GitHub InfoMagnus

GitHub Enterprise Implementation

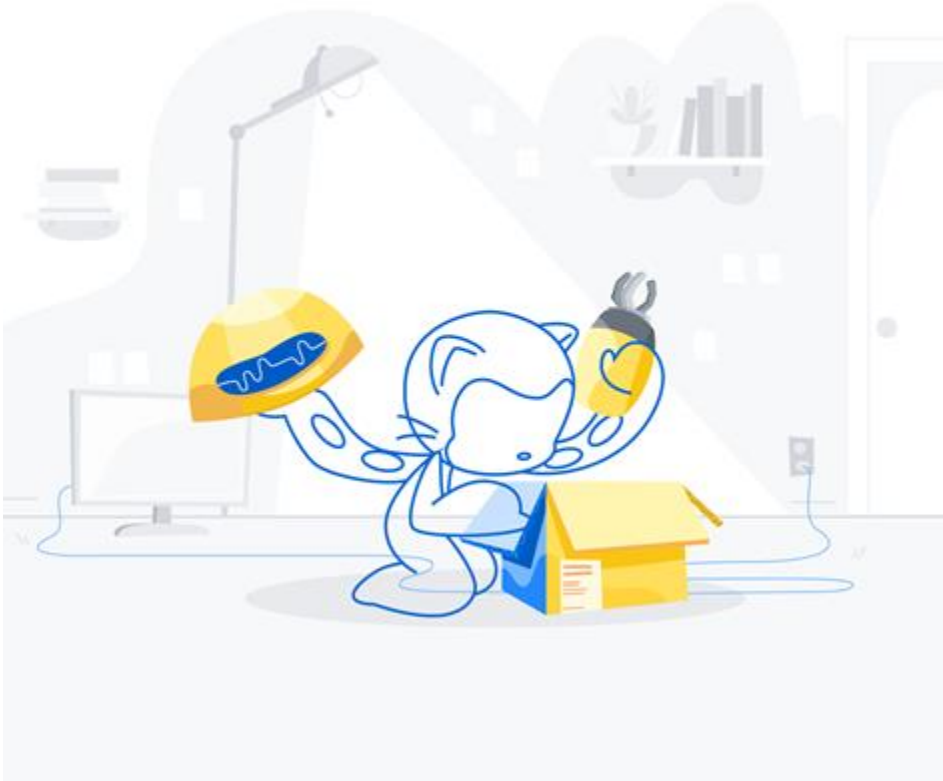
Presented by GitHub Professional Services

Objectives

- GHE platform overview
- Permission flow, visibility, base permissions
- GHE enterprise account, organization, repository
- Governance and Policies
- Q&A
- Recommended Practices for Organizations

Agenda

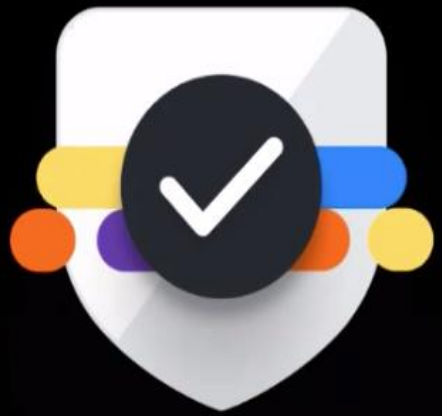
- GHE Platform Overview
- GHE Admin
  - Enterprise Account
  - Organization
  - Repository
- API and Integrations
- Custom Questions



GitHub Enterprise

single, integrated platform that empowers developers

Enterprise



Security

Developer
Experience

GitHub Enterprise

Fully integrated DevSecOps Platform

Collaboration & innersourcing


Granular permissions & custom roles


Open Source Access


Project planning


Private packaging & container registry

Automation & development tools


Actions


Codespaces


Config-as-code


GitHub copilot

Secure by design

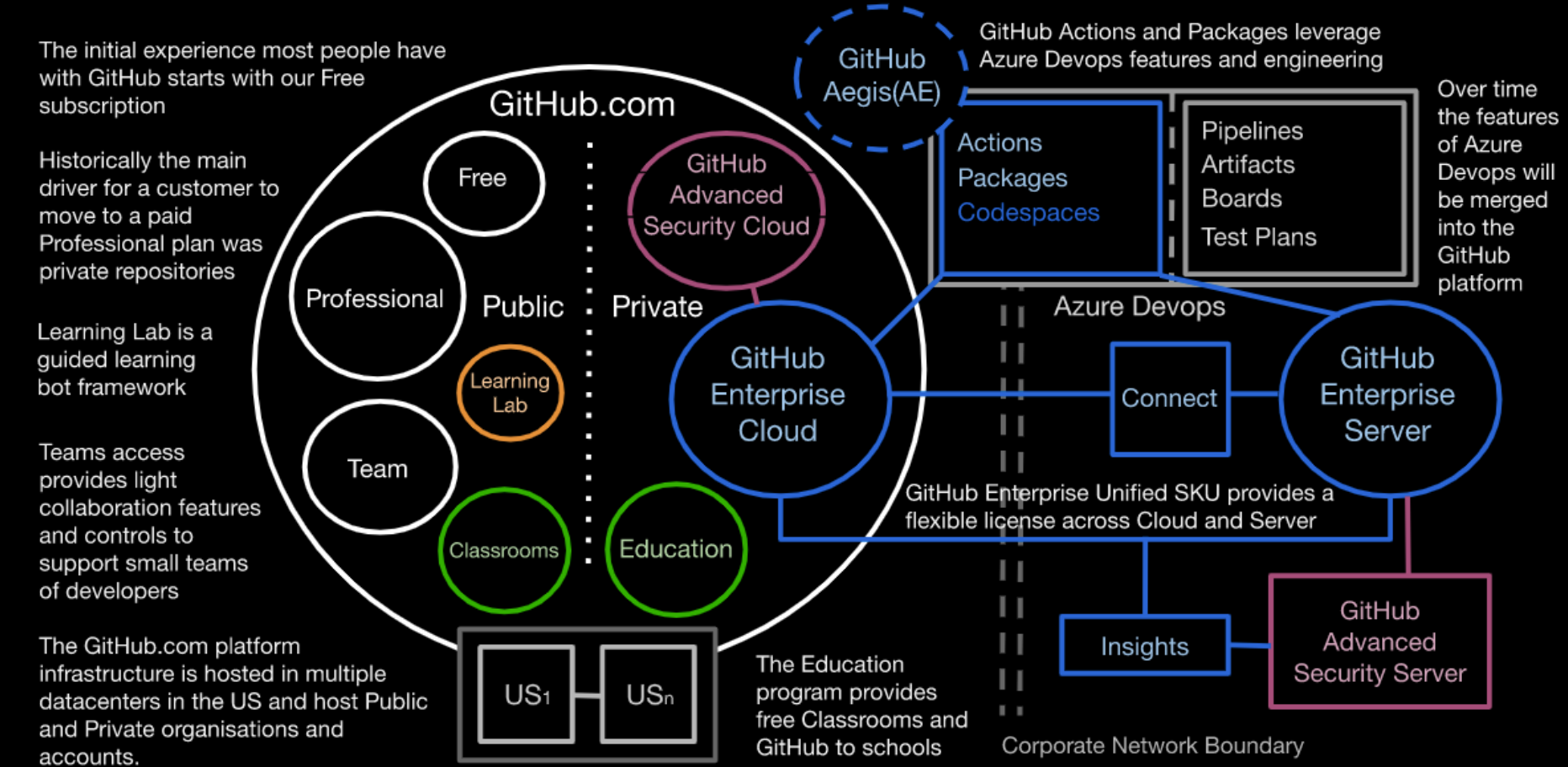

SAML SSO & SCIM


Code scanning & variant analysis


Secret scanning


Security overview

GitHub Solution Overview



GitHub Enterprise Platform Overview

Platform • Permissions Flow • GitHub.Com | GHES | GHEC

GitHub Enterprise Cloud

- GitHub Enterprise Cloud
- Software as a Service offering
- **Security** and policy features synonymous to GitHub Enterprise
- Fast onboarding of new **collaborators**
- Reduced operations overhead
- **Public repositories** on GitHub.com are viewable by anyone on the internet
- Repositories that are **private** on GitHub.com are not accessible to everyone
- Privacy is configured by **enterprise**, organization, team, or individual level

Billing and GitHub Plans

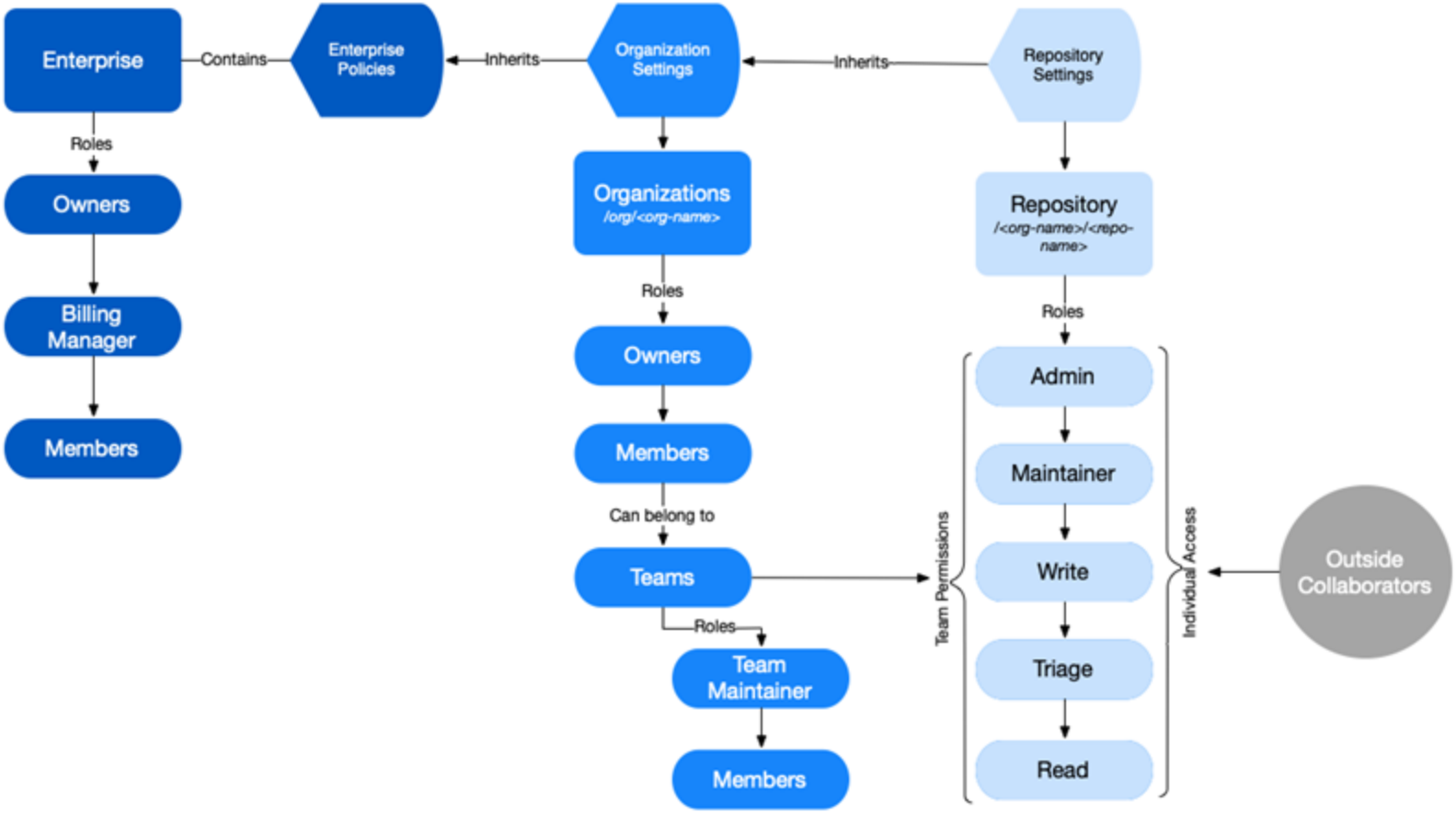
GitHub Plans
GitHub Enterprise (license seats)

Enterprise Add-ons
GitHub Advanced Security
GitHub CodeSpaces
GitHub CoPilot for Business

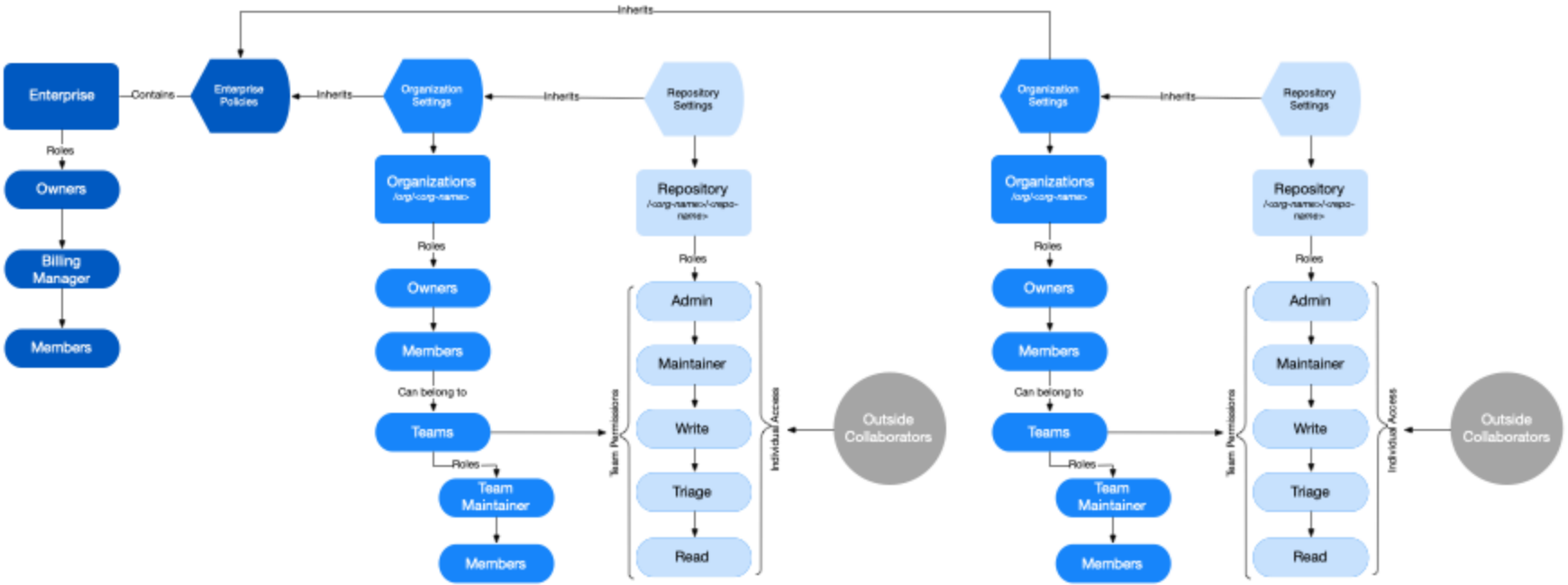
Pay per use
GitHub Actions Cloud Runners (minutes, storage and data transfer)
GitHub Packages (storage and data transfer)
Codespaces
Marketplace pay-per-use apps
Git LFS
Sponsoring projects

- Storage: 50 GB (artifacts, logs, packages)
- Runner-minutes: 50,000 min/month (GitHub-Hosted Runners)
- Self-Hosted Runners: Free (+ your own infrastructure)

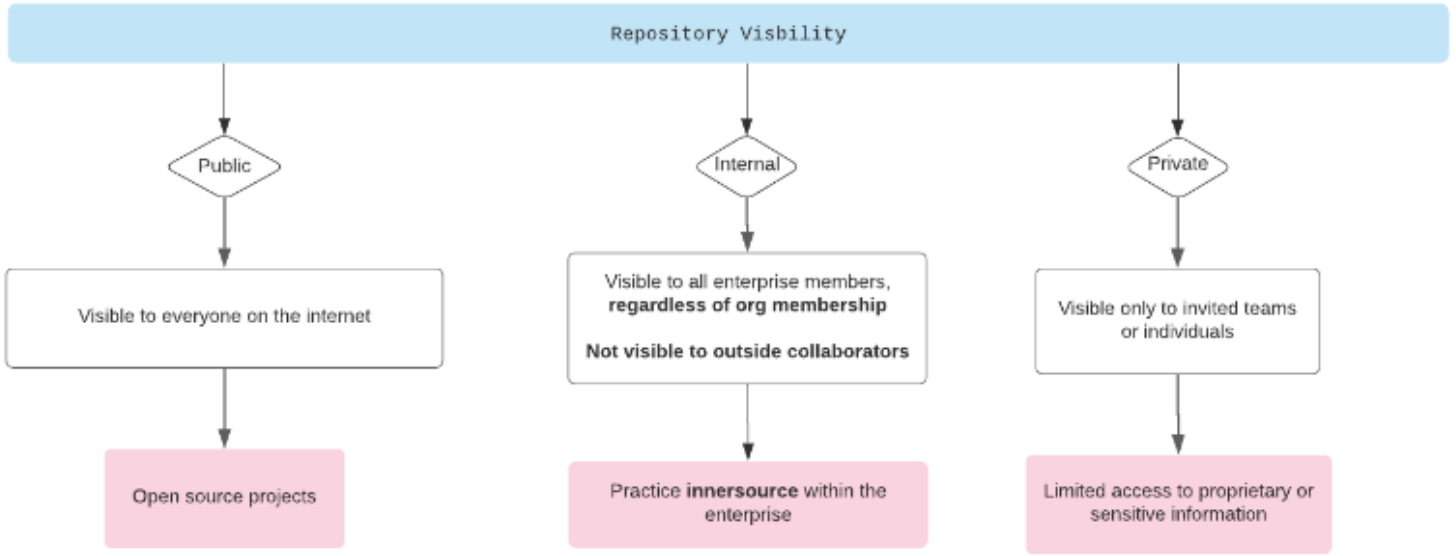
Flow of permissions



Flow of permissions - multiple orgs



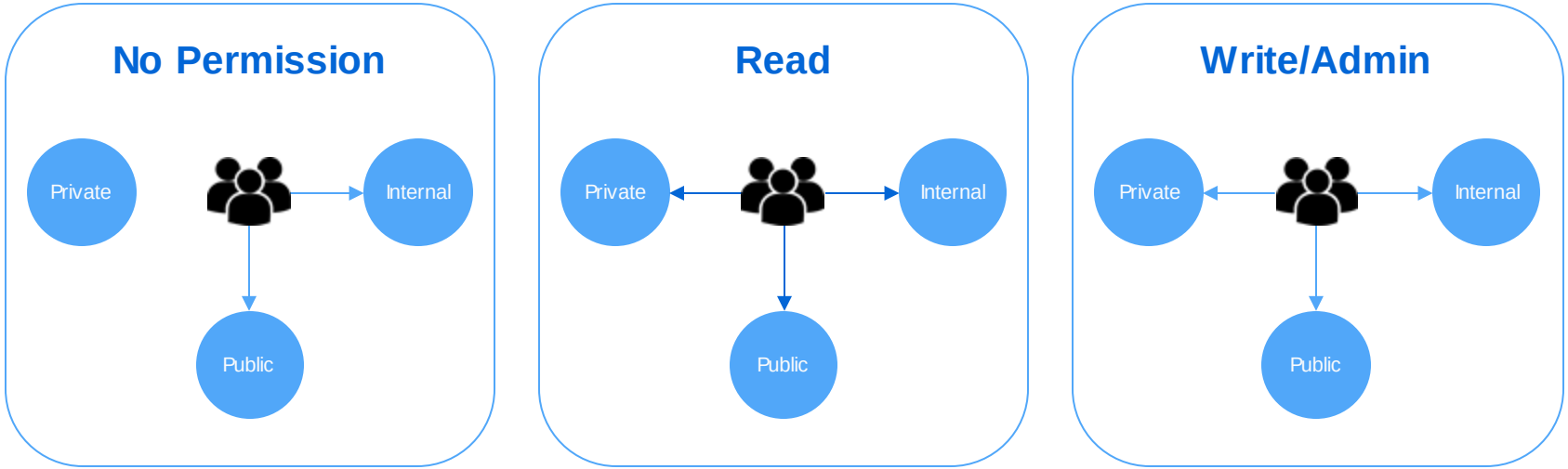
Repository Visibility



Legend

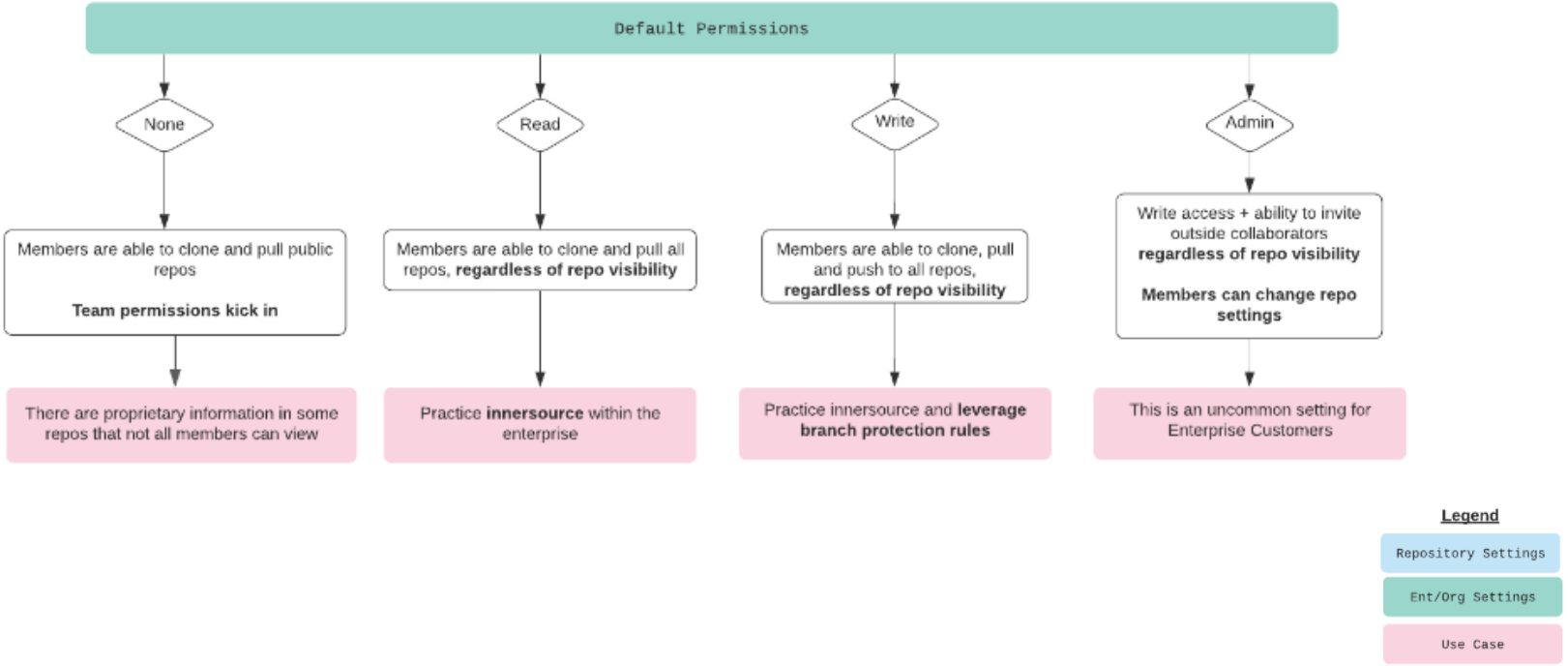
- Repository Settings
- Ent/Org Settings
- Use Case

Repository base permissions



***Users being added to an organization with **NO** other special access other than being added as a MEMBER to the organization.*

Default Permissions



Permission roles

Permission	Description
`Read`	Read-only access to Code and Actions. Can submit and comment on issues, pull requests, and discussions
`Triage`	Read-only permissions with the additional ability to manage issues, pull requests, discussions, assignments, and labels
`Write`	Gives write access to all parts of a repository project with the exception of the repository settings
`Maintain`	Ability to modify some settings of a repository including topics, enabling repository features, configuring merges and GitHub pages, pushing to protected branches
`Admin`	Has full administrative access to all features, settings and configurations of the repository project

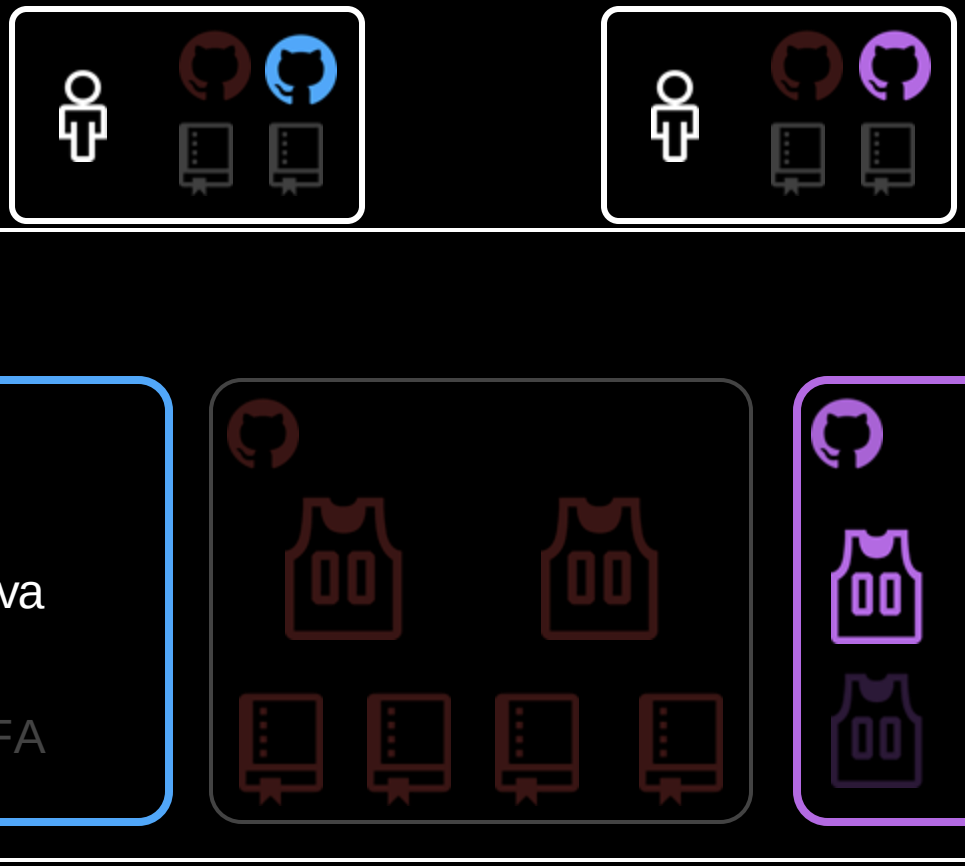
Governance and Policies

Policies

- Repositories
- Actions
- Projects
- Teams
- Organization
- Advanced Security
- Compliance

Organization

Organizations



Reasons to use teams

Collaboration

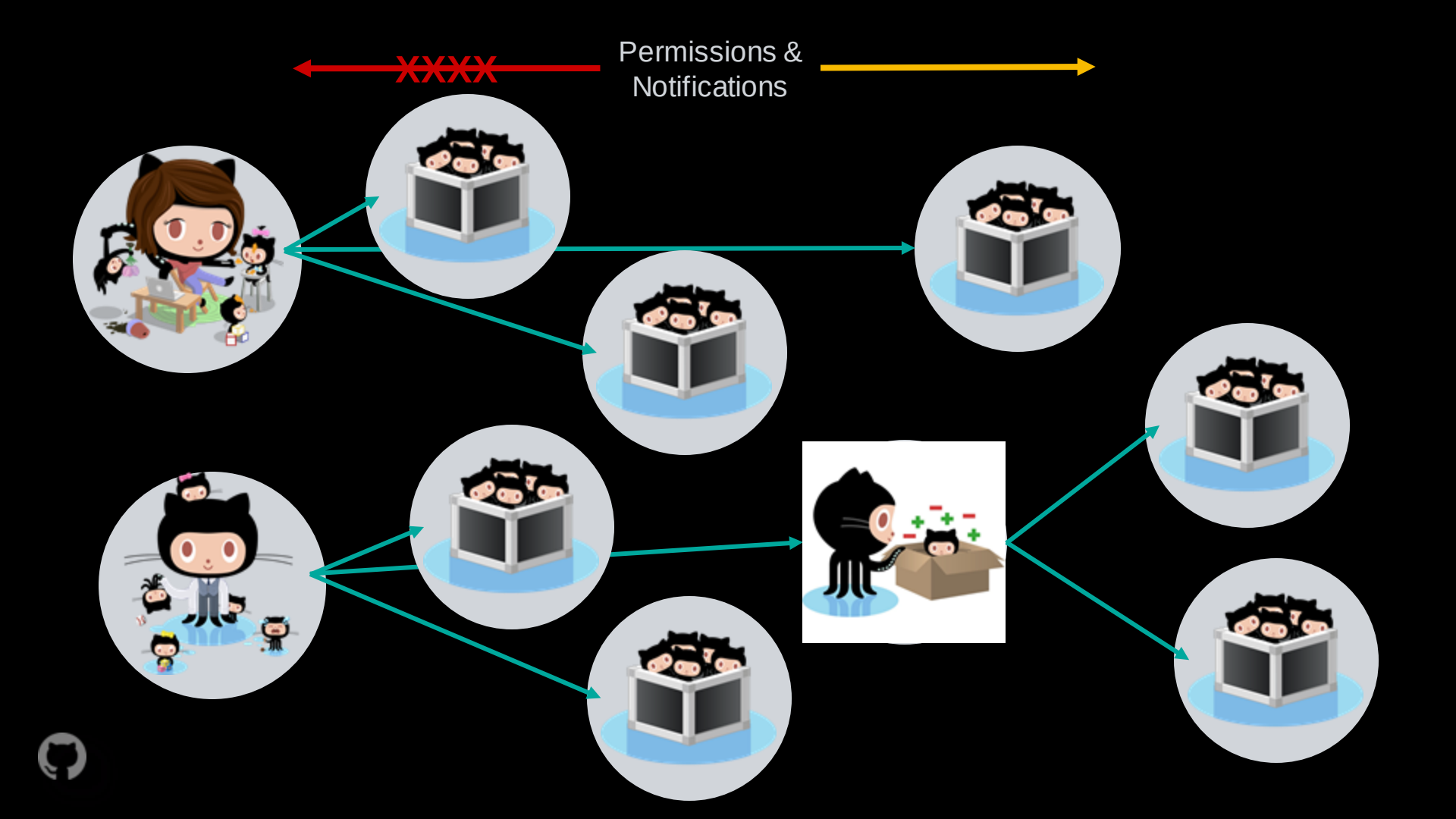
Innersource

Onboarding and offboarding

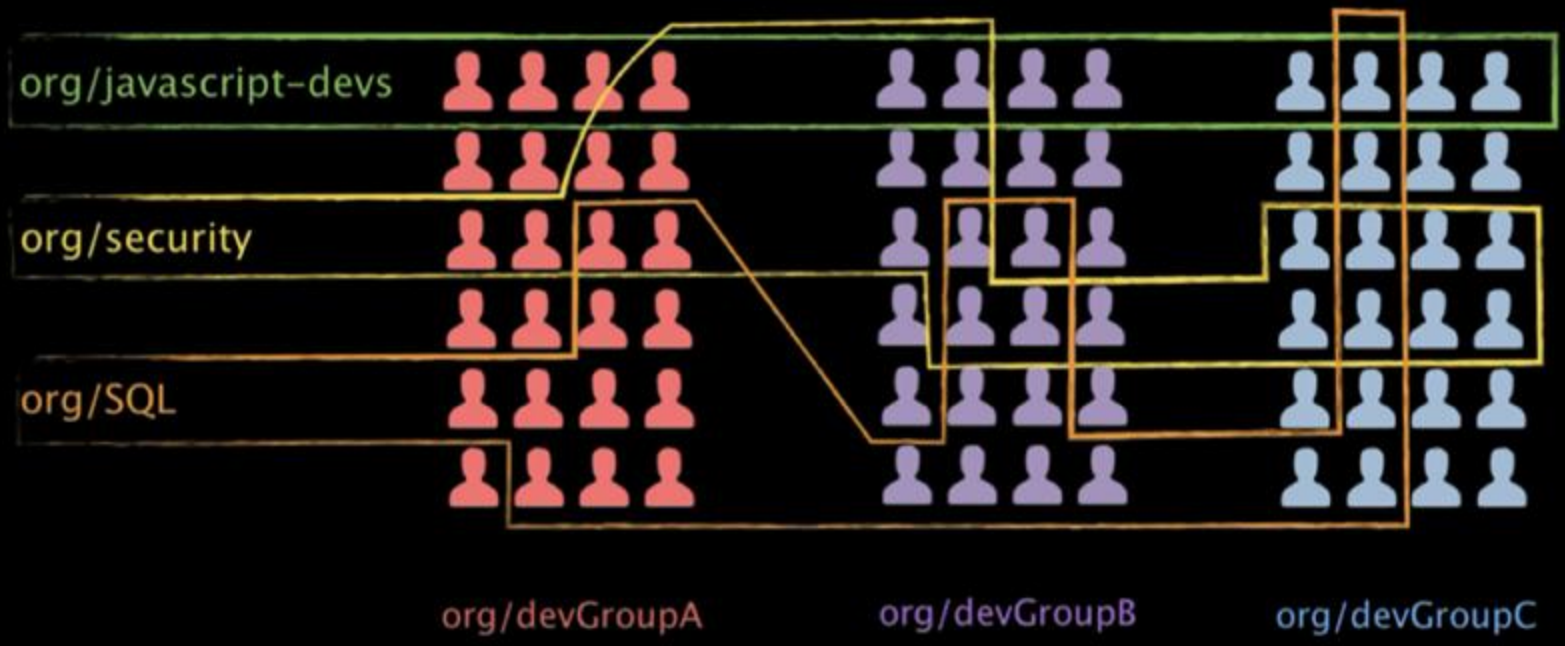
Security

Managing teams

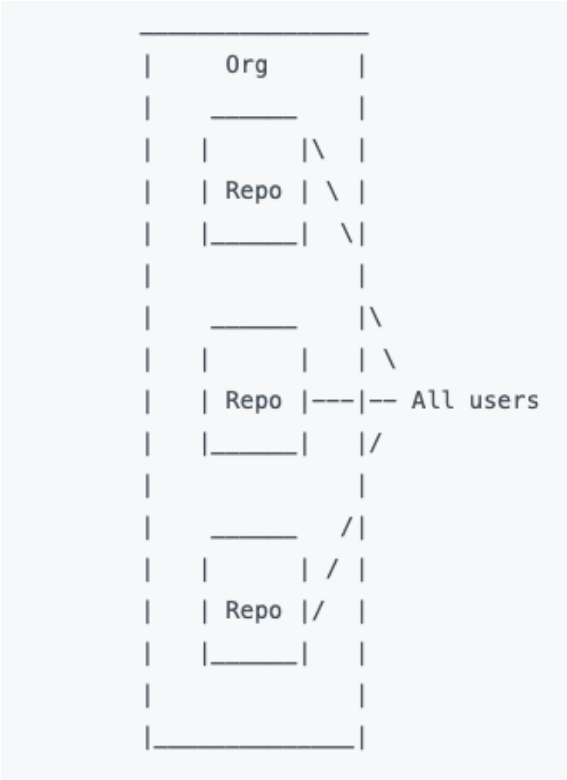
- Nested teams
  - Parents team can have more than one child
  - Child teams inherit parent's permissions
  - Children receive parent's notifications
  - Users in a child team belong also to the parent team


XXXX
Permissions &
Notifications

Team best practices



A single organization with direct organization membership for repository access (not teams)



Scenarios where this makes sense:

- Works best in small companies
  - where everyone works on everything (so they all need the same access to all repositories)
  - startup model
  - high trust
- Can work in medium sized companies as well
  - where trust is high

Scenarios where this doesn't make sense:

- Companies with siloed business divisions
- Companies with strict security constraints

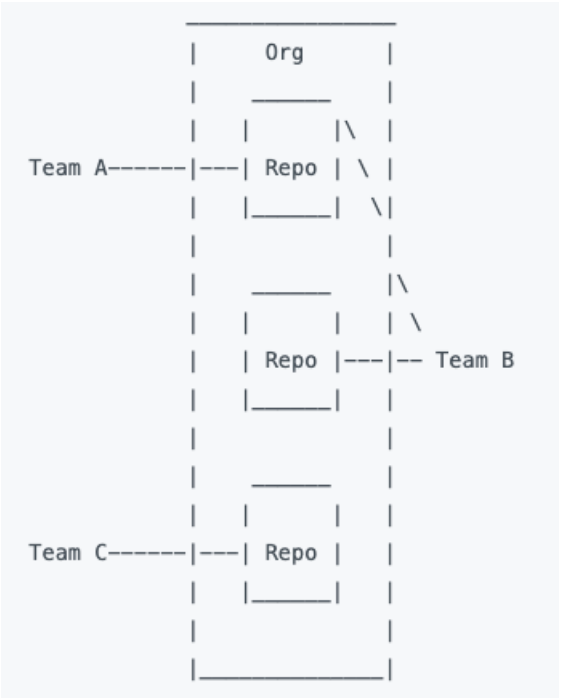
Benefit:

- All members and teams can be @mentioned to get their input on all projects
- Facilitates InnerSource projects

Drawback:

- Loose organization

A single organization with multiple teams to manage repository access



Scenarios where this makes sense:

- Works in small companies with strict development roles, or lower trust
- Medium sized companies

Scenarios where this doesn't make sense:

- Small startups: administrating teams may take more time that it is worth in a 10 or 20 seat company where everyone works on, or at least is aware of, everything
- Large companies where no single owner should have access to all repositories

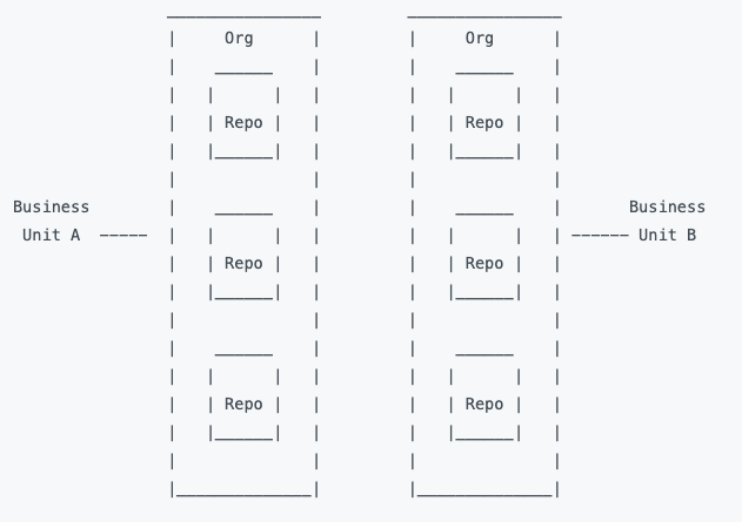
Benefit:

- All teams are able to @mention each other since they are in the same organization
- Facilitates InnerSource projects and collaboration across business units

Drawback:

- Can become unwieldy at a company with many teams at a large scale

Multiple organizations divided per business unit with direct organization membership



Scenarios where this makes sense:

- Works well when there are many repositories per business unit
- Works very well if there are strict access policies and for larger companies
- Work in companies that are large enough to have groups that don't need to work together
- Works well in companies with requirements for segregation of duties and access restrictions
- Large companies where no single owner should have access to all repositories

Scenarios where this may not make sense:

- If contribution across business units is important
- If there is a need to @mention teams from different organizations

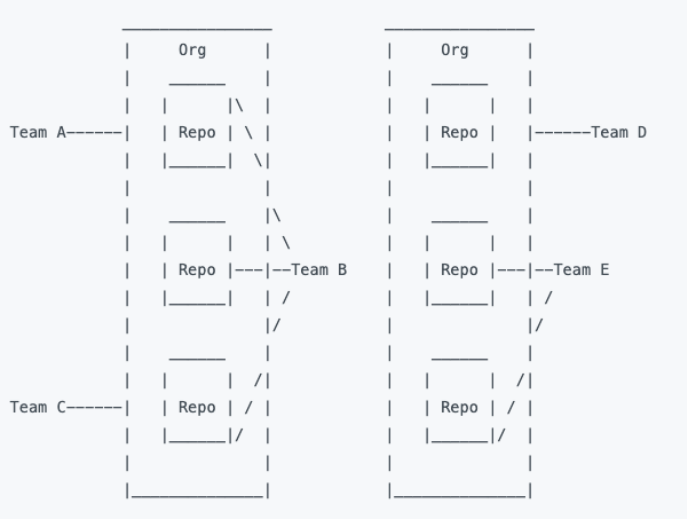
Benefit:

- Manage access without team administration

Drawback:

- May hinder collaboration or happenstance of developers discovering projects they are interested in
- Reduces searchability of resources within the company

Multiple organizations with multiple teams to manage repository access



Scenarios where this makes sense:

- Works well there are many repositories per business unit
- Works very well if there are strict access policies and for larger companies
- Work in companies that are large enough to have groups that don't need to work together
- Works well in companies with requirements for segregation of duties and access restrictions
- Large companies where no single owner should have access to all repositories

Scenarios where this may not make sense:

- If contribution across business units is important
- If there is a need to @mention teams from a different organization

Benefit:

- High level of separation and access control required in large companies

Drawback:

- May hinder collaboration or happenstance of developers discovering projects they are interested in
- Reduces searchability of resources within the company

So You Want New Organization?

You should consider a new organization if:

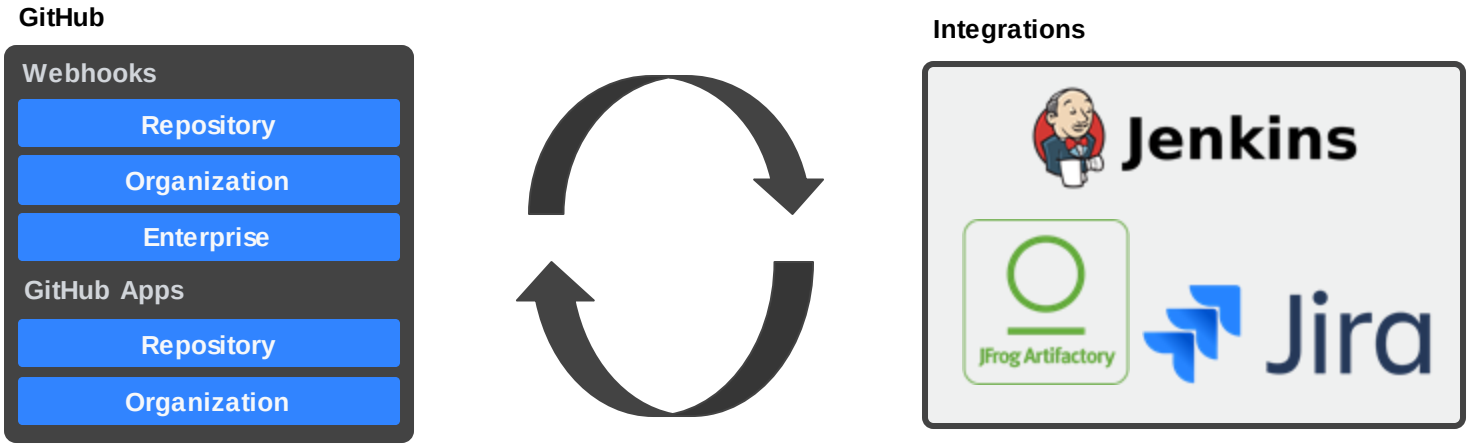
- The settings you have set on the current organization doesn't fit with the rules of the new organization. For example, having fork enabled on an OSS organization may be important while you want to have it disabled on the internal project's organization. Same would happen with default visibility
- If you want to separate billing for parts of the company that operate independent
- If collaboration between multiple orgs is irrelevant or doesn't need to happen
- For extremely sensitive projects in highly regulated environments where only a subset of the employees can have access, only if base permission is higher than None. Otherwise using the private visibility for this scenario is enough.
- When a company is divided in sub-companies that don't want to collaborate between each other. However, in this scenario it should be considered if this is a good practice and if it's intentional. We have seen scenarios where holding companies want to remove duplications by creating InnerSource practices.
- As a general rule of thumb, creating new organizations for different departments or projects is not a good practice and will make scaling GitHub adoption harder as we need to maintain all the settings separately. The work for administrators increases exponentially as new organizations get created, so always think twice before adding a new one to the Enterprise.

API and Integrations

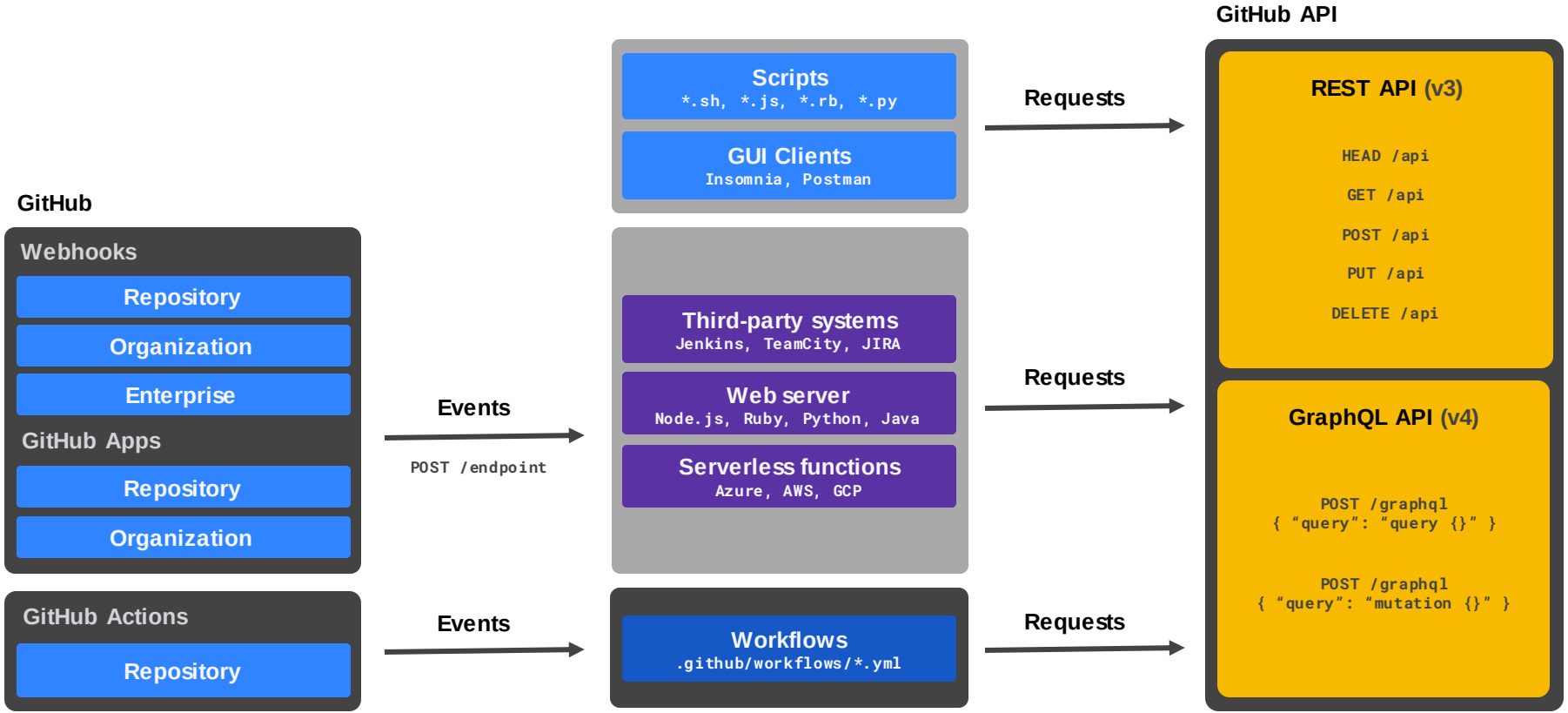
API Overview

Integration Loop

Core integration loop



Core loop overview



Additional topics

API Overview

Authentication methods

Actions overview

Marketplace overview

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 top 15k requests (enterprise)
- Permission changes require approval

Authentication methods

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

Installation (S2S)

App ID + Private key .pem + expiration = JWT

JWT + installation id = API Token

OAuth (U2S)

Client Id + callback url = auth request + code

code + client secret (stored in server) = API Token

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
- 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
- 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

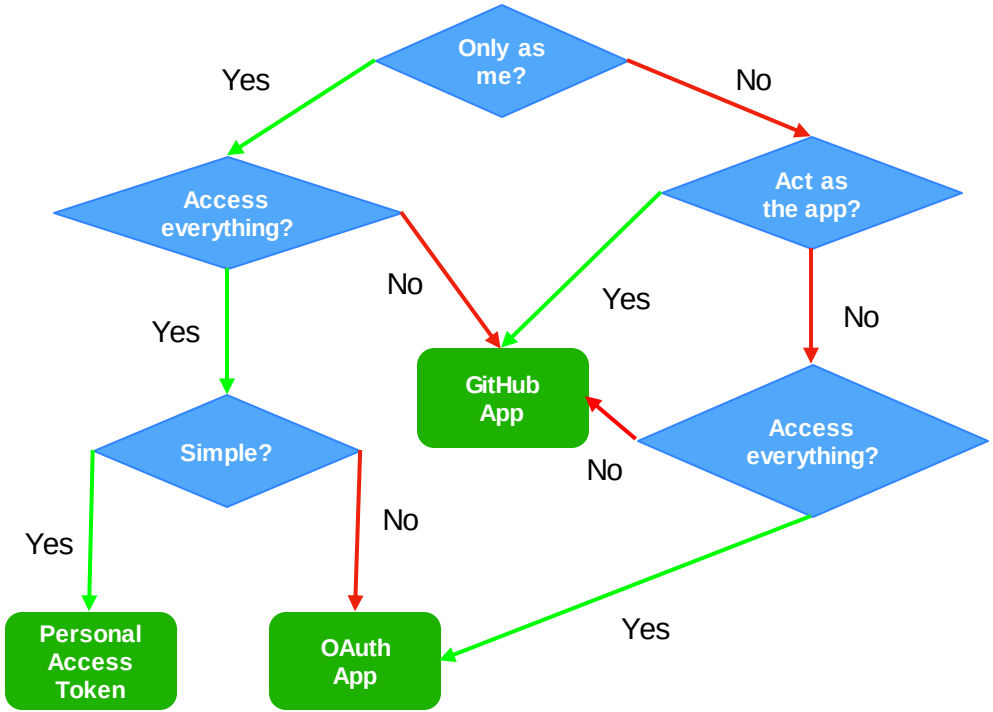
- Anyone with access to the repository and server can deploy the project
- Users don’t have to change their local SSH settings
- Deploy keys are read-only by default
- Deploy keys only grant access to a single repository
- Deploy keys are usually not protected by a passphrase

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

Which should you choose?



Additional topics

API Overview

Authentication methods

Actions overview

Marketplace overview

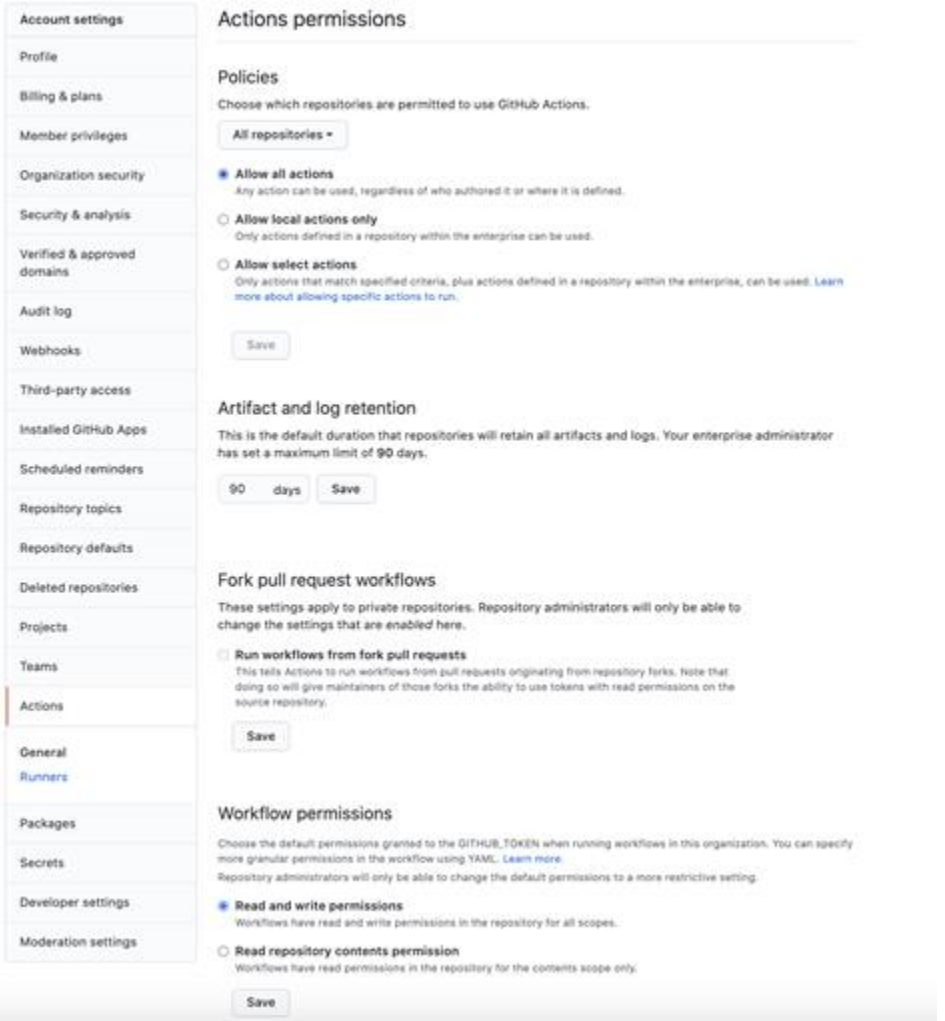
What is GitHub Actions

GitHub Actions is a GitHub product that allows you to **automate your workflows.**

- Workflows stored as `yml` files
- Fully integrated with GitHub
- Respond to GitHub events
- Live logs and visualized workflow execution
- Community-powered workflows
- GitHub-hosted or self-hosted runners
- Built-in secret store

Actions policies

- Configure Actions policies on enterprise / organization / repository level
  - Which Actions are allowed
  - Artifact retention period
  - Running workflows from fork PRs
  - Permissions of `GITHUB_TOKEN`



Sharing workflows in an organization

- Use GitHub **actions starter templates** from `.github` repository to share workflows
- Use GitHub packages and `ghcr.io` to share actions using docker execution and **package registry** permissions (only for public registries)
- Git Submodules or subtrees (not most recommended option)
- **(Upcoming)** Organization workflow execution. Open source concept: https://github.com/SvanBoxel/organization-workflows

Sharing private actions

Use a **GitHub App** to clone actions from:

- Actions in different repositories
- Actions monorepo
- Actions separate organization

```yaml
jobs:
  do-something:
    runs-on: ubuntu-latest

    steps:
      - name: Generate app installation token
        id: app
        uses: peter-murray/workflow-application-token-action@v1
        with:
          application_id: ${{ secrets.APP_ID }}
          application_private_key: ${{ secrets.PRIV_KEY }}

      - name: Checkout private repository
        id: checkout_repo
        uses: actions/checkout@v2
        with:
          repository: my-org/repo
          path: path/to/privateAction
          token: ${{ steps.app.outputs.token }}
```

Best practices on Actions in an organization

- Use the `GITHUB_TOKEN` when possible, as a second option GitHub Apps
- **Limit token permissions.** Set organization workflow permissions to read only
- Run only **trusted actions** and provide actions only the secrets that are needed. Pin untrusted actions
- Protect your secrets with **environments**
- Create **starter workflows** for reusability
- Always create meaningful `README` files for your custom actions
- Small and focused actions. Reuse them from the **marketplace**
- Use actions for CI/CD but also ***-ops**

Additional topics

API Overview

Authentication methods

Actions overview

Marketplace overview

https://github.com/marketplace

Why GitHub? Team Enterprise Explore Marketplace Pricing Search GitHub / Sign in Sign up

Extend GitHub

Find tools to improve your workflow

Explore free apps

Types

Apps

Actions

Search for apps and actions

Sort: Best Match

Apps

CircleCI
By circleci
Automatically build, test, and deploy your

CodeFactor
By codefactor-io
Automated code review for GitHub

Q&A

Thank you