GitHub

GitHub API Training

Presented by GitHub Professional Services

Objectives

As a result of this session, attendees will be comfortable with describing:

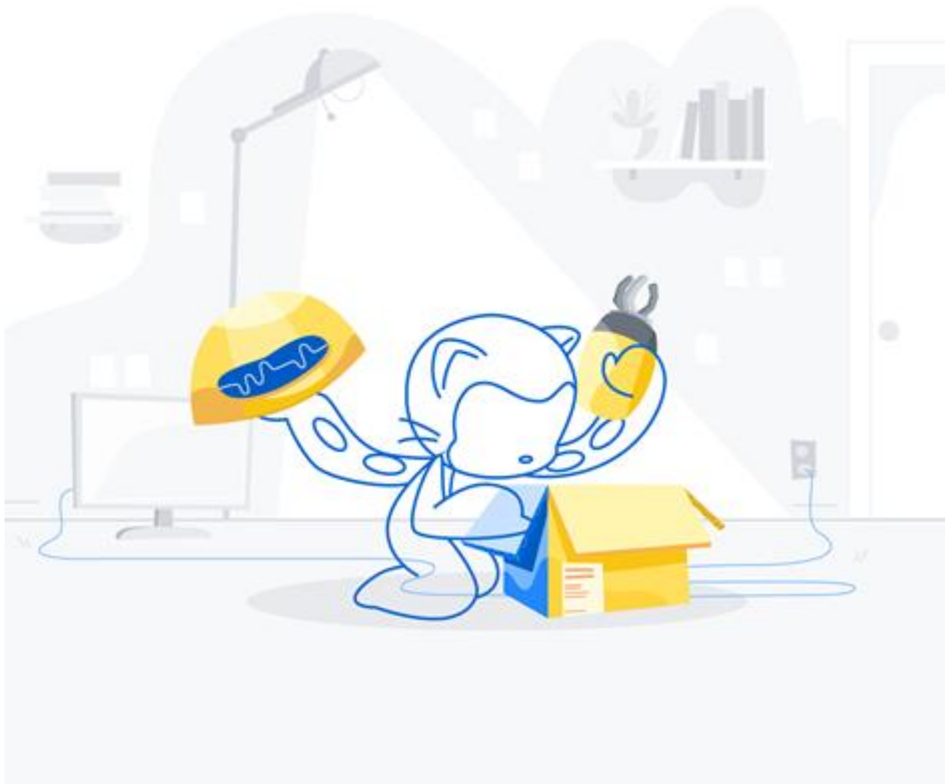
- How we use the API at GitHub
- The benefits and methods of accessing GitHub's API
- The differences between REST, GraphQL, and Webhooks
- Application of the API with topics like Octokit, Actions, and the GitHub CLI
- Additional application examples

To get there, we will need to revisit your specific learning milestones. Examples include:

- Improve operational efficiency via automated provisioning and tasking
- Making use of APIs potentially in GitHub Apps and Actions
- Your preferences toward generic enablement over specific use cases

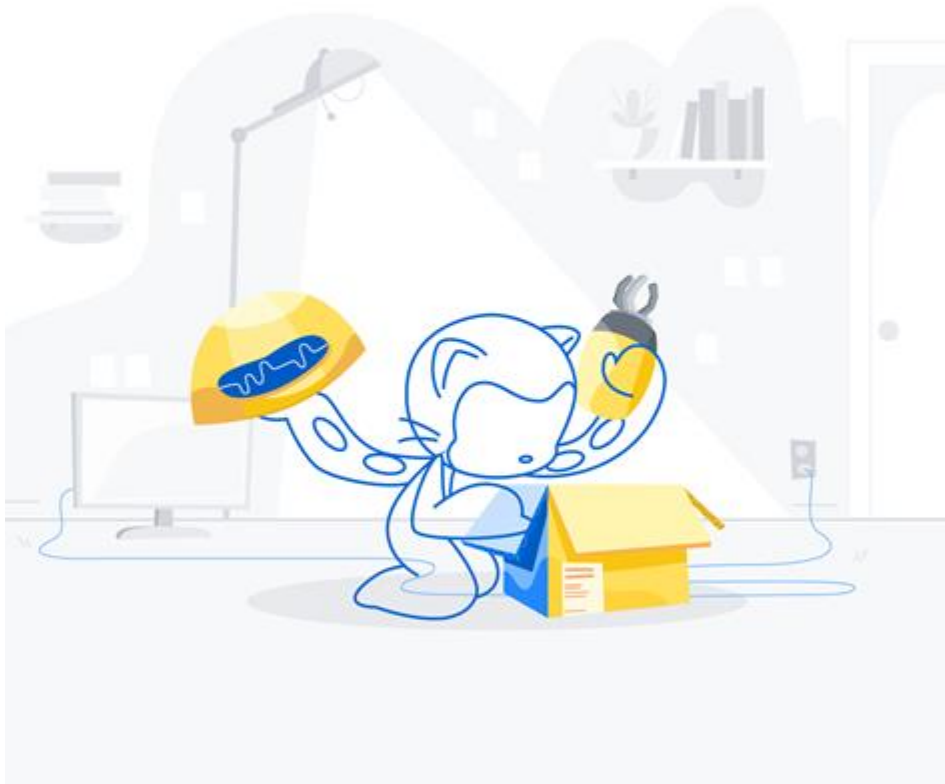
Agenda: Part 1

- GitHub Integration loop
- Tokens and authentication
- Webhooks
- REST
- GraphQL



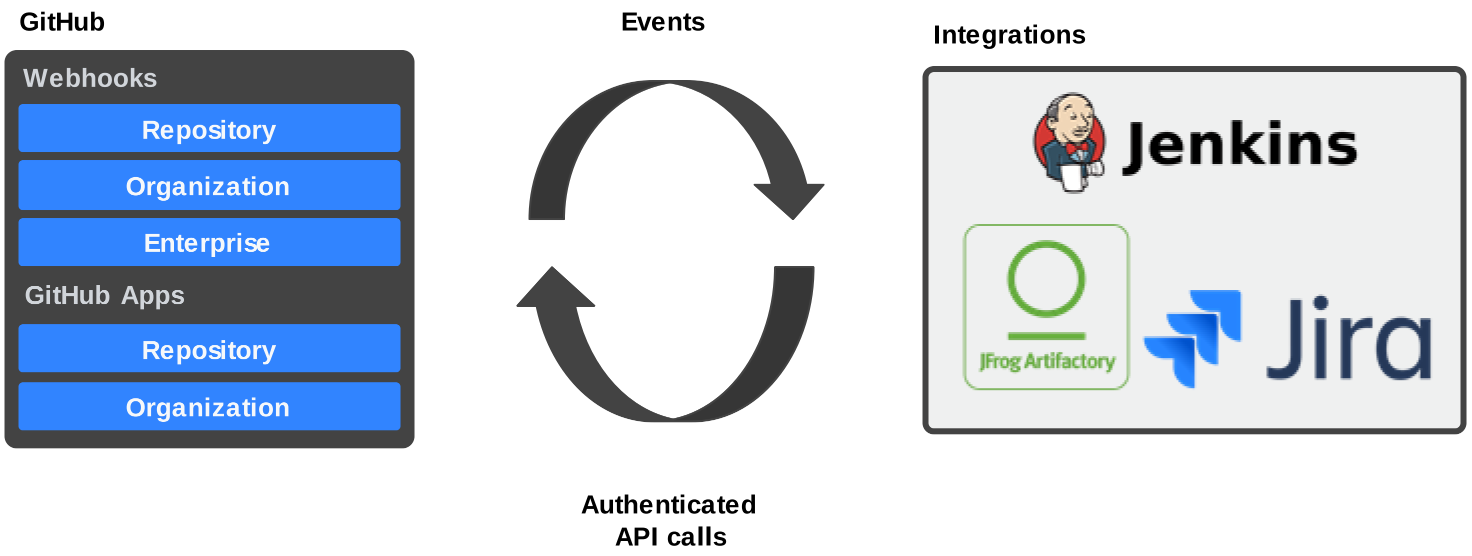
Agenda: Part 2

- Octokit
- GitHub Apps
- Probot
- Introduction to GitHub Actions

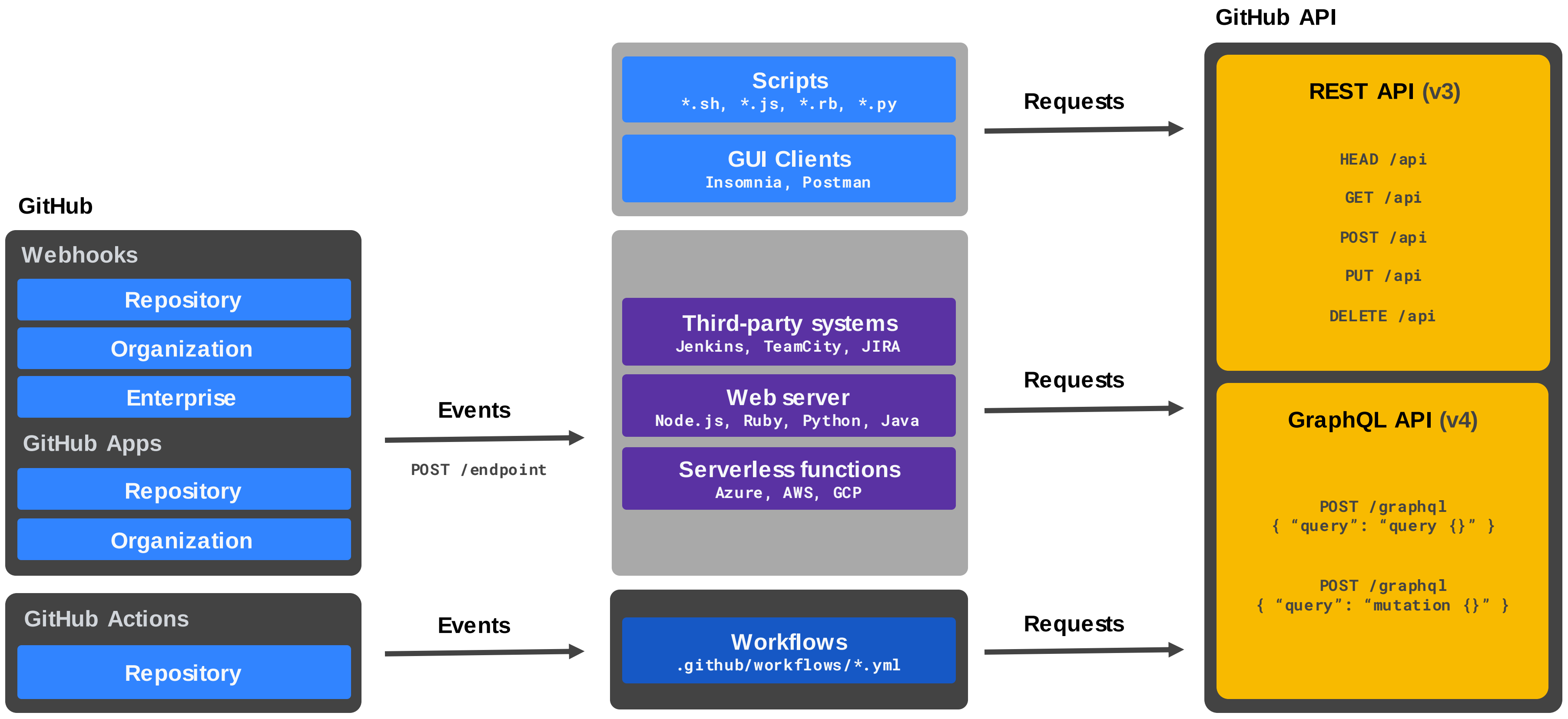


GitHub Integration loop

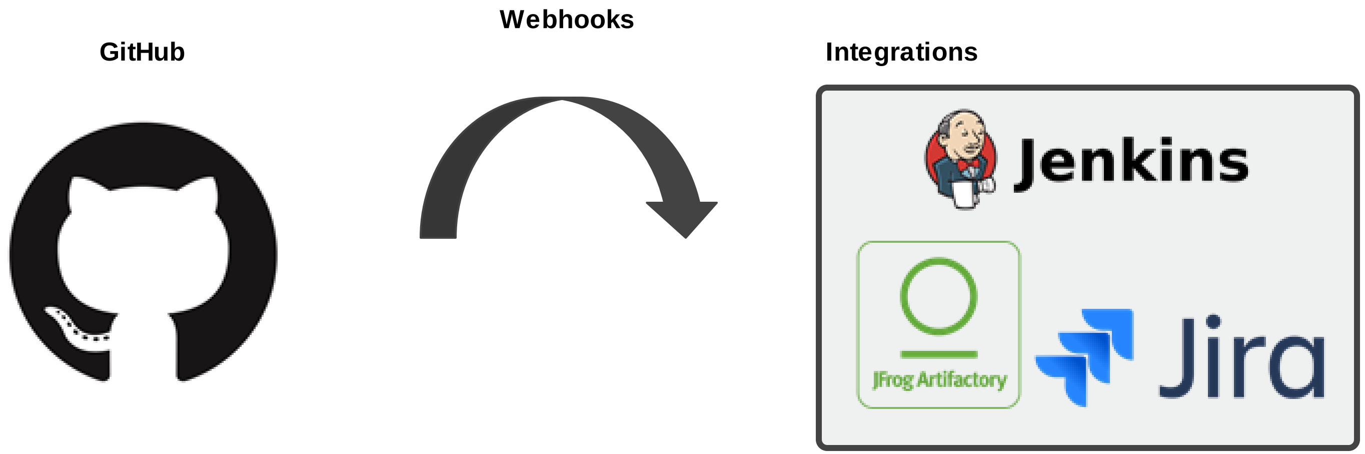
GitHub integration loop



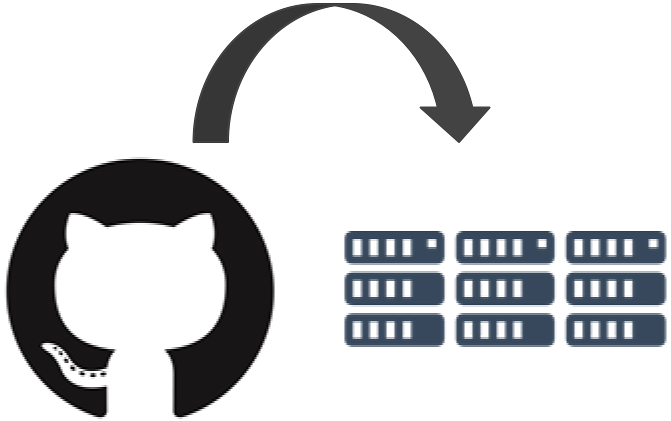
GitHub integration loop components

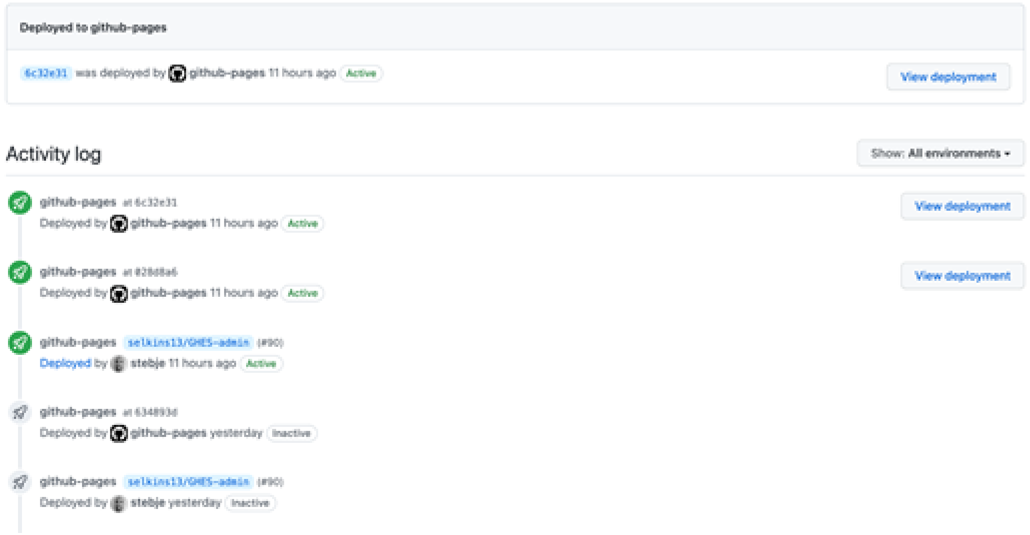


Webhooks


GitHub
Webhooks
Integrations
Jenkins
JFrog Artifactory
Jira

Deployment API (environments)





Deployed to github-pages

6c32e31 was deployed by github-pages 11 hours ago Active

View deployment

Activity log

Show: All environments

github-pages at 6c32e31
Deployed by github-pages 11 hours ago Active
View deployment

github-pages at #28d8a6
Deployed by github-pages 11 hours ago Active
View deployment

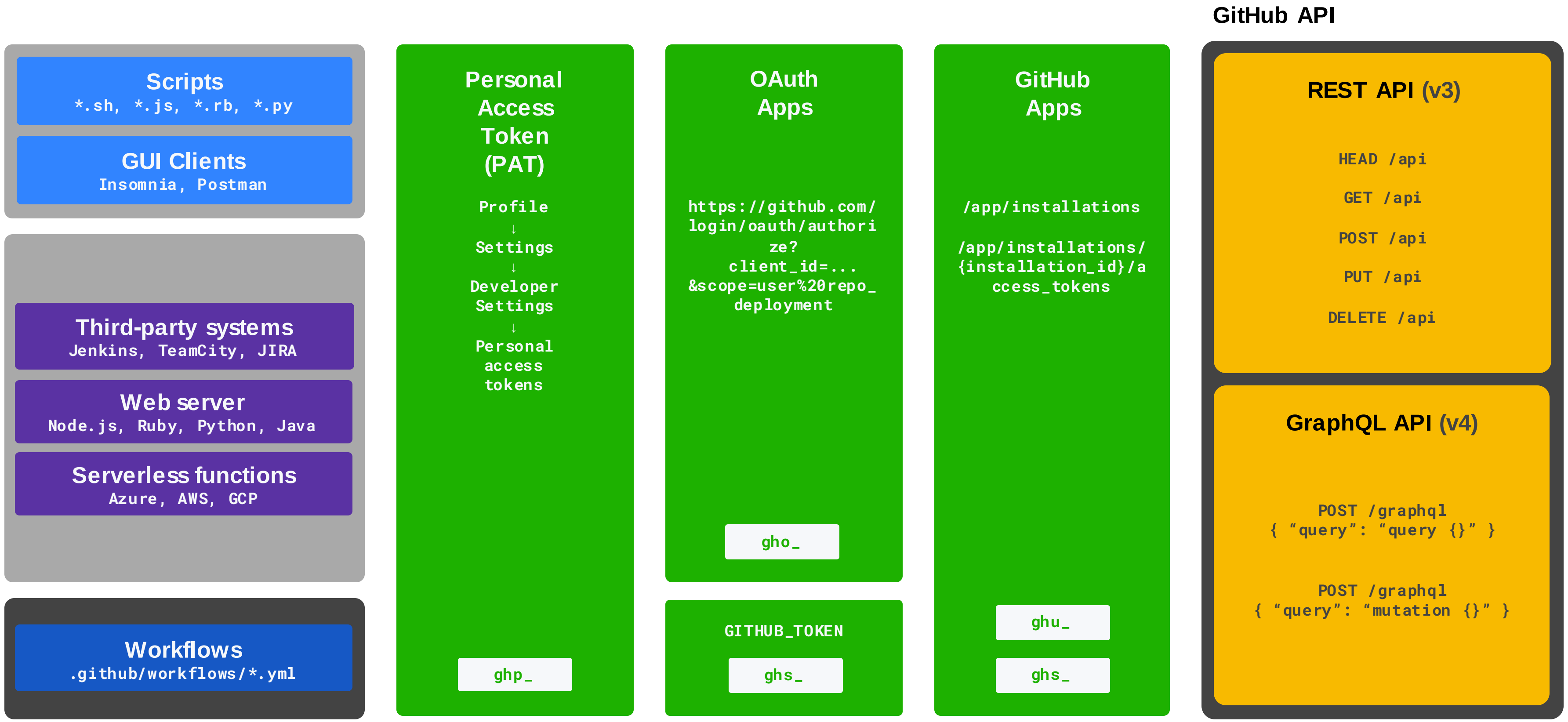
github-pages selkins13/GHES-admin (#90)
Deployed by stebje 11 hours ago Active

github-pages at 634893d
Deployed by github-pages yesterday Inactive

github-pages selkins13/GHES-admin (#90)
Deployed by stebje yesterday Inactive

Tokens and authentication

API Authentication



GitHub API

Scripts
`*.sh, *.js, *.rb, *.py`

GUI Clients
`Insomnia, Postman`

Third-party systems
`Jenkins, TeamCity, JIRA`

Web server
`Node.js, Ruby, Python, Java`

Serverless functions
`Azure, AWS, GCP`

Workflows
`.github/workflows/*.yml`

Personal Access Token (PAT)

```
Profile
↓
Settings
↓
Developer
Settings
↓
Personal
access
tokens
```

`ghp_`

OAuth Apps

```
https://github.com/
login/oauth/authori
ze?
client_id=...
&scope=user%20repo_
deployment
```

`gho_`

`GITHUB_TOKEN`

`ghs_`

GitHub Apps

```
/app/installations

/app/installations/
{installation_id}/a
ccess_tokens
```

`ghu_`

`ghs_`

REST API (v3)

```
HEAD /api
GET /api
POST /api
PUT /api
DELETE /api
```

GraphQL API (v4)

```
POST /graphql
{ "query": "query {}" }

POST /graphql
{ "query": "mutation {}" }
```

Authentication methods

- **GitHub Apps**
- OAuth Apps
- Personal access tokens
- Deploy keys
- Machine users

- A user or organization can own up to **100 GitHub Apps**
- A GitHub App should take actions **independent** of a user
- The GitHub App be installed in a personal account or an organization
- Don’t expect the GitHub App to know and do everything a user can
- Search for "**Works with GitHub Apps**" in the docs
- Can behave as OAuth apps with more permissions
- Up to 15k requests (enterprise) per hour
- Permission changes require approval

Authentication methods

- GitHub Apps
- **OAuth Apps**
- Personal access tokens
- Deploy keys
- Machine users

- A user or organization can own up to **100 OAuth apps**
- An OAuth App should always **act as the authenticated GitHub user** across all of GitHub
- An OAuth App can be used as an **identity provider** by enabling a “Login with GitHub” for the authenticated user
- OAuth Apps can act on all the **authenticated user’s** resources
- Limit of 5k requests per hour
- Requires OAuth flow (client id + auth + code + client secret)

Authentication methods

- GitHub Apps
- OAuth Apps
- **Personal access tokens**
- Deploy keys
- Machine users

- Remember to use this token to represent **yourself only**
- You can perform **one-off cURL requests**
- You can run **personal scripts**
- **Don’t set up a script for your whole team or company to use**
- Use a machine user for authentication
- Limit of 5k requests per hour
- Part of token scanning if leaked publicly
- Removed automatically after one year without use
- Limited permissions by the user

Authentication methods

- GitHub Apps
- OAuth Apps
- Personal access tokens
- **Deploy keys**
- Machine users

- Anyone with access to the repository and server can deploy the project
- Users don’t have to change their local SSH settings
- Deploy keys are read-only by default
- Deploy keys only grant access to a single repository
- Deploy keys are usually not protected by a passphrase

Authentication methods

- GitHub Apps
- OAuth Apps
- Personal access tokens
- Deploy keys
- **Machine users**

- Anyone with access to the repository and server can deploy the project
- No (human) users need to change their local SSH settings
- Multiple keys are not needed
- Only organizations can restrict machine users to read-only access
- Machine user keys, like deploy keys, are usually not protected by a passphrase

API limits

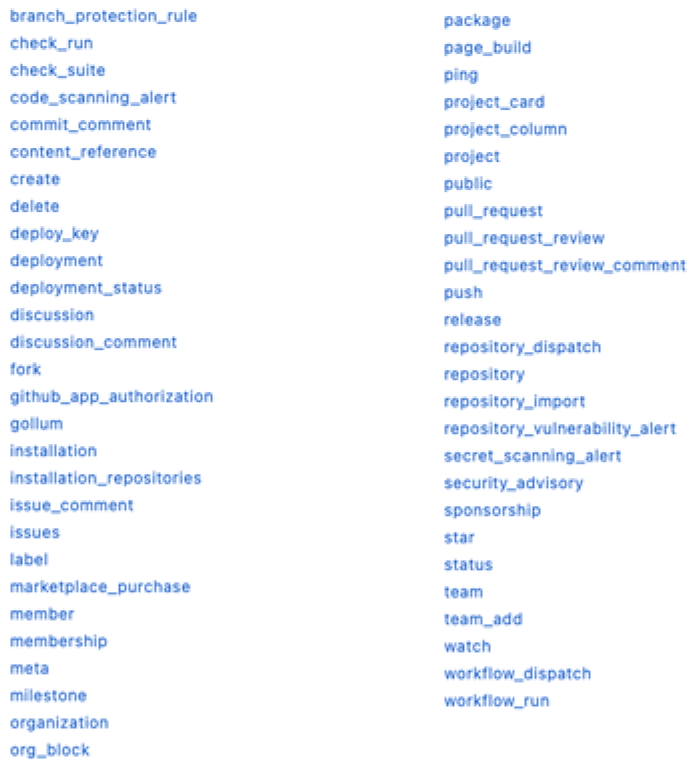
Type of request	Limit (req/h)	Limited by
Unauthenticated requests	60	IP address
Unauthenticated OAuth requests	5000 (client_id + client_secret)	IP address
Personal Access Tokens (PAT)	5000	By user
OAuth Apps	5000 (shared with other by user tokens) (client_id + client_secret)	By user
GitHub Apps (U2S)	5000 (shared with other by user tokens) (client_id + client_secret)	By user
GitHub Apps (S2S)	Up to 15000 for enterprise (client_id, private_key and installation_id)	By app

Recommended token limit remediations

- Use **webhooks** instead of polling
- If polling is required, respect the **X-Poll-Interval header** if available via the endpoint (for example, within the Events API response)
- **Keep concurrency** to a **minimum** per OAuth token
- **Limit large bursts of API calls** (cron, bots, scripts, etc.)
- **Limit bursts of content creation**
- Use **conditional requests** when possible

Webhooks

More than 50 webhooks



branch_protection_rule
check_run
check_suite
code_scanning_alert
commit_comment
content_reference
create
delete
deploy_key
deployment
deployment_status
discussion
discussion_comment
fork
github_app_authorization
gollum
installation
installation_repositories
issue_comment
issues
label
marketplace_purchase
member
membership
meta
milestone
organization
org_block

package
page_build
ping
project_card
project_column
project
public
pull_request
pull_request_review
pull_request_review_comment
push
release
repository_dispatch
repository
repository_import
repository_vulnerability_alert
secret_scanning_alert
security_advisory
sponsorship
star
status
team
team_add
watch
workflow_dispatch
workflow_run

Secure your webhooks

- Use always a **secret** when setting up a webhook
- **Verify the signature** manually or use a **framework** like probot to build your integrations
- Set network policies to allow inputs from GitHub addresses

REST API (V3)

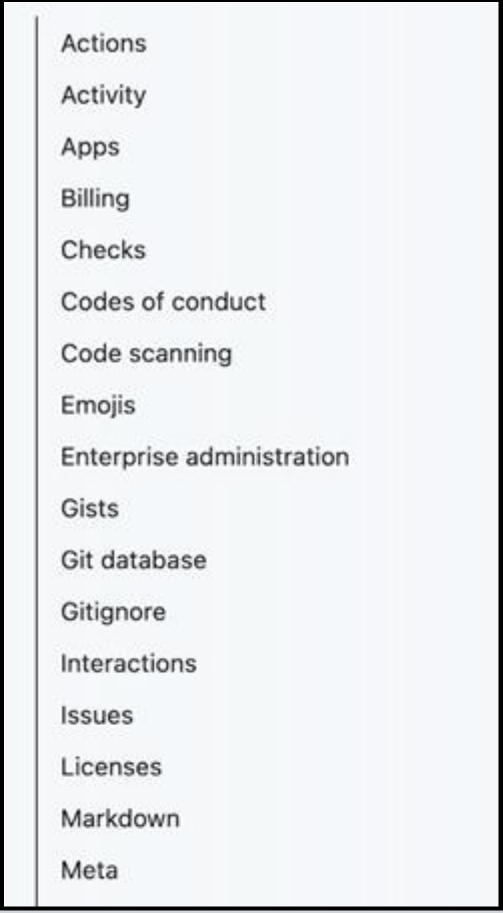
HTTP calls in REST

CREATE	POST
READ	GET
UPDATE	PATCH/POST
DELETE	DELETE

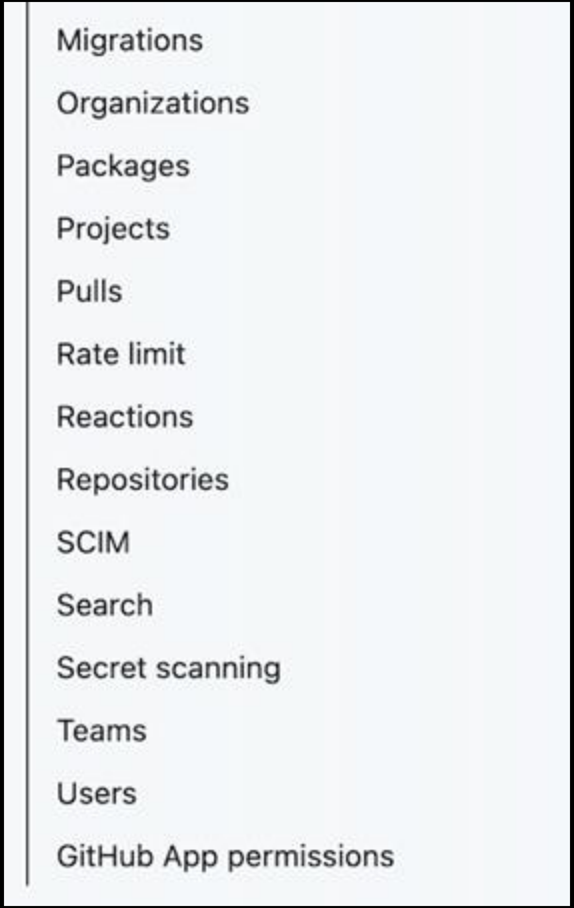
API Response codes

200/201	**302**	**401**	**404**
Ok	Not modified	Unauthorized	Not found

https://docs.github.com/en/rest

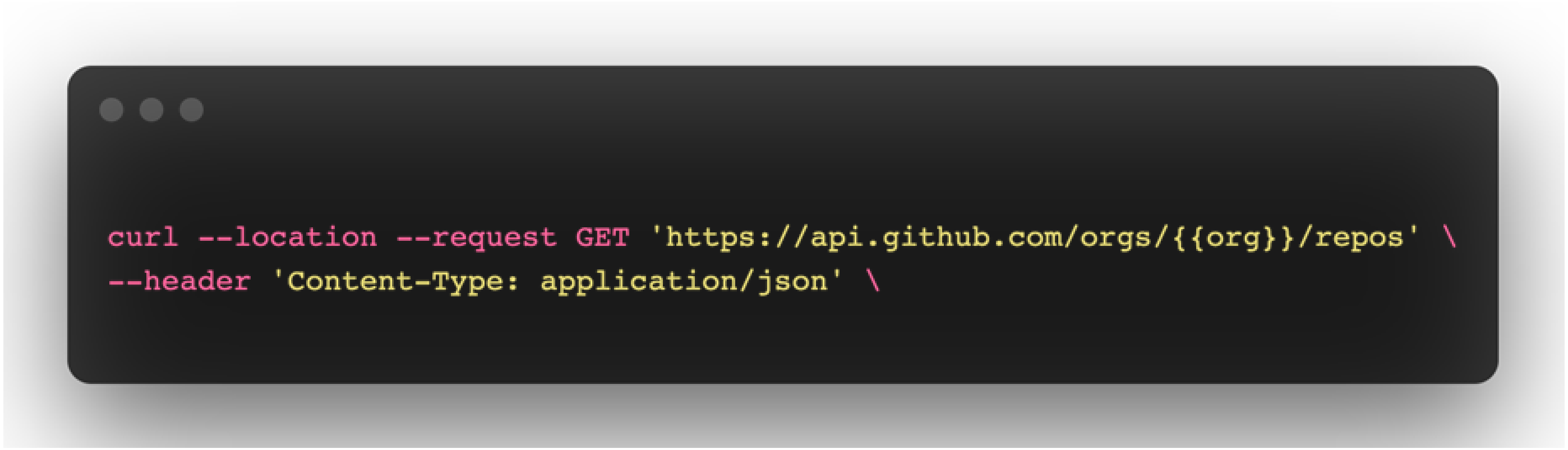


Actions
Activity
Apps
Billing
Checks
Codes of conduct
Code scanning
Emojis
Enterprise administration
Gists
Git database
Gitignore
Interactions
Issues
Licenses
Markdown
Meta



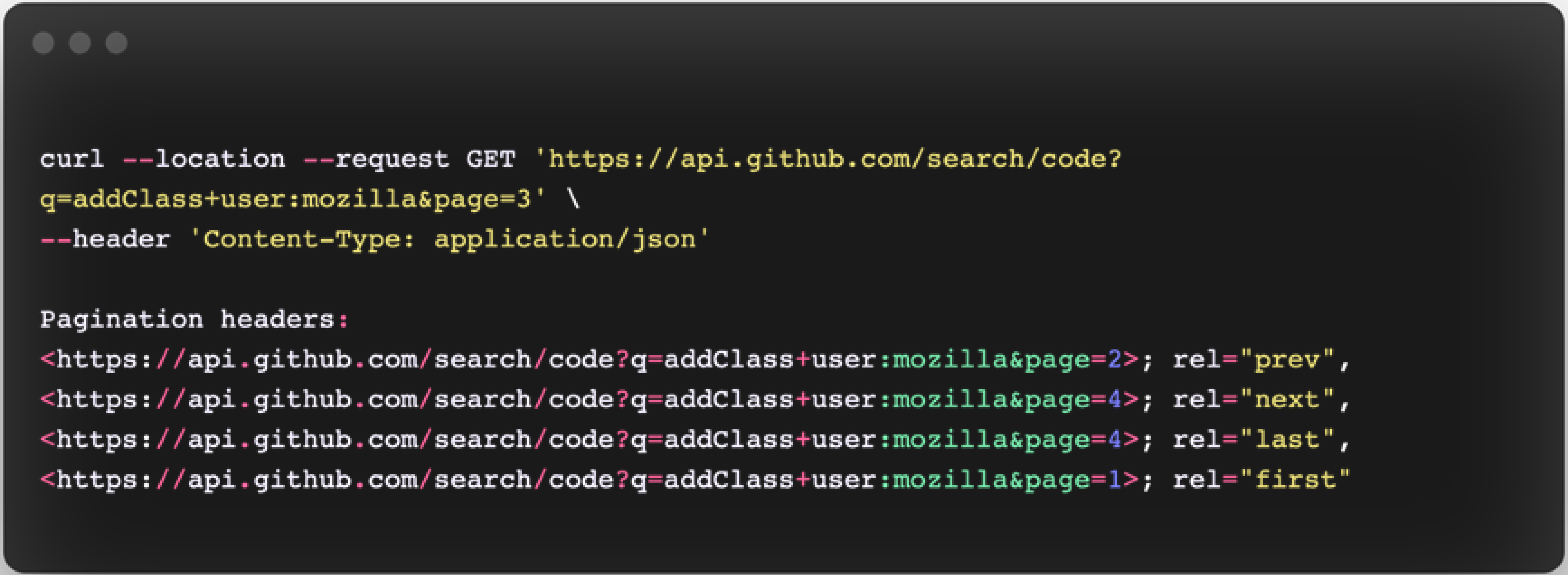
Migrations
Organizations
Packages
Projects
Pulls
Rate limit
Reactions
Repositories
SCIM
Search
Secret scanning
Teams
Users
GitHub App permissions

Example REST request



```
curl --location --request GET 'https://api.github.com/orgs/{{org}}/repos' \
--header 'Content-Type: application/json' \
```

Pagination



```
curl --location --request GET 'https://api.github.com/search/code?
q=addClass+user:mozilla&page=3' \
--header 'Content-Type: application/json'

Pagination headers:
<https://api.github.com/search/code?q=addClass+user:mozilla&page=2>; rel="prev",
<https://api.github.com/search/code?q=addClass+user:mozilla&page=4>; rel="next",
<https://api.github.com/search/code?q=addClass+user:mozilla&page=4>; rel="last",
<https://api.github.com/search/code?q=addClass+user:mozilla&page=1>; rel="first"
```

Conditional requests

- ETag and Last-Modified provides a unique id for the request resource
- If-Modified-Since and If-None-Match can be used to optimize the requests
- When not modified the response code will be 304, and does not count toward rate limit

Rate limit

- X-RateLimit-Limit, X-RateLimit-Remaining, X-RateLimit-Reset, X-RateLimit-Used, X-RateLimit-Resource are the response headers that give information about the limits
- You should review these headers always to avoid throwing errors
- Space the request execution if possible to get the best out of the rate limits

API response Headers

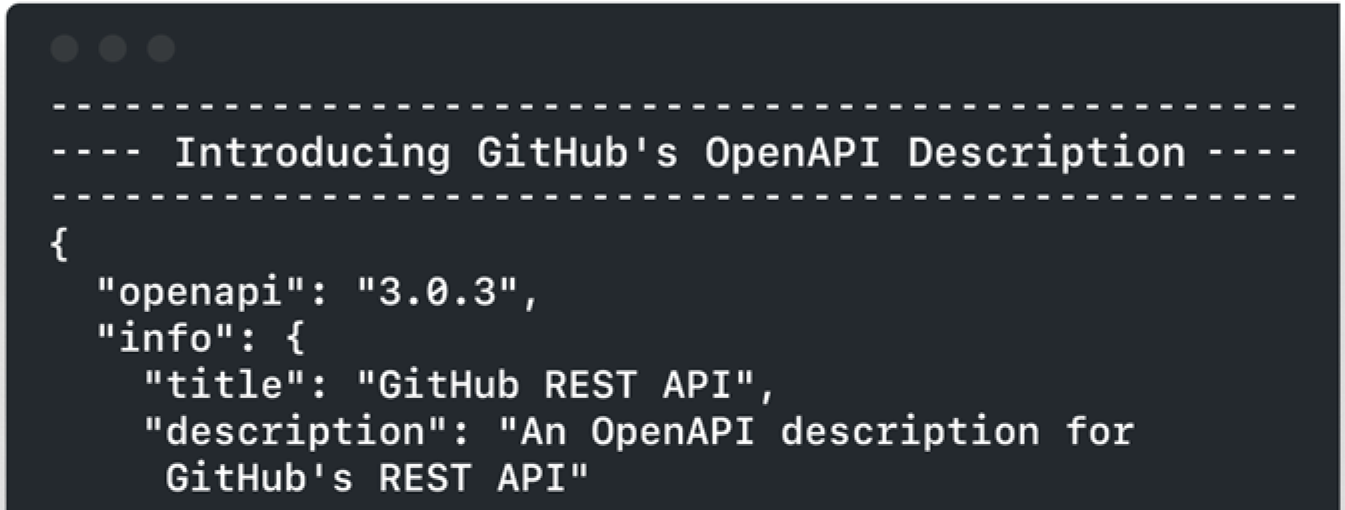
Header name	Example value	Use
X-OAuth-Scopes	repo, user	Permissions of the token
github-authentication-token-expiration	2021-09-07 22:00:00 UTC	Token expiration
Link	<https://api.github.com/search/code?q=addClass+user%3Amozilla&page=2>; rel="prev", ...	Pagination headers
X-RateLimit-Limit	30	Total requests limit
X-RateLimit-Remaining	28	Remaining requests
X-RateLimit-Reset	1630870622	Reset timestamp
X-RateLimit-Used	2	Limit used
X-RateLimit-Resource	search	Resource consumed

API response Headers

Header name	Example value	Use
X-GitHub-Request-Id	E32C:2921:3BBD7:3E2CD:61351FF8	Unique id for support
X-GitHub-Media-Type	github.v3; format=json	API and encoding
x-oauth-client-id	8e3a347e1e3763cdc30b	OAuth client id
ETag	W/"0b4d17290f20634a5ecd09b90543db3aded907cad8f812f3a3344b8d8f7e9113"	Tag for caching
Cache-Control	private, max-age=60, s-maxage=60	Caching policy
Last-Modified	Mon, 09 Aug 2021 14:37:31 GMT	Modifications to the resource

REST API compatibility

https://github.com/github/rest-api-description



```
------------------------------------------------------
---- Introducing GitHub's OpenAPI Description ----
------------------------------------------------------
{
  "openapi": "3.0.3",
  "info": {
    "title": "GitHub REST API",
    "description": "An OpenAPI description for
     GitHub's REST API"
```

API recommendations

- Follow best practices for integrators
- Adhere to rate limits
- Use pagination
- Use Octokit plugins
- Understand resource limitations and query costs

GraphQL API (V4)

GraphQL Standard

- Initially developed by facebook in 2015, and the community
- Now it has its own foundation for the project
- GraphQL is possibly the new API standard, taking over from REST

Why it was needed?

- **Extra data** provided in REST
- **Nested data** grabbing was hard and required reorganization of the endpoints
- Adding new elements to the API can be **hard** and require **versioning**
- **Lack of documentation**
- Easy to understand APIs and that support **introspection**

GraphQL benefits

- **Standard** language to query
- **Introspection** and documentation
- Once learnt **easy to use**
- Complex **queries** can be **combined**
- **No more heavy chained** requests for a single key value pair
- Rapidly **growing** API
- **Typed** API
- Better **error handling** and logging

GraphQL authentication

- It uses the **same systems** as the v3 API
- If you can **access** using the token, you can access it in GraphQL
- https://docs.github.com/en/graphql

GraphQL API (V4)

GraphQL Schema

https://docs.github.com/public/schema.docs.graphql

- Queries
- Mutations
- Objects
- Interfaces
- Enums
- Unions
- Input objects
- Scalars

GraphQL Schema

- **Queries**
- Mutations
- Objects
- Interfaces
- Enums
- Unions
- Input objects
- Scalars

The query type defines GraphQL operations that **retrieve data** from the server. They require top level root objects

https://docs.github.com/en/graphql/reference/queries

GraphQL Schema

- Queries
- **Mutations**
- Objects
- Interfaces
- Enums
- Unions
- Input objects
- Scalars

Defines GraphQL operations that **change data** on the server. It is analogous to performing HTTP verbs such as *POST*, *PATCH*, and *DELETE*. They require top level root objects

https://docs.github.com/en/graphql/reference/mutations

GraphQL Schema

- Queries
- Mutations
- **Objects**
- Interfaces
- Enums
- Unions
- Input objects
- Scalars

Objects in GraphQL represent the **resources you can access**. An object can contain a list of fields, which are specifically typed.

For example, the `Repository` object has a field called name, which is a `String`.

https://docs.github.com/en/graphql/reference/objects

GraphQL Schema

- Queries
- Mutations
- Objects
- **Interfaces**
- Enums
- Unions
- Input objects
- Scalars

Interfaces serve as parent objects from which other objects can inherit.

For example, `Lockable` is an interface because both `Issue` and `PullRequest` objects can be locked.

https://docs.github.com/en/graphql/reference/interfaces

GraphQL Schema

- Queries
- Mutations
- Objects
- Interfaces
- **Enums**
- Unions
- Input objects
- Scalars

Enums represent possible sets of values for a field. They are usually written in capital letters.

For example, the `Issue` object has a field called state. The state is an enum (specifically, of type `IssueState`) because it may be `OPEN` or `CLOSED`

https://docs.github.com/en/graphql/reference/enums

GraphQL Schema

- Queries
- Mutations
- Objects
- Interfaces
- Enums
- **Unions**
- Input objects
- Scalars

A union is a type of object representing many objects. Similar to inheritance of oop.

For example, a field marked as an `ProjectCardItem` could be an `Issue` or a `PullRequest` because each of those objects can be inside a project card

https://docs.github.com/en/graphql/reference/unions

GraphQL Schema

- Queries
- Mutations
- Objects
- Interfaces
- Enums
- Unions
- **Input objects**
- Scalars

Objects that **describe input data** for mutations to pass parameters that describe the operation.

https://docs.github.com/en/graphql/reference/input-objects

GraphQL Schema

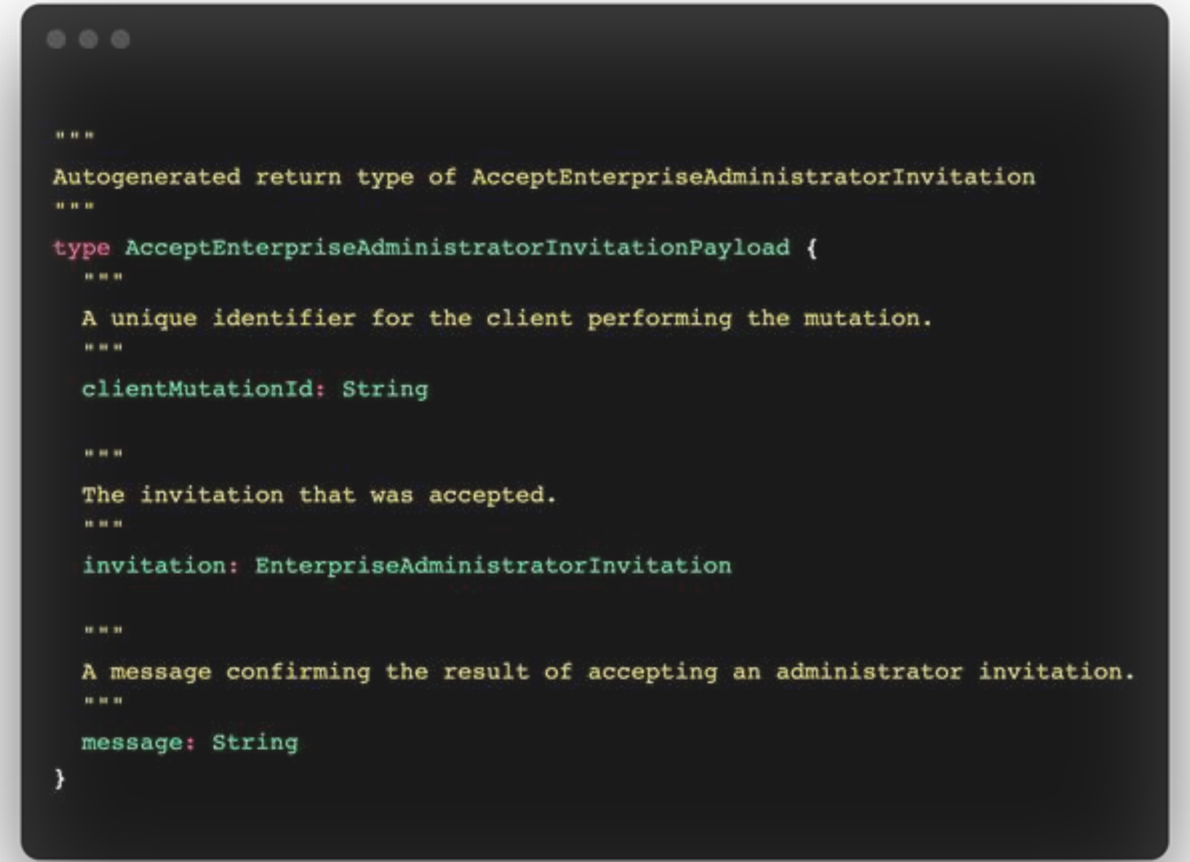
- Queries
- Mutations
- Objects
- Interfaces
- Enums
- Unions
- Input objects
- **Scalars**

Scalars are **primitive values**: `Int`, `Float`, `String`, `Boolean`, or `ID`.

When calling GraphQL, you must specify nested subfields until you return only scalars

https://docs.github.com/en/graphql/reference/scalars

Schema definition example

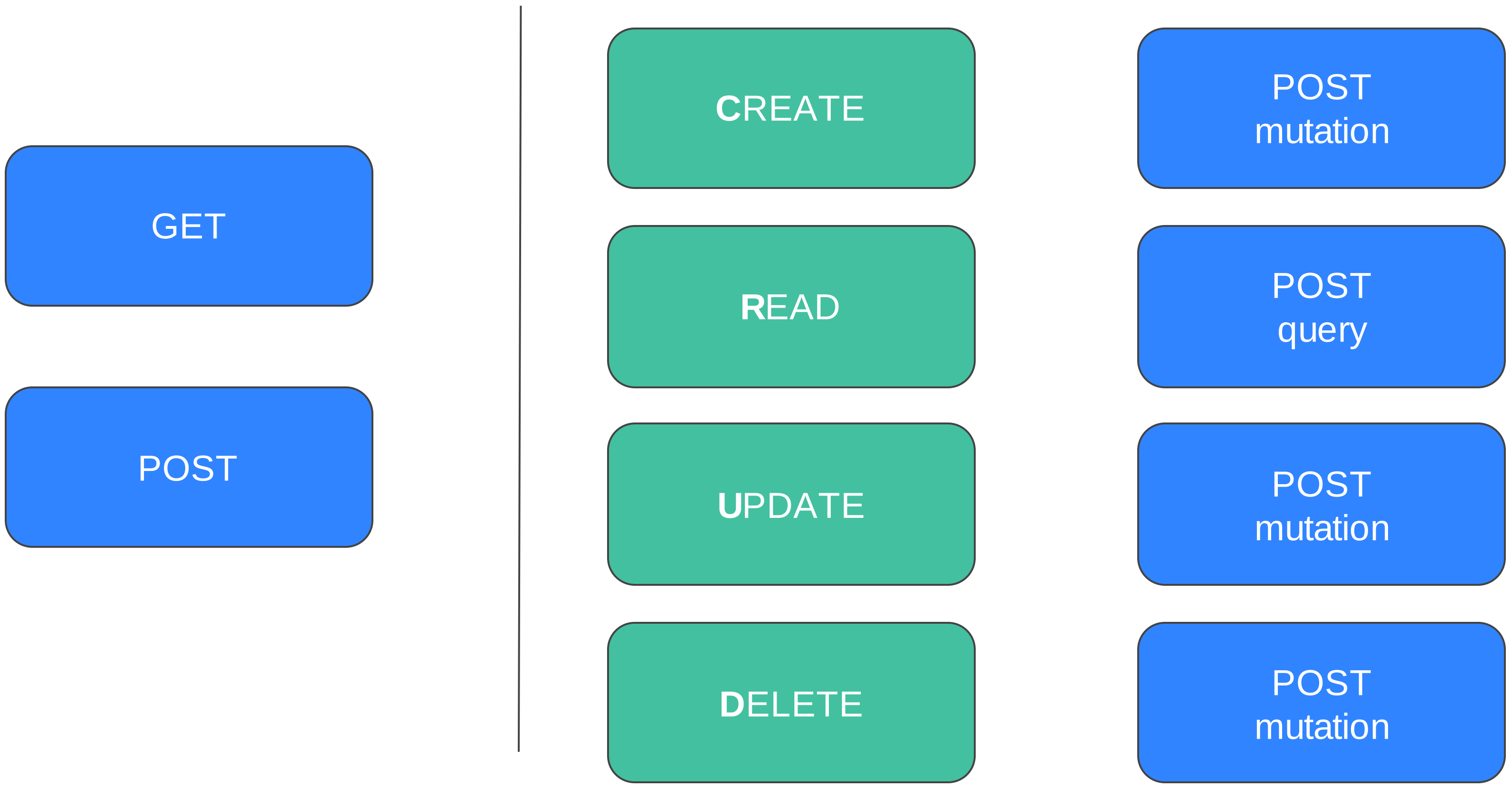


```
"""
Autogenerated return type of AcceptEnterpriseAdministratorInvitation
"""
type AcceptEnterpriseAdministratorInvitationPayload {
  """
  A unique identifier for the client performing the mutation.
  """
  clientMutationId: String

  """
  The invitation that was accepted.
  """
  invitation: EnterpriseAdministratorInvitation

  """
  A message confirming the result of accepting an administrator invitation.
  """
  message: String
}
```

GraphQL API (V4)

Introduction ○ Concepts ○ Queries and mutations ○ Limits

Http calls in GraphQL



Queries

Top level objects

- code(s)OfConduct
- enterprise
- enterprise invitations
- license(s)
- marketplace (categories, listing(s)
- meta
- organization
- rate limits
- repository and owner
- search
- security advisories
- security vulnerabilities
- sponsors
- topics
- users
- viewer

GraphQL query

Simple Viewer Query

```graphql
query {
  viewer {
    login
    bio
  }
}
```

Query result

```json
{
  "data": {
    "viewer": {
      "login": "colossus9",
      "bio": "Solution Architect @github"
    }
  }
}
```

GraphQL query

User query with pagination

```
query {
  user(login: "colossus9") {
    repositories(first: 3) {
      nodes {
        name
      }
    }
  }
}
```

Query result

```
{
  "data": {
    "user": {
      "repositories": {
        "nodes": [
          {
            "name": "bda-puppet"
          },
          {
            "name": "intro-to-github"
          },
          {
            "name": "MatchingGame"
          }
        ]
      }
    }
  }
}
```

GraphQL query

Introducing variables

```
query($user: String!) {
  user(login: $user) {
    repositories(first: 3) {
      nodes {
        name
      }
    }
  }
}
```

```
{
  "user": "colossus9"
}
```

Query result

```
{
  "data": {
    "user": {
      "repositories": {
        "nodes": [
          {
            "name": "bda-puppet"
          },
          {
            "name": "intro-to-github"
          },
          {
            "name": "MatchingGame"
          }
        ]
      }
    }
  }
}
```

GraphQL query

Fragments

```
query($org: String!) {
  organization(login: $org) {
    auditLog(first: 3){
      nodes {
        ... on AuditEntry {
          action
        }
      }
    }
  }
}
```

```
{
  "org": “colossus9-demorg”
}
```

Query result

```
{
  "data": {
    "organization": {
      "auditLog": {
        "nodes": [
          {
            "action": "org.remove_member"
          },
          {
            "action": "org.add_member"
          },
          {
            "action": "org.invite_member"
          }
        ]
      }
    }
  }
}
```

GraphQL query

Aliases

```
query($org: String!) {
 self: viewer {
  login
 }
 org: organization(login: $org) {
  auditLog(first: 3){
   nodes {
    ... on AuditEntry {
     action
    }
   }
  }
 }
}
```

```
{
 "org": “colossus9-demorg”
}
```

Query result

```
{
 "data": {
  "self": {
   "login": "colossus9"
  },
  "org": {
   "auditLog": {
    "nodes": [
     {
      "action": "org.remove_member"
     },
     {
      "action": "org.add_member"
     },
     {
      "action": "org.invite_member"
     }
    ]
   }
  }
 }
}
```

Mutations

GraphQL query

Change status mutation

```
mutation($input: ChangeUserStatusInput!) {
  changeUserStatus(input: $input) {
    clientMutationId
    status {
      message
      emoji
      indicatesLimitedAvailability
    }
  }
}
```

```
{
  "input": {
    "message": "This is a demo"
  }
}
```

Query result

```
{
  "data": {
    "changeUserStatus": {
      "clientMutationId": null,
      "status": {
        "message": "This is a demo",
        "emoji": null,
        "indicatesLimitedAvailability": false
      }
    }
  }
}
```

Exercise: Modify the name of a repository

GraphQL query

Get the repository data

```
query($org: String!) {
  organization(login: $org) {
    repositories(first: 10){
      nodes {
        id
        name
      }
    }
  }
}
```

```
{
          "org": "colossus9-demorg"
}
```

Query result

```
{
 "data": {
  "organization": {
   "repositories": {
    "nodes": [
     {
      "id": "MDEwOlJlcG9zaXRvcnkyMzMwNjMyNDc=",
      "name": "Test"
     },
     {
      "id": "MDEwOlJlcG9zaXRvcnkyMzY3NTQxMDQ=",
      "name": "template"
     },
     {
      "id": "MDEwOlJlcG9zaXRvcnkyNDQzODAzMTg=",
      "name": "another-internal"
     },
     {
      "id": "MDEwOlJlcG9zaXRvcnkyNDQ2NTExOTM=",
      "name": "runner-on-container"
     },
     {
      "id": "MDEwOlJlcG9zaXRvcnkyNDYwOTk1NTE=",
      "name": "internal-to-private"
     }
    ]
   }
  }
 }
}
```

GraphQL query

Query result

Update the repository

```
mutation($input: UpdateRepositoryInput!) {
 updateRepository(input: $input) {
  clientMutationId
  repository {
   name
  }
 }
}
```

```
{
			"input": {
						"repositoryId":
"MDEwOlJlcG9zaXRvcnkyMzMwNjMyNDc=",
						"name": "Test from graphQL"
			}
}
```

```
{
 "data": {
  "updateRepository": {
   "clientMutationId": null,
   "repository": {
    "name": "Test-from-graphQL"
   }
  }
 }
}
```

Some mutations supported

- addComment
- addAssigneesToAssignable
- addLabelsToLabelable
- changeUserStatus
- closePullRequest
- createDeployment
- createRepository
- deleteIssue
- deleteLabel
- deleteTeamDiscussion
- dismissPullRequestReview
- mergePullRequest
- removeReaction
- updateCheckRun
- updateProject
- ...and much more…

GraphQL API (V4)

Limits

- **Paginated resources** require first/last for the pagination. Max of 100 elements
- A call cannot request more than 500k nodes
- Calculation is done with the numbers multiplied also per nested loops
- Rate limits are calculated with a score. Max of 5k score points
- It uses the first/last parameter to calculate the score

GraphQL query

```graphql
query {
  viewer {
    login
  }
  rateLimit {
    limit
    cost
    remaining
    resetAt
  }
}
```

Query result

```json
{
  "data": {
    "viewer": {
      "login": "colossus9"
    },
    "rateLimit": {
      "limit": 5000,
      "cost": 1,
      "remaining": 4994,
      "resetAt": "2020-04-22T08:03:32Z"
    }
  }
}
```

Limits

- **Limit**: max points/h the token can consume
- **Cost**: the query points
- **Remaining**: remaining points/h
- **ResetAt**: moment when the limit gets restored

GraphQL query

```
query {
  viewer {
    repositories(first: 50) {
      edges {
        repository:node {
          name
          pullRequests(first: 20) {
            edges {
              pullRequest:node {
                title
                comments(first: 10) {
                  edges {
                    comment:node {
                      bodyHTML
                    }
                  }
                }
              }
            }
          }
        }
      }
    }
  }
}
```

Number of nodes

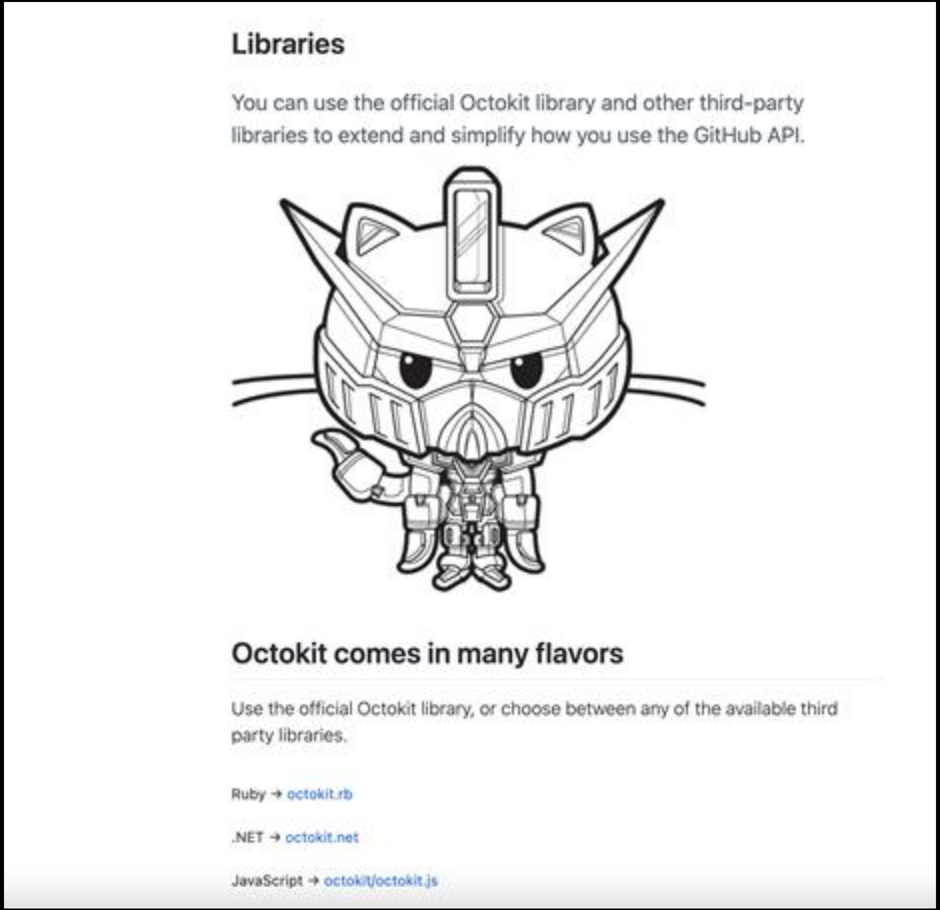
```
    50
+   50 * 20
    50 * 20 * 10
    ____________

     1 1 0 5 0
```

Query score

```
    1 (repositories)
+   50 (pr)
    50 * 20 (comments)
    ____________

    1 0 5 1
```

Octokit

https://docs.github.com/en/rest/overview/libraries ★



Libraries

You can use the official Octokit library and other third-party libraries to extend and simplify how you use the GitHub API.

Octokit comes in many flavors

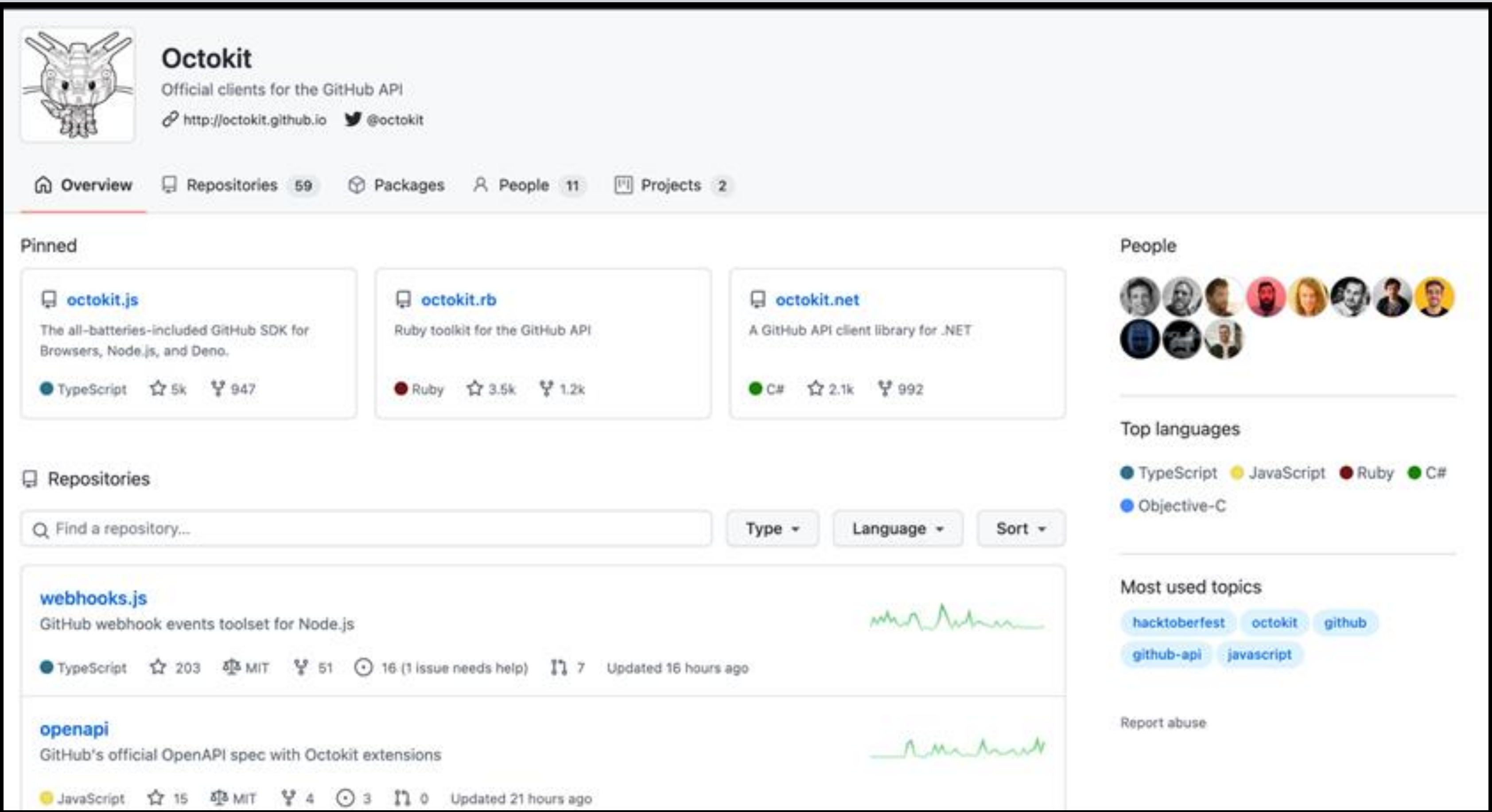
Use the official Octokit library, or choose between any of the available third party libraries.

Ruby → octokit.rb

.NET → octokit.net

JavaScript → octokit/octokit.js

https://github.com/octokit



Octokit

Official clients for the GitHub API

http://octokit.github.io @octokit

Overview Repositories 59 Packages People 11 Projects 2

Pinned

octokit.js
The all-batteries-included GitHub SDK for Browsers, Node.js, and Deno.
TypeScript 5k 947

octokit.rb
Ruby toolkit for the GitHub API
Ruby 3.5k 1.2k

octokit.net
A GitHub API client library for .NET
C# 2.1k 992

Repositories

Find a repository... Type Language Sort

webhooks.js
GitHub webhook events toolset for Node.js
TypeScript 203 MIT 51 16 (1 issue needs help) 7 Updated 16 hours ago

openapi
GitHub's official OpenAPI spec with Octokit extensions
JavaScript 15 MIT 4 3 0 Updated 21 hours ago

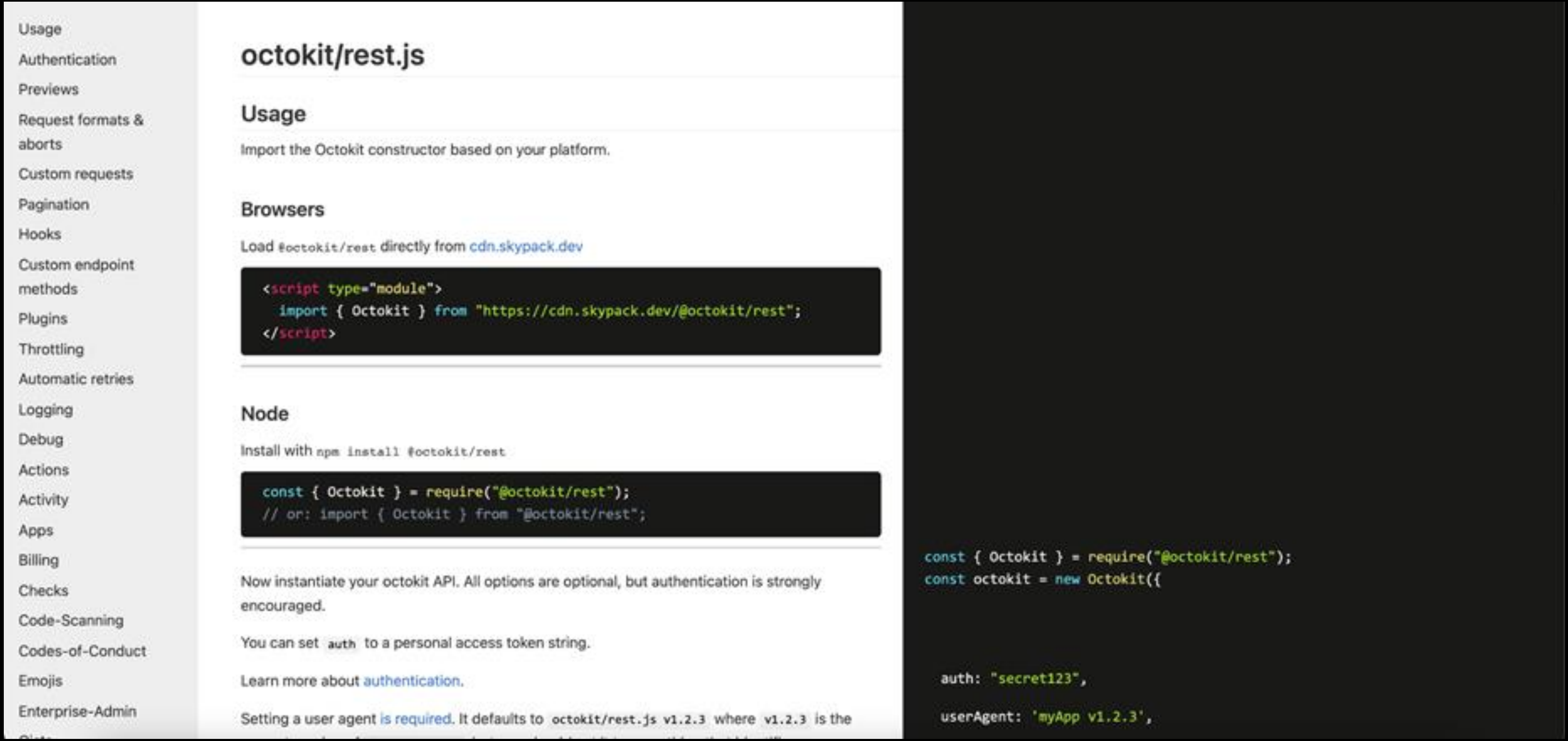
People

Top languages
TypeScript JavaScript Ruby C# Objective-C

Most used topics
hacktoberfest octokit github github-api javascript

Report abuse

https://octokit.github.io/rest.js/v18



- Usage
- Authentication
- Previews
- Request formats & aborts
- Custom requests
- Pagination
- Hooks
- Custom endpoint methods
- Plugins
- Throttling
- Automatic retries
- Logging
- Debug
- Actions
- Activity
- Apps
- Billing
- Checks
- Code-Scanning
- Codes-of-Conduct
- Emojis
- Enterprise-Admin

octokit/rest.js

Usage

Import the Octokit constructor based on your platform.

Browsers

Load `@octokit/rest` directly from cdn.skypack.dev

```
<script type="module">
  import { Octokit } from "https://cdn.skypack.dev/@octokit/rest";
</script>
```

Node

Install with `npm install @octokit/rest`

```
const { Octokit } = require("@octokit/rest");
// or: import { Octokit } from "@octokit/rest";
```

Now instantiate your octokit API. All options are optional, but authentication is strongly encouraged.

You can set `auth` to a personal access token string.

Learn more about authentication.

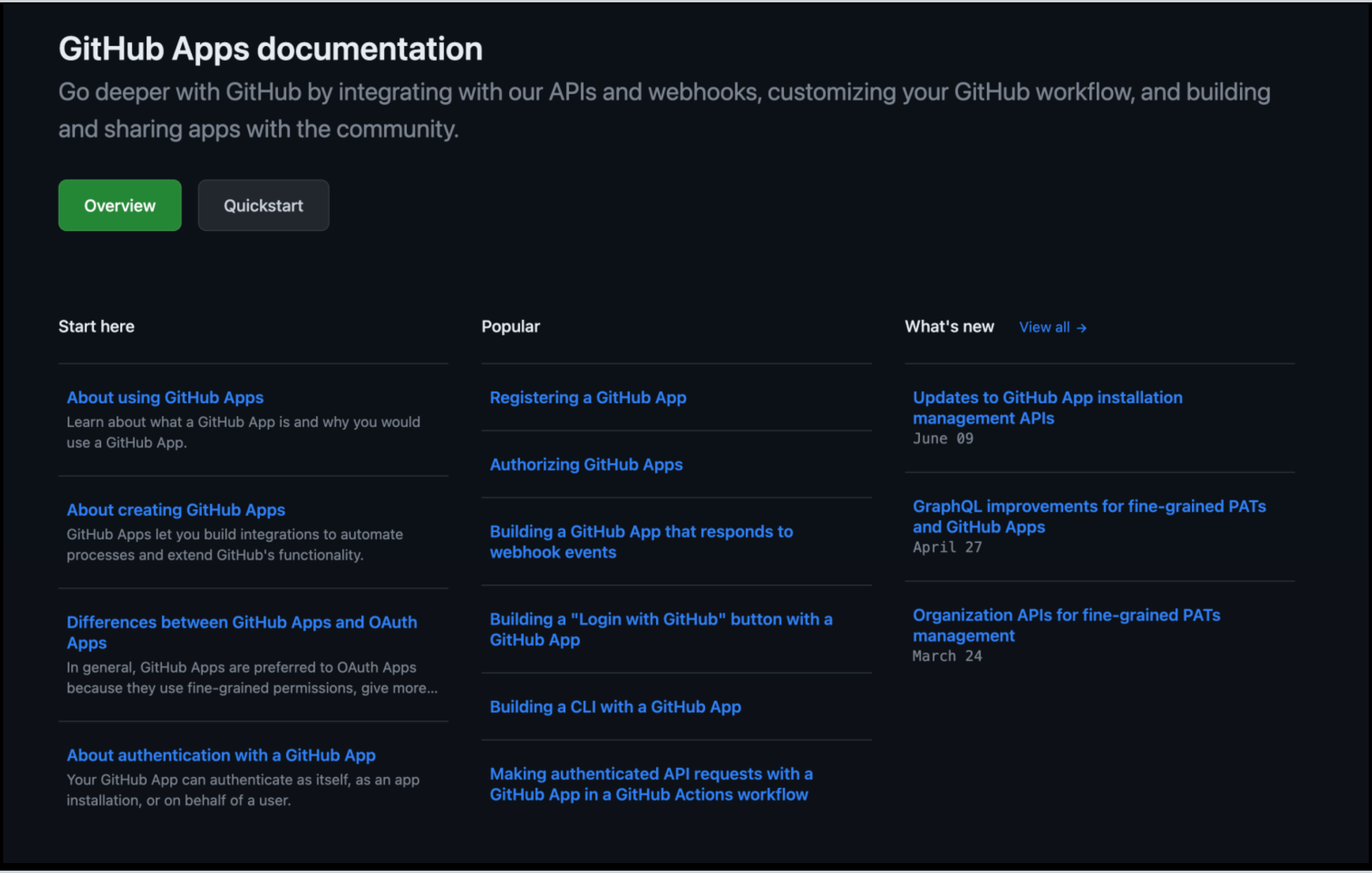
Setting a user agent is required. It defaults to `octokit/rest.js v1.2.3` where `v1.2.3` is the

```
const { Octokit } = require("@octokit/rest");
const octokit = new Octokit({



  auth: "secret123",

  userAgent: 'myApp v1.2.3',
```

Demo: build a script to view current user name using octokit for any language

GitHub Apps

https://docs.github.com/en/apps



GitHub Apps documentation

Go deeper with GitHub by integrating with our APIs and webhooks, customizing your GitHub workflow, and building and sharing apps with the community.

Start here

About using GitHub Apps
Learn about what a GitHub App is and why you would use a GitHub App.

About creating GitHub Apps
GitHub Apps let you build integrations to automate processes and extend GitHub's functionality.

Differences between GitHub Apps and OAuth Apps
In general, GitHub Apps are preferred to OAuth Apps because they use fine-grained permissions, give more...

About authentication with a GitHub App
Your GitHub App can authenticate as itself, as an app installation, or on behalf of a user.

Popular

Registering a GitHub App

Authorizing GitHub Apps

Building a GitHub App that responds to webhook events

Building a "Login with GitHub" button with a GitHub App

Building a CLI with a GitHub App

Making authenticated API requests with a GitHub App in a GitHub Actions workflow

What's new View all →

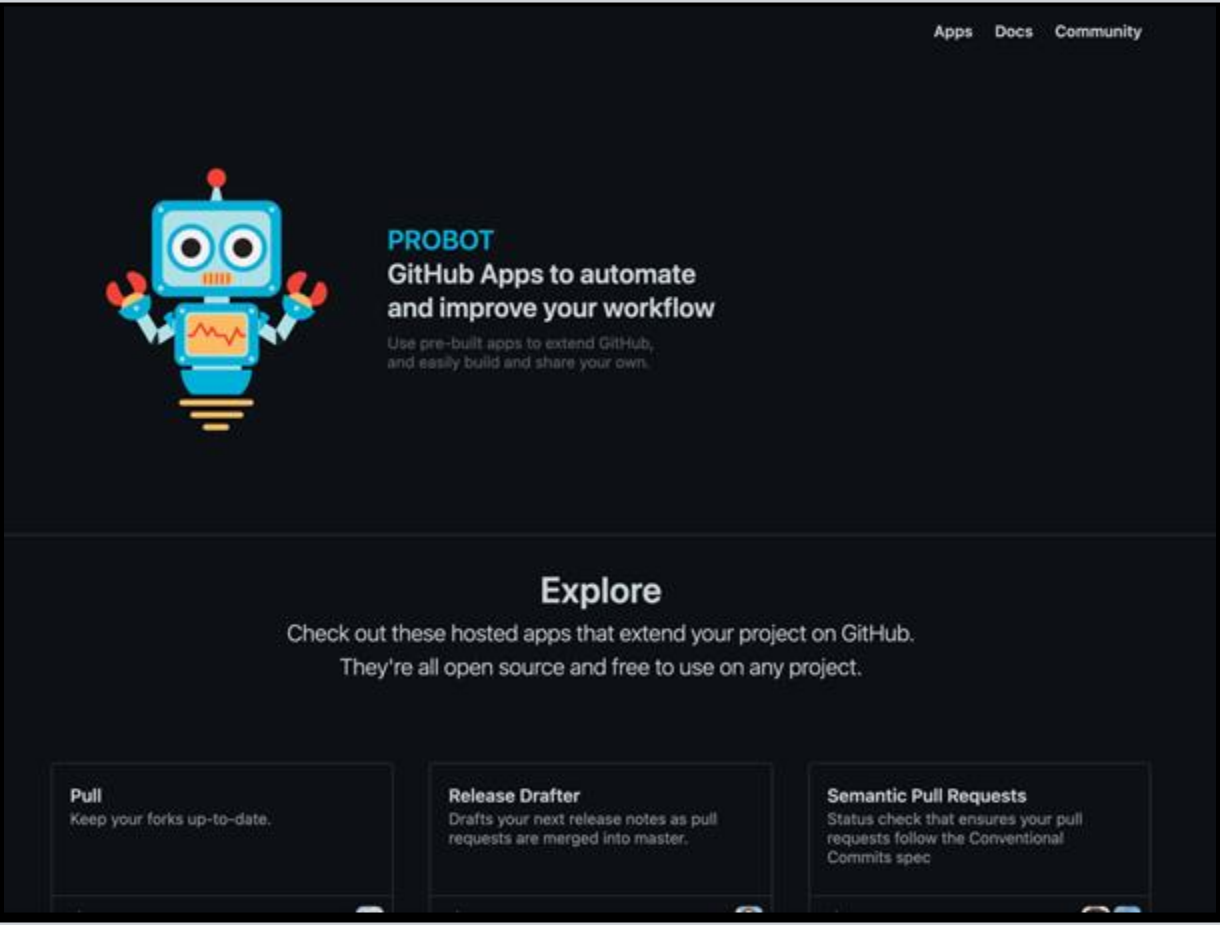
Updates to GitHub App installation management APIs
June 09

GraphQL improvements for fine-grained PATs and GitHub Apps
April 27

Organization APIs for fine-grained PATs management
March 24

Probot

https://probot.github.io/


Apps Docs Community

PROBOT
GitHub Apps to automate and improve your workflow

Use pre-built apps to extend GitHub, and easily build and share your own.

Explore

Check out these hosted apps that extend your project on GitHub.
They're all open source and free to use on any project.

Pull
Keep your forks up-to-date.

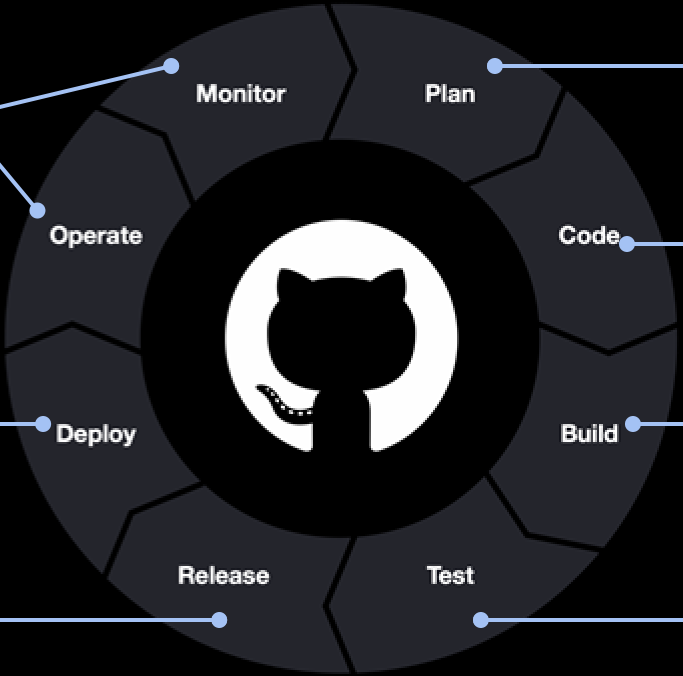
Release Drafter
Drafts your next release notes as pull requests are merged into master.

Semantic Pull Requests
Status check that ensures your pull requests follow the Conventional Commits spec

GitHub Actions

Basics ○ Build your actions

Use cases across your SDLC



Not suitable for

- Deploy application

- Publish release
- Release notes
- Upload artifacts

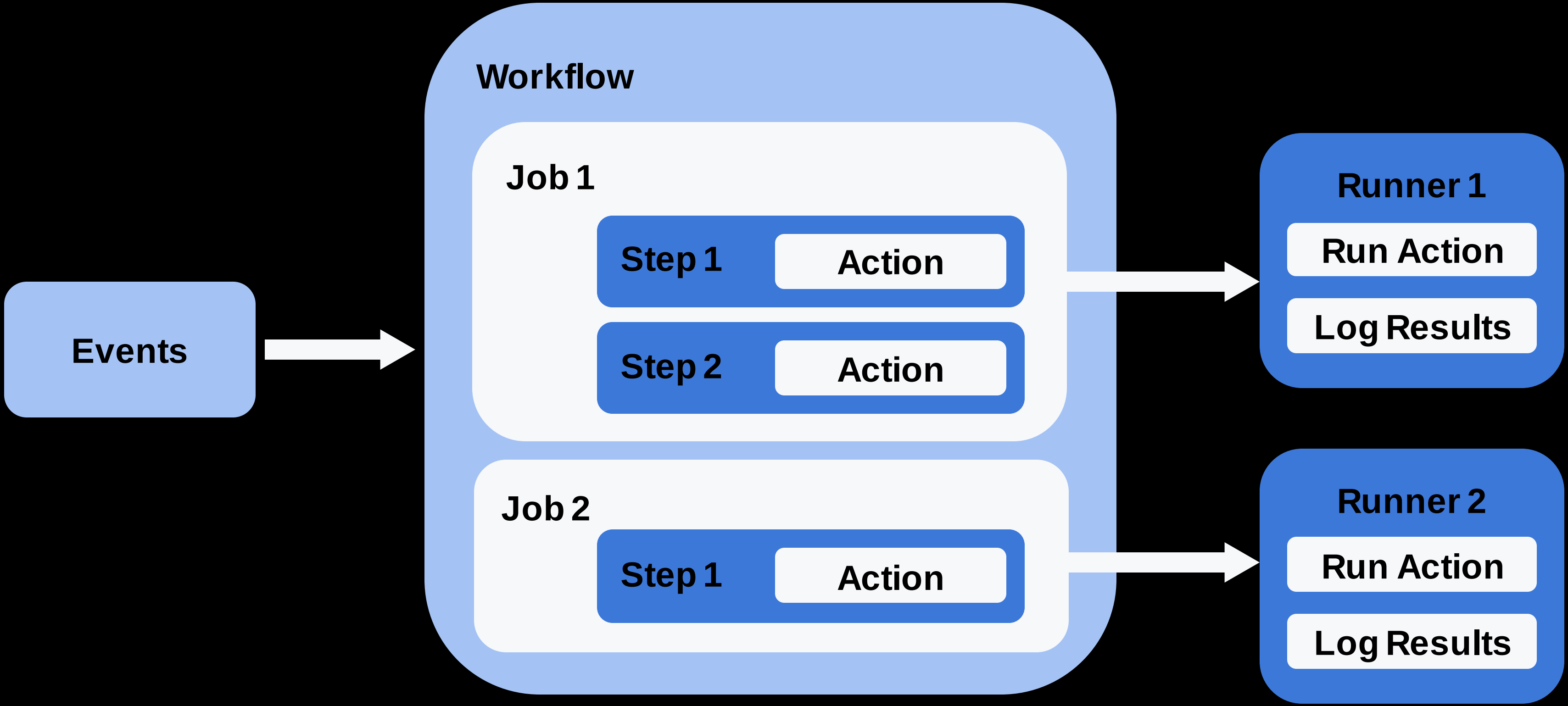
- Triage issues
- Project boards

- Code review
- Auto-merge
- Auto-labeling

- Compile and build
- Store artifacts

- Automated testing
- Linters
- CodeQL
- Consume Packages

Key components



Basic syntax

`./.github/workflows/workflow-file-name.yml`

events →
jobs →
runner →
steps →
actions →
secrets →

```yaml
name: Super Linter workflow

on:
  push:

jobs:
  lint:
    name: Lint Code Base

    runs-on: ubuntu-latest

    steps:

      - uses: actions/checkout@v2

      - uses: github/super-linter@v3
        env:
          GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
```

Events

Webhook events

- Pull request
- Issues
- Push
- Release
- ...

Scheduled events

Manual events

events

```
name: Super Linter workflow

on:
  issues:
    types: [closed, reopened]

jobs:
  lint:
    name: Lint Code Base

    runs-on: ubuntu-latest

    steps:

      - uses: actions/checkout@v2

      - uses: github/super-linter@v3
        env:
          GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
```

Events

Webhook events

- Pull request
- Issues
- Push
- Release
- ...

Scheduled events

Manual events

events

```
name: Super Linter workflow

on:
  schedule:
    - cron: ‘30 6 * * 5’ # every Friday 06:30 UTC

jobs:
  lint:
    name: Lint Code Base

    runs-on: ubuntu-latest

    steps:

      - uses: actions/checkout@v2

      - uses: github/super-linter@v3
        env:
          GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
```

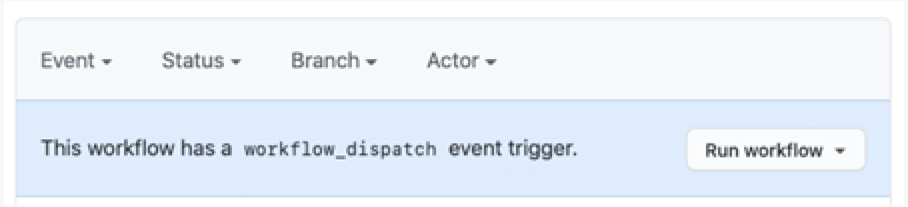
Events

Webhook events
- Pull request
- Issues
- Push
- Release
- ...

Scheduled events

Manual events
- workflow_dispatch
- repository_dispatch

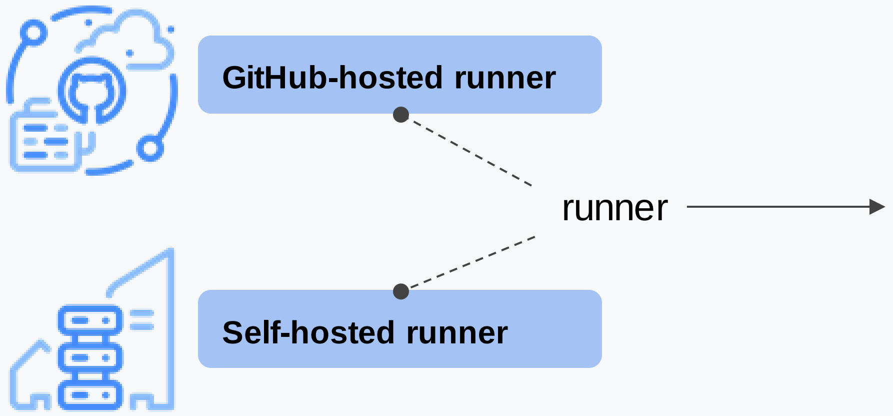
events →


Event ▾ Status ▾ Branch ▾ Actor ▾

This workflow has a `workflow_dispatch` event trigger.

```yaml
name: Super Linter workflow

on:
  workflow_dispatch:

jobs:
  lint:
    name: Lint Code Base

    runs-on: ubuntu-latest

    steps:

      - uses: actions/checkout@v2

      - uses: github/super-linter@v3
        env:
          GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
```

Runners



```yaml
name: Super Linter workflow

on:
  push:

jobs:
  lint:
    name: Lint Code Base

    runs-on: ubuntu-latest

    steps:

      - uses: actions/checkout@v2

      - uses: github/super-linter@v3
        env:
          GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
```

Runners

GitHub-hosted runner

- OS: ubuntu, windows, or macOS
- Ephemeral
- 2-core CPU (macOS: 3-core)
- 7 GB RAM (macOS: 14 GB)
- 14 GB SSD disk space
- Software installed: wget, GH CLI, AWS CLI, Java, …
- Not currently available on

runner →

```yaml
name: Super Linter workflow

on:
  push:

jobs:
  lint:
    name: Lint Code Base

    runs-on: windows-latest

    steps:

      - uses: actions/checkout@v2

      - uses: github/super-linter@v3
        env:
          GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
```

Runners

Self-hosted runner

- Custom hardware config
- Run on OS not supported on GitHub-hosted runner
- Reference runner using custom labels
- Can be grouped together
- Control which organizations/repositories have access to which runners/runner groups
- Do not use with public repositories!

runner →

```yaml
name: Super Linter workflow

on:
  push:

jobs:
  lint:
    name: Lint Code Base

    runs-on: [self-hosted, linux, ARM64]

    steps:

      - uses: actions/checkout@v2

      - uses: github/super-linter@v3
        env:
          GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
```

Actions

Reusable units of code that can be referenced in a workflow

GitHub runs them in Node.js runtime, or in Docker containers

Reference an Action, or run scripts directly

Can be referenced in three ways:

- Public repository
- The same repository as your workflow (local actions)
- A published Docker container image on DockerHub

script →

public actions →

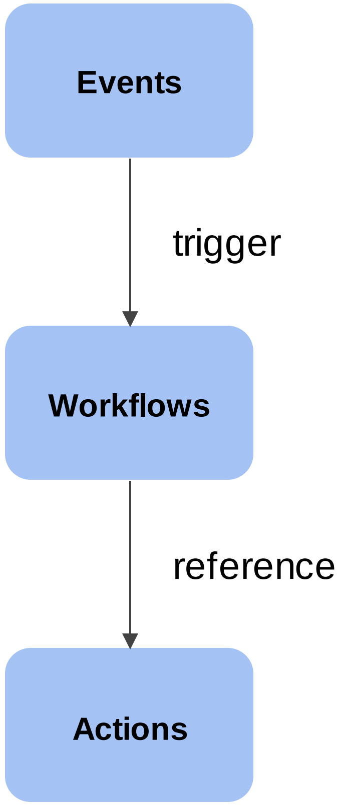
local action →

docker image →

```
name: Super workflow

on:
  push:

jobs:
  lint:
    name: Lint Code Base

    runs-on: ubuntu-latest

    steps:
      - run: echo “Hello World”

      - uses: actions/checkout@v2

      - uses: github/super-linter@v3
        env:
          GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}

      - uses: ./path/to/action

      - uses: docker://alpine:3.8
```

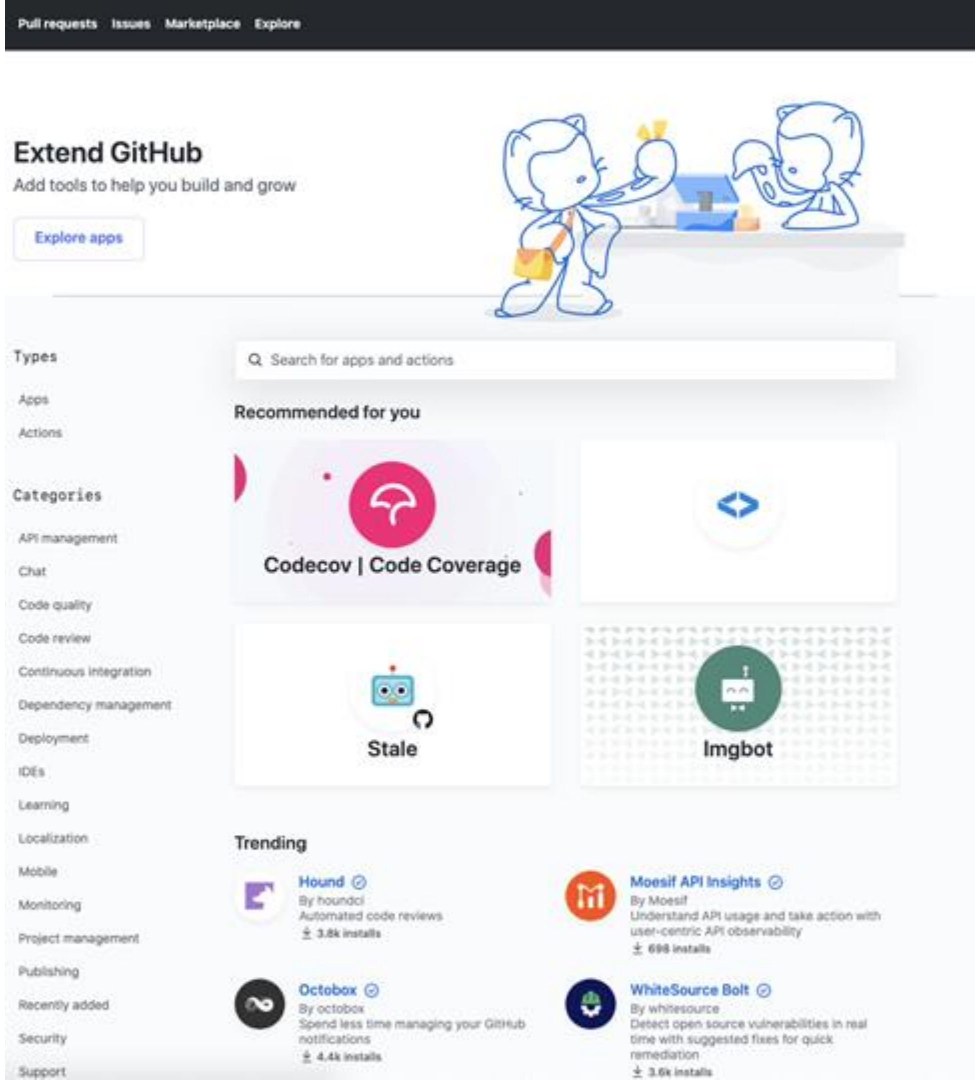
Quick summary

- Events trigger workflows, e.g. a push to a branch
- Workflows contain one or more jobs, which contains one or more steps
- These steps can reference actions or execute commands
- The term “*GitHub Actions*” include all components, not just the Actions themselves



GitHub Marketplace

- Discover open-source Actions across multiple domains
- ~9,000 Actions (and counting...)
- Verified creators ✓
- Reference these Actions directly in your workflow
- Integrated into the GitHub editor



Pull requests Issues Marketplace Explore

Extend GitHub

Add tools to help you build and grow

Explore apps

Types

Apps

Actions

Categories

API management

Chat

Code quality

Code review

Continuous integration

Dependency management

Deployment

IDEs

Learning

Localization

Mobile

Monitoring

Project management

Publishing

Recently added

Security

Support

Search for apps and actions

Recommended for you

Codecov | Code Coverage

Stale

Imgbot

Trending

Hound ✓
By houndci
Automated code reviews
3.8k installs

Moesif API Insights ✓
By Moesif
Understand API usage and take action with user-centric API observability
698 installs

Octobox ✓
By octobox
Spend less time managing your GitHub notifications
4.4k installs

WhiteSource Bolt ✓
By whitesource
Detect open source vulnerabilities in real time with suggested fixes for quick remediation
3.6k installs

GitHub Actions

Basics ○ Build your actions

Writing your own Actions

- 3 types of Actions
  - JavaScript
  - Docker
  - Composite run step
- Metadata defined in `action.yml` file
  - Inputs
  - Outputs
  - Branding
  - Pre-/post-scripts
  - ...

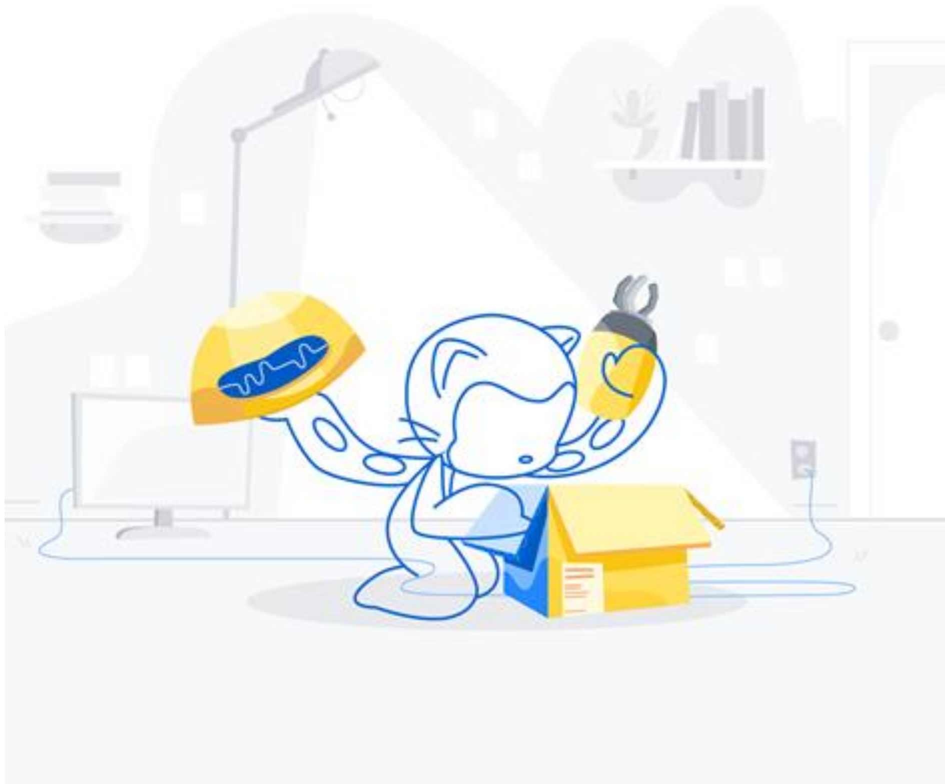
`./path/to/action/action.yml`

```yaml
name: "Hello Action"
description: "Greet someone"
author: "octocat@github.com"

inputs:
  MY_NAME:
    description: "Who to greet"
    required: true
    default: "World"

outputs:
  GREETING:
    description: "Full greeting"

runs:
  using: "docker"
  image: "Dockerfile"

branding:
  icon: "mic"
  color: "purple"
```

Writing your own Actions

Best Practices

- Design for reusability
- Write tests
- Versioning
- Documentation
- Proper `action.yml` metadata
- github.com/actions/toolkit
- Publish your Action to the Marketplace 🎉



Demo: build your own action

Q&A

Thank you