

Engineering Open Source Projects

Anton Grishin ([@alchemmיסט](https://github.com/alchemmיסט))

Introduction to the EOSP course, winter 2026. CU × CPM



Table of contents

hello

Getting to know each other better.

open-source

Diving into the World of OpenSource.

course

Figure out what awaits us in the course.

practice

Come up with a name for the project together.

live-demo

Lay the foundation for the project together.

api

Getting to know the GitHub API.

github-flow

How are Open Source development processes organized.

Let's get acquainted

hello

What do you think false here?

- I'm Anton. Study at 2nd course of SWE at Central University.
- Involved in the development of 70+ repositories and have sent ~1600 commits.
- I'm graduate of Yandex Lyceum golden certificate.
- I used Linux as main operation system for last 3 years.
- I can type 120 words per minute on qwerty keyboard.
- I used to be a professional volleyball player.
- I am using only terminal for developing.
- Author of blog: alchemmist.xyz

Tell us about yourself 

What is Open Source

open-source

Source code of project is open to see for everyone.

Open to copy? Open to appropriate? Open to sell?

Postgres DBMS (MIT/BSD)

[open-source](#)

- Full access to the source code
- Use in commercial products
- Selling as part of your own product
- Closing your code on top of PostgreSQL
- Please, save the license and author name

postgres

Overview Repositories 5 Projects Packages People 6

 **PostgreSQL**
Verified
1.8k followers Worldwide <https://www.postgresql.org/>

Pinned

 **postgres** Public
Mirror of the official PostgreSQL GIT repository. Note that this is just a *mirror* - we don't work with pull requests on github. To contribute, please see https://wiki.postgresql.org/wiki/Submitting_a_Patch
● C ⭐ 19.6k ⚡ 5.3k

Repositories

Find a repository... Type Language

 **postgres** Public
Mirror of the official PostgreSQL GIT repository. Note that this is just a *mirror* - we don't work with pull requests on github. To contribute, please see https://wiki.postgresql.org/wiki/Submitting_a_Patch
● C ⭐ 19.554 ⚡ 5.329 ○ 0 ⚡ 0 Updated 15 hours ago

 **pgweb** Public
Mirror of the code behind www.postgresql.org
● HTML ⭐ 80 ⚡ 48 ○ 0 ⚡ 0 Updated 3 days ago

 **pgcommitfest** Public
The source code for <https://commitfest.postgresql.org>
● Python ⭐ 12 ⚡ 19 ○ 13 ⚡ 1 Updated 3 days ago

Postgres PRO (EULA)

open-source

- EULA — End User License Agreement
- Distributed for a money
- OpenCore model



Весь опыт
разработчиков
PostgreSQL для вас

Angular JS (MIT)

open-source

- Fully open (as Postgres)
- Repo is exist, but deprecated
- No support
- No improves
- No bug fixes
- The reason is switched on TypeScript

angular / angular.js

Code Issues 389 Pull requests 72 Actions Projects Wiki Security Insights

This repository was archived by the owner on Apr 12, 2024. It is now read-only.

angular.js Public archive

Watch 3738 Fork 27.3k Star 59k

master Go to file Code About

AngularJS - HTML enhanced web apps!

[angularjs.org](#)

Readme MIT license Code of conduct Contributing Security policy Activity Custom properties 59k stars 3.7k watching 27.3k forks Report repository

Releases 209 tags

Commit	Message	Time
.circleci	chore(build): supp...	5 years ago
.github	chore(*): prep for ...	6 years ago
benchmarks	chore(benchpress...	8 years ago
css	style(css) separat...	9 years ago
docs	chore: update pos...	2 years ago
i18n	chore(i18n): fix U...	8 years ago
images	docs(*): optimize i...	9 years ago
lib	chore(ci): deploy t...	5 years ago
logs	creating logs/ and...	16 years ago
scripts	chore(functions): ...	5 years ago
src	docs(\$parse): fix t...	5 years ago
test	test(Angular): fix a...	5 years ago

Linux kernel (GPLv2)

open-source

- Full access to the source code
- Use in commercial products
- Forks must remain GPL if distributed: copyleft

torvalds / linux

Code Pull requests Actions Projects Security Insights

linux Public

master 1 Branch 914 Tags Go to file Add file

torvalds Merge tag 'erofs-for-6.19-rc5-fixes' of git://git.kernel.org/pub/scm/... b6151c4 · 7 hours ago

Documentation Merge tag 'soc-fixes-6.19' of git://git.kernel.org/pub/scm...

LICENSES LICENSES: Add modern form of the LGPL-2.1 tags to the ...

arch Merge tag 'arm64-fixes' of git://git.kernel.org/pub/scm/...

block Merge tag 'block-6.19-20260109' of git://git.kernel.org/...

certs sign-file,extract-cert: use pkcs11 provider for OPENSSL M...

crypto crypto: seqiv - Do not use req->iv after crypto_aead_encr...

drivers Merge tag 'block-6.19-20260109' of git://git.kernel.org/...

fs Merge tag 'erofs-for-6.19-rc5-fixes' of git://git.kernel.org/...

include Merge tag 'acpi-6.19-rc5' of git://git.kernel.org/pub/scm/...

init Merge tag 'mm-nonmm-stable-2025-12-06-11-14' of git:...

io_uring Merge tag 'io_uring-6.19-20260109' of git://git.kernel.org/...

ipc Merge tag 'mm-nonmm-stable-2025-12-06-11-14' of git:...

kernel Merge tag 'pm-6.19-rc5' of git://git.kernel.org/pub/scm/...

lib idr: fix idr_alloc() returning an ID out of range

mm mm/ksm: fix pte_unmap_unlock of wrong address in bre...

net Merge tag 'ceph-for-6.19-rc5' of <https://github.com/ceph...>

OpenSource is the foundation of modern IT

open-source

A space where technology is emerging.



*«99% of Fortune 500 companies currently use open source software. <...>
Over 56 million developers are contributing to open source projects.
<...> Due to ever-rising workloads, the Linux operating systems market
is expected to grow at the rate of 7% a year, reaching \$9.7 billion by
2024.»*

Pranay Ahlawat, Boston Consulting Group

Article «Why You Need an Open Source Software Strategy», April 2021

The idea and goal of our project

course

Course is totally practice-driven.

- Build a practical system to evaluate developer contributions based on GitHub activity
- Learn modular software design: `library` → `CLI` → `Telegram bot`
- Practice real-world Open Source workflows: issues, pull requests, reviews and so on
- Focus on clean, maintainable, and testable code
- Experience CI/CD pipelines, releases, and deployment automation
- Document, setting and organize projects properly
- Develop skills to build slide deck and pitch project publicly

Two main problem cases

course

Profile analytics

- HR wants quick insight into a developer's activity without digging into GitHub manually
- Analyze the **entire GitHub profile**: all repositories, contributions, and activity history
- Understand which languages and technologies a developer uses
- Track contributions across repositories: commits, pull requests, issues
- Generate a concise profile summary for recruitment decisions

Leader board of team

- Team leads want visibility into team productivity
- Analyze **contributions within a single repository** to compare team members fairly
- Track per-developer metrics: code quality, review participation, issue resolution
- Identify who is actively contributing and who may need support or guidance
- Provide fair, data-driven insights to improve collaboration and team performance

Three setups, three projects

course

Python library

- Calculate developer contribution metrics from GitHub data
- Provide reusable, modular functions for metrics computation
- Include comprehensive unit tests and follow TDD approach
- Serve as the core foundation for CLI and bot integrations
- Support easy extension and maintainability

CLI

- Provide command-line access to library metrics
- Support multiple commands, flags, and options
- Enable fetching, displaying, and exporting data conveniently
- Handle errors gracefully and show meaningful messages
- Integrate with CI/CD for automated releases

Telegram bot

- Provide easy access to metrics via Telegram interface
- Interact with users, handle commands and queries
- Securely manage secrets and API tokens
- Fetch data from library and format it for user-friendly display
- Support notifications, updates, and automated alerts

How should this course be perceived? course

Motivation and mindset.



«It's a bit sad to think of all the high school kids turning their backs on building treehouses and sitting in class dutifully learning about Darwin or Newton to pass some exam, when the work that made Darwin and Newton famous was actually closer in spirit to building treehouses than studying for exams.»

Paul Graham

Essay «A Project of One's Own», June 2021

It's time to come up with a name!

practice

Go to Figma board!

Let's make first step

[live-demo](#)

Creating GitHub organization and repo.

Introduction to GitHub API.

api

```
gh api /octocat
```

GitHub API is just HTTP api

Any tool that can send HTTP requests can work with GitHub API.

- `gh` — convenient wrapper around the API
- `curl` — raw HTTP from terminal
- `Python` — programmatic access for automation and logic

Official gh cli api

The cli utility for using all github functionality from terminal.

Install into your shell:

```
# Mac:  
brew install gh  
  
# Windows:  
winget install --id GitHub.cli  
  
# Arch:  
sudo pacman -S github-cli
```

And try something, for example:

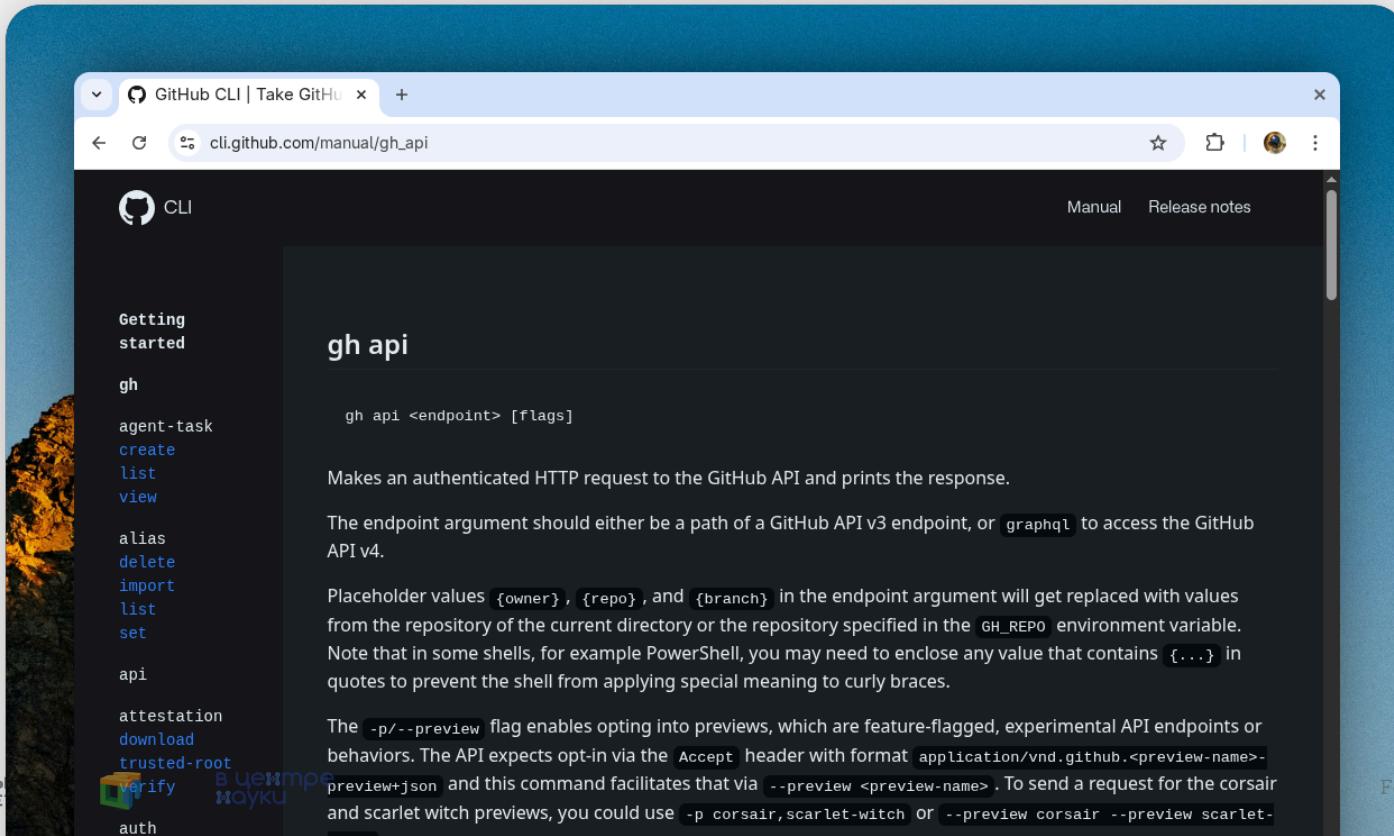
```
gh api /users/alchemmist
```

As result:

```
1  {  
2      "login": "alchemmist",  
3      "avatar_url": "https://avatars.githubusercontent.com/u/104511335?v=4",  
4      "html_url": "https://github.com/alchemmist",  
5      "followers_url": "https://api.github.com/users/alchemmist/followers",  
6      "subscriptions_url": "https://api.github.com/users/alchemmist/subscriptions",  
7      "repos_url": "https://api.github.com/users/alchemmist/repos",  
8      "type": "User",  
9      "name": "Anton Grishin",  
10     "blog": "alchemmist.xyz?utm_source=github",  
11     "location": "Russia, Moscow",  
12     "email": "anton.ingrish@gmail.com",  
13     "followers": 18,  
14     "created_at": "2022-04-27T14:12:26Z",  
15     "updated_at": "2026-01-09T06:23:55Z",  
16     ...  
17 }
```

More in documentation

api



GitHub CLI | Take GitHub api

cli.github.com/manual/gh_api

Manual Release notes

Getting started

gh

agent-task

create

list

view

alias

delete

import

list

set

api

attestation

download

trusted-root

verify

auth

gh api

```
gh api <endpoint> [flags]
```

Makes an authenticated HTTP request to the GitHub API and prints the response.

The endpoint argument should either be a path of a GitHub API v3 endpoint, or `graphql` to access the GitHub API v4.

Placeholder values `{owner}`, `{repo}`, and `{branch}` in the endpoint argument will get replaced with values from the repository of the current directory or the repository specified in the `GH_REPO` environment variable. Note that in some shells, for example PowerShell, you may need to enclose any value that contains `{...}` in quotes to prevent the shell from applying special meaning to curly braces.

The `-p/--preview` flag enables opting into previews, which are feature-flagged, experimental API endpoints or behaviors. The API expects opt-in via the `Accept` header with format `application/vnd.github.<preview-name>-preview+json` and this command facilitates that via `--preview <preview-name>`. To send a request for the corsair and scarlet witch previews, you could use `-p corsair,scarlet-witch` OR `--preview corsair --preview scarlet`.

Using curl for GitHub API

api

Raw HTTP requests from terminal.

Send GET authenticated request:

```
curl -L \
  -H "Accept: application/vnd.github+json" \
  -H "Authorization: Bearer <TOKEN>" \
  -H "X-GitHub-Api-Version: 2022-11-28" \
  https://api.github.com/repos/alchemmיסט/eosp/stats/contributors
```

According to the documentation

- `w` — Start of the week, given as a Unix timestamp.
- `a` — Number of additions
- `d` — Number of deletions
- `c` — Number of commits

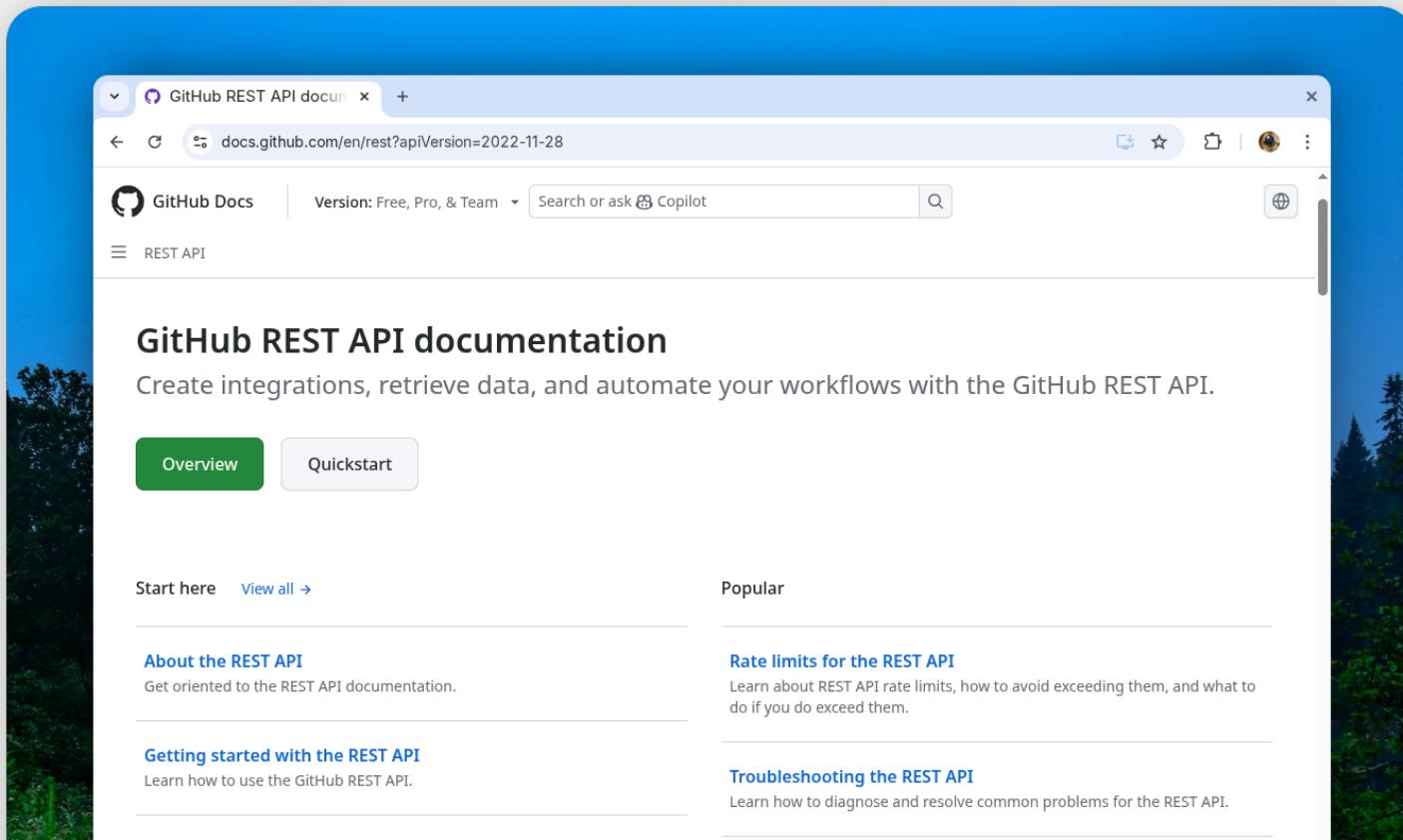
As result:

```
1  [
2    {
3      "total": 37,
4      "weeks": [
5        {
6          "w": 1765670400,
7          "a": 6477,
8          "d": 0,
9          "c": 1},
10         {
11           "w": 1768089600,
12           "a": 0,
13           "d": 0,
14           "c": 0}, ...
15       ],
16       "author": {
17         "login": "alchemmист",
18         "id": 104511335,
19         "node_id": "U_kgD0Bjq3Zw", ...
20       }
21     }
22  ]
```

Sun Dec 14 2025

More in documentation

api



The screenshot shows a web browser window displaying the GitHub REST API documentation. The URL in the address bar is `docs.github.com/en/rest?apiVersion=2022-11-28`. The page title is "GitHub REST API documentation". The main content area features a green "Overview" button and a white "Quickstart" button. Below these buttons, there are two sections: "Start here" with a "View all →" link, and "Popular" with a "Rate limits for the REST API" section. The "Rate limits for the REST API" section includes a brief description and a link to "Learn about REST API rate limits, how to avoid exceeding them, and what to do if you do exceed them." The overall layout is clean and modern, with a blue header and a white background for the main content.

Using GitHub API with Python

api

Efficient, parallel, production-ready requests.

Example with `httpx`:

```
import asyncio, httpx

async def fetch_prs(username):
    url = f"https://api.github.com/search/issues?" \
          f"q=author:{username}+type:pr+created:>2025-01-01"
    async with httpx.AsyncClient() as client:
        resp = await client.get(
            url,
            headers={
                "Authorization": "Bearer <TOKEN>"
            },
        )
        data = resp.json()
        for pr in data["items"]:
            print(f"{pr['title']}\n\t-> {pr['html_url']}")

asyncio.run(fetch_prs("alchemmist"))
```

As result:

```
Add alchemmist.xyz individual blog
-> https://github.com/kilimchoi/engineering-blogs/pull/1201
Add alchemmist.xyz personal blog
-> https://github.com/learn-anything/blogs/pull/21
Add alchemmist.xyz blog
-> https://github.com/logancyang/awesome-pers...
Add alchemmist.xyz blog
-> https://github.com/jkup/awesome-personal-blogs/pull/173
Add @alchemmist_blog to personal blogs section
-> https://github.com/goq/telegram-list/pull/992
Add @alchemmist_blog to personal blogs section
-> https://github.com/alchemmist/telegram-list/pull/1
Add a "quiet" exit (#104)
-> https://github.com/cqfn/aibolit/pull/818
```

*GraphQL - a query language for APIs that lets you request exactly the data you need in a single query, without extra fields.

GitHub API Limitations

api

Understanding these limits is essential when building scalable, reliable systems for collecting GitHub data.

- Rate limits: **5000 requests per hour** for authenticated users
- Rate limits: **60 requests per hour** for unauthenticated users
- Pagination: most endpoints return max **100 items per page**, need to handle paging
- Private data requires proper authentication and scopes
- GraphQL vs REST: some data easier via GraphQL, but query complexity may hit limits
- API responses may be delayed or cached; real-time metrics may require retries
- Some endpoints change over time; library must handle API versioning

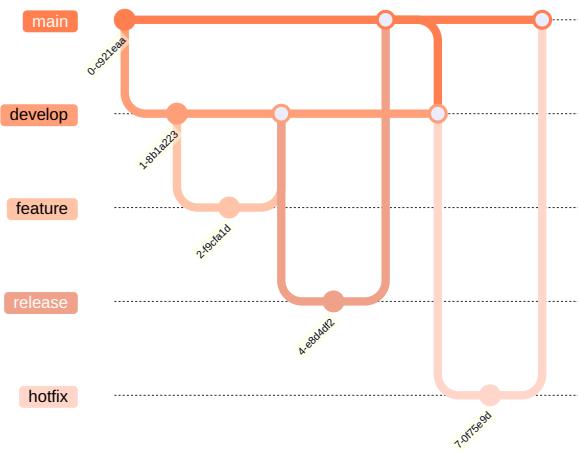
Developing flows

github-flow

Two ways of coding.

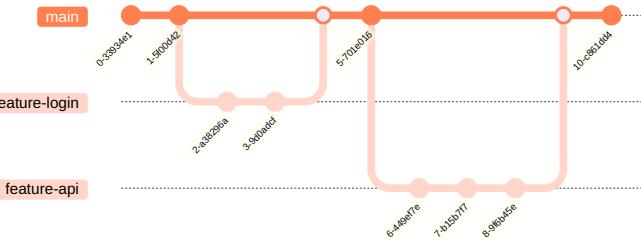
Git Flow

A structured and process-heavy branching model designed for scheduled releases. It relies on long-lived branches and explicit release management, which makes it predictable but slower to adapt.



GitHub Flow

A lightweight workflow optimized for continuous delivery and open source. The `main` branch is always deployable, and all changes flow through pull requests.



Why GitHub Flow? github-flow

- All changes go through **pull requests** → code review, CI/CD checks
- Encourages small, incremental updates rather than long-lived branches
- Clear separation: `main` branch is always deployable
- Integration with issues and project boards → planning and tracking in one place
- Transparency and collaboration: team members can comment, review, approve, or reject changes

Key Entities in GitHub Flow

github-flow

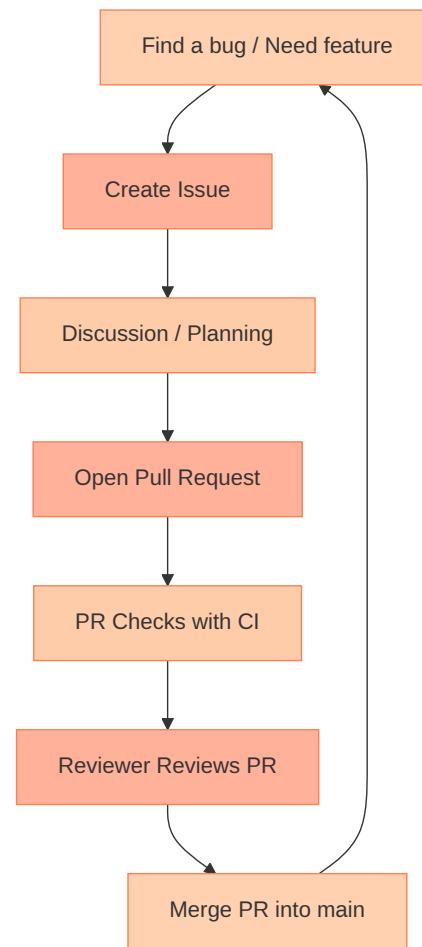
Issue	Branch	Commit
Describes a bug, feature, task, or question. Starting point for development.	Isolated workspace for a specific feature or fix.	Individual changes tracked in Git history.
Pull Request (PR)	Code Review	CI/CD Checks
Proposes changes from a branch into main. Facilitates review and discussion.	Teammates review PRs to ensure quality and maintainability.	Automated tests, linting, build, and deployment pipelines.
Merge		
Approved PR is merged into main and usually triggers deployment.		

GitHub Flow in Action

github-flow

Visual representation of the workflow:

1. Developer identifies a bug or feature → **creates an issue**
2. Creates a branch from `main` for the issue
3. Makes **commits** locally and pushes to GitHub
4. Opens a **pull request** linking to the issue
5. Team conducts **code review** and automated **CI/CD checks**
6. After approval, PR is merged into `main`
7. Deployment triggers automatically (if CI/CD is configured)



Best Practices in GitHub Flow

github-flow

- Keep branches **short-lived** → frequent integration reduces conflicts
- Write **descriptive commits** → history becomes meaningful
- Reference issues in PRs → link work to context
- Use **templates** for PRs and issues → standardize workflow
- Automate as much as possible → CI/CD, tests, linters, code quality checks
- Encourage **review culture** → better code, knowledge sharing, accountability

