Security Training with GHAS

Instructor: Andrew Scoppa
email: Andrew.Scoppa@atmosera.com
GH handle: Andrew-Scoppa

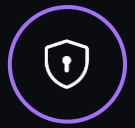
Agenda

Setting the scene

Security features and how they fit into a secure development workflow.

Configuring access

Creating teams and applying appropriate permissions.



Reviewing and analyze alerts

Use the integrated reporting facilities to identify common issues and understand risk factors.

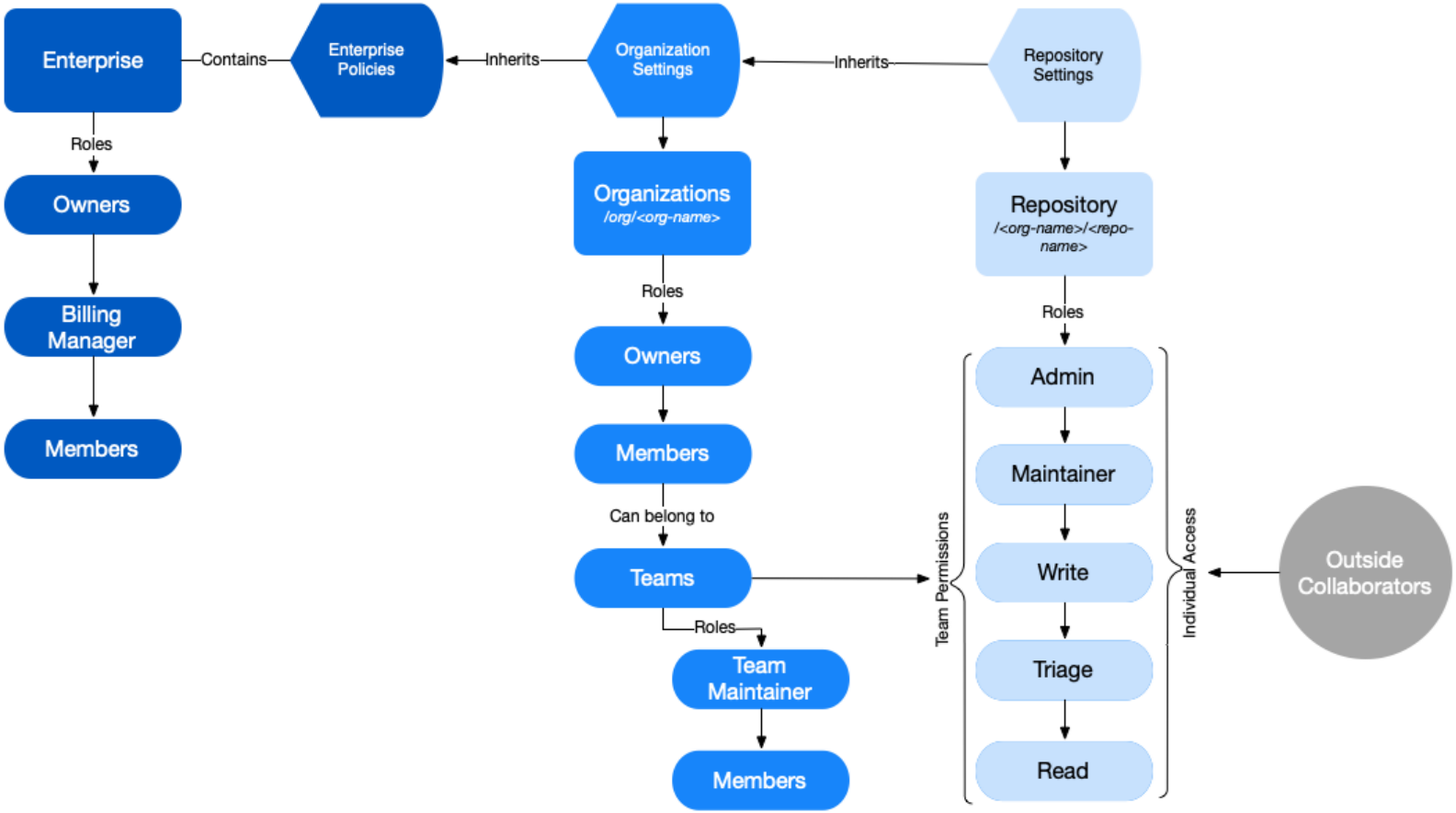
Securing your supply chain

Understanding vulnerabilities in dependencies and patching them.

Resources and Examples

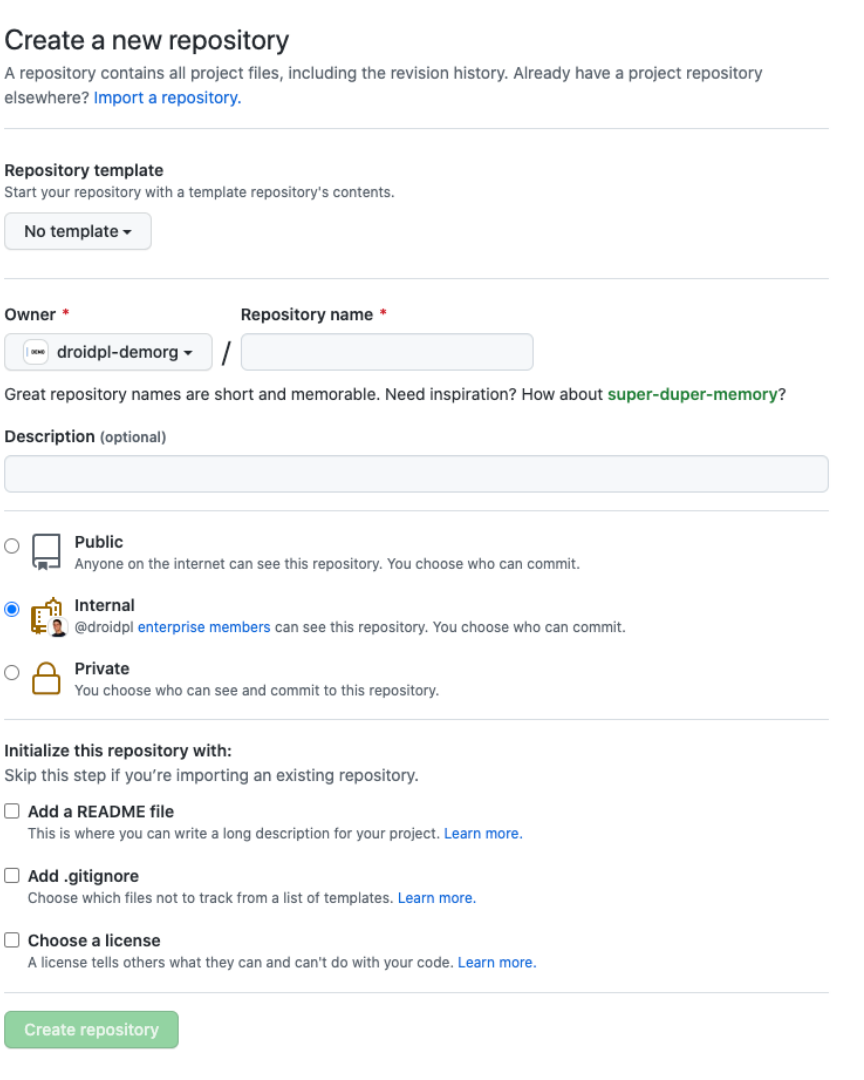
- Code security documentation
  Build security into your GitHub workflow.
- REST API endpoints for code scanning
  Use the REST API to retrieve and update code scanning alerts from a repository.
- Static Analysis Integrations
  A command line application which integrates static analysis tools into the development cycle
- Removing sensitive data from a repository
  Remove unwanted files from a repository's history
- Advanced Security Material
  A place for resources to help you understand and use GitHub Advanced Security (GHAS)
- Policy as Code
  Example application which uses the GHAS APIs to create policy engine using GitHub Actions.
- GHAS JIRA Integration
  A project showing how to integrate GitHub Advanced Security with JIRA.
- CodeQL Queries
  CodeQL queries are used to analyze code for issues related to security, correctness, maintainability, and readability.

Flow of permissions

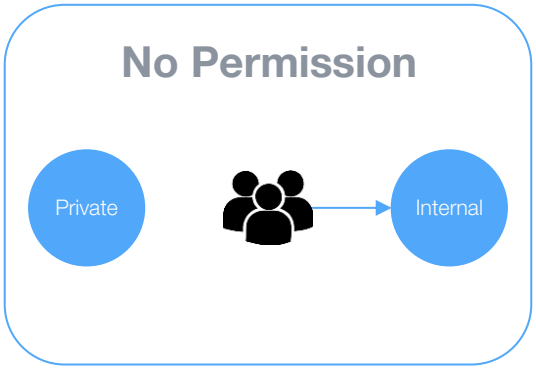


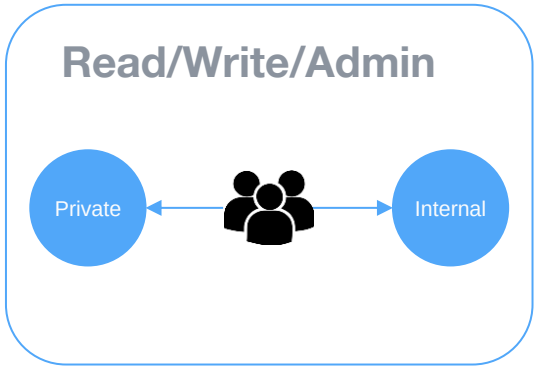
Repository visibility

- **Public** - Anyone on the internet can access (GHEC only)
- **Internal** - Organization members in the enterprise can access
- **Private** - Only people with explicit access



Repository visibility + base permission relationship





Typical enterprise scenario of **No Permission** with internal or private repositories

Roles

Role	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**

GitHub Teams

Collaboration

Innersource

Onboarding and offboarding

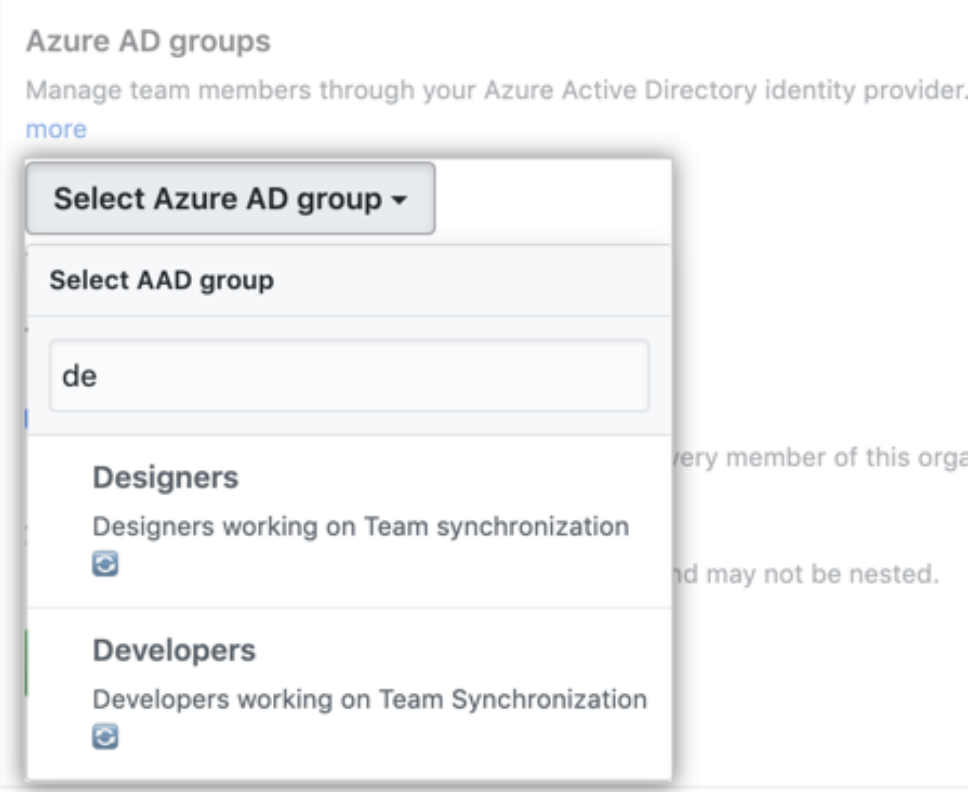
Security

Nested GitHub Teams

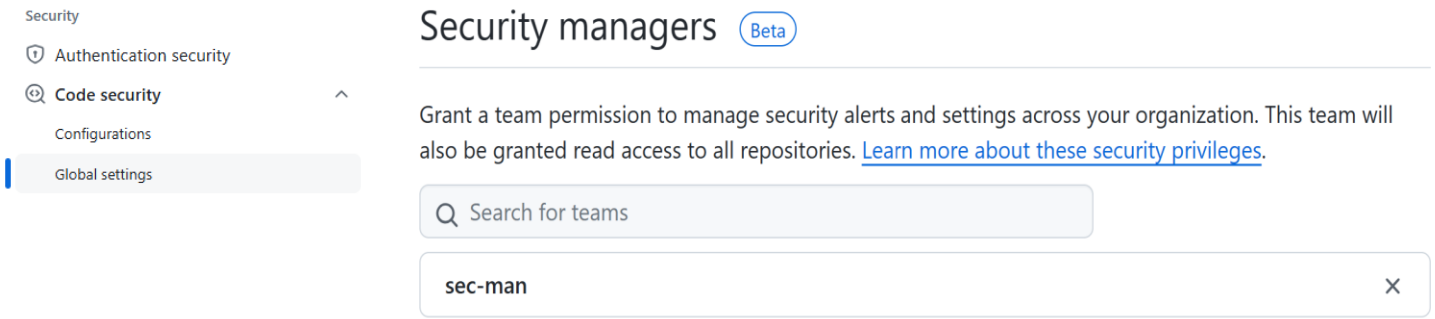
- Nested teams allow you to reflect your company's hierarchy within your org
- Parents team can have more than one child
    - Child teams inherit parent's permissions
    - Children receive parent's notifications
    - Users in a child team belong also to the parent team

Teams Synchronization

- Teams managed with Team Sync
  - Integrate IdP to synchronize groups to GitHub Teams
  - Manage rights and permissions in one place
  - Can’t connect to parent team (if using nested teams)
- Teams managed in GitHub
  - Manage membership within GitHub
  - Keep them open, reduce friction



Security Managers in your Organization



- Security manager is an organization-level role that organization owners can assign to any team in an organization.
- It gives every member of the team permissions to view security alerts and manage settings for code security across your organization, as well as read permissions for all repositories in the organization.

GitHub Actions

- Fully integrated with GitHub
- Respond to any GitHub event
- Community-powered workflows
- Any platform, any language, any cloud

Starter workflows

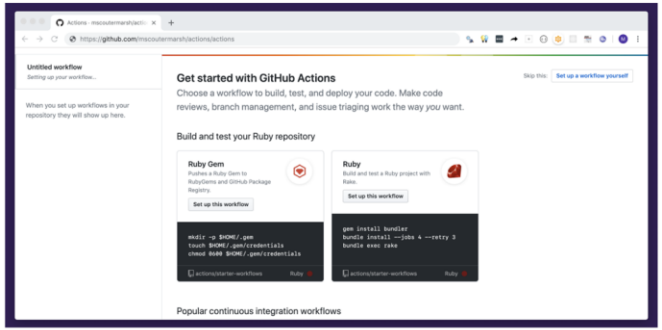
- https://github.com/actions/starter-workflows

README.md

Starter Workflows

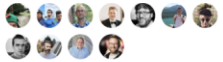
These are the workflow files for helping people get started with GitHub Actions. They're presented whenever you start to create a new GitHub Actions workflow.

If you want to get started with GitHub Actions, you can use these starter workflows by clicking the "Actions" tab in the repository where you want to create a workflow.



Directory structure:

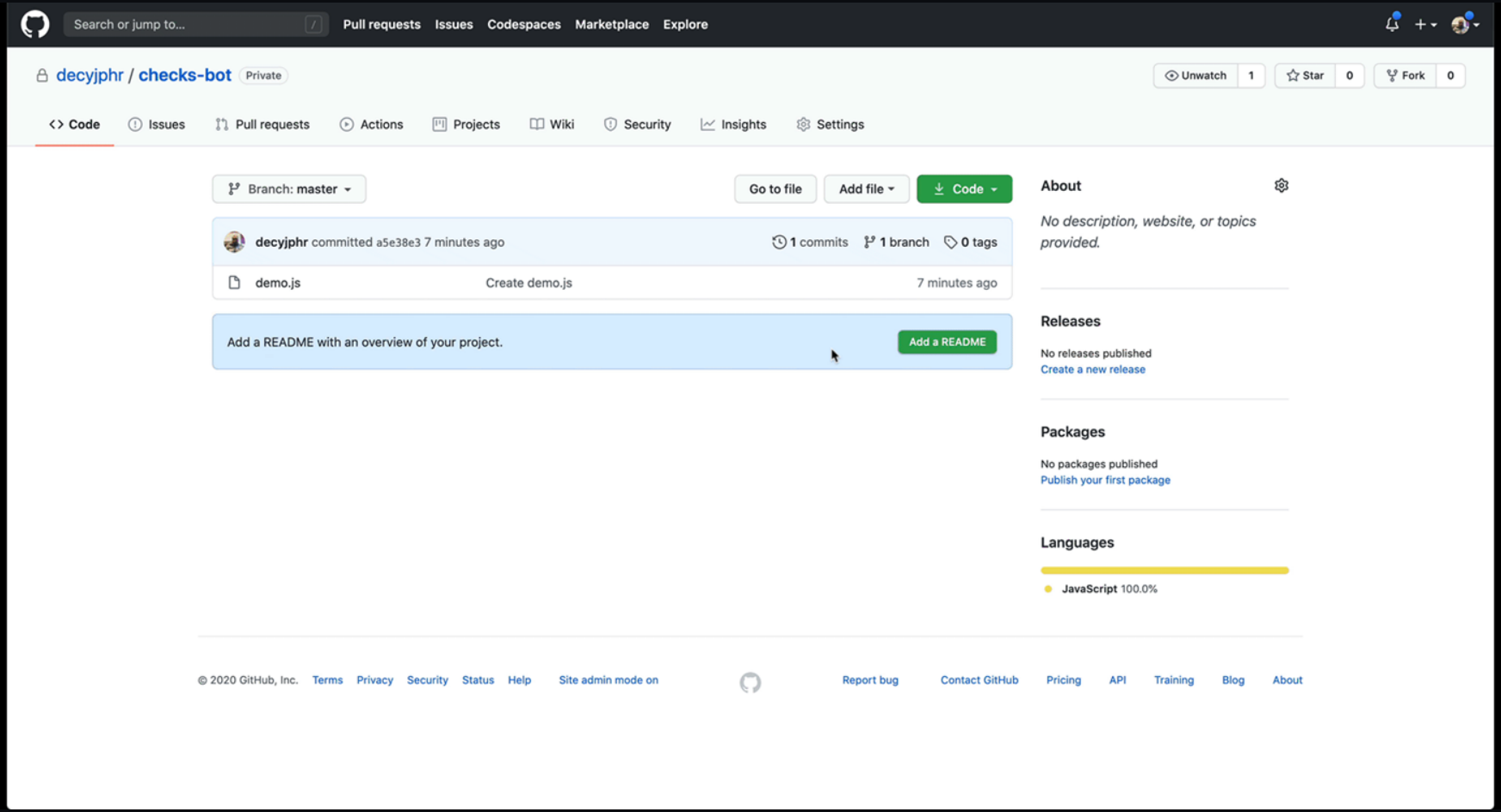
- ci: solutions for Continuous Integration
- automation: solutions for automating workflows.
- icons: svg icons for the relevant template



+ 96 contributors

Languages

- TypeScript 100.0%


Search or jump to...
Pull requests
Issues
Codespaces
Marketplace
Explore
decyjphr / checks-bot
Private
Unwatch 1
Star 0
Fork 0
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
Branch: master
Go to file
Add file
Code
decyjphr committed a5e38e3 7 minutes ago
1 commits
1 branch
0 tags
demo.js
Create demo.js
7 minutes ago
Add a README with an overview of your project.
Add a README
About
No description, website, or topics provided.
Releases
No releases published
Create a new release
Packages
No packages published
Publish your first package
Languages
JavaScript 100.0%
© 2020 GitHub, Inc.
Terms
Privacy
Security
Status
Help
Site admin mode on
Report bug
Contact GitHub
Pricing
API
Training
Blog
About

Workflows

Workflow files glue together existing actions

- Listen for particular events
- Then run shell scripts
- Or pre-existing actions
- Actions run in VMs (Linux, Win, Mac)
  - Or Docker on Linux VM
- yaml syntax
- logs streaming & artifacts
- Secret store with each repository

1
2
3

```yaml
name: Enforce repository settings

on: [push]

jobs:
  probot-settings:

    runs-on: ubuntu-latest

    steps:
    - uses: actions/checkout@v1
    - name: Run probot-settings
      uses: elstudio/actions-settings@v2-beta
      env:
        GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
    - name: Remove workflow file from master branch
      run: rm ./.github/workflows/probot-settings.yml
      if: endsWith(github.ref, '/master') && ! endsWith(git
    - name: Commit changes
      uses: elstudio/actions-js-build/commit@v2
      env:
        GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}

```

Actions are reusable components

- Live in independent repositories
  - Public repositories for now
  - Official list: https://github/actions
- Written in JavaScript (node12)
  - May use GitHub Actions Toolkit JS for command line argument parsing, passing parameters, interacting with the GitHub API
- Or Docker
  - Similar to beta 1 actions, but with updated syntax & argument passing
  - Or point to existing actions published to Docker Hub

1

```yaml
name: 'Wait'
description: 'Wait a designated number of milliseconds'
inputs:
  milliseconds:  # id of input
    description: 'number of milliseconds to wait'
    required: true
    default: '1000'
outputs:
  time: # output will be available to future steps
    description: 'The message to output'
runs:
  using: 'node12'
  main: 'index.js'
```

2

```yaml
name: 'GitHub Action to execute javascript build tools'
description: 'Executes npm install, followed by gulp or gru
inputs:
  wdPath:
    description: 'Working directory path'
    required: false
    default: ''
runs:
  using: 'docker'
  image: 'Dockerfile'
  entrypoint: 'entrypoint.sh'
  env:
    WD_PATH: ${{ inputs.wdPath }}
```

Security with self-hosted runners

Best practices for configuration of your runner nodes:

- Create a dedicated user for the Actions runner
- Enable limited sudo
- Run the Actions runner inside Docker

Self-hosted runners and Security

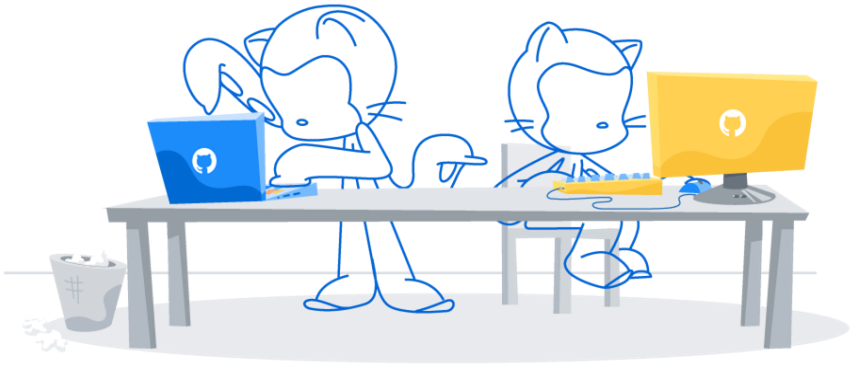
Many of your Actions may require *root* access in order to run. For example, if you are running an Action with Python natively, you may need to run *sudo* if you're checking out the `dschep/install-pipenv-action@v1` Action, as that runs *sudo* in order to install the necessary dependencies. In cases like this you may choose to avoid the Action, or you can limit the scope of your runner's permissions.

https://docs.github.com/en/actions/hosting-your-own-runners/about-self-hosted-runners

Security with self-hosted runners

Public repositories with self-hosted runners pose potential risks:

- Malicious programs running on the machine
- Escaping the machine's runner sandbox
- Exposing access to the machine's network
- Persisting unwanted or dangerous data on the machine



Self-hosted runners and Security

Forked repositories will contain the same Actions configuration as the parent repository, including the self-hosted runners. Creates the potential for a fork to run malicious code on a runner inside your network. For this reason, it is highly recommended to use self-hosted runners only with **private** repositories.

https://docs.github.com/en/actions/hosting-your-own-runners/about-self-hosted-runners

Secrets

Organization

- Allows secrets to be managed at Org level without duplication
- Effectively becomes Repository secrets
- Can be scoped to specific repositories

Repository

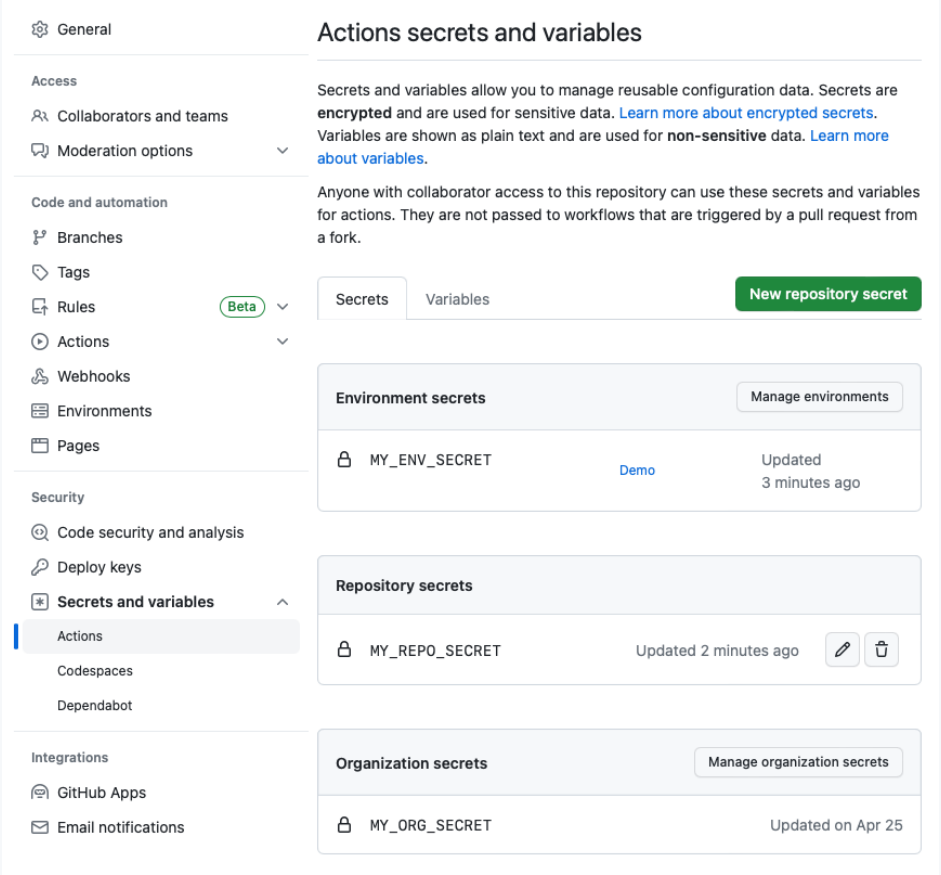
- Scoped to a repository
- Can be used to override Org secrets

Environment

- Scoped to a repository
- Can be used to override Repo secrets

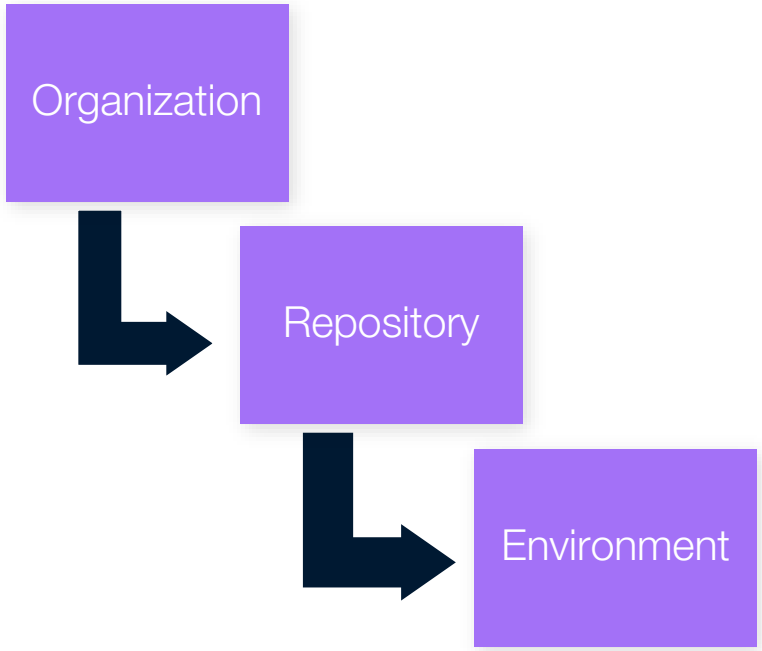
GitHub Secret store

- Built-in secret store
- Encrypted
  - LibSodium sealed box
- Use directly from your workflow
- Redacted in workflow logs
- API support
- Organization / repository / environment level secrets
- Can not read a secret value!



Secrets

- Defined on org, repo, or environment level
- Secret context
  - `${{ secrets.MY_SECRET }}`
  - Set as input (`with:`) or environment (`env:`) for actions
- Set in UI or CLI
  - `$ gh secret set MY_SECRET -body "$value"`
  - `$ gh secret set MY_SECRET --env Prod`
  - `$ gh secret set MY_SECRET --org my-org`
- Masked in log



The `GITHUB_TOKEN`

`${{ secrets.GITHUB_TOKEN }}` or `${{ github.token }}`

Authenticate to GitHub to perform automation inside the workflow's repo

Default permission read/write for all scopes (old default) or set to read

```yaml
permissions:
  contents: read
  pull-requests: write
```

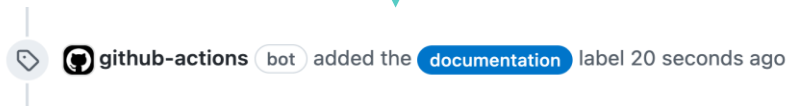
```yaml
permissions: read-all
```

```yaml
permissions:
  actions: read|write|none
  checks: read|write|none
  contents: read|write|none
  deployments: read|write|none
  issues: read|write|none
  packages: read|write|none
  pull-requests: read|write|none
  repository-projects: read|write|none
  security-events: read|write|none
  statuses: read|write|none
```

The GITHUB_TOKEN

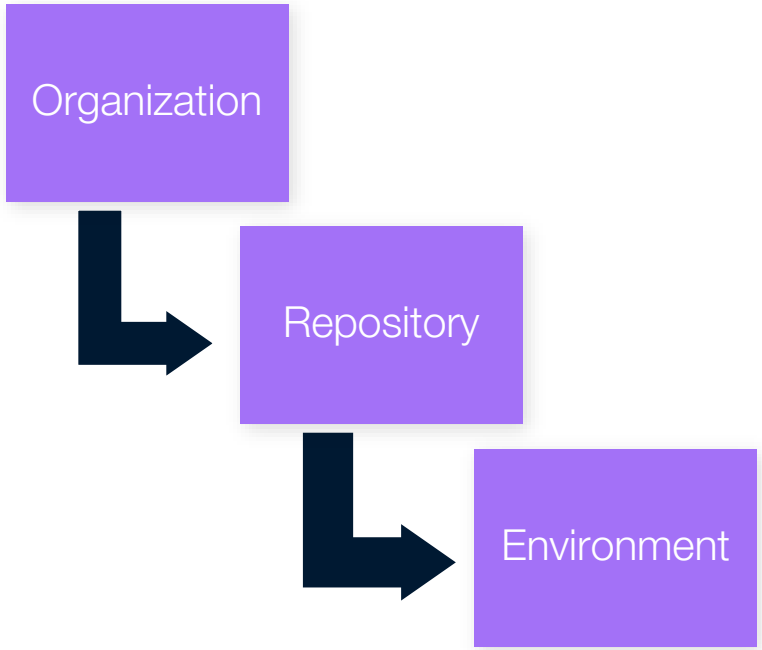
Perform actions as “github-actions”:

```yaml
permissions:
  contents: read
  issues: write


 label_issues:
   runs-on: ubuntu-latest
   if: github.event_name == 'issues'

   steps:
     - uses: andymckay/labeler@e6c4322d0397f3240f0e7e30a33b5c5df2d39e90
       with:
         add-labels: documentation
         repo-token: ${{ secrets.GITHUB_TOKEN }}
```



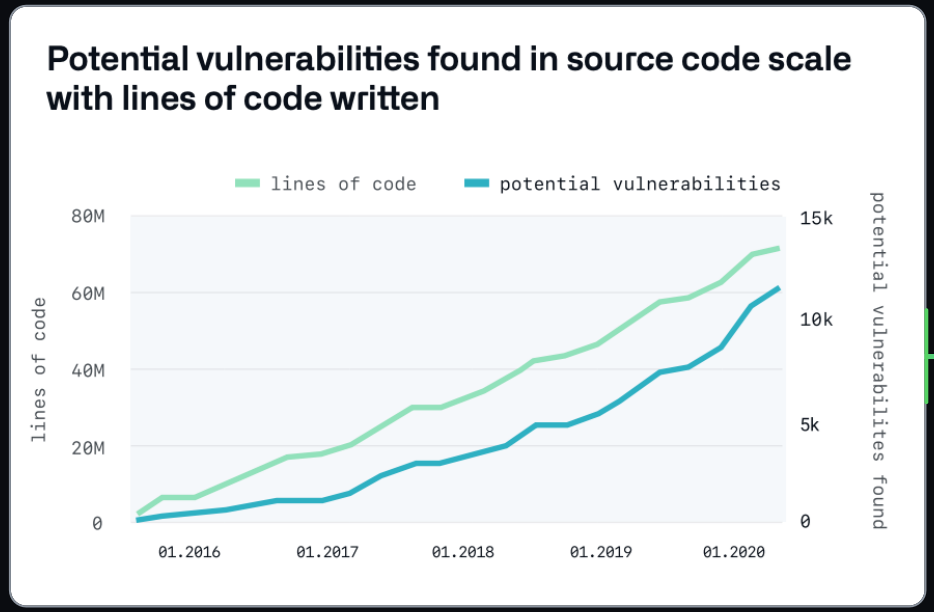
Variables

- Same setup as secrets, but no redacting
- Defined on org, repo, or environment level
- vars context
  - `${{ vars.MY_VAR }}`
  - Set as input (`with:`) or environment (`env:`) for actions
- **Not** masked in log



GHAS Application Security

The state of AppSec



Despite billions of dollars of investment…

85% of applications still contain a security issue

Code written in 2020 is just as likely to introduce a security issue as code written in 2016

Flaws in applications are consistently the #1 attack vector for breaches

Source: Verizon Data Breach Investigations reports 2016, 2017, 2018, 2019 and 2020.

The state of AppSec

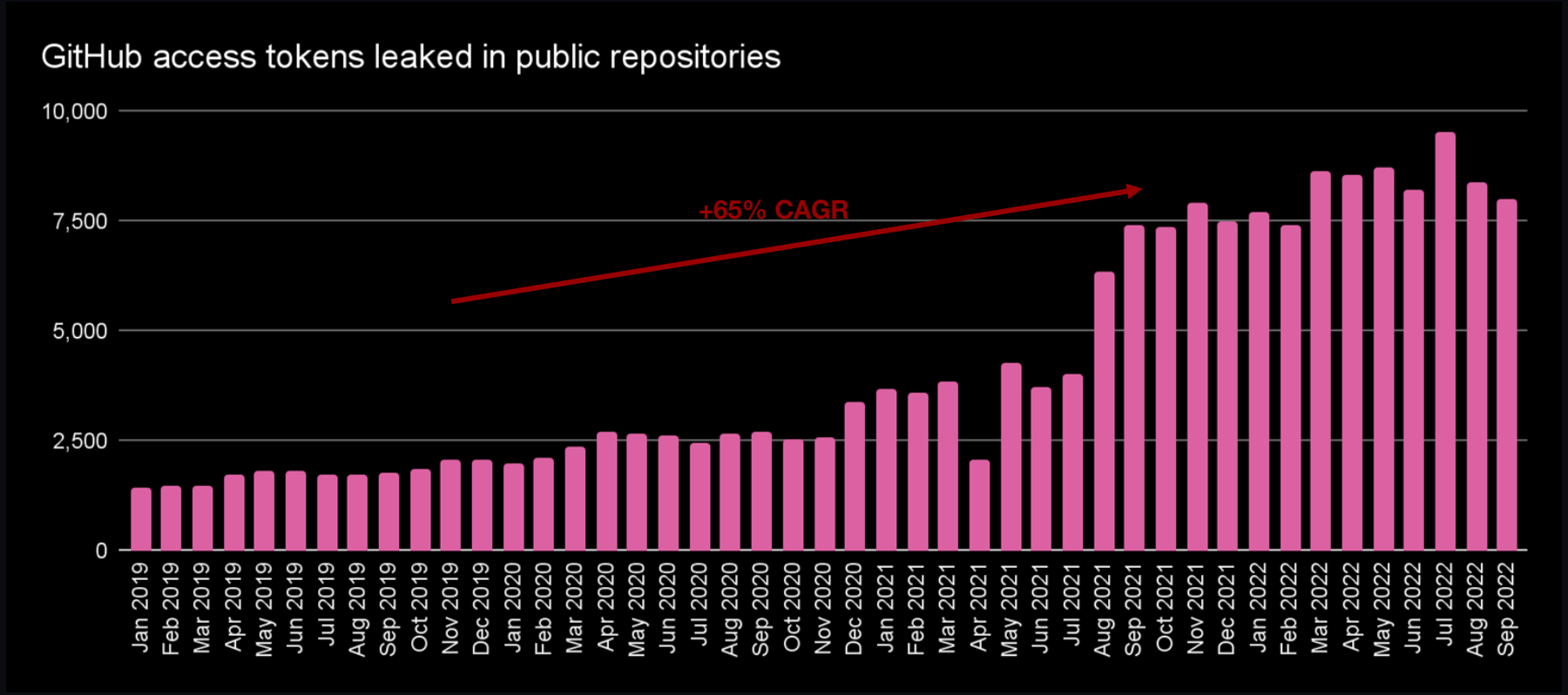
Is falling further behind the current state of Development

1:100 Security team members to developers

Lack of knowledge voted the main AppSec challenge

Remediation trends are stagnant

We’re seeing more credential leaks than ever

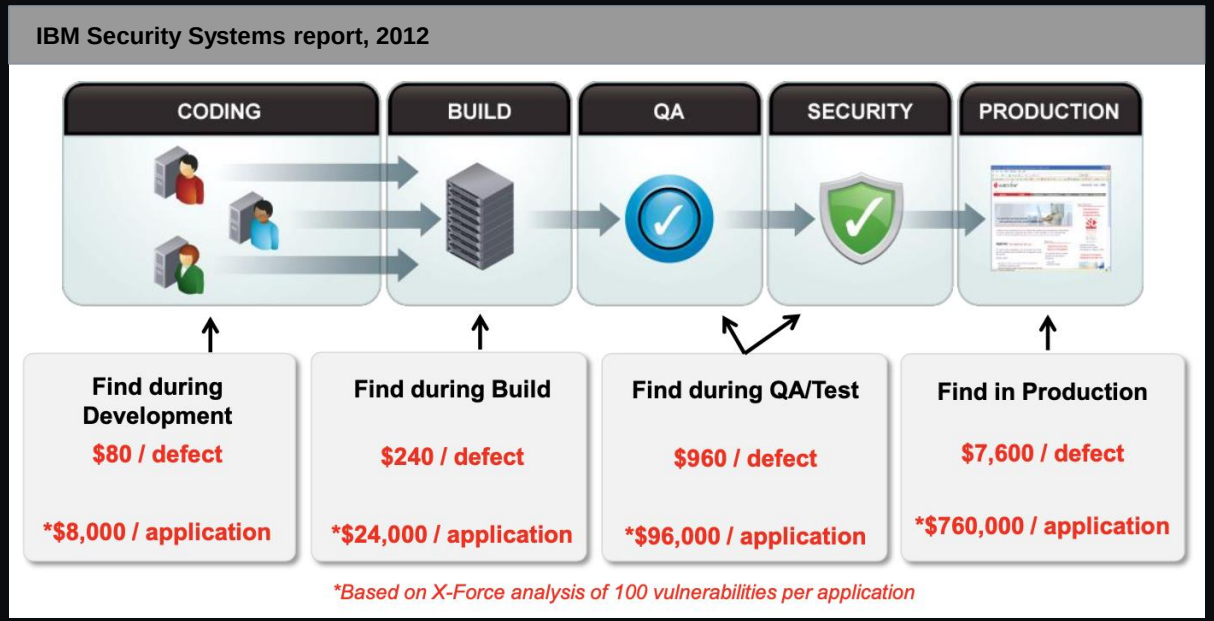


Source: GitHub data

Everyone wants to shift security left…

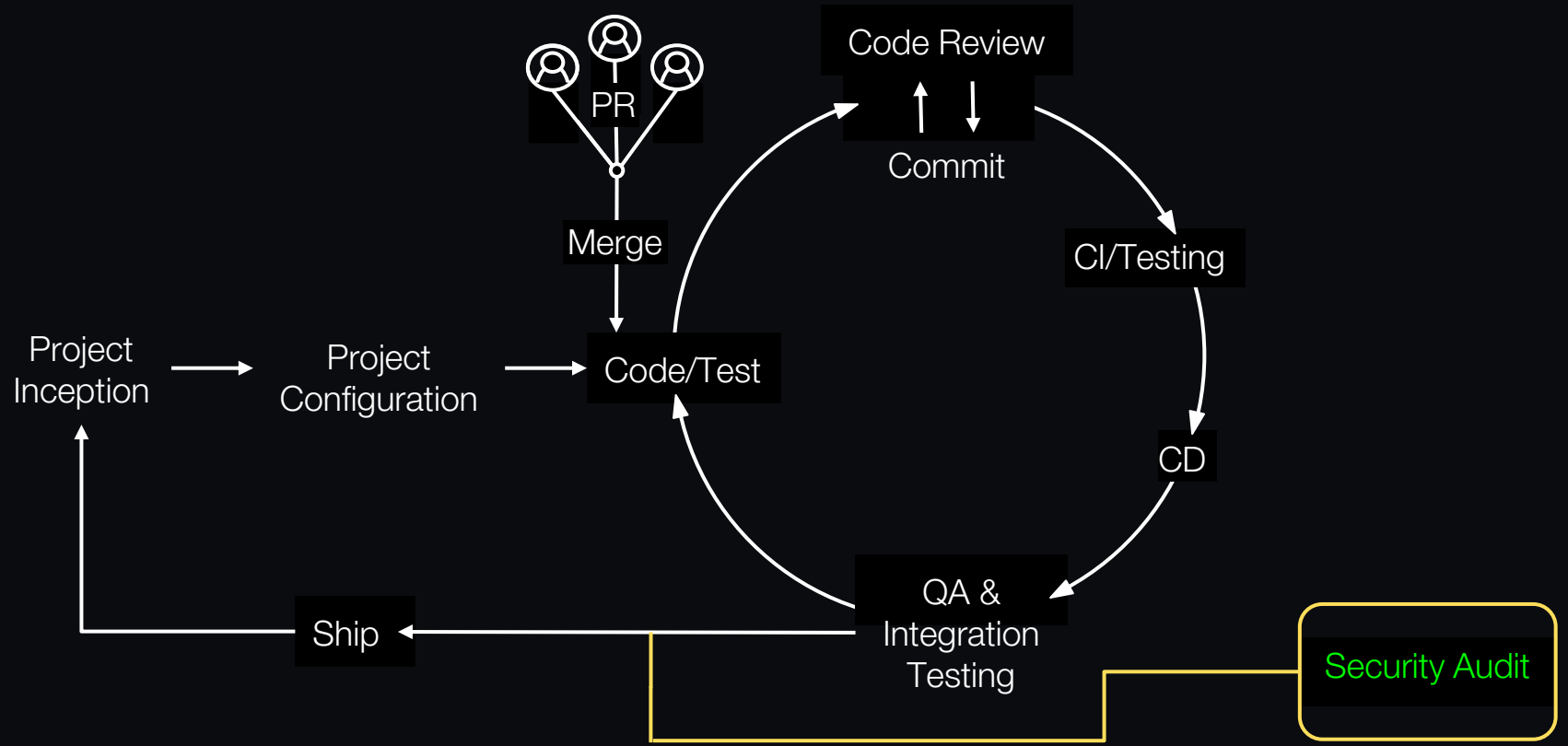
Source: Ponemon Institute Cost of a Data Breach 2020

… but the industry has been trying to shift left for at least a decade

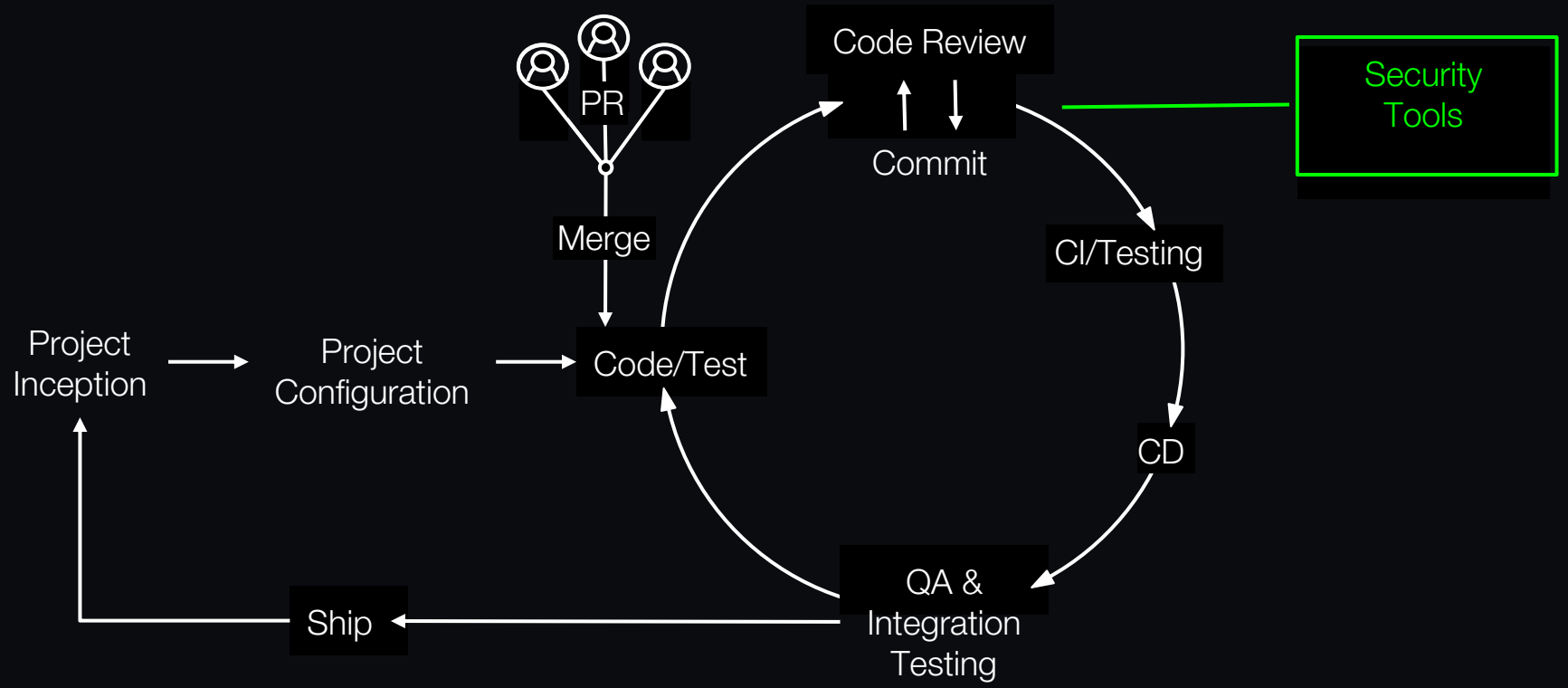


GitHub believes that making this shift requires a developer-first approach to all our security products

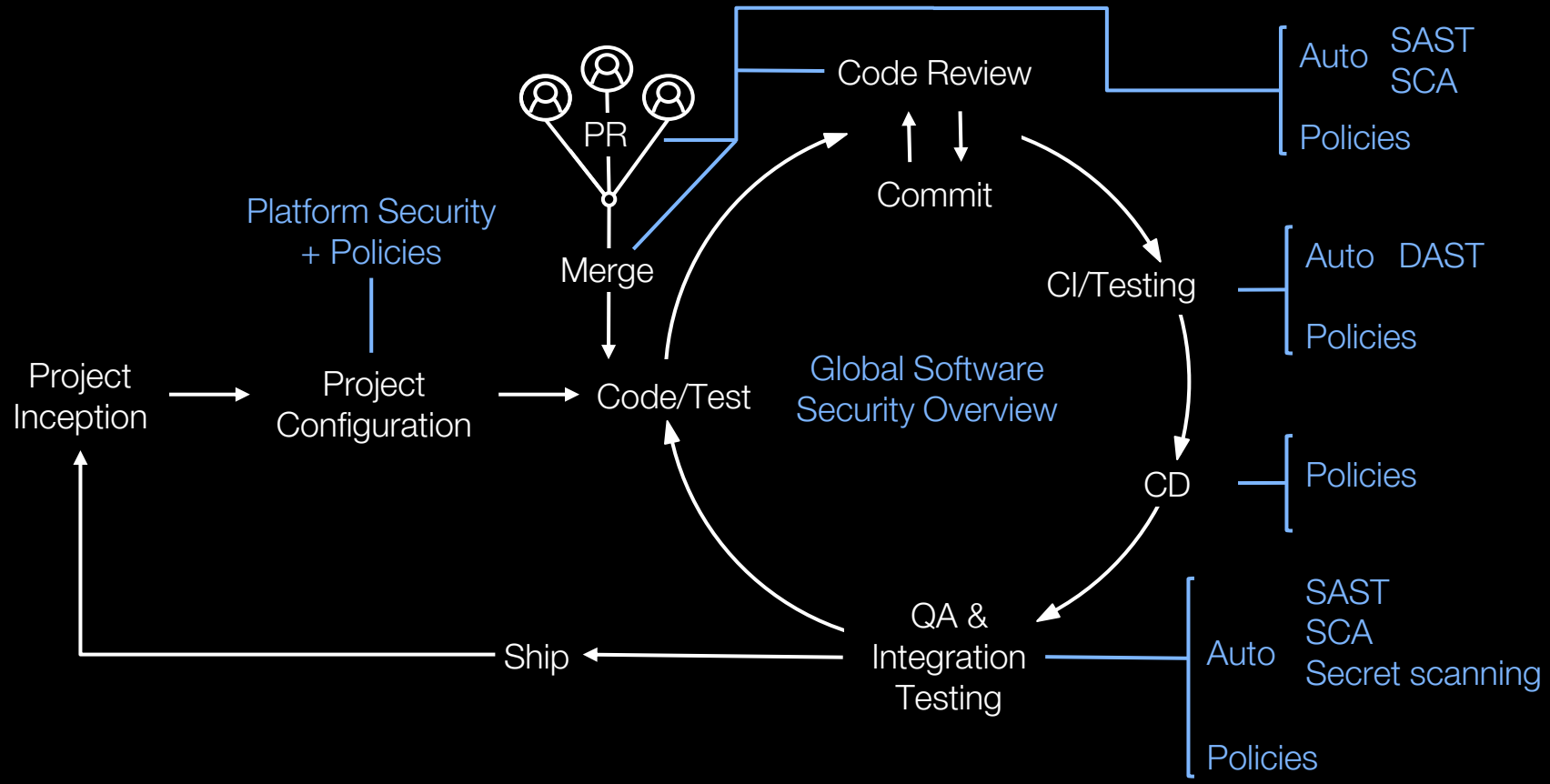
Basic Application Security scenario



Application Security scenario



Application Security - Targeted state



Developer first?

We see three key aspects to being a “developer first” tool:

Integrate *directly* into the developer workflow.

Make setup and deployment fast and easy.

Produce high quality results with low numbers of false positives.

GitHub Advanced Security: Current capabilities

supply chain

Dependency graph
View your dependencies

Advisory database
Canonical database of dependency vulnerabilities

Security alerts and updates
Notifications for vulnerabilities in your dependencies, and pull requests to fix them

Dependency review
Identify new dependencies and vulnerabilities in a PR

code

Secret scanning
Find API tokens or other secrets exposed anywhere in your git history.

Code scanning
Static analysis of every git push, integrated into the developer workflow and powered by CodeQL

platform

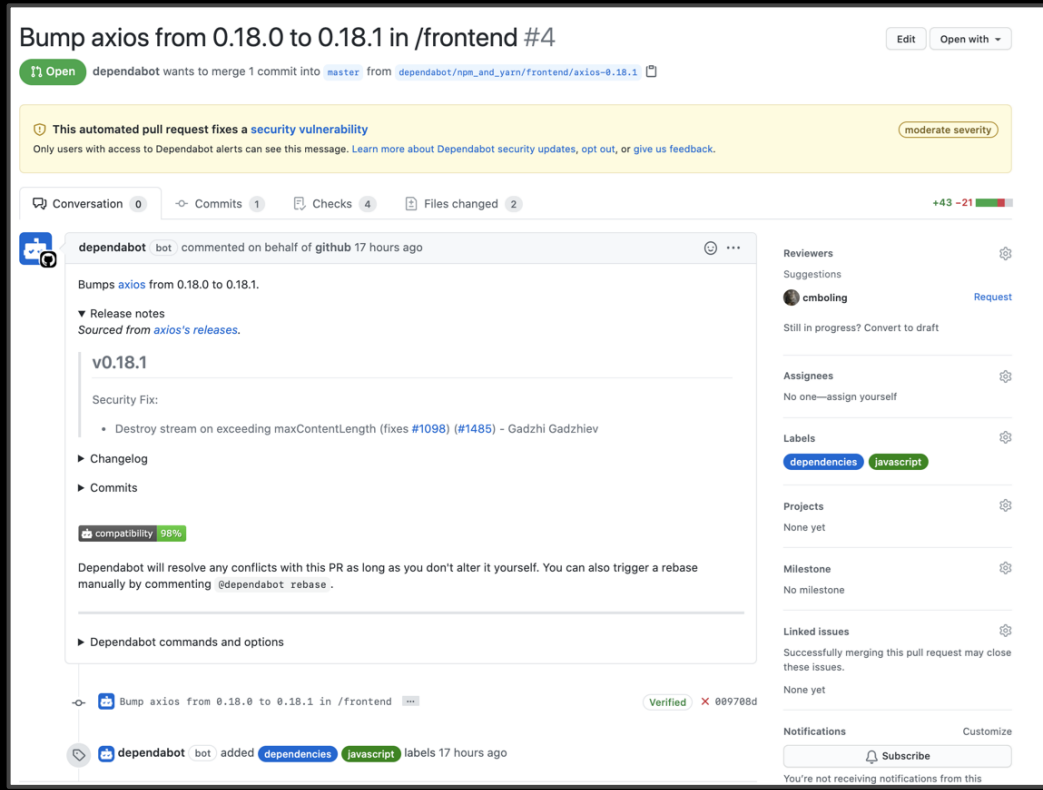
Branch protection
Enforce requirement for pushing to a branch or merging PRs

Commit signing
Enforce requirement that all commits are signed

Security overview
View security results of all kinds acros
your organization

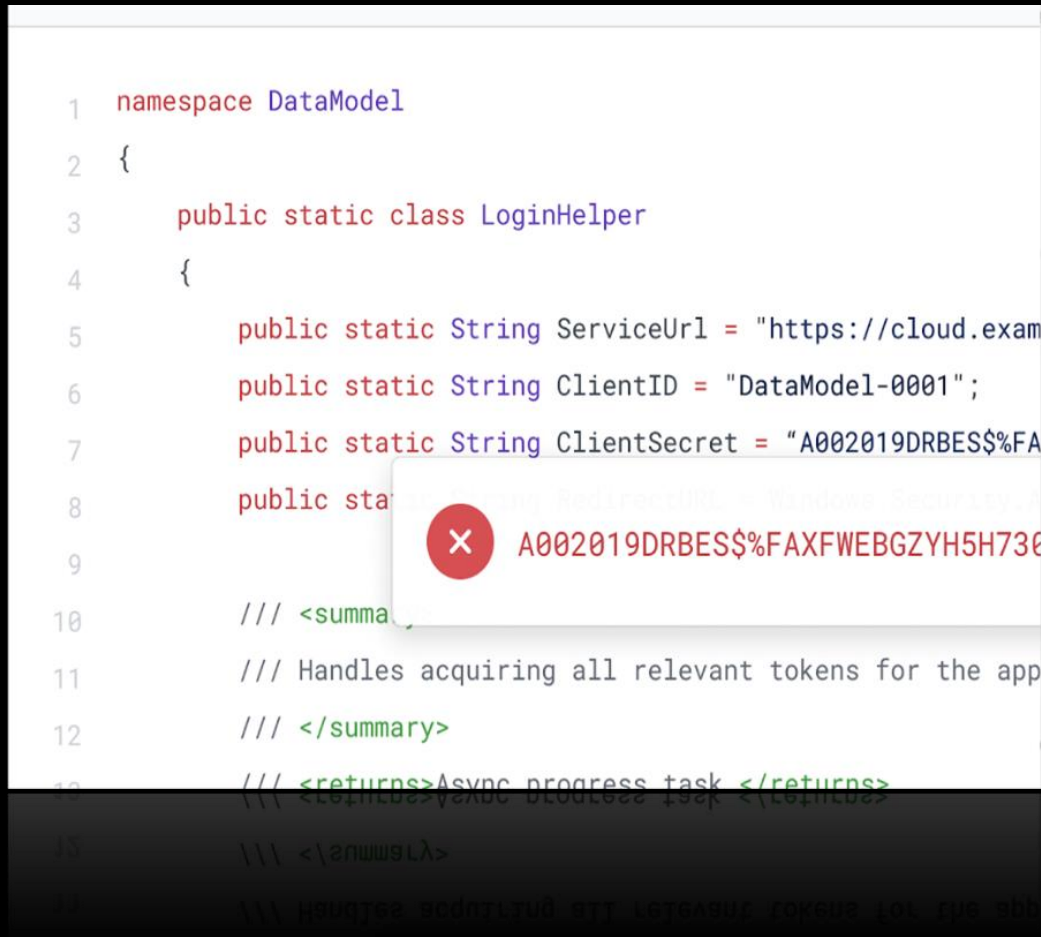
Dependabot

- Developers (and others!) notified by an alert when new vulnerable dependencies are detected.
- Automatically open pull requests to fix dependency vulnerabilities.
- Supports dependency review within PRs to prevent adding known vulnerable dependencies.



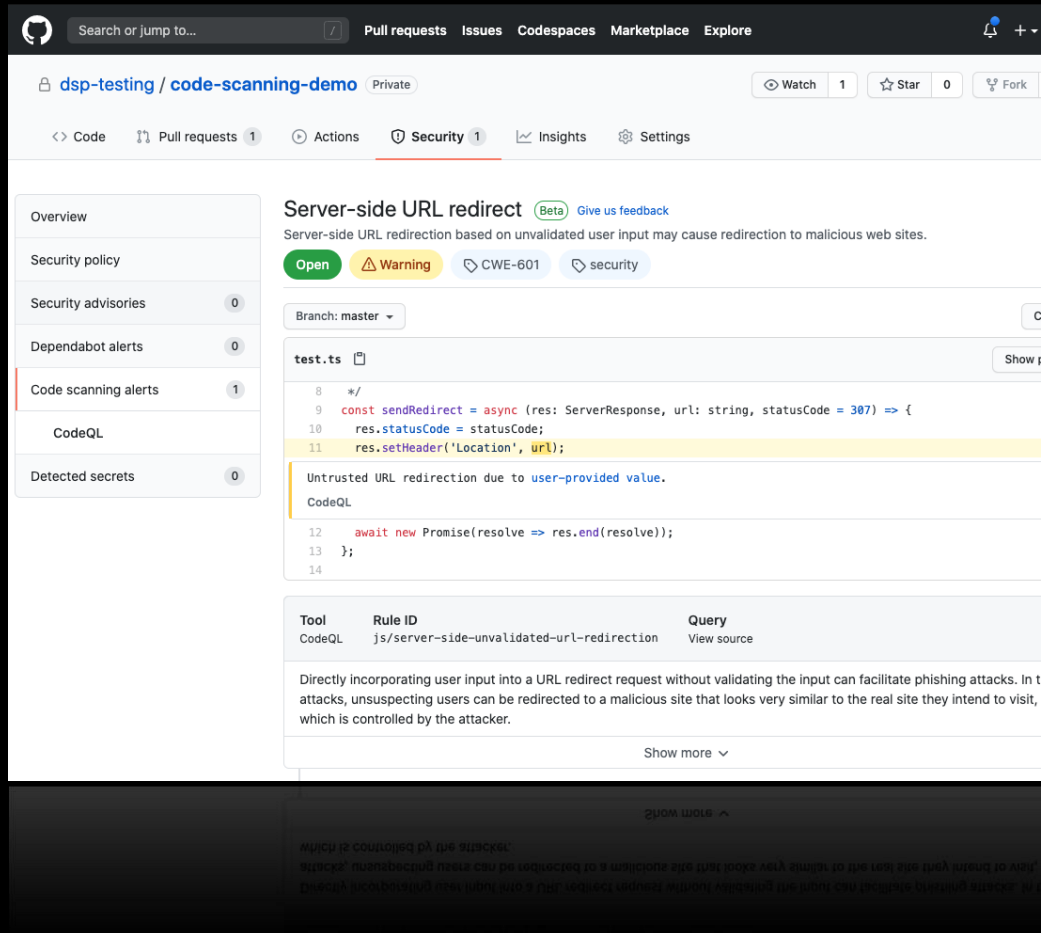
Secret scanning

- Identify secrets across your entire git history with high accuracy.
- Push protection - prevent secrets from being pushed to GitHub.
- Developers (and others!) notified by an alert if secrets are pushed.
- Automated revocation for public repositories, private repositories include a review workflow.

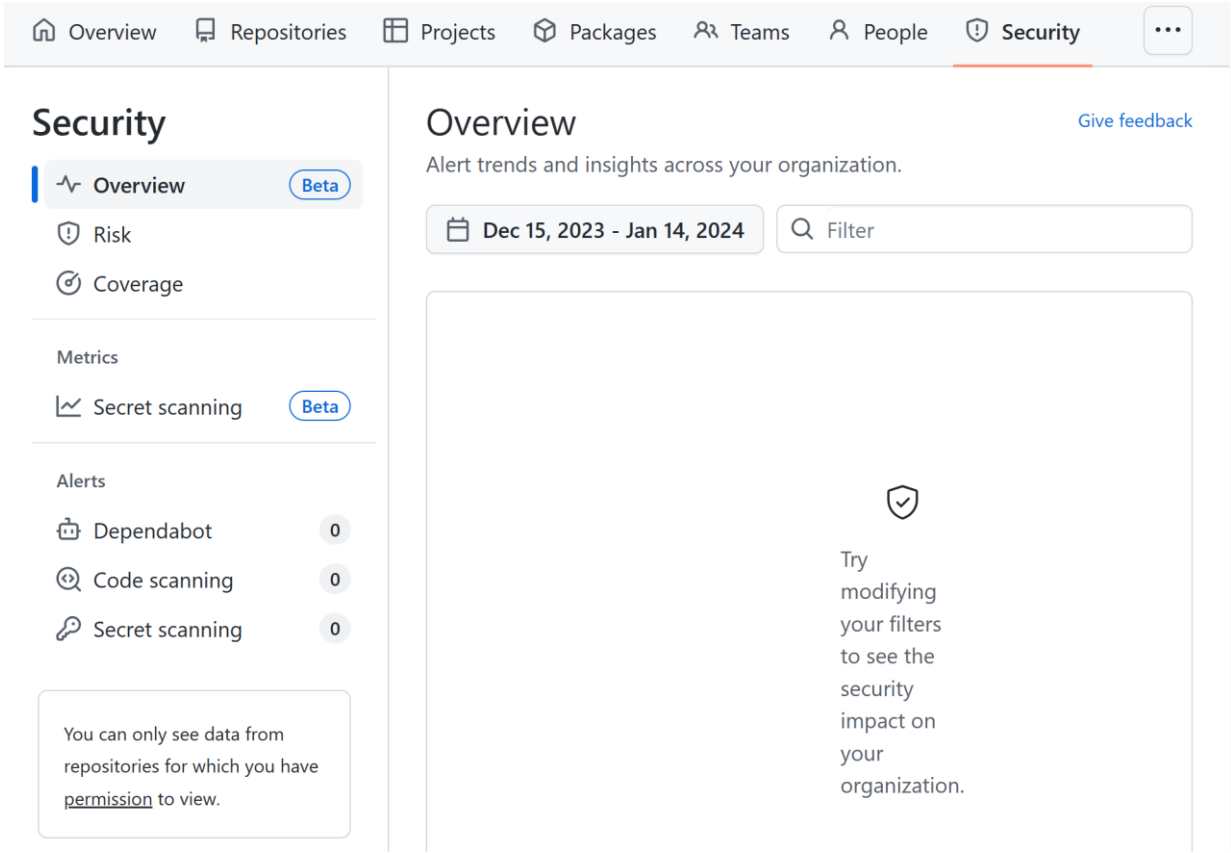


Code scanning

- Find vulnerabilities before they are merged into the code base with automated CodeQL scans
- Integrate results directly into the developer workflow
- Run custom queries and the community-powered GitHub query set
- Extensible, with support for other SAST tools



Reviewing Alerts



Monitoring and responding to alerts

Code samples for "List code scanning alerts for an organization"

Request example

GET /orgs/{org}/code-scanning/alerts

cURL | JavaScript | GitHub CLI

```
// Octokit.js
// https://github.com/octokit/core.js#readme
const octokit = new Octokit({
  auth: 'YOUR-TOKEN'
})

await octokit.request('GET /orgs/{org}/code-scanning/alerts', {
  org: 'ORG',
  headers: {
    'X-GitHub-Api-Version': '2022-11-28'
  }
})
```

Q&A