Security team training

GitHub Advanced Security

Agenda

Setting the scene

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

Configuring access

Creating teams and applying appropriate permissions.

Reviewing alerts

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

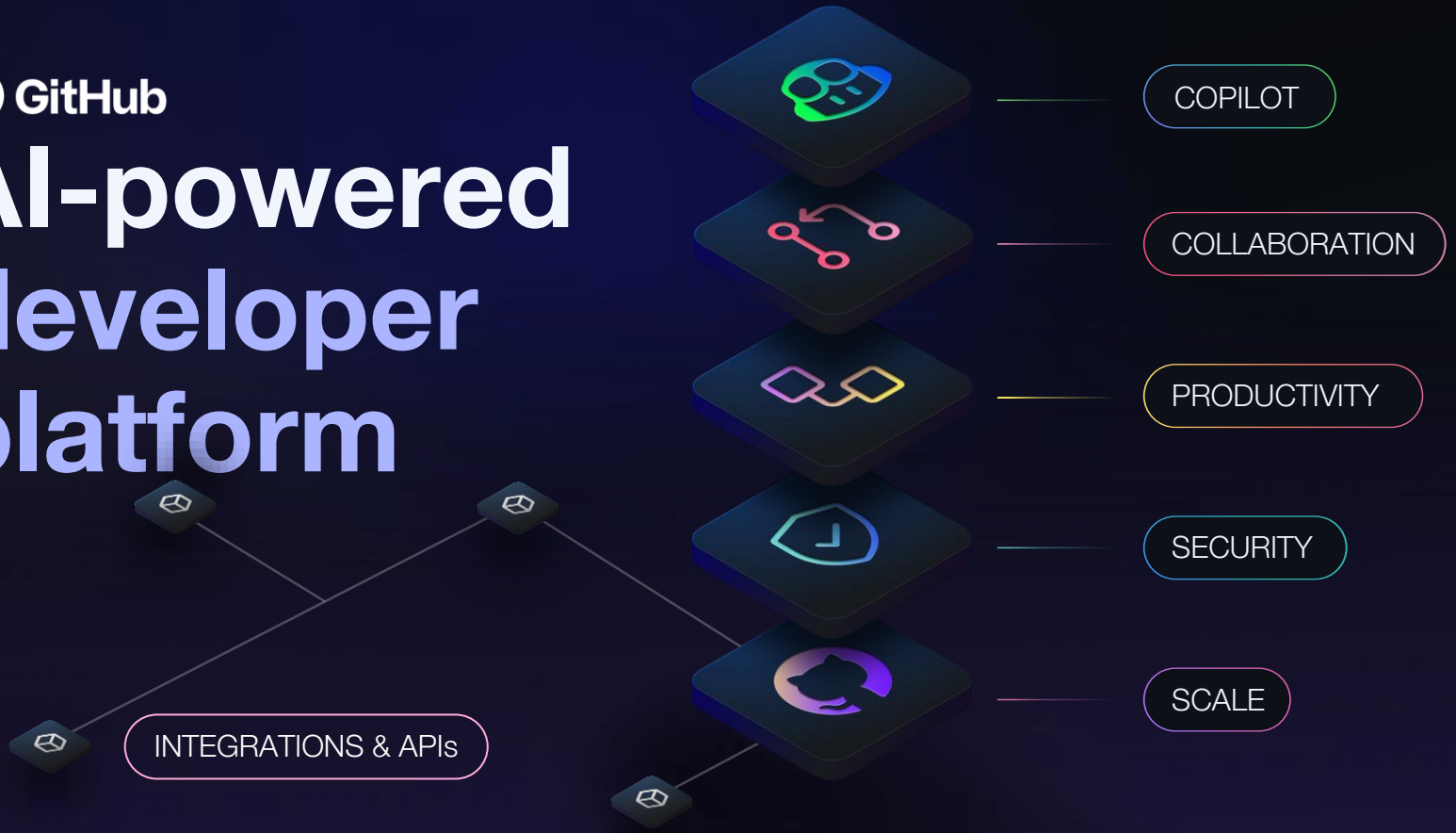
Monitoring and responding to alerts

Notifications, webhooks and APIs; building custom workflows.

Security Overview

GitHub

AI-powered developer platform



GitHub Advanced Security

Feature List

Supply Chain

Know your environment

- GitHub Advisory Database
- Dependency Graph
- Dependency Insights
- Dependency Submission API
- Software Bill of Materials (SBOM)

Manage your environment

- Dependabot Alerts
- Dependabot Security Updates
- Dependabot Version Updates
- Dependency Review

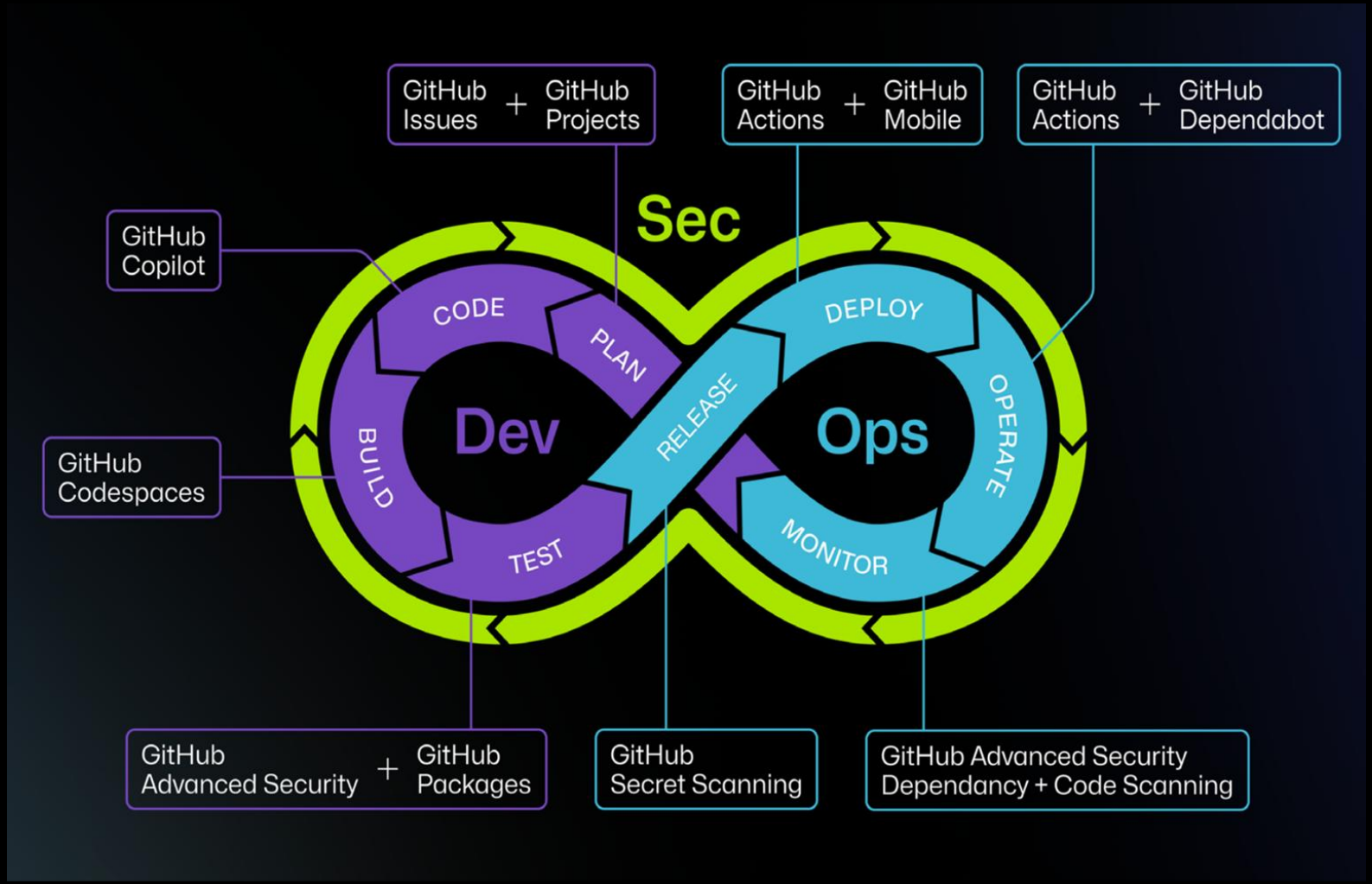
Secret Scanning

- Partner pattern and non provider pattern detection by default
- Push Protections for users and repositories
- Custom Patterns
- Regex generation with AI
- Generic Secret AI detection

Code Scanning

- Static analysis using CodeQL
- Use GitHub Actions or any other third-party CI/CD
- Run on push, pull request, schedule or manual trigger
- SARIF output
- Third-party code scanning tool support

Security Overview Dashboard

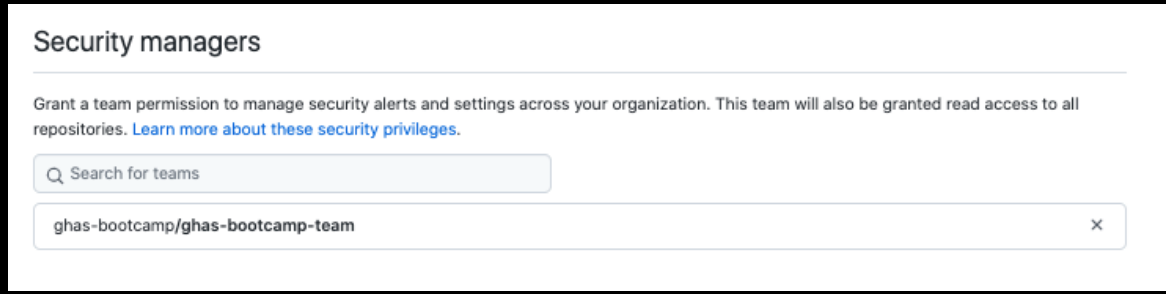

GitHub Issues + GitHub Projects
GitHub Actions + GitHub Mobile
GitHub Actions + GitHub Dependabot
Sec
GitHub Copilot
CODE
PLAN
DEPLOY
OPERATE
Dev
RELEASE
Ops
GitHub Codespaces
BUILD
TEST
MONITOR
GitHub Advanced Security + GitHub Packages
GitHub Secret Scanning
GitHub Advanced Security Dependancy + Code Scanning

Configuring Access

Security Manager Role

Members of a team with the security manager role have only the permissions required to effectively manage security for the organization.

- **Read** access on all repositories in the organization, in addition to any existing repository access
- **Write** access on all security alerts in the organization
- The ability to configure security settings at the organization level
- The ability to configure security settings at the repository level
- If your organization has a security team, you can use the security manager role to give members of the team the least access they need to the organization



Access Requirements for Security Features

- You can customize access to each repository in your organization by assigning granular roles, giving people access to the features and tasks they need

- Following are the default roles and the repository level actions that a person can perform with respect to the different security features

Repository action	Read	Triage	Write	Maintain	Admin
Receive Dependabot alerts for insecure dependencies in a repository	×	×	✓	✓	✓
Dismiss Dependabot alerts	×	×	✓	✓	✓
Designate additional people or teams to receive security alerts	×	×	×	×	✓
Create security advisories	×	×	×	×	✓
Manage access to GitHub Advanced Security features (see "Managing security and analysis settings for your organization")	×	×	×	×	✓
Enable the dependency graph for a private repository	×	×	×	×	✓
View dependency reviews	✓	✓	✓	✓	✓
View code scanning alerts on pull requests	✓	✓	✓	✓	✓
List, dismiss, and delete code scanning alerts	×	×	✓	✓	✓
View and dismiss secret scanning alerts in a repository	×	×	✓	✓	✓
Resolve, revoke, or re-open secret scanning alerts	×	×	✓	✓	✓
Designate additional people or teams to receive secret scanning alerts in repositories	×	×	×	×	✓

Permission Flow to Access Data in Security Overview

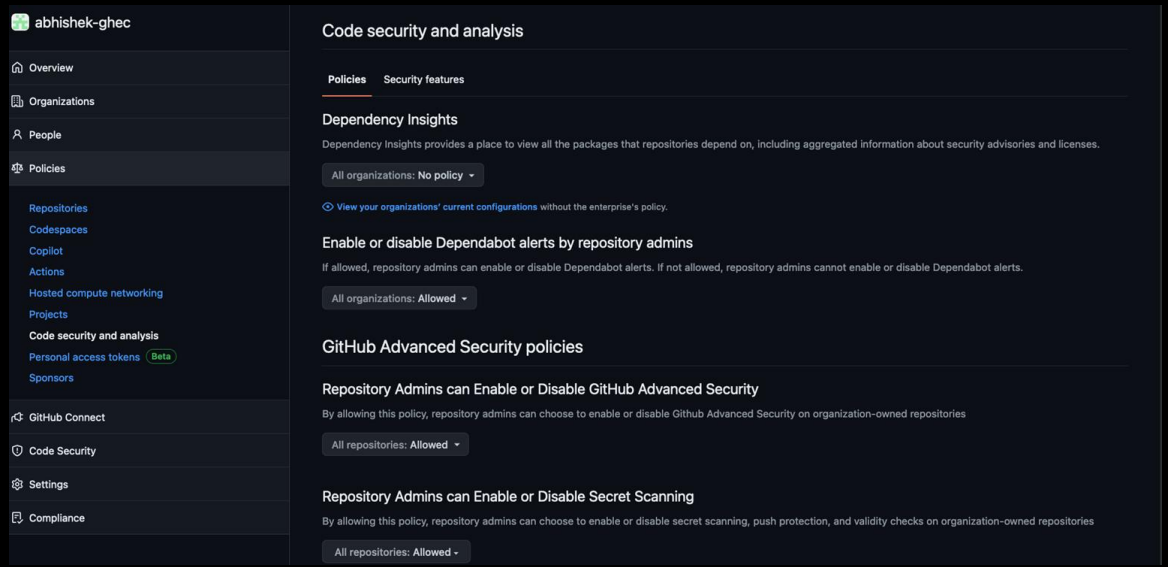
Organization member with	**Overview dashboard view**	**Risk and alerts views**	**Coverage view**
`admin` access for one or more repositories	View data for those repositories	View data for those repositories	View data for those repositories, and enable and disable security features
`write` access for one or more repositories	View code scanning and Dependabot data for those repositories	View code scanning and Dependabot data for those repositories	No access for those repositories
Security alert access for one or more repositories	View all security alert data for those repositories	View all security alert data for those repositories	No access for those repositories
Custom organization role with permission to view one or more types of security alert	View allowed alert data for all repositories	View allowed alert data for all repositories in all views	No access

GHAS Policies & Settings

GHAS Security Scanning Policies

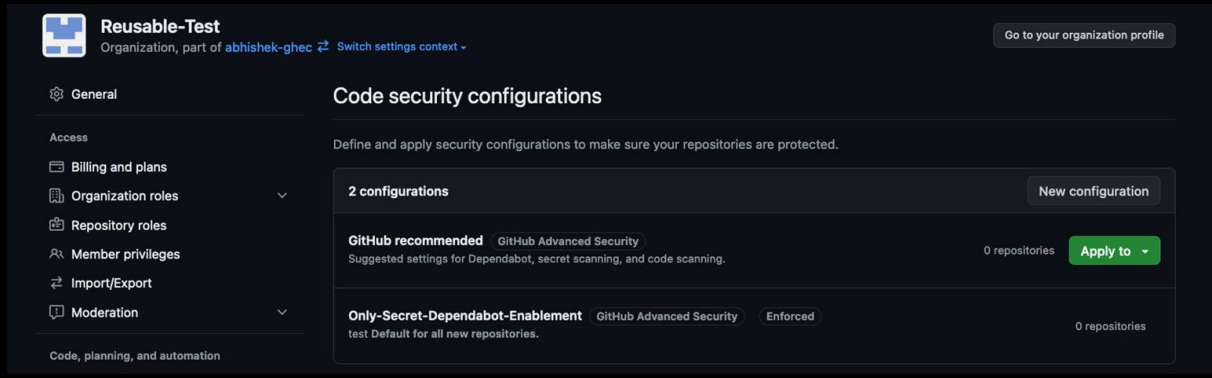
- GHAS Enablement policies can be defined at the Enterprise level or it can also be configured to delegate it to the Org and Repo Owners

- It is important to get the Policy definitions ironed out during the start of the rollout so that Security managers can have a uniform enforcement across the enterprise and there is no policy drift



GHAS Code Security Configurations

- GHAS Settings can also be applied at multiple levels in GitHub - Enterprise, Org, Repository level
- As Security Managers and GitHub Site Owners, it is important to understand at what level do you want to enforce those settings at so that you have a sustained rollout of GHAS
- With proper planning you can have an optimal usage of your GHAS license
  - Either you enable all and roll out GHAS licenses in one go
  - Or you target your most Business critical applications and corresponding Orgs and Repos



GHAS Code Security Configurations (Continued)

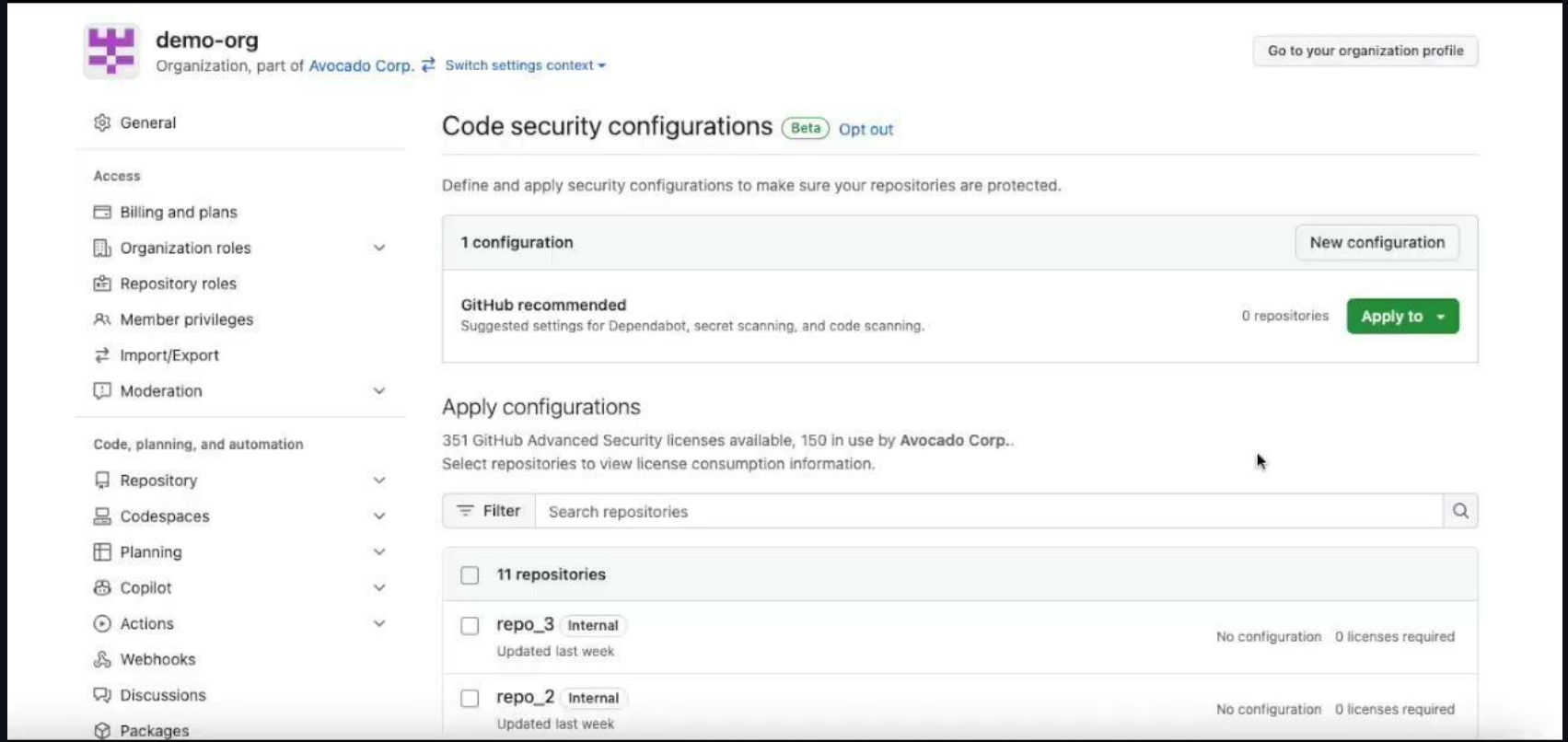
- Code security configurations simplify the rollout of GitHub security products at scale by defining collections of security settings that can be applied to groups of repositories

- Apply the ‘GitHub recommended’ security configuration, which applies GitHub’s suggested settings for Dependabot, secret scanning, and code scanning or create your own custom security configurations

- Code Configuration lets you manage security settings based on different risk profiles and security needs

- Security configurations, you can also see the additional number of GitHub Advanced Security (GHAS) licenses that are required to apply a configuration, or made available by disabling GHAS features on selected repositories. This lets you understand license usage when you roll out GitHub’s code security features in your organization.

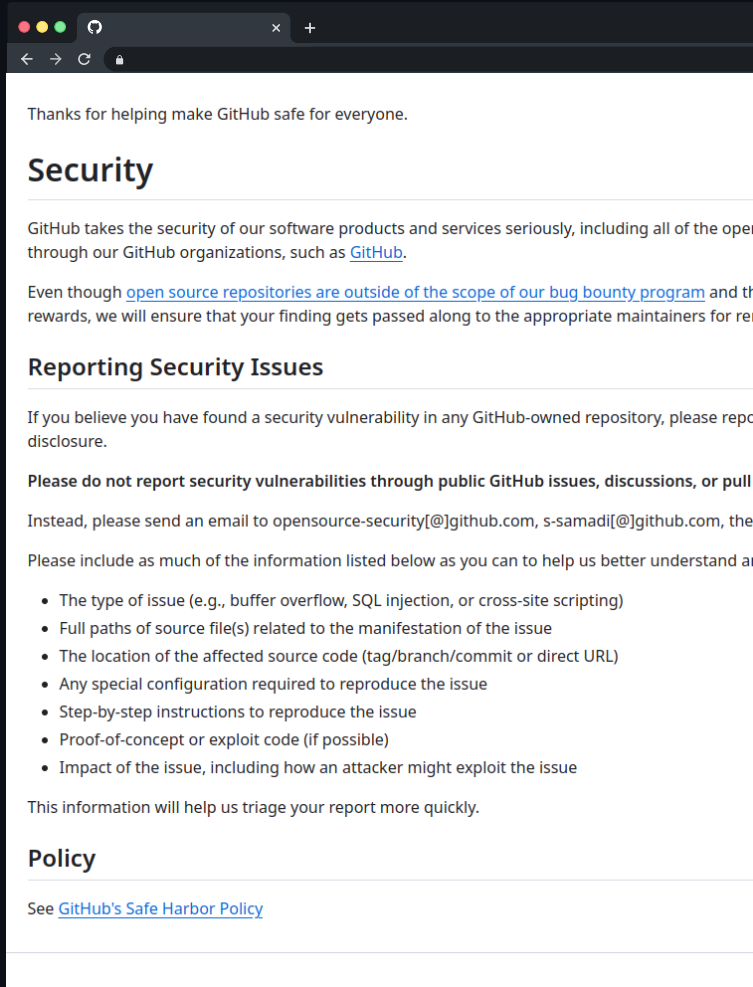
GHAS Code Security Configurations



Fix and Publish Vulnerability Information

SECURITY.md

- Instructions on how to report a security vulnerability in your repository
- May reside in the repository root, docs or .github folder
- When an issue is raised in the repository, a link to your repository's security policy will be present
- After someone reports a security vulnerability in your project, you can use GitHub Security Advisories to disclose, fix, and publish information about the vulnerability.



Thanks for helping make GitHub safe for everyone.

Security

GitHub takes the security of our software products and services seriously, including all of the oper
through our GitHub organizations, such as GitHub.

Even though open source repositories are outside of the scope of our bug bounty program and th
rewards, we will ensure that your finding gets passed along to the appropriate maintainers for re

Reporting Security Issues

If you believe you have found a security vulnerability in any GitHub-owned repository, please repo
disclosure.

Please do not report security vulnerabilities through public GitHub issues, discussions, or pull

Instead, please send an email to opensource-security[@]github.com, s-samadi[@]github.com, the

Please include as much of the information listed below as you can to help us better understand a

- The type of issue (e.g., buffer overflow, SQL injection, or cross-site scripting)
- Full paths of source file(s) related to the manifestation of the issue
- The location of the affected source code (tag/branch/commit or direct URL)
- Any special configuration required to reproduce the issue
- Step-by-step instructions to reproduce the issue
- Proof-of-concept or exploit code (if possible)
- Impact of the issue, including how an attacker might exploit the issue

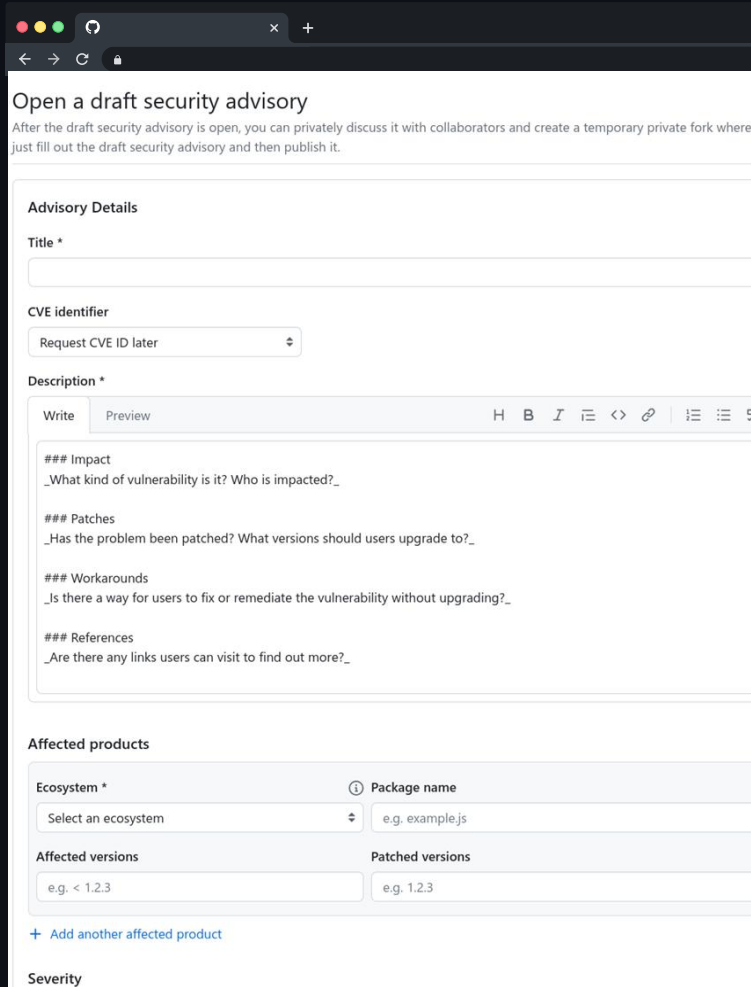
This information will help us triage your report more quickly.

Policy

See GitHub's Safe Harbor Policy

Security Advisories

- Security advisories allow repository maintainers to privately discuss and fix a security vulnerability in a project.
- GitHub Security Advisories builds upon the foundation of the Common Vulnerabilities and Exposures (CVE) list. The security advisory form on GitHub is a standardized form that matches the CVE description format.



GHAS Reusability & Enforcements

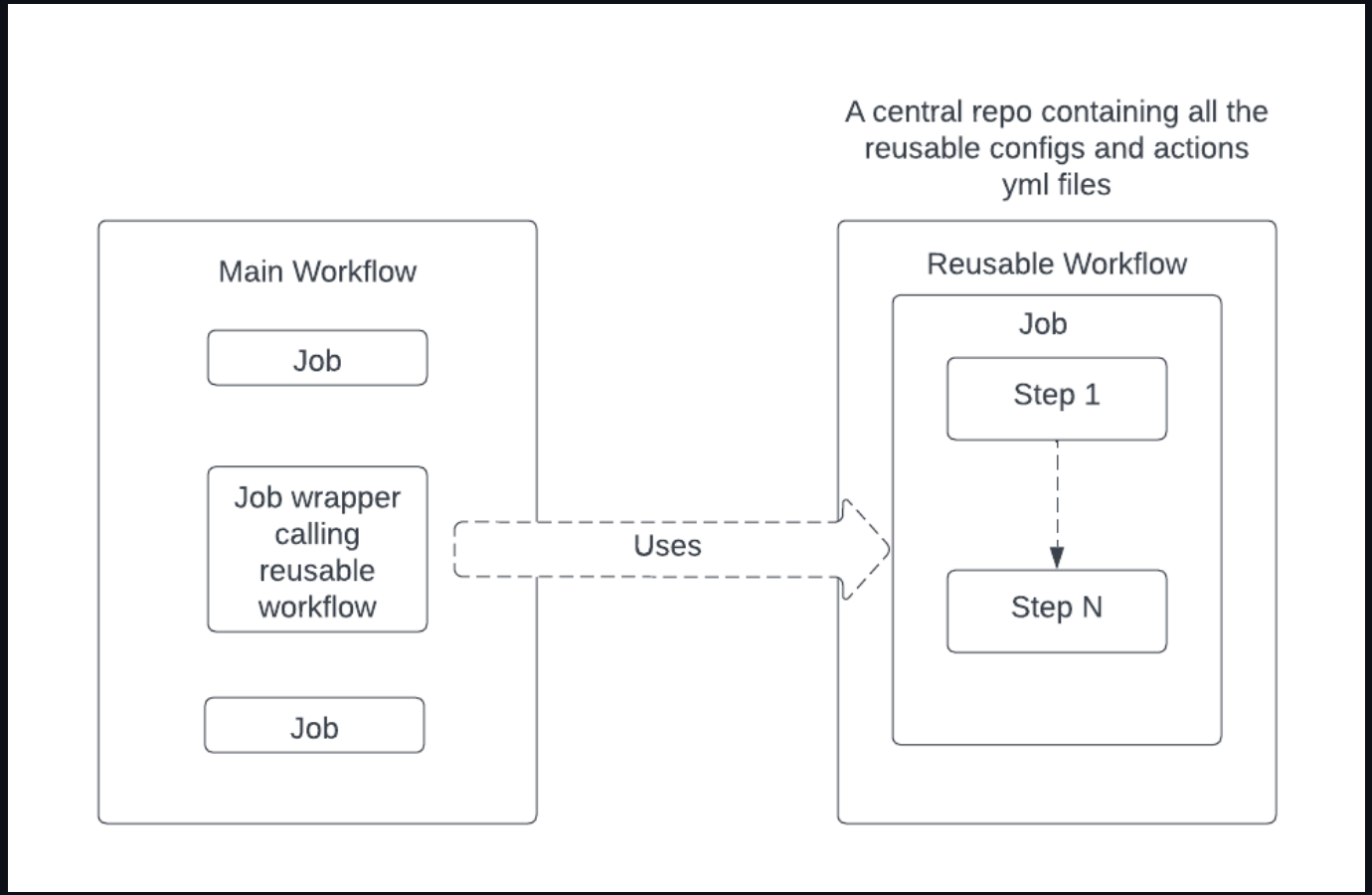
Code Scanning & Dependabot Rollout & Enforcements

- As we know CodeQL and dependabot supports a configuration based (yml) execution definition in GitHub, it may be a good discussion point to see how these configurations can be streamlined and reused to stop having configuration drifts across org and repos

- There are multiple ways to achieving reusability in GitHub and its associated features
  - **Use Reusable workflows**
  - **Use Required workflows**

- Maintaining a centralized repository, with all the CodeQL yml files dependabot yml files and having the calling repo call the centralized repository or using GitHub Repo rulesets to enforce a particular configuration on the repositories under a given Org can help streamline a standardized rollout configuration

Reusable Workflow for CodeQL



Reusable Workflow for CodeQL

```yaml
name: "CodeQL"

on:
  push:
    branches: [ "master" ]
  pull_request:
    # The branches below must be a subset of the branches above
    branches: [ "master" ]

  workflow_dispatch:

jobs:
  call-workflow-passing-data:
    uses: Reusable-Test/abhi-called-repo/.github/workflows/called-codeql-workflow.yml@main
    with:
      language: 'java'
      runner-type: 'ubuntu-latest'
      configuration-file: './.github/codeql/high-severity.yml'
```

Calling Workflow

abhi-called-repo / .github / workflows / **called-codeql-workflow.yml**

abhi-github-staff Update called-codeql-workflow.yml

Code | Blame | 122 lines (103 loc) · 4.77 KB

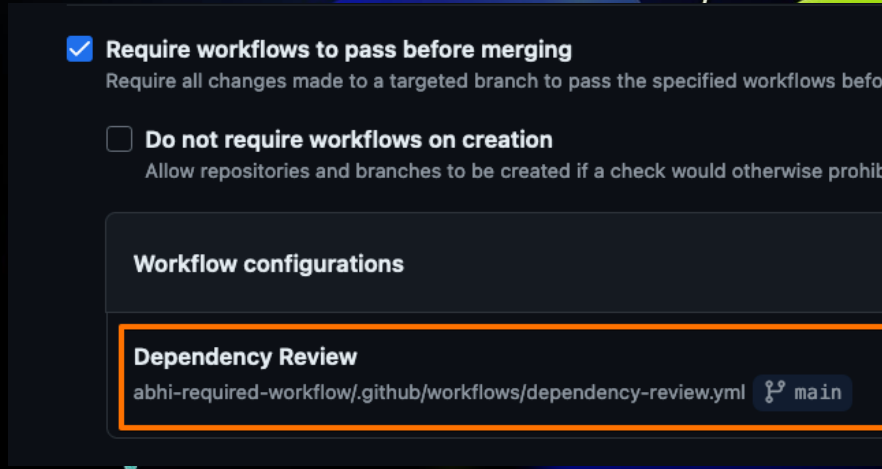
```yaml
name: Called workflow

on:
  workflow_call:
    inputs:
      language:
        required: true
        type: string
      runner-type:
        required: true
        type: string
      configuration-file:
        required: true
        type: string


jobs:
  analyze:
    name: Analyze
    # Runner size impacts CodeQL analysis time. To learn more, please see:
    #   - https://gh.io/recommended-hardware-resources-for-running-codeql
    #   - https://gh.io/supported-runners-and-hardware-resources
    #   - https://gh.io/using-larger-runners
    # Consider using larger runners for possible analysis time improvements.
    runs-on: ${{ inputs.runner-type }}
    timeout-minutes: ${{ (matrix.language == 'swift' && 120) || 360 }}
    permissions:
      # required for all workflows
```

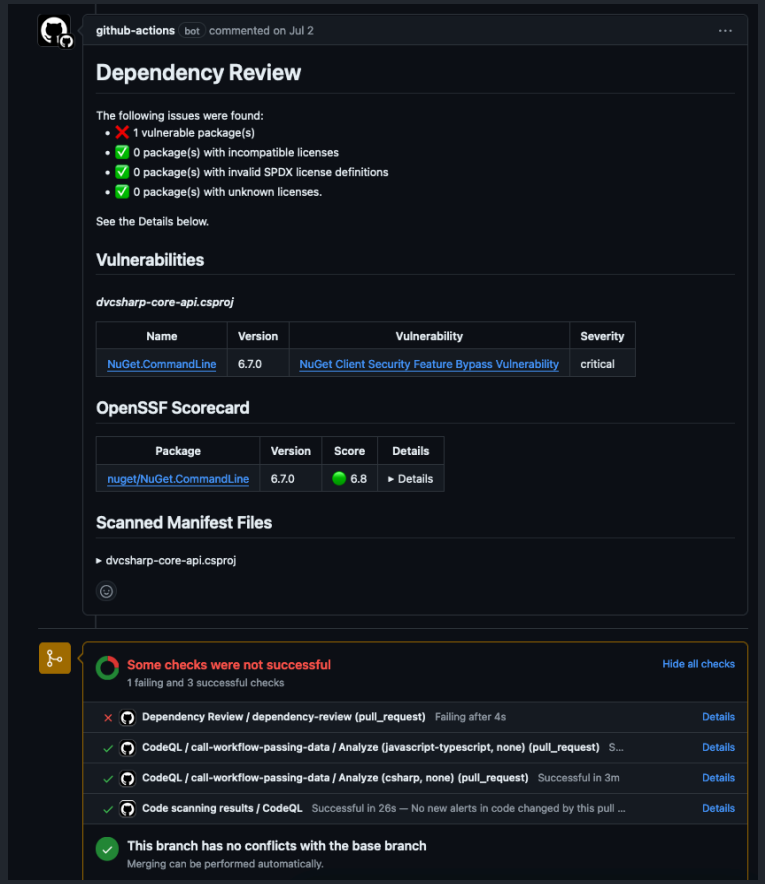
Called Reusable Workflow

Required Workflows

Another way to enforce a centralized configuration in GitHub across a set of Repositories or all repositories under an org in through the use of Required workflow (available under the repository rules in an Org or Repo level

Note: Enforcing the dependency-review.yml file in the example here will enforce dependency checks on every Pull Request





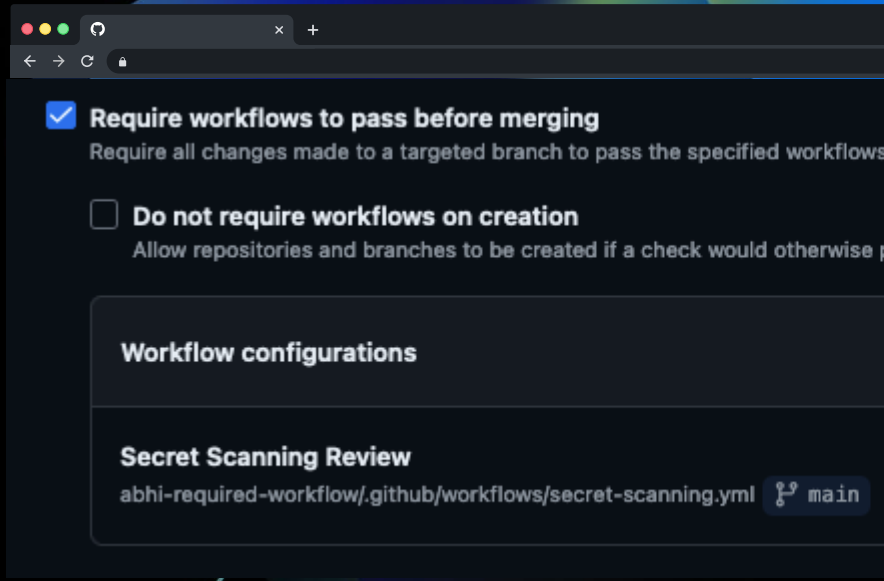
Dependency Review check enforced at the Pull Request Level based on the required workflow definition in the Repo Ruleset

Required Workflows

1. One can also enforce **Secret Scanning** checks at the Pull Request level by adding a secret-scanning.yml file (using secret scanning review action)

2. Once secret scanning yml is defined in the repo rules as part of the config “Require workflows to pass before merging”, the action will check if any new secret is leaked in the repository from the feature branches

3. Once you enforce this definition on all/selected repos under the organization, any net new secrets leaked into the repo, the PR will be blocked from merging in case of new vulnerability introduction



Workflow file for this run

.github/workflows/secret-scanning.yml at 1da9c50

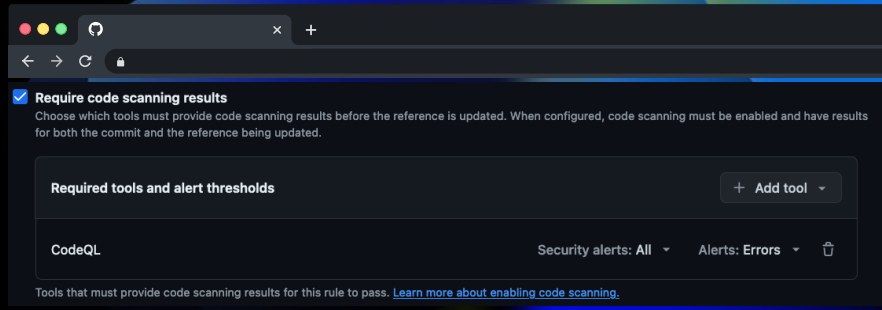
```yaml
name: 'Secret Scanning Review'
on: [pull_request]

jobs:
  secret-scanning-review:
    runs-on: ubuntu-latest
    steps:
      - name: 'Secret Scanning Review Action'
        uses: advanced-security/secret-scanning-review-action@v1
        with:
          token: ${{ secrets.SECRET_SCAN_REVIEW_GITHUB_TOKEN }}
          fail-on-alert: true
          fail-on-alert-exclude-closed: true
```

Typical secret-scanning.yml file with the policy definitions (Actions Link)

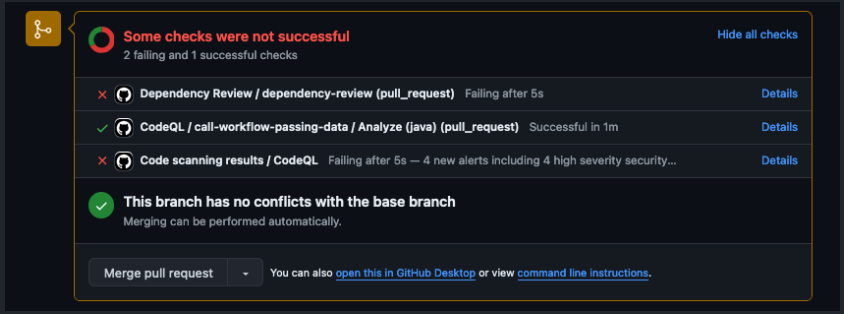
A PR level summary will be created with the list of all new secrets identified along with the PR level checks (failure in this case)

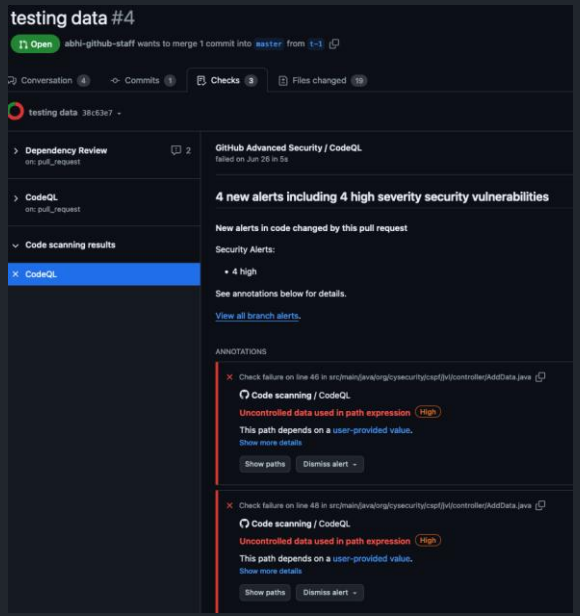
Required Workflows

Another way of **enforcing CodeQL** scans on every Pull Request is by configuring the : **Required code scanning results**
In the Repository ruleset configuration page in the GitHub UI



Note: As a Security Architect, one can define the permissible alerts threshold in the configuration and when a developer merging mode from their respective feature branches by creating a Pull Request to a protected branch, the configuration would check for any new vulnerability introduced. If new vulnerabilities are detected with the mentioned criticality, the PR would be blocked





CodeQL vulnerability threshold enforced at the Pull Request level through the Repo Rulesets

The net new vulnerability details available within the Pull Request

Best Practices and Considerations Checklist

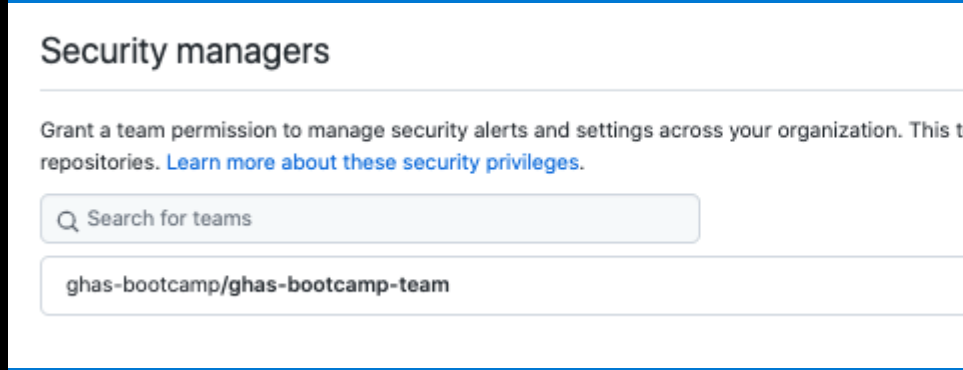
- Establish a branching strategy
- Manage repositories with custom properties
- Start with managing an inventory of the repositories by business criticality/language
- Use Security Configurations where possible
- Create Repository Rulesets for supplemental enforcement
- Consider your desired access permissions
- Create and monitor an exception process

Notifications and Alerts

Security Manager role

Members of a team with the security manager role have only the permissions required to effectively manage security for the organization.

- **Read** access on all repositories in the organization, in addition to any existing repository access
- **Write** access on all security alerts in the organization
- The ability to configure security settings at the organization level
- The ability to configure security settings at the repository level



Default Permissions

Repository action	Read	Triage	Write	Maintain	Admin
Receive Dependabot alerts for insecure dependencies in a repository	×	×	✓	✓	✓
Dismiss Dependabot alerts	×	×	✓	✓	✓
Designate additional people or teams to receive security alerts	×	×	×	×	✓
Create security advisories	×	×	×	×	✓
Manage access to GitHub Advanced Security features (see "Managing security and analysis settings for your organization")	×	×	×	×	✓
Enable the dependency graph for a private repository	×	×	×	×	✓
View dependency reviews	✓	✓	✓	✓	✓
View code scanning alerts on pull requests	✓	✓	✓	✓	✓
List, dismiss, and delete code scanning alerts	×	×	✓	✓	✓
View and dismiss secret scanning alerts in a repository	×	×	✓	✓	✓
Resolve, revoke, or re-open secret scanning alerts	×	×	✓	✓	✓
Designate additional people or teams to receive secret scanning alerts in repositories	×	×	×	×	✓

Notifications

Secret Scanning

- Incremental
- Historical
- Watch repository

Dependabot

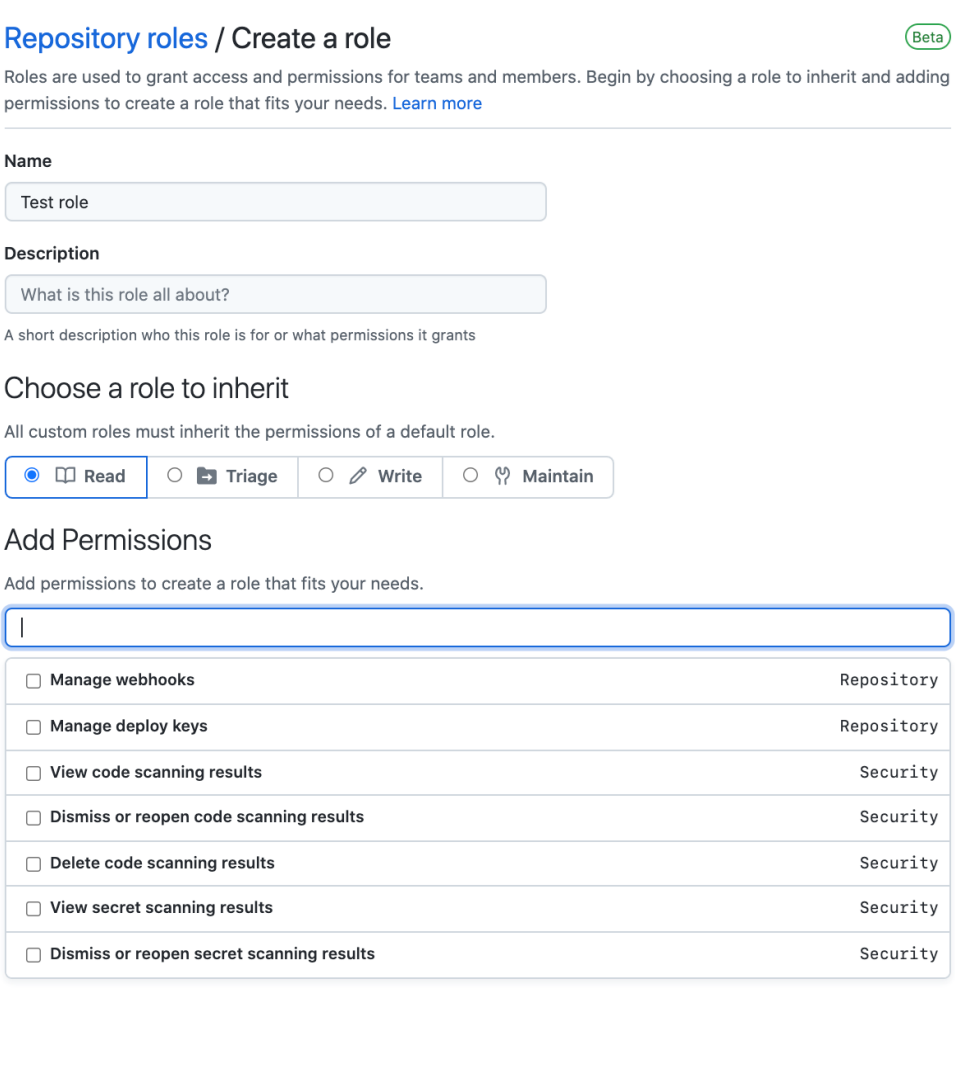
- Watch repository
- Communication channels
- Frequency

Code Scanning

- No traditional notifications
- Can be achieved via webhooks

Custom repository roles for security features

- When you create a custom repository role, you start by choosing an inherited role from a set of pre-defined options.
- The inherited role determines the initial set of permissions included in the custom role.
- Then, you can further customize the role by choosing additional permissions to give the role.



Configuring access - Demo

Reviewing Alerts Hands-on demo

Integrations - GHAS API and Webhooks

APIs

REST

/code-scanning (Enterprise, Organization & Repository)

/dependabot (Enterprise, Organization & Repository)

/dependency-graph (Repository)

/secret-scanning (Enterprise, Organization & repository)

GraphQL

DependabotUpdate

SecurityVulnerability

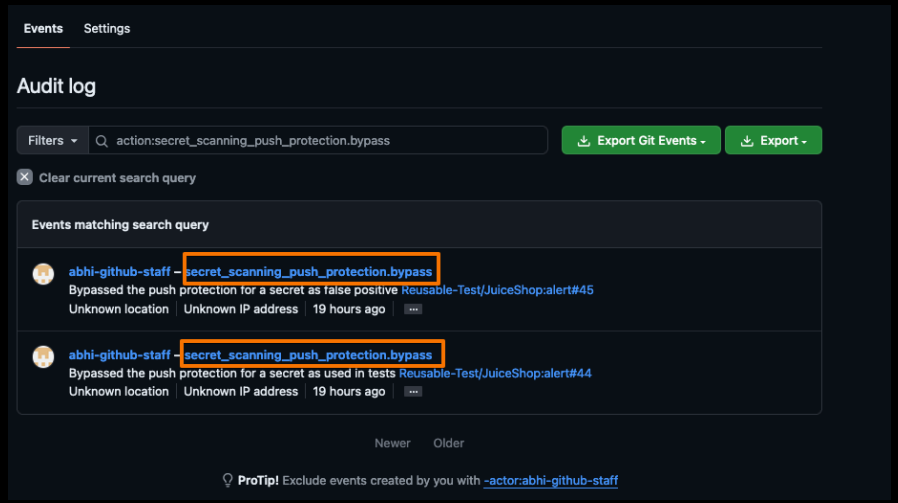
SecurityAdvisory

Preview:
DependencyGraph

Webhooks

1. *code_scanning_alert* (Repository, Organization, App)
2. *dependabot_alert* (Repository, Organization, App)
3. *secret_scanning_alert* (Repository, Organization, App)
4. *secret_scanning_alert_location* *(Repository, Organization, App)*
5. *security_advisory* *(App)*
6. *security_and_analysis* *(Repository, Organization, App)*

Audit log



1. *dependabot_**
2. *dependency_graph_**
3. *repository_secret_scanning**
4. *repository_vulnerability_**
5. **secret_scanning_**

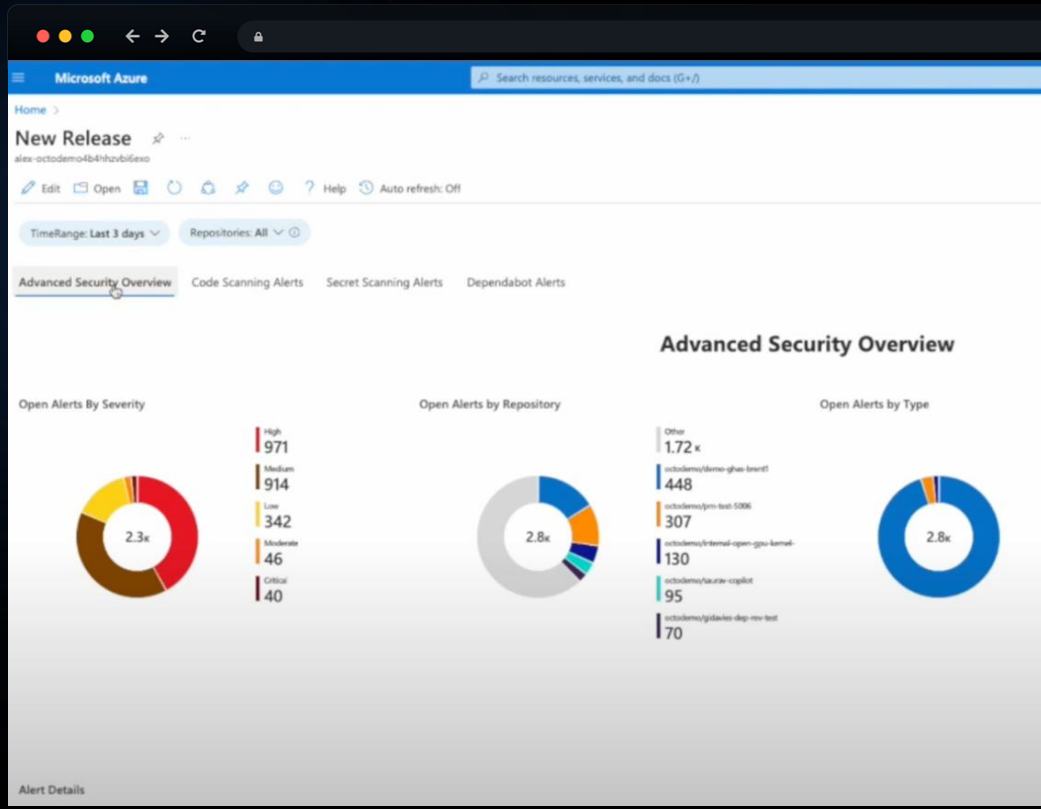
Monitoring and responding to alerts

Hands-on demo

3rd Party Integrations

SIEM Integrations

- Splunk
- Azure Sentinel
- Data Dog
- Sumo Logic
- Elastic



JIRA Integration with GHAS Vulnerabilities

JIRA now natively supports ingesting GHAS vulnerabilities into JIRA and subsequently creating issues in JIRA in an automated way

https://support.atlassian.com/jira-cloud-administration/docs/integrate-github-for-security/

https://www.youtube.com/watch?v=QA-1gjzSuYc

Integrate GitHub Advanced Security with Jira

These instructions are for connecting **GitHub Cloud** or **GitHub Enterprise Cloud** to Jira. Show me how to connect GitHub Enterprise Server

The security feature in Jira allows you to view, triage, and track security vulnerabilities from GitHub Advanced Security. To get this feature working, you'll need to:

1. Install the GitHub for Jira app.
2. Connect a GitHub organization.
3. Add GitHub Advanced Security to your Jira project.
4. Connect security containers to your project.

Before you begin

To install and set up the GitHub for Jira app, you need:

- Site administrator permission for your Jira site.
- Organization owner permission for your GitHub organization.

For some organizations, the task of integrating GitHub Advanced Security might involve multiple team members:

- A Jira site admin will install the GitHub for Jira app.
- A GitHub organization owner will connect a GitHub organization to your Jira site.
- A Jira project admin will add GitHub Advanced Security to a project and connect security containers.

3rd Party Scanner Integration and Data Ingestion

Code | Blame | 43 lines (37 loc) · 1.09 KB

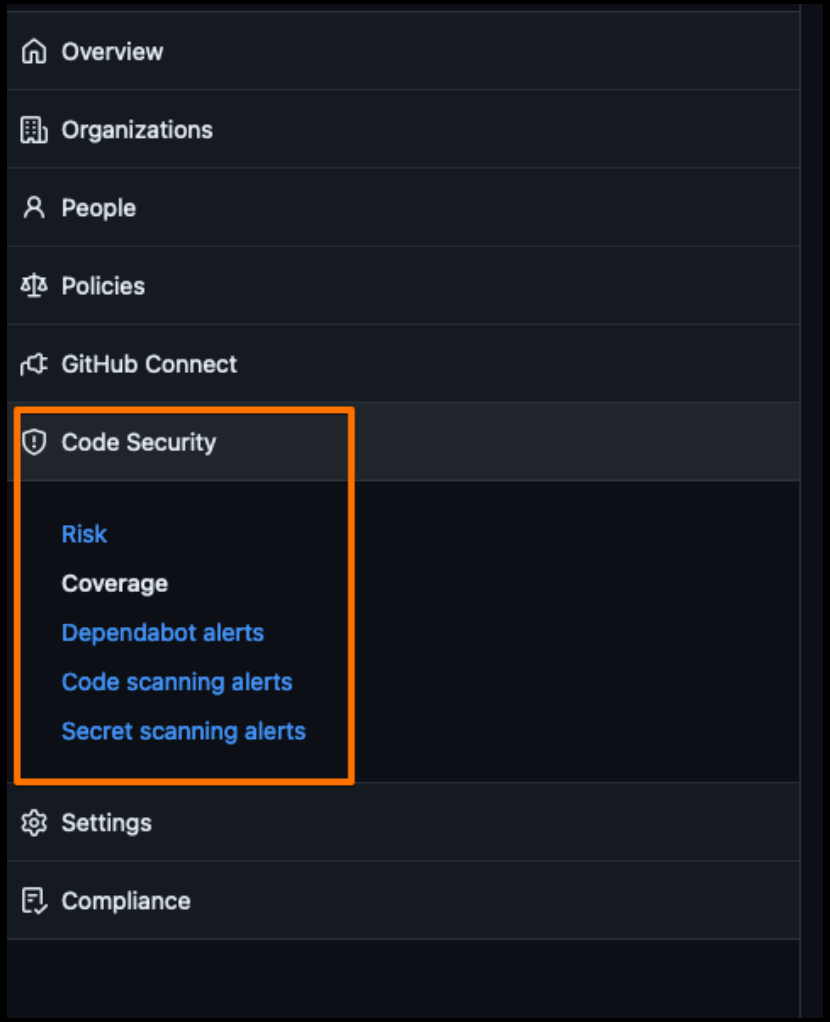
```yaml
name: scan with KICS and upload SARIF

on:
  push:
    branches: [master]

jobs:
  kics-job:
    runs-on: ubuntu-latest
    name: kics-action
    strategy:
      fail-fast: false
    steps:
      - name: Checkout repo
        uses: actions/checkout@v3
      - name: Mkdir results-dir
        # make sure results dir is created
        run: mkdir -p results-dir

      - name: Run KICS Scan with SARIF result
        uses: checkmarx/kics-github-action@v2.1.0
        with:
          path: 'terraform'
          output_path: results-dir
          platform_type: terraform
          output_formats: 'json,sarif'
          ignore_on_exit: results

      - name: Show results
        run: |
          cat results-dir/results.sarif
          cat results-dir/results.json

      - name: Archive code coverage results
        uses: actions/upload-artifact@v4
        with:
          name: result
          path: results-dir/results.sarif

      - name: Upload SARIF file
        uses: github/codeql-action/upload-sarif@v1
        with:
          sarif_file: results-dir/results.sarif
```

1. GitHub Code Scanning allows 3rd party scanner data ingestion
2. Output data format should be SARIF
3. If running the scan (ex. IaC) in GitHub Actions, use **upload-sarif**
4. Upload can also be done through REST API
5. Vulnerabilities can be viewed in Security Dashboard in GitHub

Security Overview

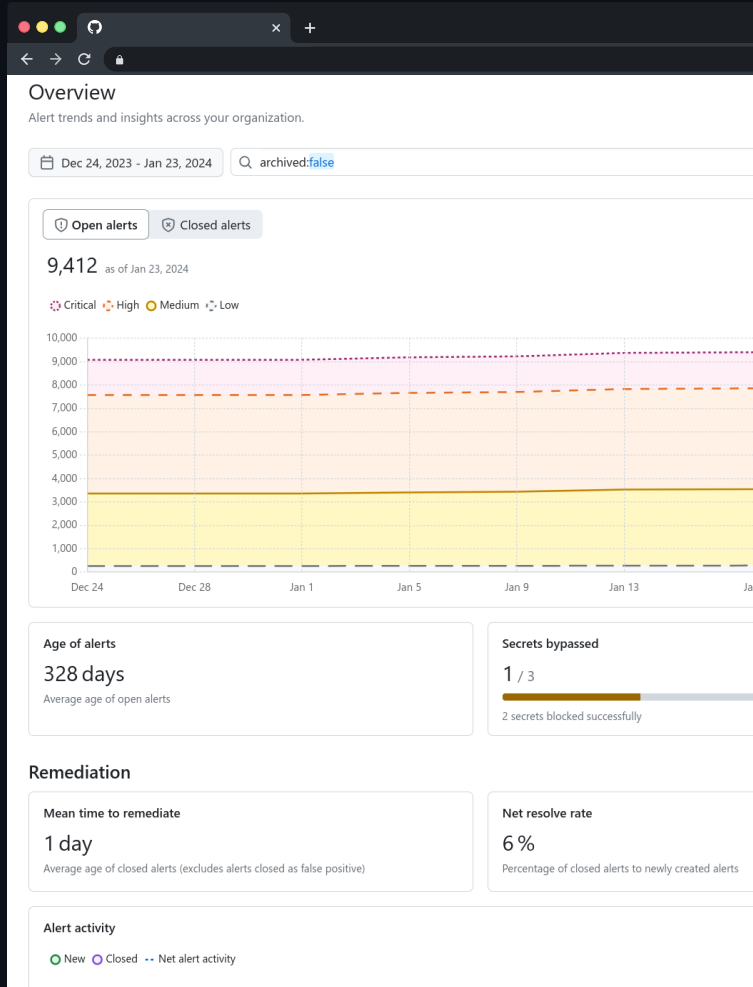
Enterprise Level View

- You can find security overview on the Code Security tab for your enterprise. Each page displays aggregated and repository-specific security information for your enterprise.
- In the enterprise-level security overview, you can see data for all organizations where you are an organization owner or security manager.
- As a security team member, you cannot use the enterprise-level security overview to enable and disable security features



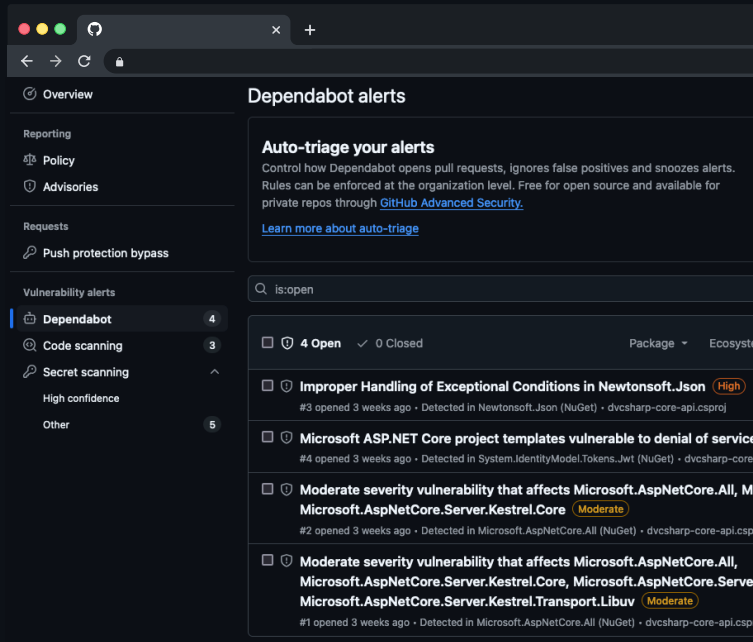
Org Level View

- Provides high-level summaries of the security landscape of an organization or enterprise and makes it easy to identify repositories that require intervention
- Shows metrics on default branches
- At organisation level, 3 summary views:
  - Overview
  - Coverage
  - Risk
- Show data only for high confidence alerts. Code scanning alerts from third-party tools, and secret scanning alerts for ignored directories and non-provider alerts are all omitted from these views



Repo Level View

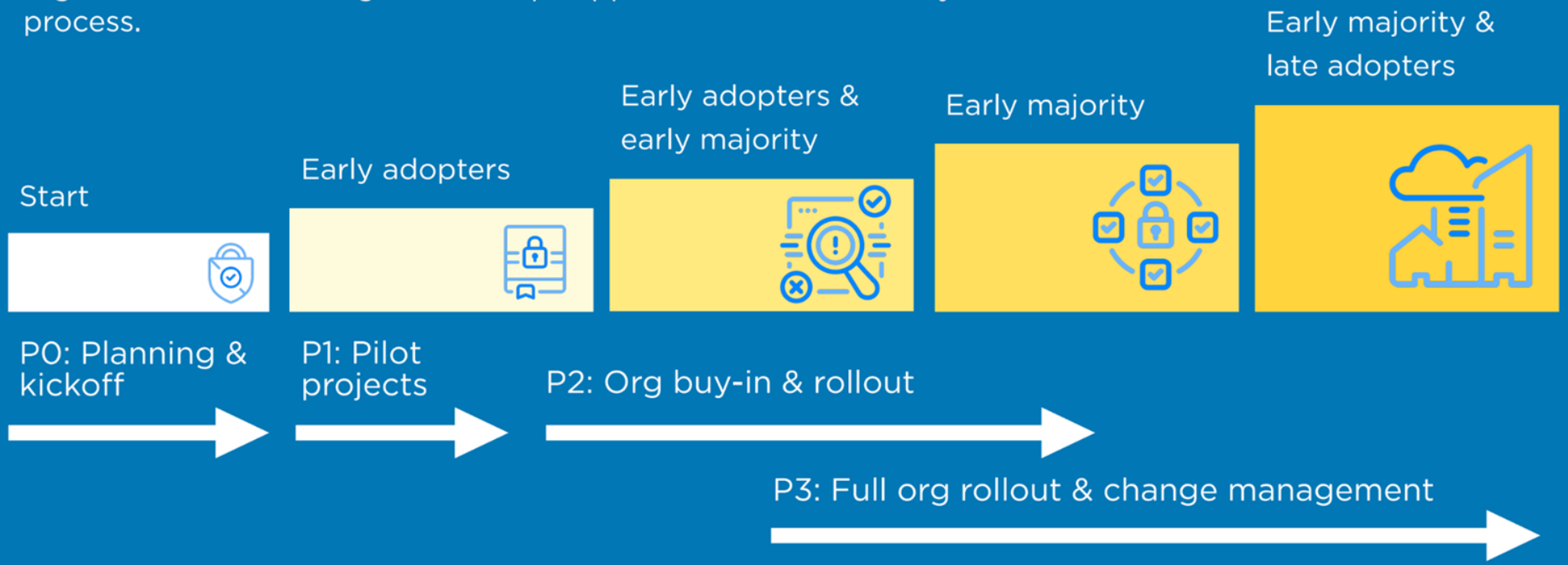
- At the repo level, drill down at the vulnerabilities across the 3 pillars of GHAS
  - Dependabot
  - Code Scanning
  - Secret Scanning
    - High Confidence
    - Other



Scaling GitHub Advanced Security

GHAS phased rollout & deployment

Throughout the phases of the GHAS rollout, groups within your organization can be targeted to help support the rollout and buy-in process.



Resources/Appendix

Docs/Resources

- Whitepaper - keys to a successful rollout
- Adopting and scaling GHAS in your company
- GHAS Enablement automation
- GitHub Professional Services
- Advanced-security repo - a collection of tools for managing GHAS
- GitHub Skills - a GitHub learning platform
- GitHub Advanced Security learning path
- Code scanning API
- Secret scanning API
- CodeQL Repository

Thank you