GitHub

Introduction to GitHub Apps

GitHub Partner Engineering

Introducing GitHub Apps 🤖

GitHub Apps are a tool to build comprehensive integrations with GitHub:

- First class actors on GitHub -- operating independently of any user identity
- Offer fine-grained permissions
- Installed on a user's or organization's repos
- Replace and offer many advantages over OAuth apps
- Come with built-in webhooks
- Work on GitHub.com and GitHub Enterprise Server
- Compatible with web technologies and standards, such as HTTP-based APIs and OAuth-like flows
- Rich open source tooling and libraries available, eg. octokit

Advantages for customer

- **Confidence** in granting third parties access to their assets in GitHub due to fine-grained and repo-centric permissions model
- **Convenience** through user-friendly (un)installation flow
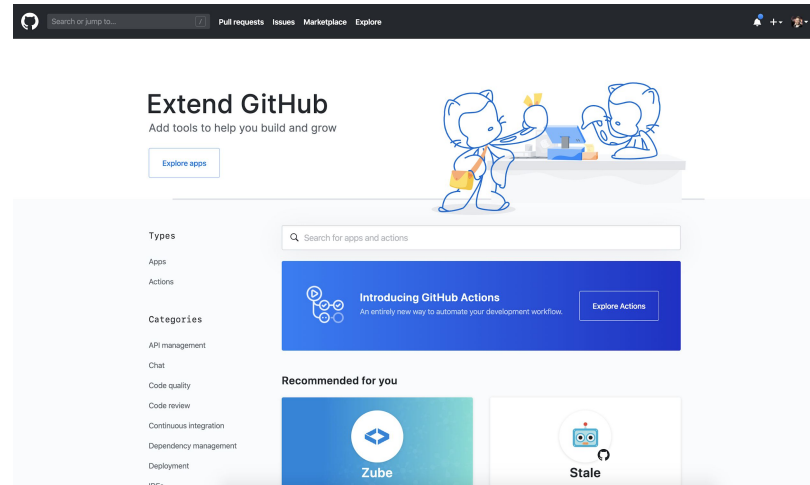
Advantages for integrator

- Can **decouple** integration from GitHub user identities due to first class actor model of GitHub Apps.
- Can take advantage of **dedicated, scalable [rate limits](#)**, as opposed to the shared rate limit model offered by OAuth apps.
- Can utilize **modern GitHub APIs** like [Checks](#) and [Content Attachments](#)

🧪 Hands on with GitHub Apps

1. Creating a GitHub App
2. Authenticating as that app
3. Installing the app on repositories
4. Authentication as that installation
5. Receiving a webhook
6. Creating content as an installation

Authentication overview

| Authentication Scheme	Also Known As	Description	How to Get It	Available Endpoints	Examples
JSON Web Token	JWT (pronounced "jot")	Authenticates as the GitHub App	GitHub docs, Octokit	List	Fetching application installation details or exchanging the **JWT** for an **installation access token**.
Installation access token	Server-to-server requests	Authenticates as a specific installation of the GitHub App	GitHub docs, Octokit	List	Opening an issue or providing feedback on a pull request
OAuth access token	User-to-server requests	Authenticates as a user of the GitHub App	GitHub docs	List	Authenticating as a user when a GitHub App needs to verify a user's identity or act on a user's behalf

Authentication at a glance

Deciding which authentication type to use comes down to:

- What resource do I need to access?
- Who do I need to access it as?
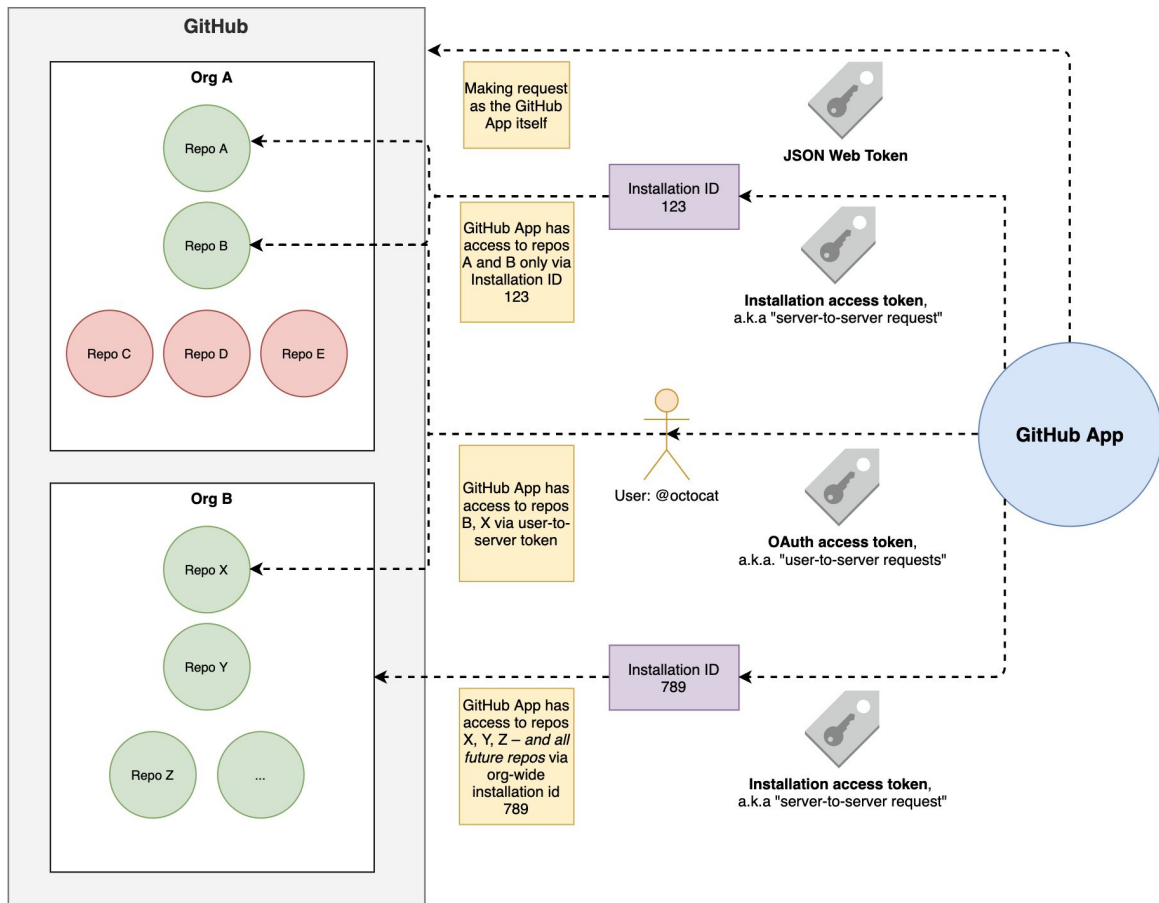
Onboarding users

Users can be directed through various flows to create/update installations and identify themselves to your GitHub App.
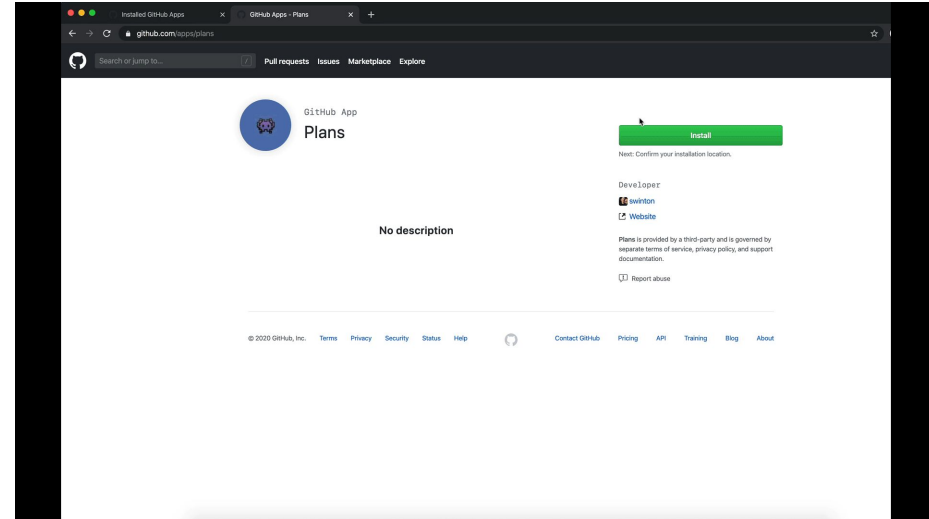
Notable APIs for GitHub Apps

- GitHub App information
    - Get the authenticated GitHub App (JWT)
- Identify installation information
    - List installations (JWT)
    - Get an organization installation (JWT)
    - Get a user installation (JWT)
- Token creation / revocation
    - Create a new installation token (JWT)
    - Revoke an installation token (installation access token)
- Identify installation resources
    - List repositories (installation access token)
- Identify user-accessible resources
    - List installations for a user (user-to-server OAuth access token)
    - List repositories accessible to the user for an installation (user-to-server OAuth access token)

GitHub Apps best practices for integrators

✅ **Do:**

- Cache and re-use installation tokens
- Use [webhooks](#) for real-time data
- Throttle requests to stay within rate limits
- Consider if REST or GraphQL APIs (or both) are best for your use case
- Use [conditional requests](#) wherever possible
- Subscribe to this [RSS feed](#) for Platform updates
- Include a descriptive [User-Agent header](#)
- Save the X-GitHub-Request-Id response header value, especially for error responses
- Consider other best practices listed [here](#)

❌ **Don't:**

- Depend on concurrent requests, this can trigger [secondary rate limits](#)
- Poll, use webhooks where possible

Libraries and resources

- [Developer Documentation](#)
- GitHub [REST](#) and [GraphQL](#) APIs
- [API route specifications](#) for Insomnia
- [Migrating OAuth Apps to GitHub Apps](#)
- [GitHub Apps](#)
- [GitHub Webhooks](#)
- [Octokit](#)
- [Probot](#)
- [smee.io](#) to test webhooks
- [GitHub Marketplace](#)