GitHub Integration(Elastic)

GitHub Integration

Introduction

Elastic’s GitHub integration allows you to ingest GitHub logs, alerts, and developer activities into the Elastic Stack for centralized analysis. This supports use cases like vulnerability management, compliance auditing, and DevSecOps monitoring.

Note: This integration is only compatible with **GitHub Enterprise Cloud** and is **not supported on GitHub Enterprise Server**.

GitHub Data Streams Overview

Feature	Integration Type	Description
Audit Logs	PAT	Track org-level admin and security events
Code Scanning	GitHub App / PAT	Pull static analysis results from GitHub Advanced Security
Secret Scanning	GitHub App / PAT	Detect exposed secrets (API keys, tokens, etc.) in repositories
Dependabot Alerts	GitHub App / PAT	Retrieve alerts on insecure open-source dependencies
Issues & PRs	GitHub App / PAT	Sync issues, pull requests, comments, labels, and milestones

---

Option 1: GitHub Audit Logs

Description:
Audit logs contain records of all administrative and security events within a GitHub organization.

Requirements

- GitHub Enterprise Cloud
- You must be an organization owner
- Use a Personal Access Token (PAT) with `read:audit_log` scope

What It Does

- Captures repository creation, permission changes, team updates, and more
- Helps detect suspicious or non-compliant behavior

Setup Steps

1. **Create a PAT**
   - Go to GitHub → Developer Settings → Personal Access Tokens
   - Click "Generate new token"
   - Select `read:audit_log` scope
   - Save the token securely
2. **Configure Integration in Elastic**
   - Navigate to Integrations in Kibana
   - Search for "GitHub" and click "Add GitHub integration"
   - Select the "Audit Logs" data stream
   - Enter your organization name and paste your PAT
3. **Test and Deploy**
   - Click "Test integration" to verify connectivity
   - Choose a data stream name and index settings
   - Click "Save and Deploy"
4. **Verify in Kibana**
   - Navigate to Discover
   - Use the index pattern `logs-github.audit-*`
   - Filter using fields such as `actor`, `action`, or `created_at`

Option 2: Code Scanning Alerts

Description:
Collect static code analysis results from GitHub Advanced Security Code Scanning.

Requirements

- Code Scanning must be enabled per repository
- Use either:
  - GitHub App with `security_events` read permission
  - PAT with:
    - `security_events` (for private repositories)
    - `public_repo` (for public repositories)

What It Does

- Ingests vulnerabilities and insecure code patterns
- Supports SARIF format scan results

Setup Steps

1. **Enable Code Scanning in GitHub**
   - Go to your repository → Security → Code scanning alerts
   - Enable GitHub Advanced Security
   - Configure workflows such as CodeQL
2. **Generate PAT or GitHub App**
   - If using a PAT, ensure it includes `security_events` or `public_repo` scope
3. **Configure Integration in Elastic**
   - Open Integrations in Kibana
   - Add GitHub integration and select "Code Scanning"
   - Input organization name and credentials
4. **Test and Configure**
   - Test the integration
   - Set polling frequency (e.g., every 5 minutes)
   - Save and deploy
5. **Monitor in Kibana**
   - Use Discover with the index pattern `logs-github.code_scanning-*`
   - Filter by fields such as `severity`, `rule_id`, or `repository.name`

Option 3: Secret Scanning Alerts

Description:
Detect and alert on exposed secrets in source code repositories.

Requirements

- Secret Scanning must be enabled in repository settings
- You must be a repository or organization administrator
- Use either:
  - GitHub App with `secret_scanning_alerts` read permission
  - PAT with:
    - `repo` or `security_events` (for private repos)
    - `public_repo` (for public repos)

What It Does

- Flags exposed API keys, tokens, and credentials
- Helps prevent credential leaks

Setup Steps

1. **Enable Secret Scanning**
   - Go to GitHub repo → Settings → Code Security and Analysis
   - Enable "Secret scanning alerts"
2. **Generate Access**
   - Create a PAT with appropriate scopes
   - Or set up a GitHub App with necessary permissions
3. **Configure in Elastic**
   - Go to the GitHub integration in Kibana
   - Enable the "Secret Scanning" stream
   - Provide token and repository/org details
4. **Test and Save**
   - Test the connection
   - Select desired polling interval (e.g., 10 minutes)
   - Save and deploy
5. **Analyze Alerts**
   - Open Discover and use `logs-github.secret_scanning-*`

- Use filters such as `alert_type`, `secret_type`, and `state`

Option 4: Dependabot Alerts

Description:
Monitor dependency vulnerabilities in GitHub repositories using Dependabot.

Requirements

- Dependabot must be enabled in repository settings
- You must be a repository or organization administrator
- Use either:
  - GitHub App
  - PAT with:
    - `repo`, `security_events`, or `public_repo` scope

What It Does

- Identifies and alerts on known insecure packages
- Includes CVE metadata and suggested fixes

Setup Steps

1. **Enable Dependabot in GitHub**
   - Go to Repository → Settings → Code Security and Analysis
   - Enable "Dependency Graph" and "Dependabot alerts"
2. **Generate GitHub App or PAT**
   - Ensure scopes include `repo`, `security_events`, or `public_repo`
3. **Configure in Elastic**
   - Go to GitHub integration
   - Enable "Dependabot"
   - Enter org/repo and credentials
4. **Test and Deploy**
   - Test the integration
   - Select polling interval
   - Save settings
5. **Monitor in Kibana**
   - Use Discover → `logs-github.dependabot-*`
   - Filter by `dependency_name`, `ecosystem`, `severity`, etc.

Option 5: Issues & Pull Requests

Description:
Ingest GitHub issues, pull requests, comments, labels, milestones, and other metadata.

Requirements

- Use a GitHub App or PAT with:
  - `repo` (for private repositories)
  - `public_repo` (for public repositories)
  - Optional: `read:org` for org-wide access

What It Does

- Collects all issue and PR activity
- Enables filtering of pull requests with `github.issues.is_pr = true`

Setup Steps

1. **Create or Use PAT / GitHub App**
   - Ensure appropriate access to repositories
2. **Enable GitHub Integration in Elastic**
   - Choose "Issues" as the data stream
   - Enter credentials and repository/organization name
3. **Customize Settings**
   - Set state filter (e.g., `state=open` for open issues only)
   - Configure sync interval
4. **Test and Activate**
   - Verify GitHub API connectivity
   - Deploy integration
5. **View Data in Kibana**
   - Go to Discover → `logs-github.issues-*`
   - Use filters such as `assignees`, `labels`, `state`, or `is_pr`

Comparison Table

Feature	GitHub App	PAT Support	Required Scopes	Public Repos	Private Repos
Audit Logs	No	Yes	`read:audit_log`	No	Yes
Code Scanning	Yes	Yes	`security_events`, `public_repo`	Yes	Yes
Secret Scanning	Yes	Yes	`repo`, `security_events`, `public_repo`	Yes	Yes
Dependabot Alerts	Yes	Yes	`repo`, `security_events`, `public_repo`	Yes	Yes
Issues & PRs	Yes	Yes	`repo`, `public_repo`, `read:org`	Yes	Yes

Documentation References

- Elastic GitHub Integration: Cytech Docs
- GitHub Official Docs:
  - Audit Logs
  - Code Scanning
  - Secret Scanning
  - Dependabot
  - Issues API

Revision #1
Created 25 June 2025 13:31:25 by Kent Lauron
Updated 25 June 2025 13:33:37 by Kent Lauron