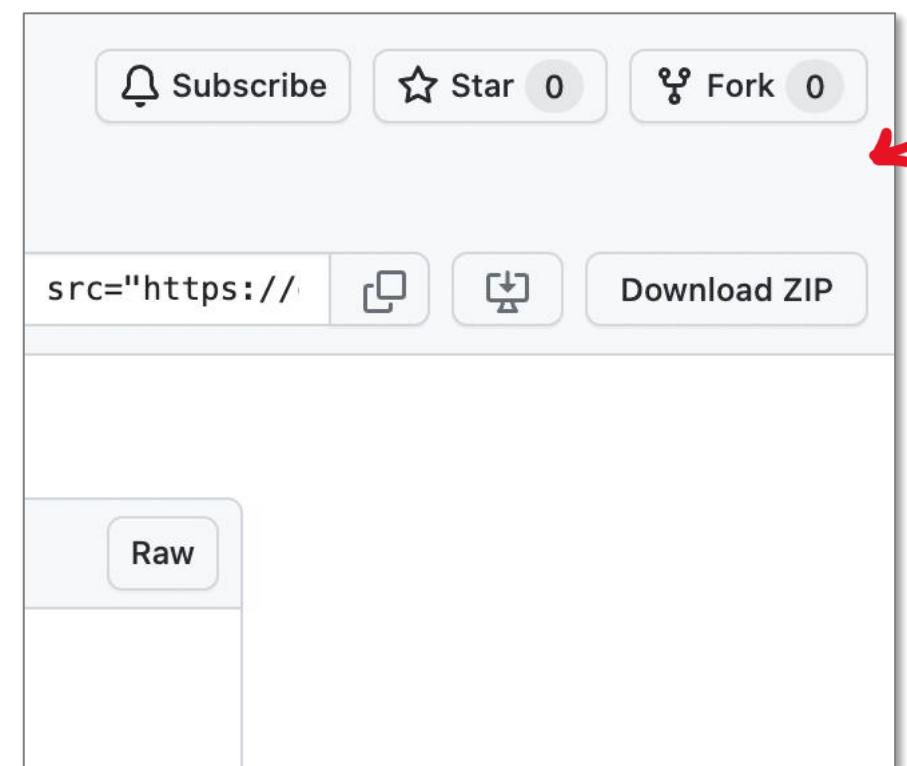
Set it as either **secret** or **private**

# Fork and Clone Gists

You can **clone URL** by dropping down embed



For Gist repos other than your own you **fork** them.



# GitHub Wiki Pages

A **Wiki** is collaborative way to quickly **create documentation with multiple nested pages**

A wiki would serve as a **knowledge base** for your repo or organization.

## Configuration files

Justin MacCarthy edited this page on Aug 23, 2021 · 2 revisions

### Files used by Guard

**Project-specific Guard DSL files:**

- `Guardfile` (in current directory) - can contain DSL statements, evaluated by Guard during startup
- `.Guardfile` (in current directory) - if it exists, it is used instead of the `Guardfile`

**Global Guard DSL files:**

- `~/.guard.rb` (in home directory) - can contain DSL statements. If it exists, it is always evaluated by Guard (before evaluating the project-specific DSL).

### Pry-only config

Guard supports the `XDG_CONFIG_HOME` environment value if it is set .

- `~/.guardrc` (in home directory) - if it exists, it is evaluated by the Pry interator (and should not contain DSL statements)

This wiki and the Guard [README](#) document contains a lot of information, please take your time and read these instructions carefully.

If you run into any trouble, you may start by [understanding how Guard works](#).

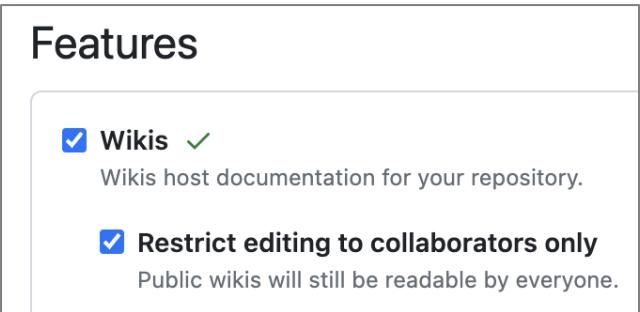
We provide [detailed changes for each Guard release](#).

**Search pages**

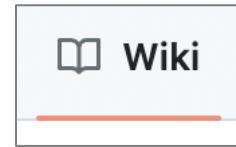
**Nested pages**

# GitHub Wiki Pages

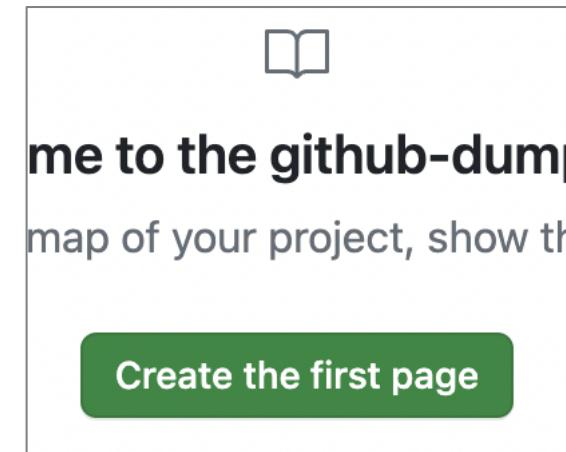
Turn **on the Wiki feature** for your GitHub Repo.



Go to your **Wiki page** of your Repo

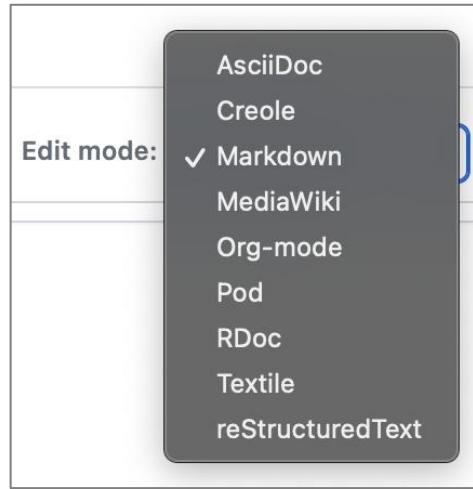


Create your **first page!**



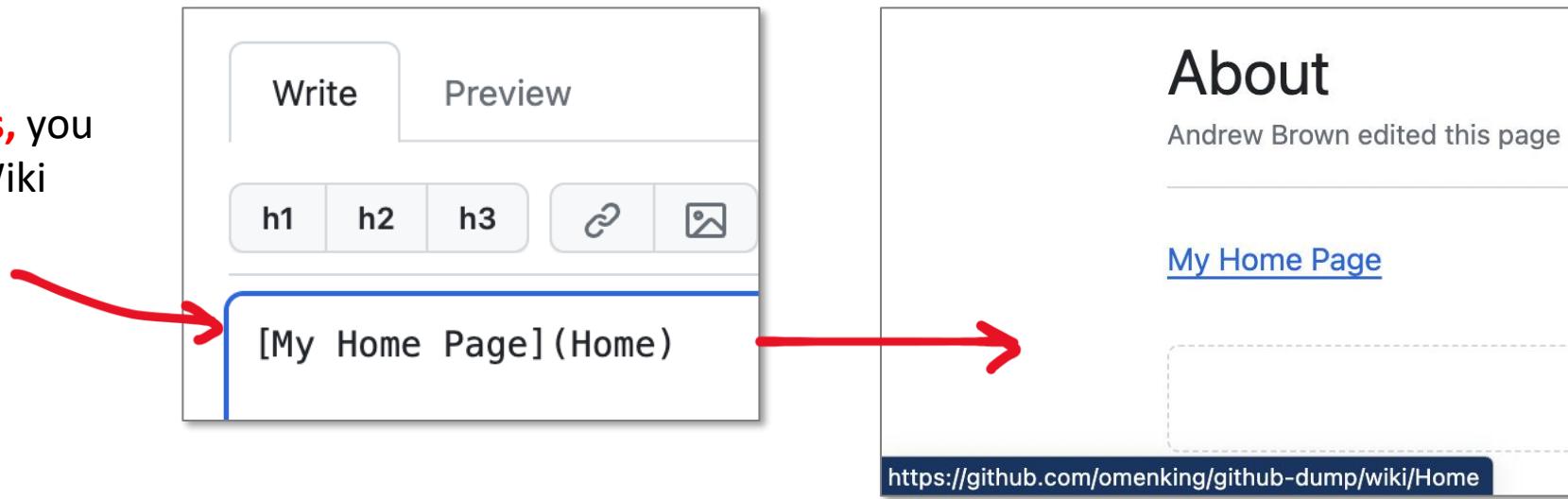
GitHub Wiki's can be used for free on Public repos.

# Managing GitHub Wiki Pages



When you create a page, you can use  
**various markup languages.**

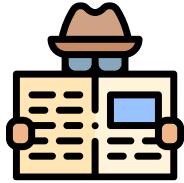
To **link to other pages**, you provide the GitHub Wiki pages title as the link



# GitHub Wiki pages visibility



If you create a wiki in a **public repository**,  
the wiki is available to the public.



If you create a wiki in a **private repository**,  
only people with access to the repository  
can access the wiki.

# GitHub Pages

**GitHub Pages** allows you to directly host a static website via a GitHub repository.

You need to create a **public repo** with the repo name being **<username>.github.io**

Owner \* omenking

Repository name \* omenking.github.io

✓ omenking.github.io is available.

Great repository names are short and memorable. Need ins...

Description (optional)

Public

Anyone on the internet can see this repository. You cho...

You simply **add HTML files and assets** to your repo.



**Jekyll** is static site generator which allows you to build complex static sites or blogs with ease.

- Jekyll uses yaml files with frontmatter format.
- Jekyll utilizes markdown files
- Jekyll directly integrates with GitHub Pages and is a GitHub project

Your website will be available the domain name: **<username.github.io>**

# Introduction to GitHub Actions



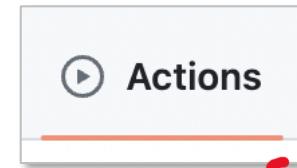
**GitHub Actions** is a **CI/CD pipeline directly integrated with your GitHub repository.**

GitHub Actions allows you to **automate**:

- Running test suites
- Building images
- Compiling static sites
- Deploying code to servers
- and more...

GitHub Actions has **templates** you can use to get started.

A screenshot of a GitHub Actions template card. It shows the title "Deploy Node.js to Azure Web App" by Microsoft Azure. Below the title, it says "Build a Node.js project and deploy it to an Azure Web App." At the bottom, there are two buttons: "Configure" and "Deployment". A red arrow points from the text "GitHub Actions has templates you can use to get started." to this card.



Within a GitHub repo you'll have a **tab for Actions**.

GitHub Actions files are defined as YAML files located in the **.github/workflow** folder in your repo.

You can have multiple workflows in repo triggered by different events.

```
on:
 push:
 branches: [$default-branch]
 workflow_dispatch:
env:
 AZURE_WEBAPP_NAME: your-app-name
 AZURE_WEBAPP_PACKAGE_PATH: '.'
 NODE_VERSION: '14.x'
permissions:
 contents: read
jobs:
 build:
 runs-on: ubuntu-latest
 steps:
 - uses: actions/checkout@v3
 ...
```

# Introduction to GitHub Actions

When you run GitHub Actions, you'll get **a history of workflow runs** where it will indicate if it was success a failure, and how long it took to run.



| All workflows                                                                                                                                             | Filter workflow runs | Event ▾       | Status ▾ | Branch ▾ | Actor ▾ |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------|---------------|----------|----------|---------|
| Showing runs from all workflows                                                                                                                           |                      |               |          |          |         |
| 113 workflow runs                                                                                                                                         |                      |               |          |          |         |
| <span>✓ Merge pull request #116 from ExamProCo/main</span><br>Deploy production #113: Commit <a href="#">8cac2e9</a> pushed by omenking                   | production           | 6 months ago  | 10m 10s  | ...      |         |
| <span>✓ Merge pull request #114 from ExamProCo/main</span><br>Deploy production #112: Commit <a href="#">6d2e4e8</a> pushed by omenking                   | production           | 6 months ago  | 10m 55s  | ...      |         |
| <span>✓ Merge pull request #113 from ExamProCo/main</span><br>Deploy production #111: Commit <a href="#">af69015</a> pushed by bayko                      | production           | 9 months ago  | 11m 1s   | ...      |         |
| <span>✓ Merge pull request #112 from ExamProCo/vimeo-submission-feedb...</span><br>Deploy production #110: Commit <a href="#">7c4cd93</a> pushed by bayko | production           | 10 months ago | 11m 24s  | ...      |         |
| <span>✓ Merge pull request #111 from ExamProCo/main</span><br>Deploy production #109: Commit <a href="#">f0152e7</a> pushed by bayko                      | production           | 10 months ago | 12m 34s  | ...      |         |
| <span>✗ Merge pull request #110 from ExamProCo/main</span><br>Deploy production #108: Commit <a href="#">6ed15e7</a> pushed by bayko                      | production           | 10 months ago | 18m 1s   | ...      |         |

# Finding GitHub Actions

GitHub has a repo of example workflows you can use to get you started

<https://github.com/actions/starter-workflows>

A screenshot of a GitHub repository page for 'starter-workflows'. The repository is public, as indicated by the 'Public' button at the top right. The main branch is 'main', and there are 23 branches and 0 tags. A search bar is present at the top right. Below the header, a pull request from 'orhantoy' is shown, merging pull request #2275 from 'aeisenberg/patch-1'. The repository contains several folders: '.github', '.vscode', 'automation', 'ci', 'code-scanning', 'deployments', 'icons', and 'pages'. A red arrow points from the URL bar above to the 'ci' folder in the repository listing.

A screenshot of a specific workflow file, 'ci/go.yml', within the 'starter-workflows' repository. The file is a YAML configuration for GitHub Actions. It defines a workflow named 'Go' that triggers on push or pull request events to the default branch. The workflow includes a job named 'build' that runs on an Ubuntu 20.04 LTS machine and uses the 'actions/checkout@v3' action to check out the code.

```
1 # This workflow will build a golang project
2 # For more information see: https://docs.github.com/en/actions
3
4 name: Go
5
6 on:
7 push:
8 branches: [$default-branch]
9 pull_request:
10 branches: [$default-branch]
11
12 jobs:
13
14 build:
15 runs-on: ubuntu-latest
16 steps:
17 - uses: actions/checkout@v3
```

# Different Types of GitHub Actions

Event Triggers causes a GitHub Action to run.

The **on** attribute specifies  
the **event trigger** to be used:

GitHub Actions has 35+ event triggers

```
name: Example Workflow
on: [push]
jobs:
 build:
 runs-on: ubuntu-latest
 steps:
 - uses: actions/checkout@v2
 - name: Run a script
 run: echo "This script runs on every push!"
```

Examples of common GitHub Actions that could be triggered:

- **Pushes:** Trigger an action on any push to the repository.
- **Pull Requests:** Run actions when pull requests are opened, updated, or merged.
- **Issues:** Execute actions based on issue activities, like creation or labeling.
- **Releases:** Automate workflows when a new release is published.
- **Scheduled Events:** Schedule actions to run at specific times.
- **Manual Triggers:** Allow manual triggering of actions through the GitHub UI.

# Introduction to GitHub Copilot

GitHub Copilot is **AI developer tool** that can be used with multiple IDEs via an extension

- **Code Completion**
- Chat in IDE and Mobile
- CLI assistance
- Code referencing
- Public code filter
- IP indemnity



```
game.rb
A terminal-based black jack game with weird scifi theme and new rules.

def game
 puts "Welcome to the game of Black Jack!"
 puts "What is your name?"
 name = gets.chomp
 player = Player.new(name)
 dealer = Dealer.new
 deck = Deck.new
 deck.shuffle
 player.hand = Hand.new
 dealer.hand = Hand.new
 player.hand.add_card(deck.deal)
 dealer.hand.add_card(deck.deal)
 player.hand.add_card(deck.deal)
 dealer.hand.add_card(deck.deal)
 puts "The dealer is showing #{dealer.hand.cards[0].rank} of #{dealer.hand.cards[0].suit}"
 puts "You have a #{player.hand.cards[0].rank} of #{player.hand.cards[0].suit}"
 puts "Your total is #{player.hand.total}."
 puts "Would you like to hit or stay?"
 hit_or_stay = gets.chomp.downcase
 while hit_or_stay == "hit"
 player.hand.add_card(deck.deal)
 puts "You have a #{player.hand.cards[0].rank} of #{player.hand.cards[0].suit}"
 puts "Your total is #{player.hand.total}."
```



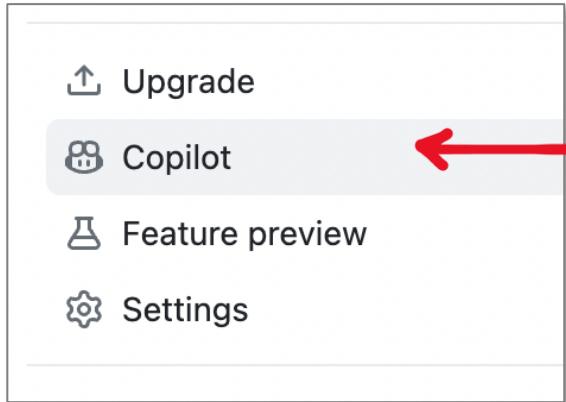
You can try GitHub Copilot for free with a one-time 30-day trial.

GitHub Copilot is free to use for verified students, teachers, and maintainers of popular open source projects.

# GitHub Copilot Individuals vs Business

|                                            | Copilot Individuals                      | Copilot Business                    |
|--------------------------------------------|------------------------------------------|-------------------------------------|
| Pricing                                    | \$10 USD per month<br>\$100 USD per year | \$19 USD per user per month         |
| Types of GitHub accounts                   | Personal accounts                        | Organization or enterprise accounts |
| Telemetry                                  | ✓                                        | ✗                                   |
| Blocks suggestions matching public code    | ✓                                        | ✓                                   |
| Plugs right into your editor               | ✓                                        | ✓                                   |
| Offers multi-line function suggestions     | ✓                                        | ✓                                   |
| Organization-wide policy management        | ✗                                        | ✓                                   |
| Exclude specified files                    | ✗                                        | ✓                                   |
| Audit logs                                 | ✗                                        | ✓                                   |
| HTTP proxy support via custom certificates | ✗                                        | ✓                                   |

# Setting up Copilot

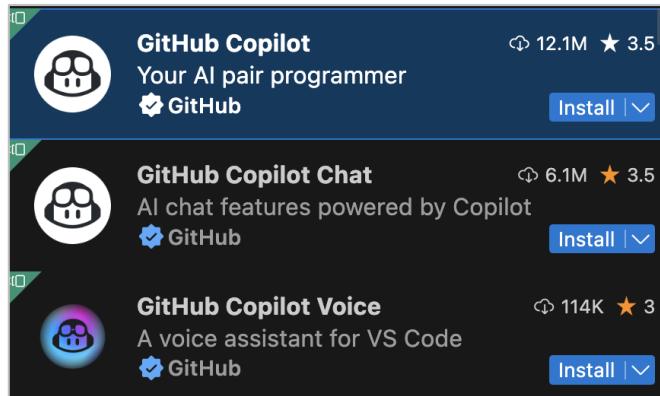


Under User Account dropdown there is a **Copilot** option.

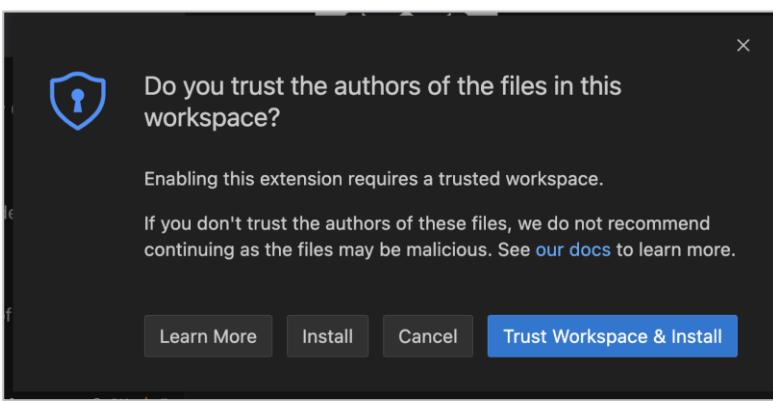
A screenshot of the GitHub Copilot landing page. It features a section titled "Get started with Copilot" with a sub-section "Start using Copilot to secure your seat." Below it, a text says "Select your IDE and follow the easy steps to get started with your AI pair-programmer". At the bottom, there are four buttons: Visual Studio Code, JetBrains, Visual Studio, and Vim/Neovim.

GitHub has various instructions to get started with GitHub Copilot based on your IDE.

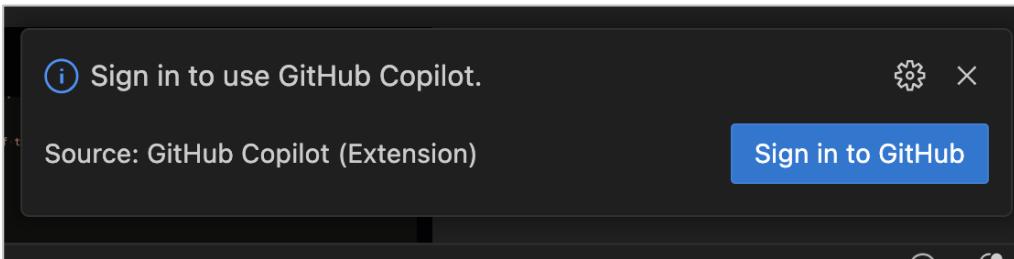
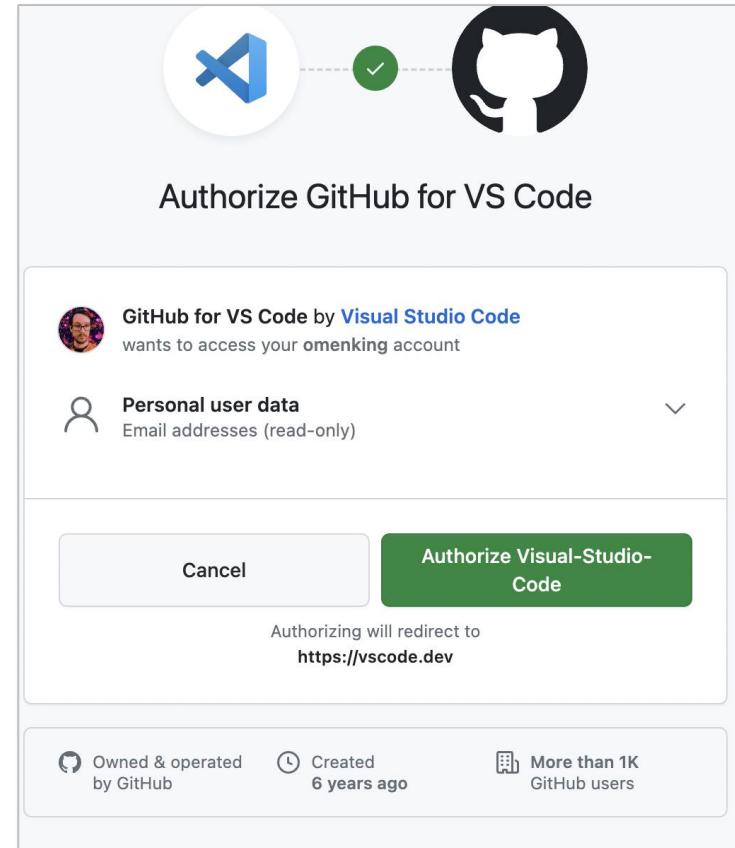
# Setting up Copilot



Install the **GitHub Copilot extension** in your VS Code

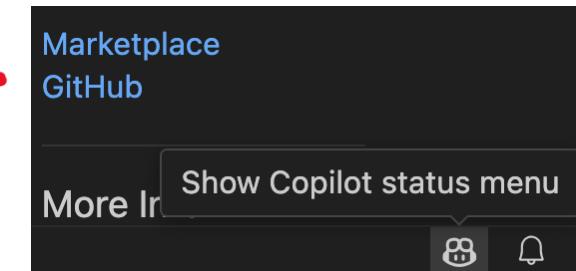


Choose **Trust Workspace & Install**



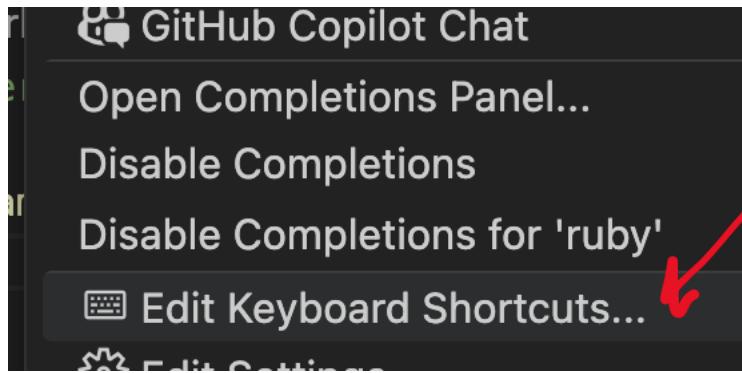
**Sign in to GitHub**

Copilot will show up  
in **bottom right corner**



# Using Copilot

Copilot is best used when you  
**know or set hotkeys** to  
perform suggestions



| Command                                        | Keybinding |
|------------------------------------------------|------------|
| GitHub <b>Copilot</b> : Accept panel sugg...   | ^ /        |
| github.copilot.acceptCursorPane...             |            |
| GitHub <b>Copilot</b> : Navigate to the nex... | ⌘ ]        |
| github.copilot.nextPanelSolution               |            |
| GitHub <b>Copilot</b> : Navigate to the pre... | ⌘ [        |
| github.copilot.previousPanelSol...             |            |
| GitHub <b>Copilot</b> : Open Completions ...   | ^ Enter    |
| github.copilot.generate                        |            |
| GitHub <b>Copilot</b> : Suggest Terminal C...  | ⌘          |
| github.copilot.terminal.suggest...             |            |
| Trigger Inline Suggestion                      | ⌘ \        |
| Chat: Focus on GitHub <b>Copilot</b> View      |            |

Its best to check what hotkeys you have since they may vary based on the documentation stated hotkeys

# Using Copilot Chat

CHAT: GITHUB COPILOT

card sum wins.

Here's a basic implementation of this logic in Ruby:

```
class Card
 attr_reader :value

 def initialize(value)
 @value = value
 end

 class Deck
 def initialize
 @cards = (1..10).to_a * 4
 @cards.shuffle!
 end

 def deal
 Card.new(@cards.pop)
 end
 end

 def game
 deck = Deck.new
 player_cards = [deck.deal, deck.deal]
 end
end
```

⚡ How can I handle user input for the player's actions in the game?

Ask Copilot or type '/' for command ▶

You need to have **GitHub Copilot Chat feature** enabled in your personal account or for your organization.

Features

**Copilot Chat in the IDE** Beta Enabled

If enabled, members of this organization will have access to [GitHub Copilot Chat in IDE](#).

**Copilot in the CLI** Beta Disabled

If enabled, members of this organization will get [GitHub Copilot assistance in terminal](#).

Once enabled you can  
**chat with Google Copilot**

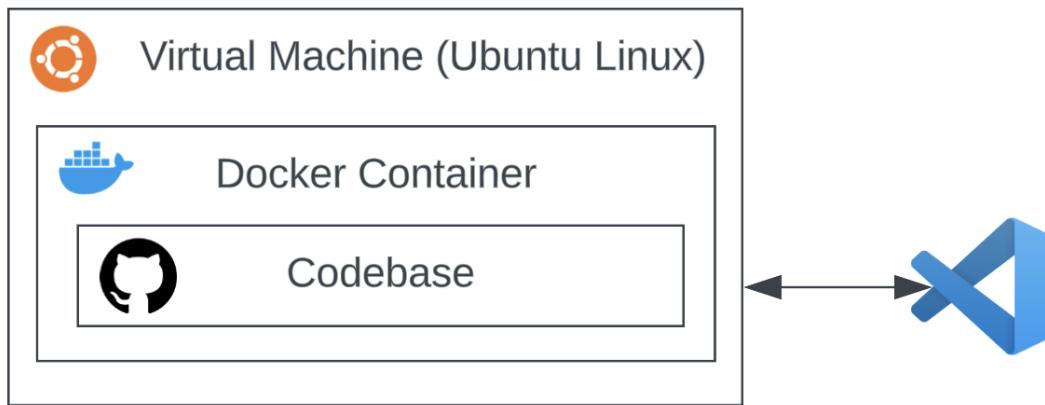
# Introduction to GitHub Codespaces

**GitHub Codespaces is Cloud Developer Environment (CDE) integrated with your GitHub repo.**

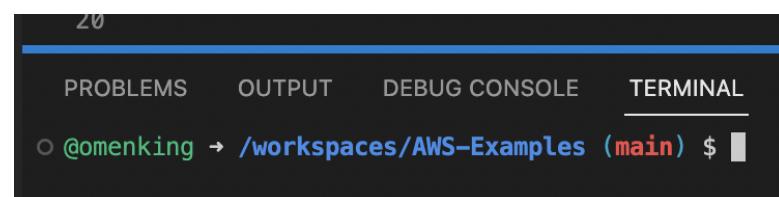
An environment is known as a **codespace**.

A codespace runs:

- in Ubuntu Linux docker container
  - on a virtual machine
  - hosted and managed by GitHub



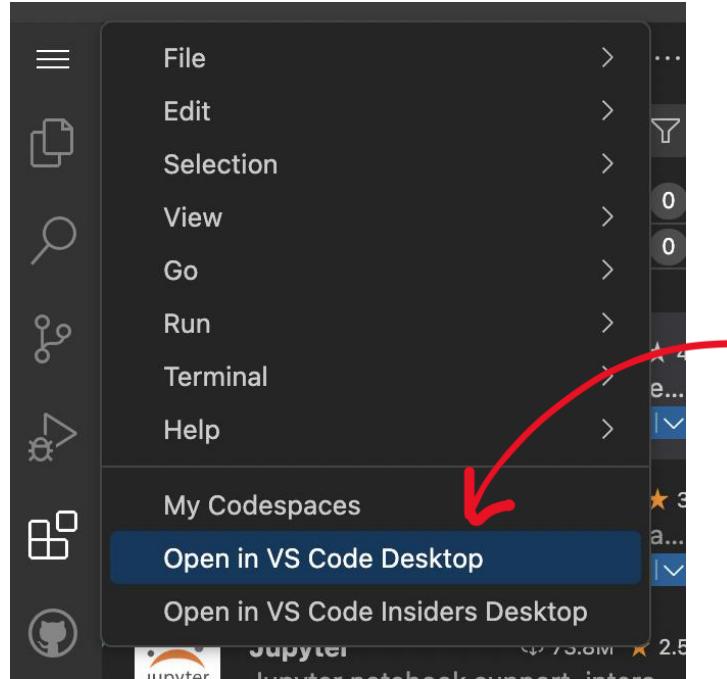
Your codebase will be cloned into **/workspaces**



A screenshot of a web browser window titled "ideal-acorn-7vxgr5q4cp74j.github.dev". The address bar also shows "AWS-Examples [Codespaces]". The browser displays a dark-themed interface of the AWS Examples repository. On the left is the Explorer sidebar with a tree view of files and folders. The main area shows a terminal window with the command "s3-versioning-commands.sh" and its output, which includes AWS CLI commands for creating an S3 bucket, uploading files, and listing objects. A red arrow points from the text "An environment is known as a codespace." down to the browser window.

```
$ s3-versioning-commands.sh
s3 > s3-versioning > $ s3-versioning-commands.sh
1 # create a bucket without versioning
2 aws s3 mb s3://aws-s3-versioning-ab5345
3
4 # create a unversioned file
5 echo "Hello World" > myfile.txt
6
7 # copy the unversioned file to s3
8 aws s3 cp myfile.txt s3://aws-s3-versioning-ab5345
9
10 # show the contents of the s3 bucket notice that s3 ls does
11 aws s3 ls s3://aws-s3-versioning-ab5345
12
13 ## Get a more detailed listing of the bucket, noticed it doe
14 aws s3api list-objects --bucket aws-s3-versioning-ab5345
15
16 # get object versions and see what happens on an unversioned
17 aws s3api list-object-versions --bucket aws-s3-versioning-ab
18 # make note that versionid is null
19 # "VersionId": "null"
20
21 # turn on versioning for our s3 bucket
22 aws s3api put-bucket-versioning --bucket aws-s3-versioning-a
23
24 # confirm bucket versioning is turned on via the cli
25 aws s3api get-bucket-versioning --bucket aws-s3-versioning-a
26 # {
27 # "Status": "Enabled"
28 # }
```

# Introduction to GitHub Codespaces



In the VS Code Browser, you can tell it to connect this codespace To your locally installed VS Code

You can interact with your codespace via:

- Command Line
  - **SSH via GitHub CLI**
- Code Editors
  - VS Code Browser
  - **VS Code Desktop**
  - JetBrains IDE

The GitHub CLI commands available for Codespaces:

- gh codespace code
- gh codespace cp
- gh codespace create
- gh codespace delete
- gh codespace edit
- gh codespace jupyter
- gh codespace list
- gh codespace logs
- gh codespace ports
- gh codespace rebuild
- gh codespace ssh
- gh codespace stop
- gh codespace view

# Introduction to GitHub Codespaces

When you create a new codespace you **choose**:

- The repo
- The branch
- The Region
- Machine type



Create a new codespace

|                                                                 |                       |
|-----------------------------------------------------------------|-----------------------|
| <b>Repository</b><br>To be cloned into your codespace           | Select a repository ▾ |
| <b>Branch</b><br>This branch will be checked out on creation    | Default branch ▾      |
| <b>Region</b><br>Your codespace will run in the selected region | US East ▾             |
| <b>Machine type</b><br>Resources for your codespace             | None ▾                |
| <b>Create codespace</b>                                         |                       |

The **regions** you can choose:

- US East
- US West
- Europe West
- Southeast Asia
- Australias

The capacity of your environment is based on the virtual **machine type** you choose:

- 2-Core (8GB RAM, 32 GB)
- 4-Core (16GB RAM, 32 GB)
- 8-Core (32GB RAM, 64 GB)
- 16-Core (32GB RAM, 128 GB)

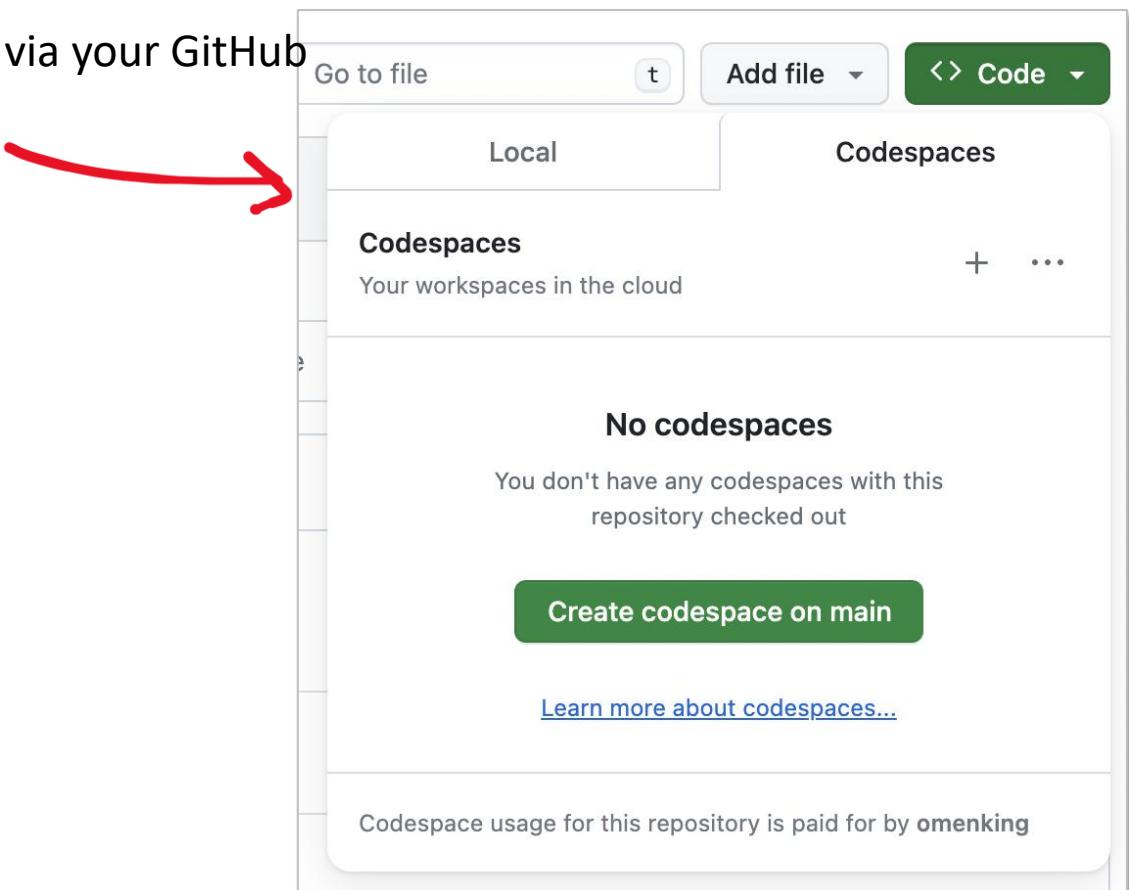
# Introduction to GitHub Codespaces

You can see all of your codespaces across all your repos at <https://github.com/codespaces>

The screenshot shows the GitHub Codespaces interface. At the top left, there are filters for 'All' (selected) and 'Templates'. Below this, a section titled 'Your codespaces' shows a repository named 'omenking/terraform-beginner-bootcamp-2023'. A red arrow points from the text above to this repository name. On the right side of the interface, there are four 'Explore quick start templates': 'Blank' (by github), 'React' (by github), 'Jupyter Notebook' (by github), and '.NET' (by github). Each template has a 'See all' link and a 'Use this template' button. At the bottom, there's a section titled 'Owned by omenking' which lists the repository 'dev-env2' (main branch, 0 changes). On the far right, resource details are shown: 2-core, 8GB RAM, 32GB storage, 0.92 GB used, Active status, and three dots for more options.

# Starting GitHub Codespaces

You can start a Codespace via your GitHub repo via the **Code** button



You can also use the **GitHub CLI** to create an open a codespace

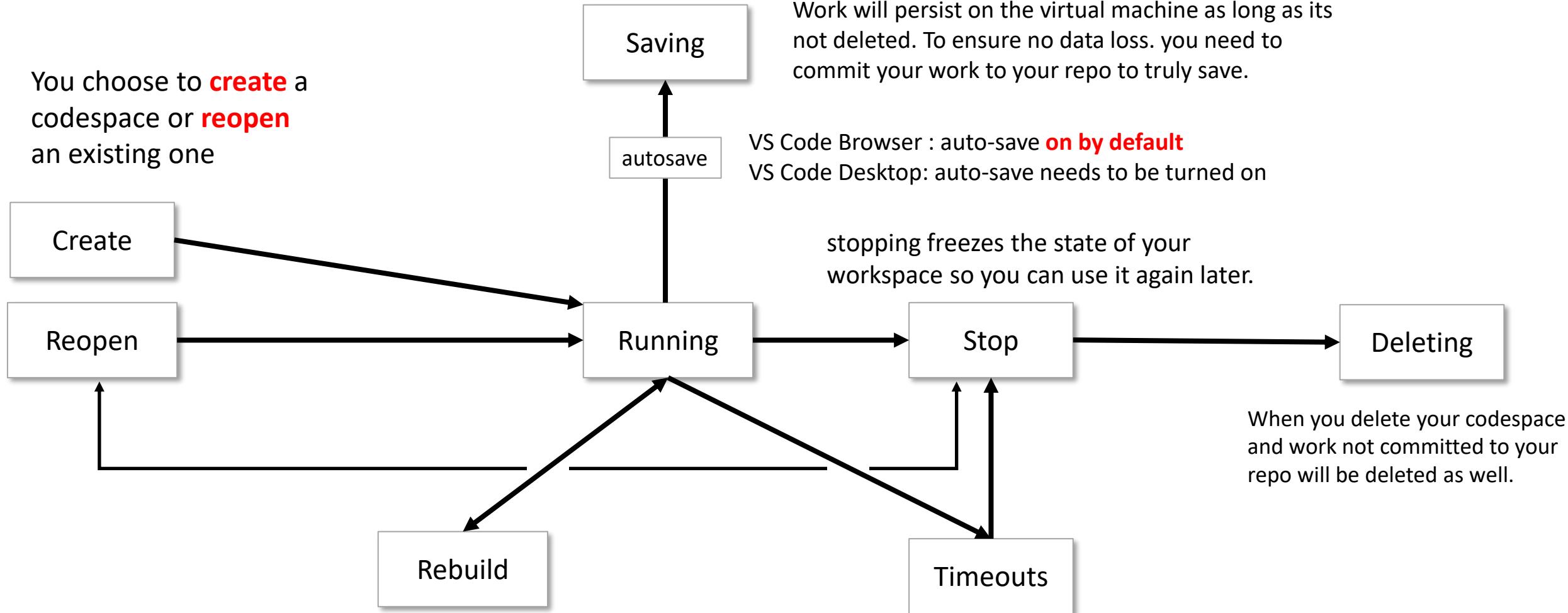
```
gh codespace create
gh codespace open
```



There is a limit of concurrent codespaces allowed to run.  
If you hit the limit you may need to stop other codespaces first.

# GitHub Codespace Lifecycle

You choose to **create** a codespace or **reopen** an existing one



If you made changes to your dev container configuration. You need to rebuild your codespace for them to take affect

Codespace by default will timeout after 30 mins of activity. The codespace will **stop** to conserve spend. Your changes will persist on the Virtual Machine

# Renaming Codespaces

When you create a codespace it's assigned an **auto-generated display name**.

shiny-space-chainsaw-jjgx4w5r3q9xj



You can rename the auto-generated name in the case you have multiple codespaces to easily identify the purpose or state of your codespace.

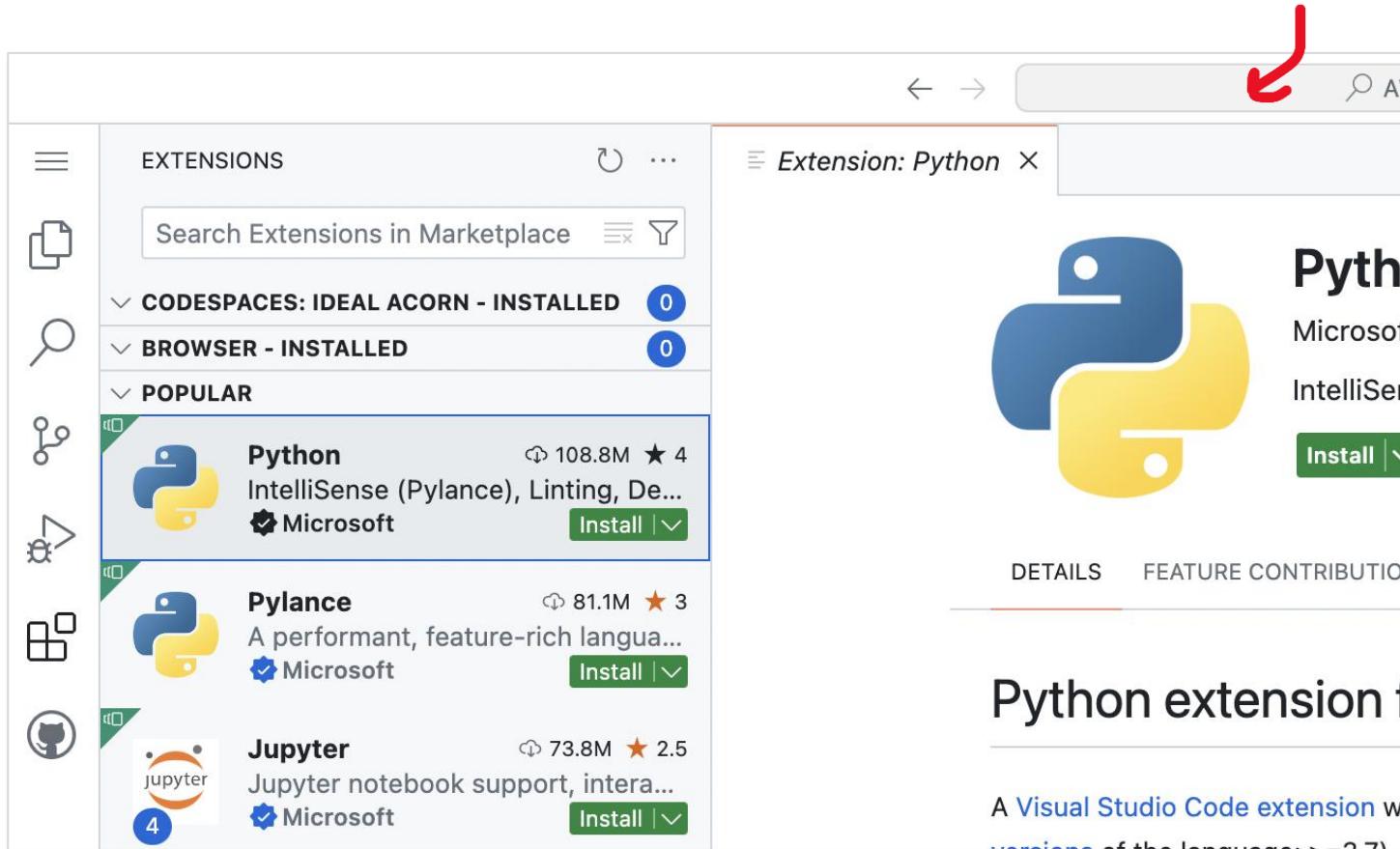
You can see all of your codespaces at

<https://github.com/codespaces>

The screenshot shows the GitHub Codespaces interface. At the top, there are tabs for 'Local' and 'Codespaces'. Below the tabs, the 'Codespaces' section is titled 'Codespaces' and describes 'Your workspaces in the cloud'. It lists two items: 'On current branch' and 'shiny space chainsaw'. The 'shiny space chainsaw' item is active, indicated by a green dot and the word 'Active'. To the right of this item are three dots. Below the list, there are buttons for 'Rename' (highlighted with a blue border) and 'Export changes to a branch'. A red arrow points from the text 'auto-generated display name' in the previous slide to the 'shiny space chainsaw' text in the screenshot.

# VS Code Extensions

GitHub Codespaces **should persist your VS Code extensions** that you change in your codespace.



**Most VSCode Marketplace extensions are available** to install in VSCode Browser for GitHub codespaces  
You can ensure your extensions reliably load every time by setting them in your devcontainer.json

# Retention Period and Timeout Configuration

Under **GitHub Settings > Codespaces** you can change the **timeout** and **retention period**.

## Default idle timeout

A codespace will suspend after a period of time if no activity occurs. You can change the default timeout after the default is changed. You will be charged for the duration of the codespace. The minimum is 5 minutes (4 hours).

30 minutes

Save

How long should codespaces wait to stop a codespace when there has been no activity.

- Minimum: 5 minutes
- Maximum 240 minutes (4 hours)
- Default: 30 minutes

## Default retention period

Inactive codespaces are automatically deleted after a period of time. This applies to all codespaces created going forward.

30 days

Save

How many days should pass until inactive codespaces are automatically deleted?

- Minimum: 0 days
- Maximum 30 days
- Default: 30 days

# Codespaces Sync Settings

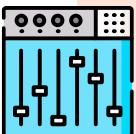
## Settings Sync

By enabling, your codespaces will be able to pull from VS Code Settings Sync service. Only enable this for repositories that you trust.

**Enable**

VS Code Settings Sync will be available in Codespaces

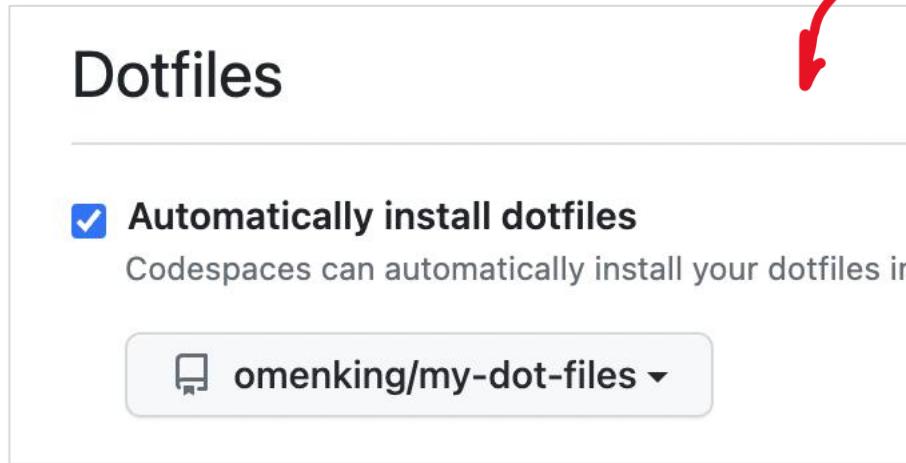
You can have codespaces able to pull from  
**VS Code Settings Sync service**



**VS Code Settings Sync** lets you share your Visual Studio Code configurations such as settings, keybindings, and installed extensions across your machines so you are always working with your favorite setup.

# Installing Dotfiles

You can configure GitHub Codespaces to **use dotfiles from a specific repository** that will be used for all future codespaces you start.



When you create a new codespace with this feature, GitHub does two things:

1. GitHub clones your selected dotfiles repository to the codespace environment.
2. Looks for one of the following files to set up the environment.
  - *install.sh*
  - *install*
  - *bootstrap.sh*
  - *bootstrap*
  - *script/bootstrap*
  - *setup.sh*
  - *setup*
  - *script/setup*

# GitHub Codespaces Deep Links

Github Codespaces Deep Link is an easy to way to generate a shareable link that will launch a Codespace for a specific Github Repo.

<https://codespaces.new/omenking/git-examples>

The screenshot illustrates the process of generating a GitHub Codespaces deep link. It consists of three main panels:

- Left Panel (Codespaces Overview):** Shows a list of workspaces under "Codespaces". A red arrow points from the "Share a deep link" button in the sidebar to the central "Share codespace configuration" panel.
- Middle Panel (Share codespace configuration):** Allows selecting settings for sharing. It includes a "Quick start" checkbox, a "Snippets" section with URL, HTML, and Markdown options, and a prominent "Open in GitHub Codespaces" button. A red arrow points from this panel to the generated URL in the top bar.
- Right Panel (Create a new codespace):** Provides configuration options for a new workspace, including repository ("omenking/git-exa..."), branch ("main"), region ("US East"), and machine type ("2-core"). A red arrow points from this panel to the "Create codespace" button at the bottom right.

The top bar displays the generated URL: <https://codespaces.new/omenking/git-examples>.

# Codespaces Secrets

Under **Github Settings > Codespaces** you can set GitHub Secrets

## Codespaces secrets

New secret

Development environment secrets are environment variables that are **encrypted**. They are available to any codespace you create using repositories with access to that secret.

|                                                                                                                                                |                         |                                                 |
|------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------|-------------------------------------------------|
|  <b>AWS_ACCESS_KEY_ID</b><br>Available to 1 repository.       | Updated on Nov 23, 2023 | <button>Update</button> <button>Delete</button> |
|  <b>AWS_REGION</b><br>Available to 1 repository.              | Updated on Nov 23, 2023 | <button>Update</button> <button>Delete</button> |
|  <b>AWS_SECRET_ACCESS_KEY</b><br>Available to 1 repository. | Updated on Nov 23, 2023 | <button>Update</button> <button>Delete</button> |

Secrets allows you to **set sensitive environment variables that will be loaded into your codespace**  
This avoids hardcoding secrets, passwords or sensitive values into your codebase.

# Dev Containers Configuration

Visual Studio Code Dev Containers allows you to **configure your docker container via json file.**

**devcontainer.json** file is expected in the root of your project.



```
{
 "name": "My Development Container",
 "image": "mcr.microsoft.com/vscode/devcontainers/base:0-focal",
 "settings": {
 "terminal.integrated.shell.linux": "/bin/bash"
 },
 "extensions": [
 "ms-python.python",
 "ms-vscode.cpptools"
],
 "forwardPorts": [3000],
 "postCreateCommand": "echo 'Container ready!',
 "remoteUser": "vscode"
}
```

# Dev Containers Configuration

You can **provide remote image** from a repo

```
"image": "mcr.microsoft.com/vscode/devcontainers/base:0-focal"
```

```
"build": {
 "dockerfile": "Dockerfile",
 "args": {
 "VARIANT": "3.8",
 "INSTALL_NODE": "true"
 },
},
```

You can **build your own image** by including the Dockerfile with your project

# Dev Containers Configuration

```
"features": {
 "github-cli": "latest",
 "node": {
 "version": "lts"
 },
 "python": {
 "version": "3.8"
 }
},
```

**Features** allows you to quickly install programs on top of a common base image.



You can create your own features to quickly install on top of your own base image.

To **custom configure program** generally you will install these via your custom Dockerfile

```
FROM python:3.8-slim
WORKDIR /usr/src/app
ENV TF_VER=1.0.0
ENV TF_PREFIX=https://releases.hashicorp.com/terraform
RUN apt-get update && apt-get install -y --no-install-recommends \
curl \
unzip \
git \
&& rm -rf /var/lib/apt/lists/*

RUN curl -o terraform.zip ${TF_PREFIX}/${TF_VER}/terraform_${TF_VER}_linux_amd64.zip \
&& unzip terraform.zip \
&& mv terraform /usr/local/bin/ \
&& rm terraform.zip

COPY . .
EXPOSE 80
ENV NAME World
CMD ["python", "app.py"]
```

You could use the `postCreateCommand` to **install a bunch of programs** ontop of your base container.

```
"postCreateCommand": "bash ./devcontainer/postCreateCommand.sh",
```

# Dev Containers Configuration

Settings allows you to configure the **settings of VS Code**

Eg. Setting editor font and terminal size

```
"settings": {
 "terminal.integrated.shell.linux": "/bin/bash",
 "python.pythonPath": "/usr/local/bin/python"
},
"extensions": [
 "ms-python.python",
 "ms-vscode.cpptools",
 "github.vscode-pull-request-github"
],
```

**Extensions** allows you to install VS Code extensions  
eg. VIM style keyboard controls

```
"forwardPorts": [3000, 5000],
```

You can list **ports you always want to forward**

# Dev Containers Configuration

Command flags and their values we want to pass to when **docker run** command.

```
"runArgs": [
 "-e", "MY_ENV_VAR=myvalue",
 "-e", "ANOTHER_ENV_VAR=anothervalue",
 "-v", "/my/host/folder:/container/folder",
 "--network=host",
 "--user=$(id -u):$(id -g)",
 "--cpus=2.0",
 "--memory=2g",
 "--shm-size=1g",
 "--add-host=example.com:192.168.1.10",
 "--privileged",
 "--security-opt", "seccomp=unconfined",
 "--network-alias=myalias",
 "-e", "TZ=Europe/London",
 "-v", "/var/run/docker.sock:/var/run/docker.sock"
]
```

# Dev Containers Configuration

## Environment Variables

we want to pass to the running container



```
"containerEnv": {
 "NODE_ENV": "development",
 "PYTHONUNBUFFERED": "1",
 "TZ": "America/New_York",
 "LANG": "en_US.UTF-8",
 "HTTP_PROXY": "http://proxy.example.com:8080",
 "HTTPS_PROXY": "http://proxy.example.com:8080",
 "NO_PROXY": "localhost,127.0.0.1,.example.com",
 "JAVA_HOME": "/usr/lib/jvm/java-11-openjdk-amd64",
 "MAVEN_HOME": "/usr/share/maven",
 "DOCKER_TLS_VERIFY": "1",
 "DOCKER_HOST": "tcp://docker.example.com:2376",
 "DOCKER_CERT_PATH": "/certs/client",
 "DATABASE_URL": "postgresql://user:password@localhost:5432/mydatabase",
 "REDIS_URL": "redis://localhost:6379",
 "API_KEY": "your-api-key",
 "LOG_LEVEL": "debug"
}
```

# Dev Containers Configuration

Mounts allows you to **map files and folders to your containers**

```
"mounts": [
 "source=${localWorkspaceFolder}/data,target=/container/data,type=bind,consistency=cached",
 "source=myvolume,target=/container/volume,type=volume",
 "source=/var/run/docker.sock,target=/var/run/docker.sock,type=bind",
 "source=${localWorkspaceFolder}/config/settings.json,target=/container/settings.json,type=bind,consistency=cached",
 "source=${localWorkspaceFolder}/docs,target=/container/docs,type=bind,consistency=cached,readonly"
]
```

- **Bind Mount a Local Directory:** Mount a local directory for direct file access.
- **Mount a Docker Volume for Persistent Storage:** Use a Docker volume for data that persists across container rebuilds.
- **Mount the Docker Socket:** Allow the container to interact with the host's Docker daemon.
- **Mount a Specific File:** Mount a specific configuration file into the container.
- **Mount a Host Directory as ReadOnly:** Mount a directory from the host as read-only.

# Dev Containers Configuration

command that **runs after creating the container**.

```
"postCreateCommand": "echo 'Container ready!',
"remoteUser": "vscode"
```

the **default username** that VS Code should use when connecting to the container.

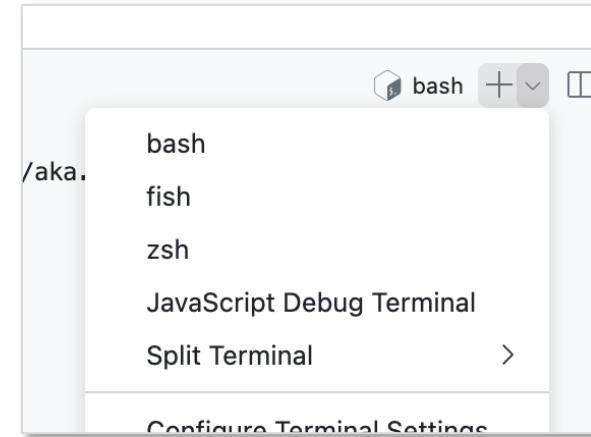
You could use the postCreateCommand to **install a bunch of programs** ontop of your base container.

```
"postCreateCommand": "bash ./devcontainer/postCreateCommand.sh",
```

# Changing Shell

GitHub Codespaces VS Code Browser default devcontainer images comes with:

- Bash (default)
- Zsh
- Fish



If you build or provide an image to devcontainer.json with other shells installed, you can define and provide the path to that shell for use:



```
{
 "terminal.integrated.profiles.linux": {
 "csh": {
 "path": "csh"
 }
 },
 "terminal.integrated.defaultProfile.osx": "zsh",
 "terminal.integrated.defaultProfile.linux": "bash",
 "terminal.integrated.defaultProfile.windows": "PowerShell"
}
```

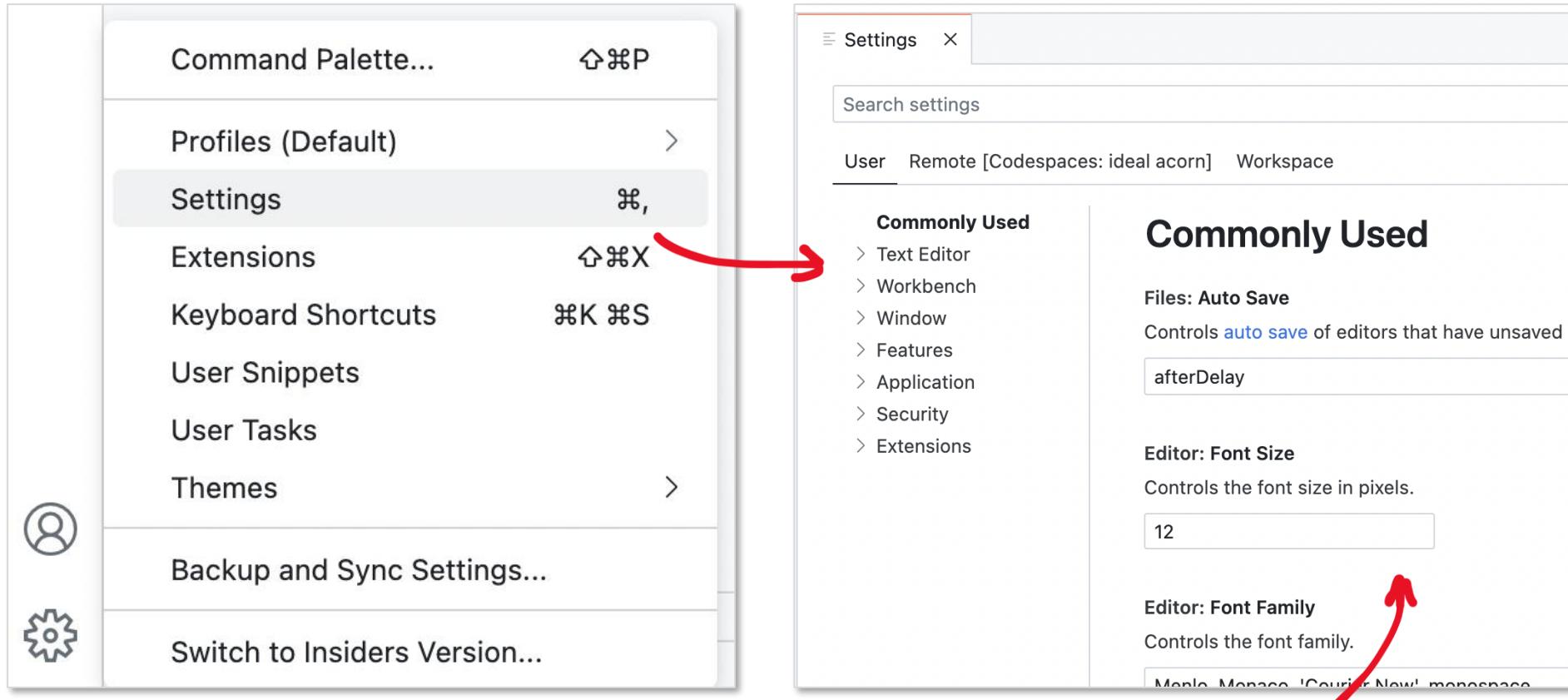
You can configure the settings.json in the VSCode to change the defaults based on Operating System.



*settings.json is directly edited in the UI of VS Code.*

# VS Code Settings

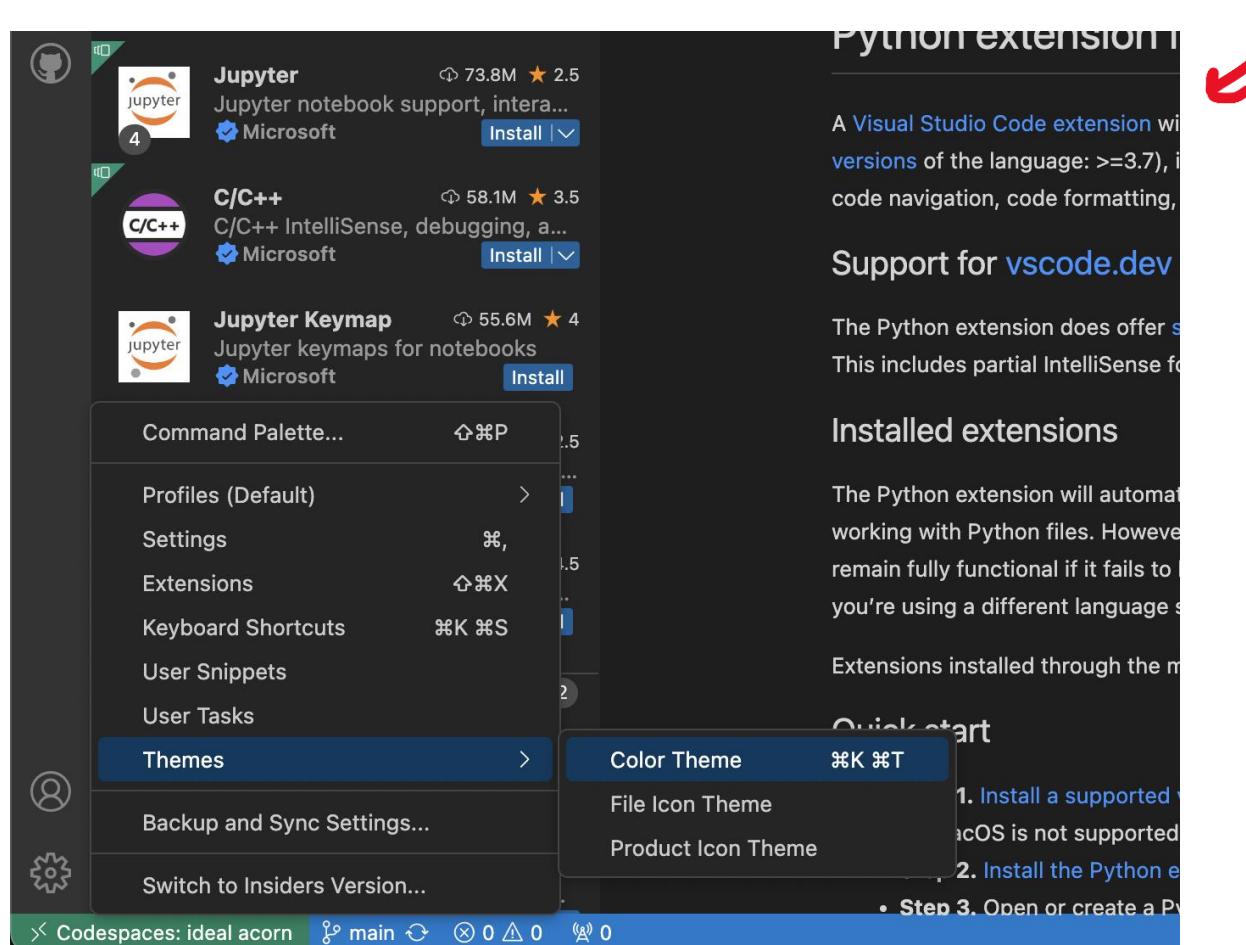
Github Codespaces **should persist your VS Code settings** that you change in your codespace.



When you edit Settings in the UI you are **actually modifying a settings.json file**.  
It gets synced and backed up hosted by GitHub  
You can override setting.json options in your devcontainer.json file

# VS Code Theme

Github Codespaces **should persist your VS Code theme** that you change in your codespace.



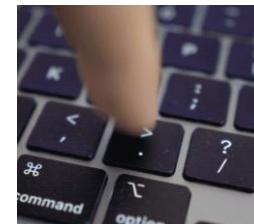
user interface decorators or themes are personal choices that should not be put in the devcontainer.json file

# GitHub.dev Editor

**Github.dev Editor** is a **VS Code browser** that instantly loads that has no attached compute.

There are two ways you can open your GitHub repo in Github.dev:

1. Press the period (.) hotkey on your keyboard when you have your GitHub Repo page opened.



2. Place your GitHub Path after the URL:

github.dev/ **<user/org>** / **<repo name>**



Github.dev is intended for **quick changes to file and committing code** in an IDE experience.

```
1 terraform {
2 required_providers {
3 terratowns = {
4 source = "local.providers/local/terratowns"
5 version = "1.0.0"
6 }
7 }
8 #backend "remote" {
9 # hostname = "app.terraform.io"
10 # organization = "ExamPro"
11 }
12 # workspaces {
13 # name = "terra-house-1"
14 }
15 #}
16 cloud {
17 organization = "ExamPro"
18 workspaces {
19 name = "terra-house-1"
20 }
21 }
22
23 provider "terratowns" {
24 endpoint = var.terratowns_endpoint
25 user_uuid = var.teacherseat_user_uuid
26 token = var.terratowns_access_token
27 }
28
29}
```

# GitHub.dev vs GitHub Codespaces

|                 | GitHub.dev                                                                                                          | Codespaces                                                                                                                                                                                 |
|-----------------|---------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Cost            | FREE                                                                                                                | <ul style="list-style-type: none"><li>• Free monthly quota</li><li>• Storage and Core hours per month<ul style="list-style-type: none"><li>• Machine type changes cost</li></ul></li></ul> |
| Availability    | Available to everyone on GitHub.com.                                                                                |                                                                                                                                                                                            |
| Start up        | <ul style="list-style-type: none"><li>• Instantly launches.</li><li>• No devcontainer.json configuration</li></ul>  | <ul style="list-style-type: none"><li>• Takes a few minutes to startup</li><li>• Does load devcontainer.json</li></ul>                                                                     |
| Compute         | <ul style="list-style-type: none"><li>• There is no attached compute</li><li>• You can't run code or apps</li></ul> | <ul style="list-style-type: none"><li>• Dedicated attached VM</li><li>• Run and debug your code and apps</li></ul>                                                                         |
| Terminal access | No terminal                                                                                                         | Terminal access                                                                                                                                                                            |
| Extensions      | Limited subset of extensions                                                                                        | Most extensions for them VS Code Marketplace                                                                                                                                               |

# Open Source



**Open-source** is **source code made freely available** for possible modification and redistribution.

Open source benefits include:

- Encourages global collaboration.
- Speeds up innovation.
- Offers adaptability and customization.
- Reduces software costs.
- Enhances learning for developers.
- Typically high-quality and reliable.
- Provides transparency for trust and security.

Open-source software has often have **free-community versions** which makes it easy for personal developers or small organizations to quickly adopt technology.

| Ranking | Project                       | Leading company          | Market value   |
|---------|-------------------------------|--------------------------|----------------|
| 1       | <a href="#">Linux</a>         | <a href="#">Red Hat</a>  | \$16 billion   |
| 2       | <a href="#">Git</a>           | <a href="#">GitHub</a>   | \$2 billion    |
| 3       | <a href="#">MySQL</a>         | <a href="#">Oracle</a>   | \$1.87 billion |
| 4       | <a href="#">Node.js</a>       | NodeSource               | ?              |
| 5       | <a href="#">Docker</a>        | Docker                   | \$1 billion    |
| 6       | <a href="#">Hadoop</a>        | <a href="#">Cloudera</a> | \$3 billion    |
| 7       | <a href="#">Elasticsearch</a> | Elastic                  | \$700 million  |
| 8       | <a href="#">Spark</a>         | Databricks               | \$513 million  |
| 9       | <a href="#">MongoDB</a>       | MongoDB                  | \$1.57 billion |
| 10      | <a href="#">Selenium</a>      | Sauce Labs               | \$470 million  |

# Open Source



The Open Source Initiative (OSI) is a non-profit organization is the steward of the Open Source Definition, the set of rules that define open source software

THE OSI maintains **a list of open-source licensing documents** for all different use-cases which are readability available for project owners to adopt

The Top five most popular open-source licenses:

- **MIT License:**
  - Known for its simplicity and permissiveness.
- **GNU General Public License (GPL):**
  - Ensures that modified versions remain open.
- **Apache License 2.0:**
  - Provides explicit patent rights grants.
- **GNU Lesser General Public License (LGPL):**
  - Similar to GPL but with some allowances for linking to proprietary software.
- **BSD License:**
  - Characterized by minimal restrictions on redistribution.

| License Name                                         |
|------------------------------------------------------|
| <a href="#">1-clause BSD License</a>                 |
| <a href="#">Academic Free License v. 3.0</a>         |
| <a href="#">Adaptive Public License 1.0</a>          |
| <a href="#">Apache License, Version 2.0</a>          |
| <a href="#">Apache Software License, version 1.1</a> |

# GitHub Sponsors

GitHub Sponsors allows GitHub users **to collect donations for their GitHub hosted open-source projects.**

Sponsorship payments are facilitated two ways:

- Payments through GitHub on Github.com
- Payments through Patreon on Github.com

You can turn sponsor on for a **specific repo**

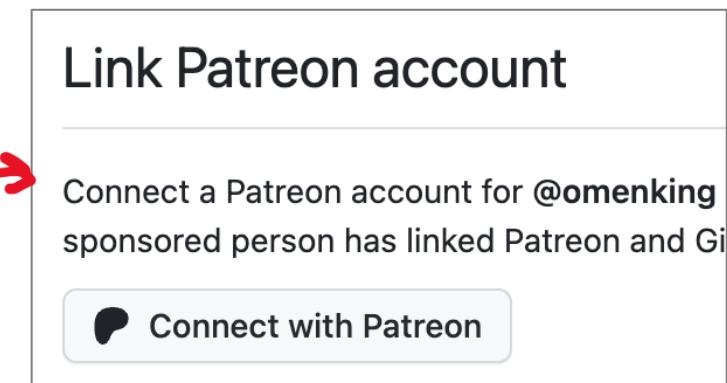
**Sponsorships** ✓  
Sponsorships help your community know how to financially support this repository.

Display a "Sponsor" button  
Add links to GitHub Sponsors or third-party methods your repository accepts for financial contributions to your project.

**Set up sponsor button**



You can **connect Patreon** via your Account settings



# GitHub Sponsors

To receive sponsorship through GitHub on Github.com you'll need to get accepted into **GitHub Sponsors**.

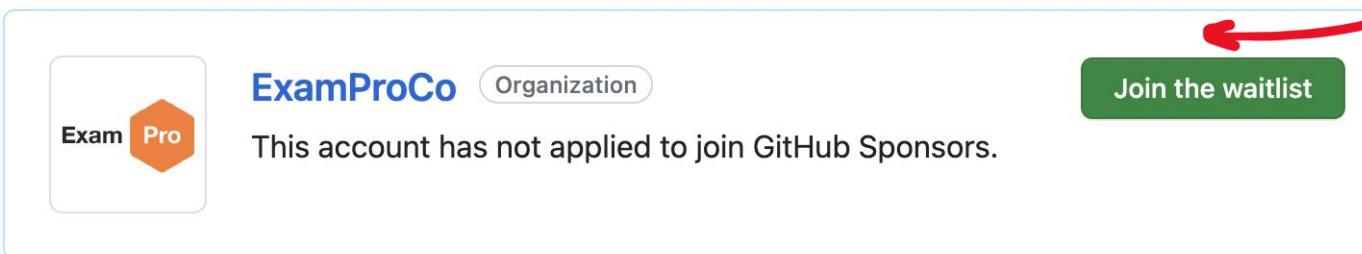
<https://github.com/sponsors>



omenking  
This account has not applied to join GitHub Sponsors.

Join the waitlist

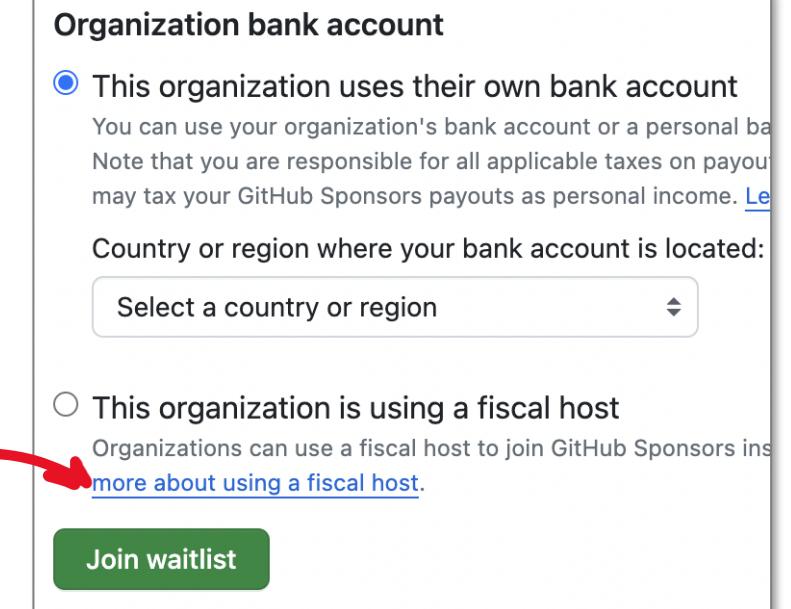
**Personal accounts** can apply



ExamProCo Organization  
This account has not applied to join GitHub Sponsors.

Join the waitlist

**Organizational accounts** can apply



Organization bank account

This organization uses their own bank account  
You can use your organization's bank account or a personal bank account. Note that you are responsible for all applicable taxes on payouts from GitHub Sponsors. GitHub may tax your GitHub Sponsors payouts as personal income. [Learn more](#)

Country or region where your bank account is located:

Select a country or region

This organization is using a fiscal host  
Organizations can use a fiscal host to join GitHub Sponsors instead of connecting their own bank account. [Learn more about using a fiscal host.](#)

Join waitlist

Setting up GitHub sponsors via GitHub requires you to connect your bank account and may have **other fiscal requirements**.

# GitHub Sponsors

GitHub makes it easy to locate people who maintains your dependencies so you can **support open-source contributors**

<https://github.com/sponsors/explore>

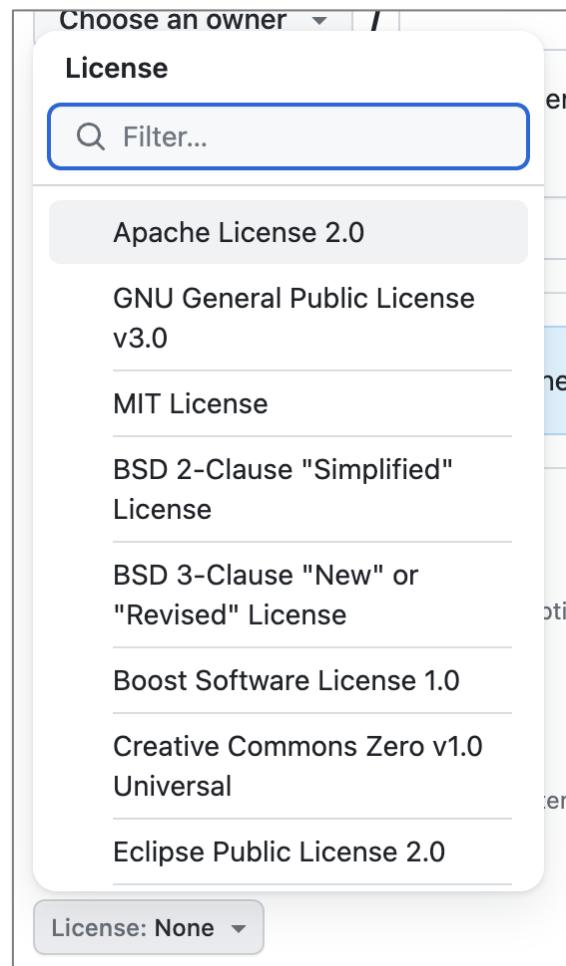
The screenshot shows the GitHub Sponsors explore page. At the top, it displays "123 developers who maintain your dependencies". A red arrow points from the URL bar down to this text. Below this, there's a "Bulk Sponsor" button and a "Get started" button. The main content area lists three maintainers with their profiles and sponsorship buttons:

- sindresorhus**: You depend on 195 repositories they own or maintain. Sponsor button.
- jonschlinkert**: You depend on 88 repositories they own or maintain. Sponsor button. Progress bar shows 34% towards goal.
- ljharb**: You depend on 77 repositories they own or maintain. Sponsor button.

On the left sidebar, under "Explore as", the user "omenking" is selected. Other options include "ExamProCo", "teacherseat", and "monsterboxpro". There's also a "See more" link. Under "Ecosystem", links are provided for "All ecosystems", "Cargo", "composer", "GitHub Actions", and "go".

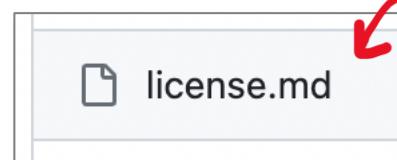
# GitHub and Open source projects

When creating a new repo GitHub makes it easy to  
**quickly add** a selected Open source license to your repo



They will **display** this on the GitHub repo page

A screenshot of a GitHub repository page. The top navigation bar includes 'README', 'Code of conduct', 'MIT license' (which is underlined in orange), and 'Security'. An edit icon is located at the top right. The main content area displays the 'The MIT License (MIT)' text, the copyright notice '(c) 2024 Vercel, Inc.', and the full legal text of the MIT license, which is identical to the one shown in the creation step above.



The file in the repo will be listed as either:  
• license.md, LICENSE or LICENSE.md

# Open source discovery

GitHub **search** makes it easy to find public repo's based on Open source license

A screenshot of the GitHub search interface. At the top is a search bar with the placeholder "license:". Below it is a dropdown menu titled "Values" containing the following options: "BSD Zero Clause License", "MIT License", "Apache License 2.0", "Creative Commons", and "GNU General Public License". A red arrow points from the text "GitHub search makes it easy to find public repo's based on Open source license" to the search bar.

Many open-source GitHub repos are hosted or mirrored on GitHub.

Just type the open-source project name into search

A screenshot of the GitHub search results for the project "expressjs/express". The search bar at the top shows "expressjs/express". The results summary is "645k results (422 ms)". The top result is "expressjs/express" with the description "Fast, unopinionated, minimalist web framework for node.". It includes tags for "nodejs", "javascript", "express", and "server". Below the description are the language "JavaScript", stars "62.7k", and the last update "Updated 6 days ago". A red arrow points from the text "Just type the open-source project name into search" to the search bar.

In Explore:

- GitHub Topics allows you to find repos for specific categories eg. Azure
- GitHub Trending: shows you most popular repos

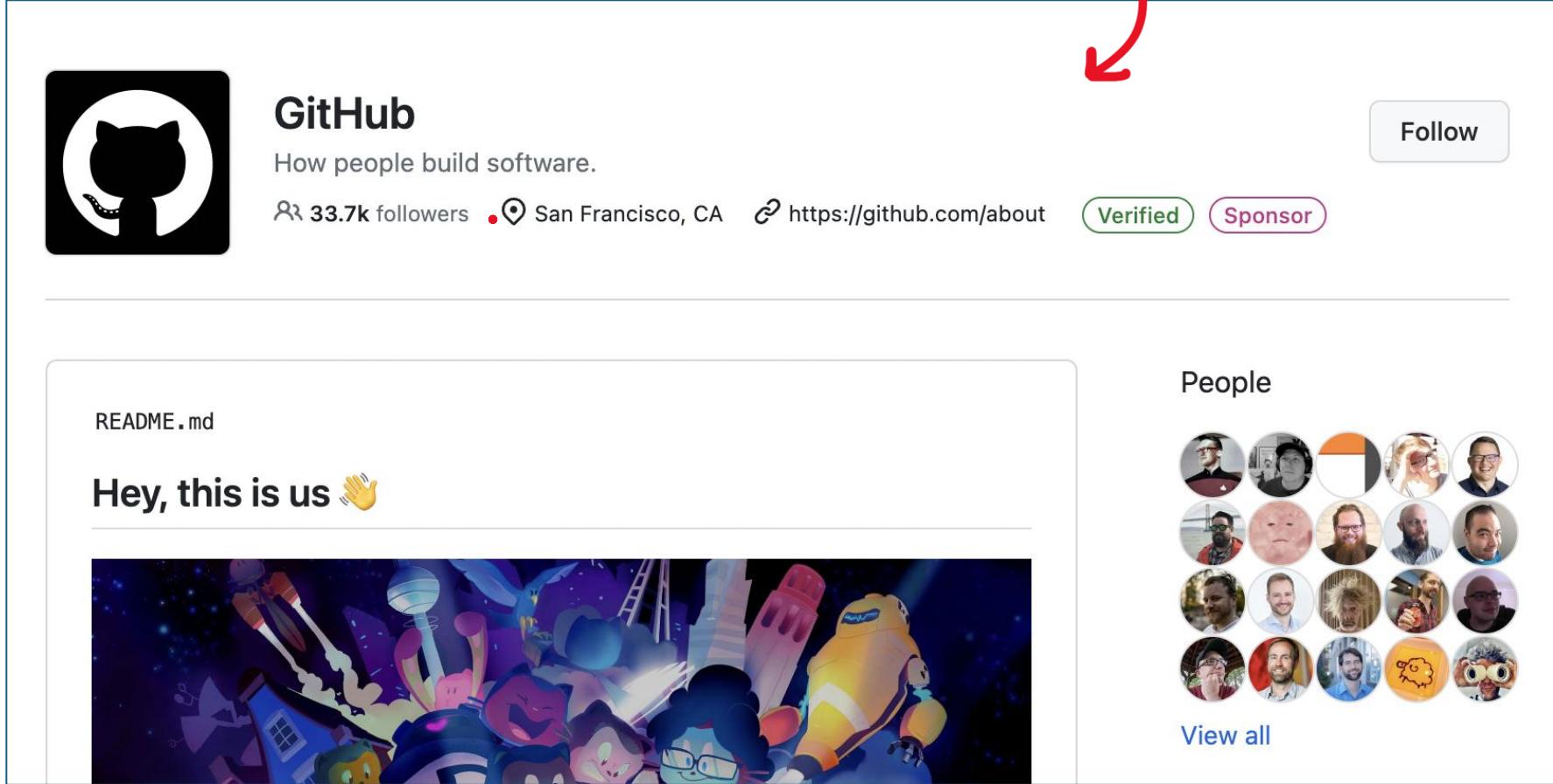
A screenshot of the GitHub Trending page. The navigation bar at the top includes "Explore", "Topics", "Trending" (which is underlined), "Collections", "Events", and "GitHub Sponsors". The main heading is "Trending" with the subtext "See what the GitHub community is most excited about today.". Below this is a list of trending repositories, starting with "mckaywrigley / chatbot-ui". The interface includes filters for "Repositories" and "Developers", and dropdowns for "Spoken Language: Any", "Language: Any", and "Date range: Today". A red arrow points from the text "In Explore:" to the "Trending" tab in the navigation bar.

# Following Organizations

On a GitHub Organizations public profile, you can **choose to follow** them to notifications of their public activity on your personal dashboard

This activity includes:

- new discussions
- sponsorships
- repositories



A screenshot of a GitHub Organization profile page for "GitHub". The profile features the GitHub logo, the text "GitHub How people build software.", and statistics like 33.7k followers and a location in San Francisco, CA. A prominent "Follow" button is visible on the right. A large red arrow points from the text above to this "Follow" button. Below the main profile area, there's a "README.md" section containing the text "Hey, this is us 🙌" and a colorful illustration of various cartoonish characters. To the right, there's a "People" section showing a grid of user profiles and a "View all" link.

GitHub  
How people build software.

33.7k followers • San Francisco, CA • https://github.com/about

Follow

Verified Sponsor

README.md

Hey, this is us 🙌

People

View all

# GitHub Marketplace

GitHub Marketplace are apps that **integrate with your GitHub repos** to provide additional functionality.

Apps can be **free or paid**:



**Types**

- Apps
- Actions

**Categories**

- API management
- Chat
- Code quality
- Code review
- Continuous integration
- Dependency management
- Deployment
- IDEs
- Learning

Search for apps and actions

## Apps

**CircleCI**  
By circleci  
Automatically build, test, and deploy your project in minutes  
Recommended

**CodeFactor**  
By codefactor-io  
Automated code review for GitHub  
Recommended

**Codecov | Code Coverage**  
By codecov  
Automatic test report merging for all CI and languages into a single code coverage report directly into your pull request

**Pricing and setup**

| Plan        | Price                                                      |
|-------------|------------------------------------------------------------|
| Open Source | \$0<br>Free for open source projects                       |
| Pro         | \$180<br>per user / year<br>Unlimited private repositories |

**Codacy**  
**Open Source**  
Free for open source projects  
Account:  Search by name  Next: Confirm your installation location.

Codacy is provided by a third-party and is governed by separate [terms of service](#), [privacy policy](#), and [support contact](#).

**Rewind Backups for GitHub**  
By backhub  
Automatic daily backups of your GitHub repos and metadata with on-demand restores to protect your business  
Recommended

**Zenhub**  
By ZenHubIO  
Agile Task Boards, Epics, Estimates and Reports, all within GitHub's UI

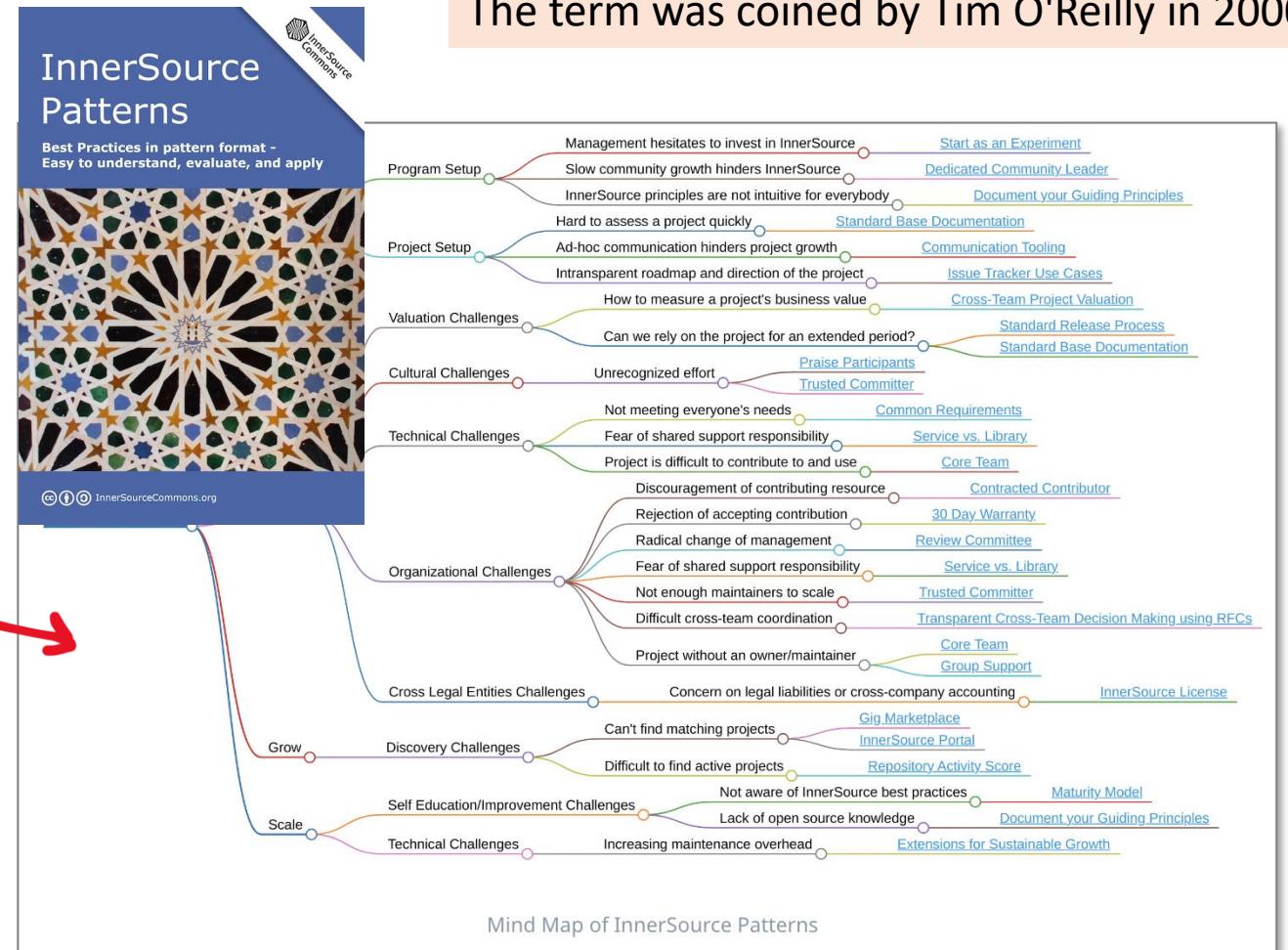
# InnerSource

InnerSource is organization and development best practices of non-open-source and/or proprietary software.



InnerSource is **not a strict guideline** but **a loose strategy** to establish an open source-like culture within organizations

Innersourcecommons.org provide several **free structured resources** to help understand how to adopt InnerSource



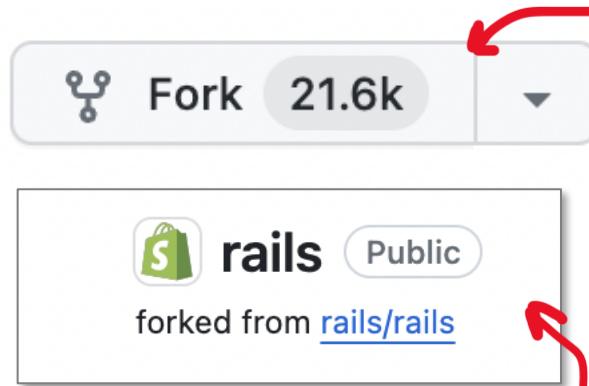
<https://patterns.innersourcecommons.org/explore-patterns>

The term was coined by Tim O'Reilly in 2000

# InnerSource vs Open source

|               | InnerSource                                            | OpenSource                                                   |
|---------------|--------------------------------------------------------|--------------------------------------------------------------|
| Scope         | Within an organization                                 | Public, global community                                     |
| Accessibility | Limited to company employees                           | Open to anyone                                               |
| Purpose       | Improve collaboration and efficiency internally        | Share and collaborate on projects globally                   |
| Contribution  | Employees of the organization                          | Anyone (developers, users, etc.)                             |
| Visibility    | Code and discussions often private to the organization | Publicly available and visible                               |
| Governance    | Dictated by internal policies and culture              | Usually governed by open source licenses and community rules |

# Forking



A forked repo is indicated **underneath its repo name.**

You may prefer using a forked repo over the original.

You may find original projects abandon, and a forks becoming the go-to repo for the project.

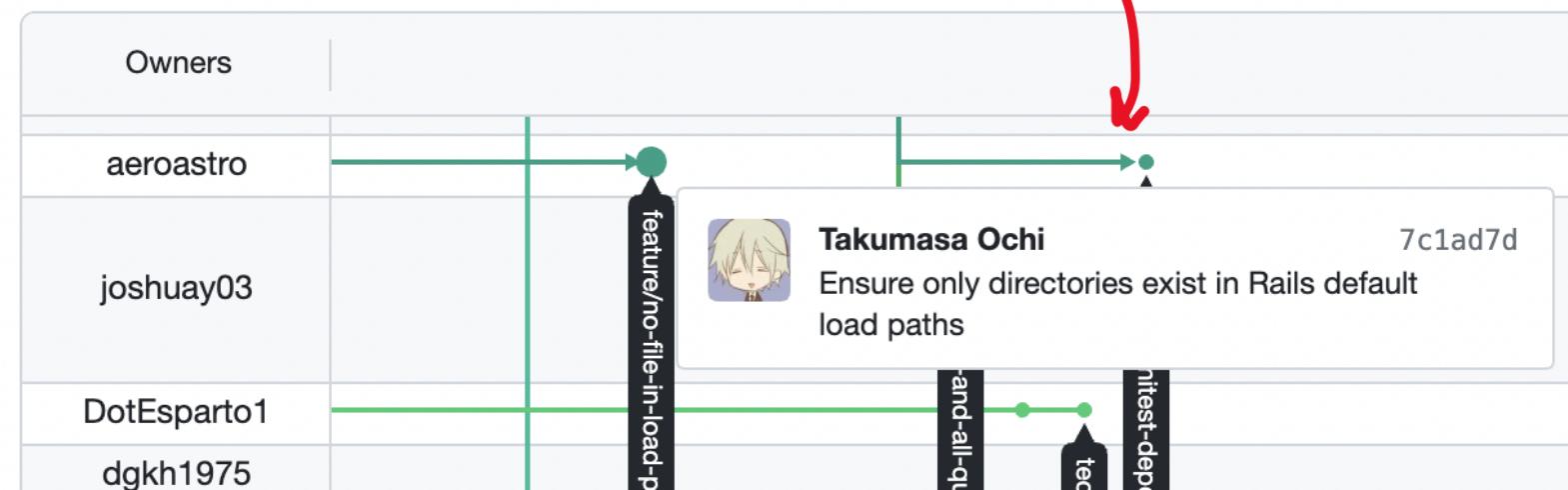
Fork's help keep public projects alive, on the edge and collaborative

**Forking repo's** allows you to create a copy of a repo.

Forking allows you to:

- Take an open-source or source available repo and go your own way
- Quickly apply bugs and patch fixes
- A separate repo to work on community contributions
  - You can create cross-repo pull-requests to get accepted in the original repo

Using **Network Graph** under Repo Insights you can often find community solutions to bugs or missing features.



# Discoverable repos

GitHub Repos can be set as public, making repos easily searchable on GitHub and via search engines. Public repos serve a community purpose for knowledge sharing, educational resources and open-source projects.

Besides just search GitHub curates content via their community pages:

<https://github.com/explore>

The screenshot shows the GitHub Explore page. On the left, there's a sidebar with a profile picture of Andrew Brown, his name, and links for starred topics (0) and repositories (109). The main content area features a large banner for "GitHub Unwrapped Best Projects 2023" with a video thumbnail of a woman speaking. Below the banner, there's a section titled "Here's what we found based on your interests..." with several repository cards. One card for "mckaywrigley / chatbot-ui" is highlighted. At the bottom, there are sections for "Trending developers" and "Based on repositories you've viewed".

<https://github.com/trending>

The screenshot shows the GitHub Trending page. It has a header with tabs for Explore, Topics, Trending, Collections, Events, and GitHub Sponsors. The "Trending" tab is selected. The main content area is titled "Trending" and says "See what the GitHub community is most excited about today." It features a list of repositories with cards for "mckaywrigley / chatbot-ui", "aaamoon / copilot-gpt4-service", and "subquery / subql". Each card includes the repository name, description, programming language, star count, and other details like built-by icons and stars for the day.

# Discoverable repos

GitHub provides a **robust search** that lets us not only repo's names but the contents of their repos.

A screenshot of the GitHub search interface. The search bar at the top contains the query "rails". Below the search bar, there are filter options under "Filter by" and "Languages". The "Repositories" filter is selected, showing 479k results. The main area displays four repository cards:

- rails/rails**: Ruby on Rails, 54.2k stars. Description: Ruby on Rails (Rails) is a web application framework written in Ruby. Links: rubyonrails.org, Wikipedia, rails.
- capistrano/rails**: Official Ruby on Rails specific tasks for Capistrano, 854 stars. Description: Official Ruby on Rails specific tasks for Capistrano. Links: rails, deployment, capistrano.
- railsadminteam/rails\_admin**: RailsAdmin is a Rails engine that provides an
- Sponsor open source projects you depend on

GitHub also has several **advanced search** options

A screenshot of the GitHub advanced search interface. It shows a list of search filters:

- + Owner
- + Size
- + Number of followers
- + Number of forks
- + Number of stars
- + Date created
- + Date pushed
- + Topic
- + License
- + Archived
- + Public
- + Private

At the bottom, there is a link to "Advanced search". A red arrow points from the text "GitHub also has several advanced search options" to this link.

# Discoverable repos – Search Syntax

GitHub global search box has a **search syntax** to help quickly apply advanced search options

A screenshot of the GitHub search interface. In the top search bar, the query `repo:omenking` is entered. Below the search bar, a list of repositories owned by the user `omenking` is displayed. The repositories listed are: `omenking/ampt-example`, `omenking/aws-cli-labs`, `omenking/aws-bootcamp-cruddur-2023`, `omenking/ampt-javascript-express-api`, and `omenking/terraform-beginner-bootcamp-2023`. A red arrow points from the text "Repositories owned by a specific user" in the table below to the search bar.

## Search Syntax

`language:python`

`stars:>100`

`forks:>=50`

`size:<1000`

## **user:username**

`repo:username/repository`

`filename:file.ext`

`path:/docs/`

`extension:py`

`is:issue is:open`

`is:pr is:closed`

## Description

Finds repositories written in Python.

Repositories with more than 100 stars.

Repositories forked at least 50 times.

Repositories smaller than 1000 KB.

Repositories owned by a specific user.

Specific repository from a user.

Search for a specific file.

Files in a specific directory.

Files with a .py extension.

Open issues.

Closed pull requests.



Search syntax also supports : **regular expressions, wildcards, NOT, OR AND, multiple terms and more**

# Labels

GitHub Labels are used to **categorize** issues, pull requests, and discussions

The image shows a screenshot of the GitHub Labels interface on the left and a GitHub issue card on the right. The interface includes a sidebar with filters like Labels, Milestone, Status, etc., and a main area for selecting labels. A search bar at the top says 'enhancement'. Below it is a list of labels: enhancement (checked), bug, documentation, duplicate, good first issue, help wanted, invalid, question, and wontfix. Red arrows point from the text 'You can apply multiple Labels to a labeled item' to the checked 'enhancement' label in the interface and to the label on the issue card.

Labels

enhancement

Milestone

Status

Linked pull requests

Repository

Priority

Size

Estimate

Iteration

Start date

End date

Select items

Search labels

enhancement

bug

documentation

duplicate

good first issue

help wanted

invalid

question

wontfix

You can **apply multiple Labels** to a labeled item

Finish the Github Foundations Certification course **enhancement**

#1 opened 18 hours ago by omenking

# Labels

You can **manage your labels** or create new labels on the Labels page for your repo

<https://github.com/<username/org>/<repo-name>/labels>



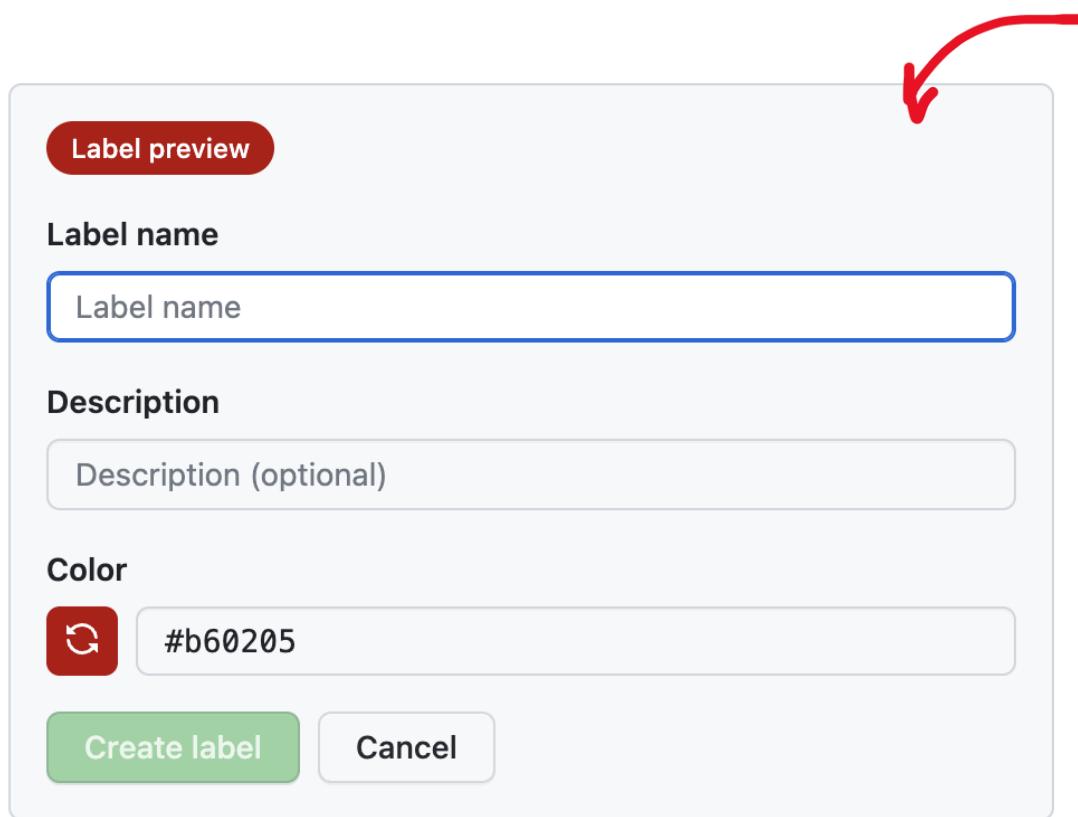
The screenshot shows the GitHub repository interface. The top navigation bar includes links for Code, Issues (with 2 notifications), Pull requests (with 1 notification), Discussions, Actions, Projects (with 1 notification), Wiki, Security, Insights (which is highlighted with a red arrow), and Settings.

The main content area is titled "Labels" and contains a search bar with "Search all labels". A green "New label" button is located in the top right corner of this section.

The "Labels" section displays 9 labels:

- bug**: Something isn't working. Actions: Edit, Delete.
- documentation**: Improvements or additions to documentation. Actions: Edit, Delete.
- duplicate**: This issue or pull request already exists. Actions: Edit, Delete.
- enhancement**: New feature or request. Actions: Convert to discussions, Edit, Delete. A circular icon indicates 1 issue associated with this label.
- good first issue**: Good for newcomers. Actions: Edit, Delete.

# Labels



**Creating a label** as simple as providing a:

- Name
- Description
- Colour

Github CLI provides commands for labels

- gh label clone
- gh label create
- gh label delete
- gh label edit
- gh label list

```
gh label create bug \
--description "Something isn't working" \
--color E99695
```

# Milestones

Milestones allow you to **group multiple issues into an end goal** which show completion towards the goal for each closed issue

The screenshot shows a project management interface with the following elements:

- Header:** Code, Issues (1), Pull requests (1), Discussions, Actions, ...
- Filter Bar:** Labels, Milestones (selected).
- Status Summary:** 1 Open, 0 Closed.
- Issue Card:** **Launch this Course!**
  - Due by January 16, 2024
  - Last updated less than a minute ago
  - I need to get this course done. Pronto!
  - Progress: 50% complete (1 open, 1 closed)
  - Actions: Edit, Close, Delete
- Milestone Dialog:** Set milestone, Filter milestones, Open, Closed, Launch this Course!, Due by January 16, 2024.

Annotations with red arrows:

- A red arrow points from the text "group multiple issues into an end goal" to the "Milestones" button in the filter bar.
- A red arrow points from the text "show completion towards the goal for each closed issue" to the progress bar and status summary.
- A red arrow points from the text "You associate Issues to Milestones" to the "Set milestone" field in the dialog.

# Introduction to GitHub Projects

GitHub Projects is a **planning and tracking tool** when on working on GitHub Repo.

A project has an adaptable view that can be changed at anytime between:

- Spreadsheet (Table)
- Taskboard (Board)
- Roadmap (Roadmap)

Projects **directly integrate** with Issues and Pull Requests.

Projects have built-in workflows to **automate common actions**.

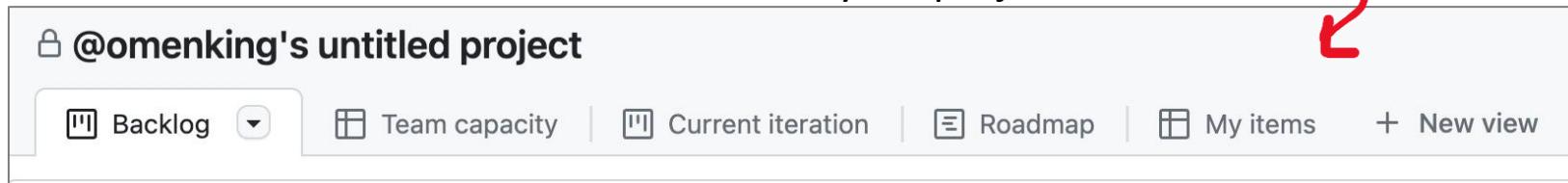
The screenshot shows the 'Create project' interface on GitHub. On the left, there's a sidebar with 'Project templates' and 'Featured' sections. Below that are three options: 'Start from scratch', 'Table', 'Board', and 'Roadmap'. A red arrow points to the 'Project templates' section. To the right, there are four cards, each showing a different project template:

- Team planning • GitHub**: Manage your team's work items, plan upcoming cycles, and understand team capacity. It shows a Kanban board with columns like Backlog, In Progress, and Done.
- Feature release • GitHub**: Manage your team's prioritized work items when planning for a feature release. It shows a table with rows for different tasks.
- Kanban • GitHub**: Visualize the status of your project and limit work in progress. It shows a Kanban board with columns like Backlog, Ready, In Progress, and In Review.
- Bug tracker • GitHub**: Track and triage your bugs. It shows a table with rows for different bugs.

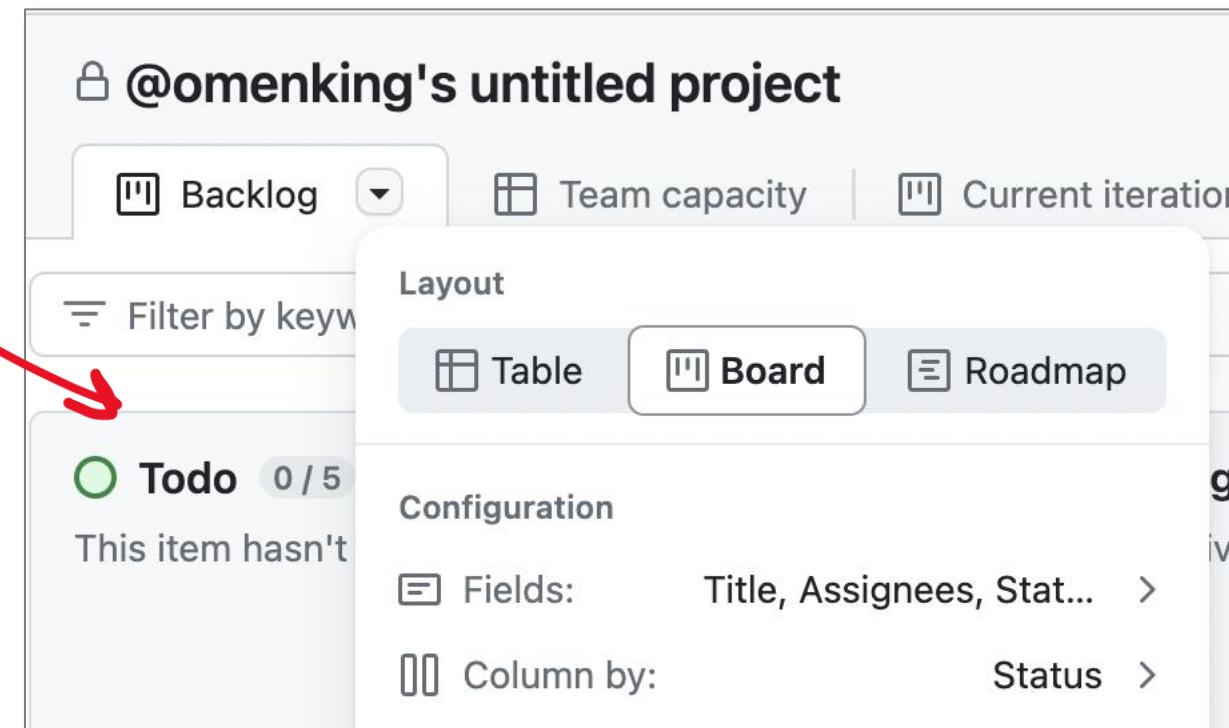
# GitHub Projects Layout Options

A GitHub Project is **composed of multiple views**

- You **add more views** to your project



Each **view's layout can be changed at anytime** to accommodate the project use-case:



# GitHub Projects Layout Options

**Table** layout is great for your traditional ticket tracker and large amount of tasks



**Feature release**  
Manage your team's prioritized work items when planning for a feature release 🚀

Prioritized backlog | Status board | Roadmap | Bugs | In review | + New view

Filter by keyword or field

| Title                                                      | Status      | Assignees      | Size | Estimate |
|------------------------------------------------------------|-------------|----------------|------|----------|
| > P0 2 Estimate: 2                                         |             |                |      |          |
| ▽ P1 4 Estimate: 6                                         |             |                |      |          |
| 3 Updates to velocity of the ship and alien movements #802 | In progress | Dedac          | S    |          |
| 4 Create user account and subscription #820                | In progress | j0siepy        | S    |          |
| 5 Finalize upsell model #1323                              | In progress | marywhite      | M    |          |
| 6 Add a leaderboard for Europe #1378                       | In progress | rileybroughten | L    |          |
| + Add item                                                 |             |                |      |          |
| ▽ P2 2 Estimate: 3                                         |             |                |      |          |
| 7 Save score across levels #800                            | To-do       | Mattamorphic   | M    |          |
| 8 Review engagement metrics once posts are live #26        | To-do       | ohiosveryown   | S    |          |
| + Add item                                                 |             |                |      |          |

# GitHub Projects Layout Options

**Board** layout is great for your agile tasks and understand clearly the state of Issues



**Team planning**  
Manage your team's work items, plan upcoming cycles, and understand team capacity

Backlog | Team capacity | Current iteration | Roadmap | My items | + New view

Filter by keyword or by field

**Todo** 3 / 5 Estimate: 6  
This item hasn't been started

- planning-tracking-demo #802 Updates to velocity of the ship and alien movements
- planning-tracking-demo #820 Create user account and subscription
- planning-tracking-demo #1378 Add a leaderboard for Europe

**In Progress** 2 / 5 Estimate: 2  
This is actively being worked on

- planning-tracking-demo #800 Save score across levels
- planning-tracking-demo #1323 Finalize upsell model

**Done** 1 Estimate: 2  
This has been completed

- OctoArcade #3 Integrate with Commerce Service

+ Add item

The screenshot shows a 'Team planning' board with three columns: 'Todo', 'In Progress', and 'Done'. Each column has a title, a summary, and a list of items. The 'Todo' column has 3 items, 'In Progress' has 2, and 'Done' has 1. Each item includes a link to its GitHub issue page and a small profile picture. At the bottom of each column is a '+ Add item' button.

# GitHub Projects Layout Options

**Roadmap** layout is great for For planning based on a timeline



Roadmap

Roadmap + New view

Filter by keyword or by field

October 2023

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27

**GA Launch**

Markers Start date Date fields More

**Squad 1** 7

| Item                                                       | Start Date   | End Date     |
|------------------------------------------------------------|--------------|--------------|
| 1 Updates to velocity of the ship and alien movements #802 | Oct 6, 2023  | Oct 24, 2023 |
| 2 Create user account and subscription #820                | Oct 9, 2023  | Oct 16, 2023 |
| 3 Finalize upsell model #1323                              | Oct 12, 2023 | Oct 30, 2023 |
| 4 Blowing in to the cartridge does not fix the bl...       | Oct 14, 2023 | Oct 18, 2023 |
| 5 Add a leaderboard for Europe #1378                       | Oct 16, 2023 | Oct 23, 2023 |
| 6 Create the design for login screen #39                   | Oct 16, 2023 |              |
| 7 Support light and dark themes                            | Oct 17, 2023 |              |

+ Add item

**Squad 2** 14

Tue, Jan 31 - Thu, Oct 26

# GitHub Projects Layout Options

For your layouts you can **configure** **your data** based on multiple options



Configuration

- Group by: none >
- Markers: none >
- Sort by: manual >
- Dates: Iteration >
- Zoom level: Month >
- Slice by: none >

- Fields:** Add custom data fields to each project item (e.g., status, assignees).
- Column by:** Organize board columns based on specific fields (e.g., status).
- Group by:** Group items into rows based on a chosen field (e.g., assignee, label).
- Sort by:** Define the order of items within columns or groups (e.g., by priority).
- Fields sum:** Show a summary or total of a particular field for all items in a column/group.
- Slice by:** Filter visible items on the board by a specific field (e.g., milestone, label).
- Zoom Level (Roadmap only):** Adjust the view of the roadmap timeline (daily, monthly, quarterly).
- Dates (Roadmap only):** Set and view start and end dates for each item on the roadmap.

# GitHub Projects Configuration Options

Your GitHub Project Items have **custom fields**

Custom fields

- Status
- Priority
- Size
- Estimate
- Iteration
- Start date
- End date

+ New field

You can create **new fields**

Field name \*

Agileness

Field type

Aa Text

✓ Aa Text

Number

Date

Single select

Iteration

And provide **options** to easily fill in the fields.

Options

- Todo This item hasn't been started
- In Progress This is actively being worked on
- Done This has been completed
- Failed

Add option...

Add

# Projects vs Projects Classic

| GitHub Projects |                                                            | GitHub Projects <i>Classic</i>             |
|-----------------|------------------------------------------------------------|--------------------------------------------|
| Interface       | New, more dynamic interface with tables, boards, and views | Traditional board view similar to Trello   |
| Flexibility     | Highly customizable with different views and fields        | Limited customization options              |
| Automation      | Supports automated workflows and field updates             | Basic automation capabilities              |
| Reporting       | Enhanced reporting features and insights                   | Basic reporting and tracking               |
| Integration     | Deeper integration with GitHub issues and pull requests    | Basic integration with GitHub repositories |

GitHub Projects offers more advanced and flexible project management tools compared to GitHub Classic Projects, which is simpler and more straightforward.

# Project Workflows

GitHub built-in Project Workflows allows you to **automate what happens based on specific events**

The screenshot shows the GitHub Project Workflows interface. On the left, there's a sidebar with icons for Workflows, Archived items, and Settings. A red arrow points from the 'Workflows' icon to the main workflow editor area. The main area is titled 'Pull request merged' with 'Discard' and 'Save and turn on workflow' buttons. It contains a step titled 'When a pull request is merged' with a 'Set value' action. A dropdown menu is open under 'Select an item' with options: Todo (selected), In Progress, Done (checked), and Failed.

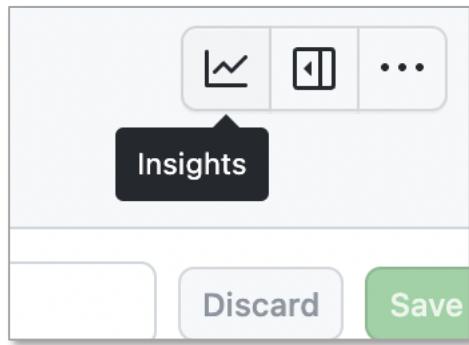
## Built-in Workflows

- Item added to project
- Item reopened
- Item closed
- Code changes requested
- Code review approved
- Pull request merged
- Auto-archive items
- Auto-add to project



For more advanced automation of GitHub Projects, you can use GitHub Actions

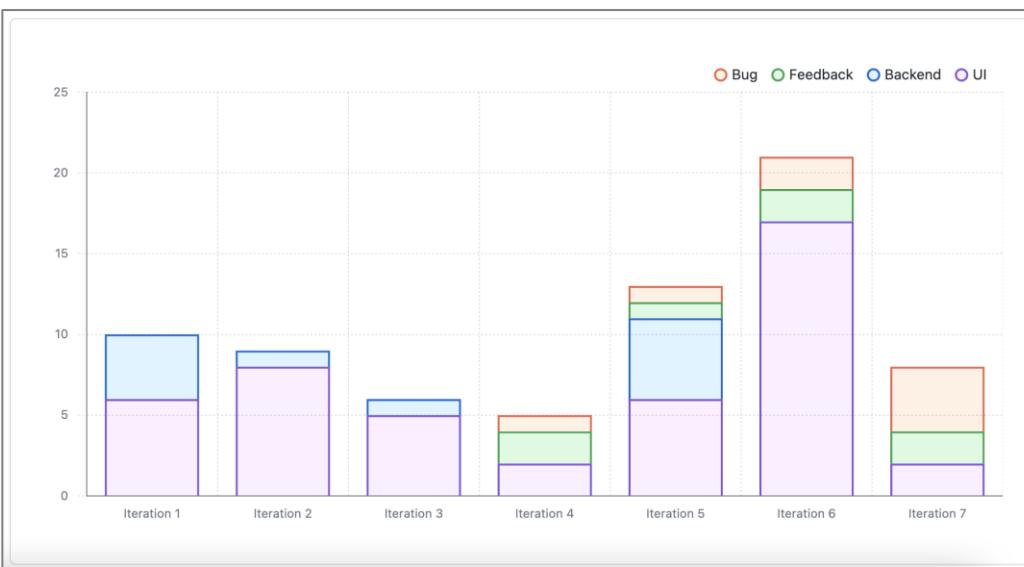
# Project Insights



Project Insights lets you **create charts about your GitHub Project.**

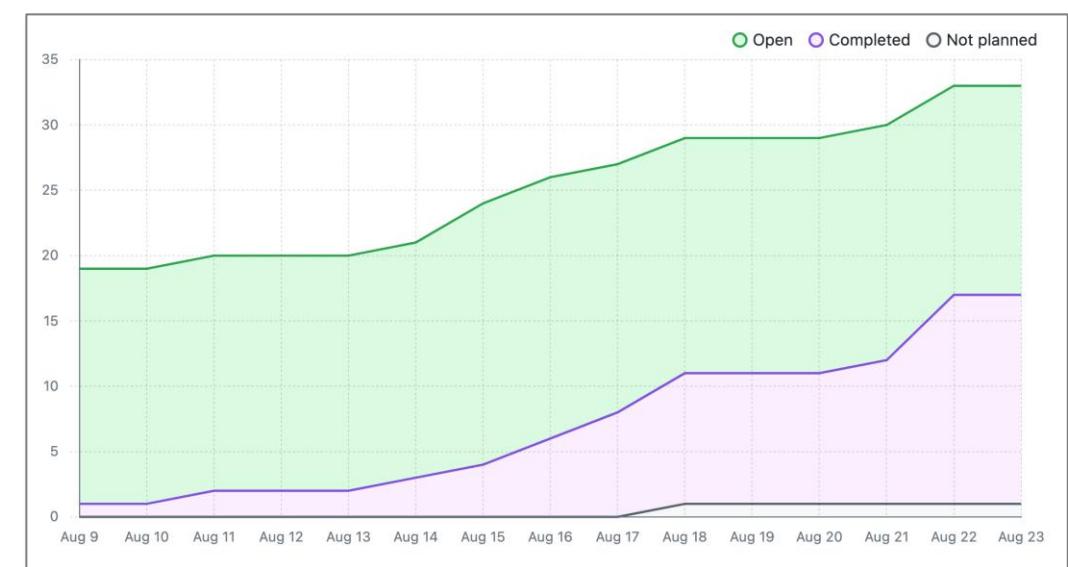
## Current Charts

eg. how many items are assigned to each individuals



## Historical Charts

track changes to the state of your project items



# Managing Saved Replies

When commenting on an issue or pull request, you can **add a saved reply that you've already set up.**

Saved replies

Add a saved reply

Saved reply title

Kick-the-can

Write Preview

I think this feature request looks great.  
Let's make sure everyone gets a chance to get input on this feature before we approve it.  
Can you in your off-time put together long-form documentation and I'll share it was leadership.

Markdown is supported

Paste, drop, or click to add files

Add saved reply

# Assigning Issues and Pull Requests

The screenshot shows the 'Assignees' section of a GitHub issue or pull request page. It includes a search bar labeled 'Type or choose a user', a 'Clear assignees' button, and a list of assignees. A red arrow points from the text 'You can assign issues and pull requests to specific users.' to the gear icon in the top right corner of the assignees box.

Assignees

Assign up to 10 people to this issue

Type or choose a user

Clear assignees

✓ omenking Andrew Brown

omenking's untitled project

You can **assign issues** and pull requests to specific users.  
That's it.

The screenshot shows the GitHub search interface with a filter modal open. The filter is set to 'Filter by who's assigned'. A red arrow points from the text 'You can filter Issues and PRs based on assignees.' to the 'Sort' dropdown menu in the top right of the filter modal.

Projects ▾ Milestones ▾ Assignee ▾ Sort ▾

Filter by who's assigned

Filter users

Assigned to nobody

omenking Andrew Brown

You can **filter** Issues and PRs based on assignees.

# Securing Account with 2FA



**2-Factor Authentication (2FA)** adds an extra layer of security by requiring a second device (e.g. phone) to confirm your identity during sign-in or sensitive actions.

**Preferred 2FA method**  
Set your preferred method to use for 2FA

GitHub Mobile

Choose your preferred **UFA (User-Facing Authentication)** method:

- GitHub Mobile
- SMS/Text message
- Authenticator app



You can download third-party authenticator apps via your phone

- Authy
- Google Authenticator
- Microsoft Authenticator

Get UFA code sent as text message



Use **hardware security keys** like YubiKeys

**GitHub's Mobile app**

**Recovery codes** are long secret codes you need to print and store somewhere safe in case you get locked out.

| Two-factor methods                         |            |                                                                     |
|--------------------------------------------|------------|---------------------------------------------------------------------|
| <input type="checkbox"/> Authenticator app | Configured | Use an authentication app or browser extension                      |
| <input type="checkbox"/> SMS/Text message  | Configured | You will receive one-time codes at this phone number                |
| <input type="checkbox"/> Security keys     |            | Security keys are hardware devices that can be used for 2FA         |
| <input type="checkbox"/> GitHub Mobile     | Configured | 1 device<br>GitHub Mobile can be used for two-factor authentication |
| Recovery options                           |            |                                                                     |
| <input type="checkbox"/> Recovery codes    | Viewed     | Recovery codes can be used to access your account                   |

# Access Permissions – Personal Accounts

Personal account has **two permission levels**:

- **Repository owner** – the person who owns the account, that's you!
  - Repositories owned by personal accounts have one owner
  - Ownership permissions can't be shared with another personal account.
- **Collaborators** – people you add as collaborators

## Repository owner actions — **has full control of the repository**

- Invite collaborators
- Change the visibility of the repository
- Limit interactions with the repository
- Rename a branch, including the default branch
- Merge a pull request on a protected branch, even if there are no approving reviews
- Delete the repository
- Manage the repository's topics
- Manage security and analysis settings for the repository
- Enable the dependency graph for a private repository
- Delete and restore packages
- Customize the repository's social media preview
- Create a template from the repository
- Control access to Dependabot alerts
- Dismiss Dependabot alerts in the repository
- Manage data use for a private repository
- Define code owners for the repository
- Archive the repository
- Create security advisories
- Display a sponsor button
- Allow or disallow auto-merge for pull requests
- Manage deploy keys
- Manage webhooks

# Access Permissions – Personal Accounts

**Collaborators actions — can pull (read) the contents of the repository and push (write) changes to the repository**

- Fork the repository
- Rename a branch other than the default branch
- Create, edit, and delete comments on commits, pull requests, and issues in the repository
- Create, assign, close, and re-open issues in the repository
- Manage labels for issues and pull requests in the repository
- Manage milestones for issues and pull requests in the repository
- Mark an issue or pull request in the repository as a duplicate
- Create, merge, and close pull requests in the repository
- Enable and disable auto-merge for a pull request
- Apply suggested changes to pull requests in the repository
- Create a pull request from a fork of the repository
- Submit a review on a pull request that affects the mergeability of the pull request
- Create and edit a wiki for the repository
- Create and edit releases for the repository
- Act as a code owner for the repository
- Publish, view, or install packages
- Remove themselves as collaborators on the repository

# Access Permissions – Personal Accounts

## Repository owner actions

has full control of the repository

VS

## Collaborators actions

can pull (read) the contents of the repository  
and push (write) changes to the repository

# Access Permissions – Organization Accounts

Organization members have the permission levels

- **Owner**
  - complete administrative access to your organization
  - This role should be limited, but to no less than two people, in your organization
- **Billing manager** — can manage billing settings
- **Member roles** — the default role for everyone else

<Orgnaziation roles conflict, this makes no sense maybe ignore this slide for now>

# EMUs

Enterprise Managed Users (EMUs) allows you to **manage the lifecycle and authentication of your users on GitHub.com** from an **external identity management system**, or IdP.

- GitHub partners with some developers of identity management systems to provide a "paved-path" integration with EMUs
- These IdPs mostly provide authentication using SAML.
- Microsoft Entra ID (aka Azure AD) also offers OIDC for authentication.
- The IdP applications provision users with System for Cross-domain Identity Management (SCIM).

| Partner IdP                                                                                           | SAML | OIDC | SCIM |
|-------------------------------------------------------------------------------------------------------|------|------|------|
|  Microsoft Entra ID |      |      |      |
|  Okta              |      |      |      |
|  PingFederate      |      |      |      |

# Managing Features

GitHub Repo have **multiple features we can enable or disable:**

- Wikis
- Issues
- Sponsorships
- Preserve this repo
  - Goes into GitHub Arctic Code Vault
- Discussions
- Projects
- and possibly more...



**Features**

- Wikis**  
Wikis host documentation for your repository.
- Restrict editing to collaborators only**  
Public wikis will still be readable by everyone.
- Issues**  
Issues integrate lightweight task tracking into your repository. Keep projects on track with issue labels and milestones, and reference them in commit messages.
- Sponsorships**  
Sponsorships help your community know how to financially support this repository.
- Preserve this repository**  
Include this code in the [GitHub Archive Program](#).
- Discussions**  
Discussions is the space for your community to have conversations, ask questions and post answers without opening issues.
- Projects**  
Projects on GitHub are created at the repository owner's level (organization or user) and can be linked to a repository's Projects tab. Projects are suitable for cross-repository development efforts such as feature work, complex product roadmaps or even Issue triage.

# Repo permission levels

A screenshot of the GitHub repository permission settings interface. On the left, a sidebar lists roles: Read, Triage, Write (selected), Maintain, and Admin. Each role has a description and a 'Remove' button. A red arrow points to the 'Remove' button for the 'Write' role. At the top, there's a 'Role: Write' dropdown and another 'Remove' button.

| Role     | Description                                                                                                                                   | Action |
|----------|-----------------------------------------------------------------------------------------------------------------------------------------------|--------|
| Read     | Can read and clone this repository. Can also open and comment on issues and pull requests.                                                    | Remove |
| Triage   | Can read and clone this repository. Can also manage issues and pull requests.                                                                 | Remove |
| Write    | Can read, clone, and push to this repository. Can also manage issues and pull requests.                                                       | Remove |
| Maintain | Can read, clone, and push to this repository. They can also manage issues, pull requests, and some repository settings.                       | Remove |
| Admin    | Can read, clone, and push to this repository. Can also manage issues, pull requests, and repository settings, including adding collaborators. | Remove |

When you add a collaborator (and they accept the invite) you can choose from **predefined roles** with different level of access

- **Read:**
  - View and clone the repository.
  - Open and comment on issues and pull requests.
  - Download releases.
- **Triage:**
  - Manage issues and pull requests without write access.
  - Label and assign issues and pull requests.
  - Close and reopen issues and pull requests.
- **Maintain:**
  - Push to the repository.
  - Manage issues, pull requests, labels, and project boards.
  - Create and publish releases.
  - Configure repository settings for non-sensitive fields (like assigning collaborators).
- **Admin:**
  - Full control over the repository.
  - Push to, delete, or archive the repository.
  - Change repository settings, including sensitive fields like security or billing.
  - Manage collaborators and their permissions.

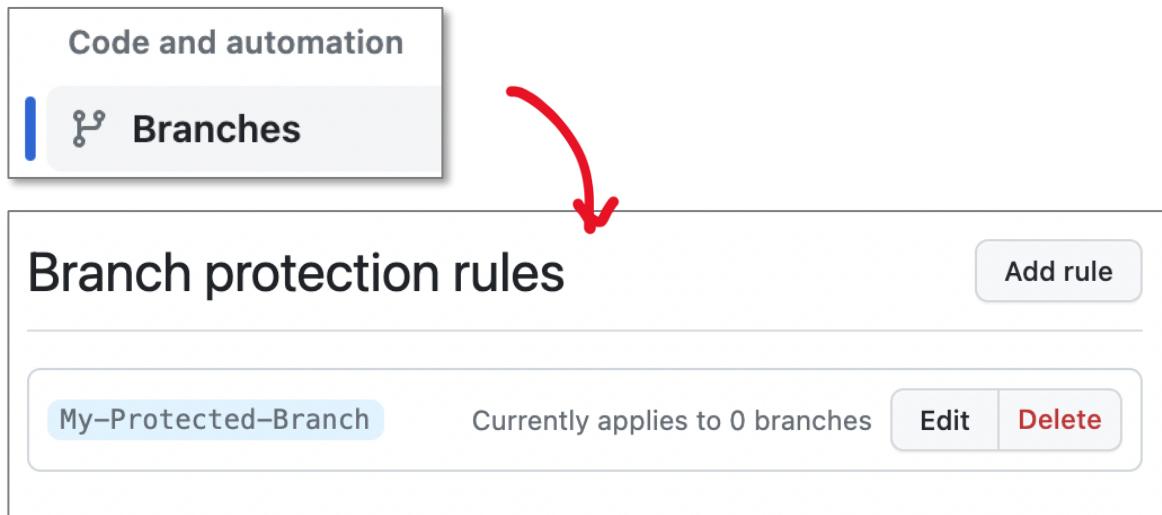


Enterprise accounts can have custom roles.

Access to all functionalities provided in the Maintain role.

# Branch Protection Rules

**Branch protection rules** are used to **enforce certain workflows or requirements** before changes can be merged into a branch.



The screenshot shows the 'Branches' section of the GitHub settings. A red arrow points from the 'Branches' link in the sidebar to the 'Branch protection rules' section below. The 'Branch protection rules' section contains a table with one row for 'My Protected Branch'. The row shows 'My Protected Branch' in the name column, 'Currently applies to 0 branches' in the status column, and 'Edit' and 'Delete' buttons. An 'Add rule' button is located at the top right of the table.

| Name                | Status                          | Action | Action |
|---------------------|---------------------------------|--------|--------|
| My Protected Branch | Currently applies to 0 branches | Edit   | Delete |

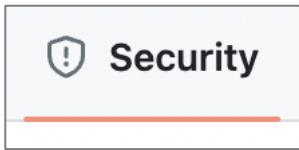
## Protect matching branches rules

- Require a pull request before merging
- Require status checks to pass before merging
- Require conversation resolution before merging
- Require signed commits
- Require linear history
- Require deployments to succeed before merging
- Lock branch
- Do not allow bypassing the above settings

## Rules applied to everyone including administrators

- Allow force pushes
- Allow deletions

# Security Tab features and options



The security tab will present a **repo security checklist** of security options to configure

## Security policy

Create markdown file that explains how security vulnerabilities should be reported

## Security advisories

privately discuss, fix, and publish information about security vulnerabilities in your public repository

## Private vulnerability reporting

Allow your community to privately report potential security vulnerabilities to maintainers and repository



## Dependabot alerts

A bot that alerts you of vulnerabilities due to out-of-date dependencies.

Dependabot can automatically create PRs to update dependencies for you to approve.

## Code scanning alerts

Automatically detect common vulnerabilities and coding errors. Via CodeQL or third-party tools

## Secret scanning alerts

Get notified when a secret is pushed to this repository

# Managing Collaborators

Collaborators allows you let other GitHub users have access to your repo based on the permission levels you provide

You search their **username** to invite

A screenshot of a GitHub search results page. A red arrow points from the search bar at the top left to the user profile of Andrew Bayko below it. The search bar contains the text "bayko". The user profile for Andrew Bayko includes a small circular profile picture, the name "Andrew Bayko", and the text "bayko • Invite collaborator". Below this are two more user profiles: "Evgeniya Baykova" and "Baykodrom77", each with their respective profile pictures and "Invite collaborator" links.

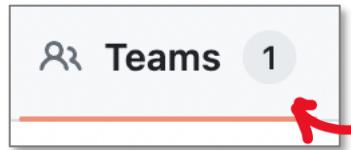
A screenshot of the "Manage access" page on GitHub. A red arrow points from the "Pending Invite" status next to Andrew Bayko's name in the list below to a blue button at the bottom of the page. The page title is "Manage access". It features a search bar labeled "Find a collaborator..." and a list of users with checkboxes. Andrew Bayko is listed with the status "Pending Invite" and a "Remove" link. The "Add people" button is located in the top right corner.

<https://github.com/<username/org>/<repo-name>/invitations>

The user need to **accept** the invite.

*Its best to share the link directly and manually remind them to accept*

# Organization Teams Tab



An organization can group organization members in **teams**.

A screenshot of a team management interface. It includes a search bar 'Find a team...', a 'New team' button, and filters for 'Select all' and 'core Secret'. Below these are two user icons, 'Visibility' and 'Members' dropdowns, and summary statistics: '2 members', '0 roles', and '0 teams'.

A screenshot of a 'Team visibility' section. It shows two options: 'Visible Recommended' (selected) and 'Secret'. Descriptions for each are provided: 'A visible team can be seen and' for Visible, and 'A secret team can only be seen' for Secret. A red arrow points from the text 'Teams can be public or private' to the 'Secret' option.

Teams can be **public** or **private**

- Teams can be assigned to projects and issue
- Request reviews from teams
- Teams can be mentioned in discussions, issues, and pull requests
- Control team access to repositories
- New members can be added to teams, instantly giving them access to all relevant repositories and discussions

# GitHub Connect



**GitHub Connect** enhances GitHub Enterprise Server (GHES) by allowing your GHES instance **access some of GitHub.com cloud only offerings**

You can **individually enable these features**:

- Automatic user license sync
- Dependabot
- GitHub.com actions
- Server Statistics
- Unified search
- Unified contributions

# GitHub Internal Repos

**GitHub Enterprise Cloud** accounts have a third special GitHub Repo type called **Internal**

- All enterprise members have **read permissions** to the internal repository
- Internal repositories are not visible to people outside of the enterprise, including outside collaborators on organization repositories.
- Unless your enterprise uses Enterprise Managed Users (EMUs), members of the enterprise can fork any internal repository owned by an organization in the enterprise.

Internal repositories are the **default setting** for all new repositories created in an organization owned by an enterprise account.

# GitHub Advanced Security

GitHub makes **extra security features available** to customers under a GitHub Advanced Security (GHAS) license.

GitHub Advanced Security is available for enterprise accounts on:

- GitHub Enterprise Cloud
- GitHub Enterprise Server
- And some features of for public repositories on GitHub.com

GitHub Advanced Security license provides the following additional features:

|                   |
|-------------------|
| Code scanning     |
| Secret scanning   |
| Dependency review |

| Public repository | Private repository without Advanced Security | Private repository with Advanced Security |
|-------------------|----------------------------------------------|-------------------------------------------|
|-------------------|----------------------------------------------|-------------------------------------------|

|  |  |  |
|--|--|--|
|  |  |  |
|  |  |  |
|  |  |  |

# SAML and SCIM

**SAML and SCIM** is available for both GitHub Enterprise Cloud and GitHub Enterprise Server

## SAML

**Security Assertion Markup Language (SAML)** is an open standard for exchanging authentication and authorization between an identity provider and a service provider.

An important use case for SAML is **Single-Sign-On via web browser**.

## SCIM

**System for Cross-domain Identity Management (SCIM)** is an open standard for automating the exchange of user identity information across different identity management systems. A key use case for SCIM is to enable scalable and automated user provisioning and deprovisioning, often integrated with enterprise identity services.

SAML is about **securely transmitting user authentication and authorization data** for single sign-on purposes.

vs

SCIM is about **managing user identities across different systems** to simplify account maintenance and provisioning



They can complement each other in an ecosystem where you need both federated identity for authentication and a standardized way to manage user accounts across various systems.