How to use GitHub Actions with security in mind

NDC { Security }

https://myoctocat.com

@robbos81

How to use GitHub Actions with security in mind

NDC { Security }

Rob Bos
DevOps Consultant – Xpirit
The Netherlands
https://devopsjournal.io

@robbos81

https://myoctocat.com

What are GitHub workflows?

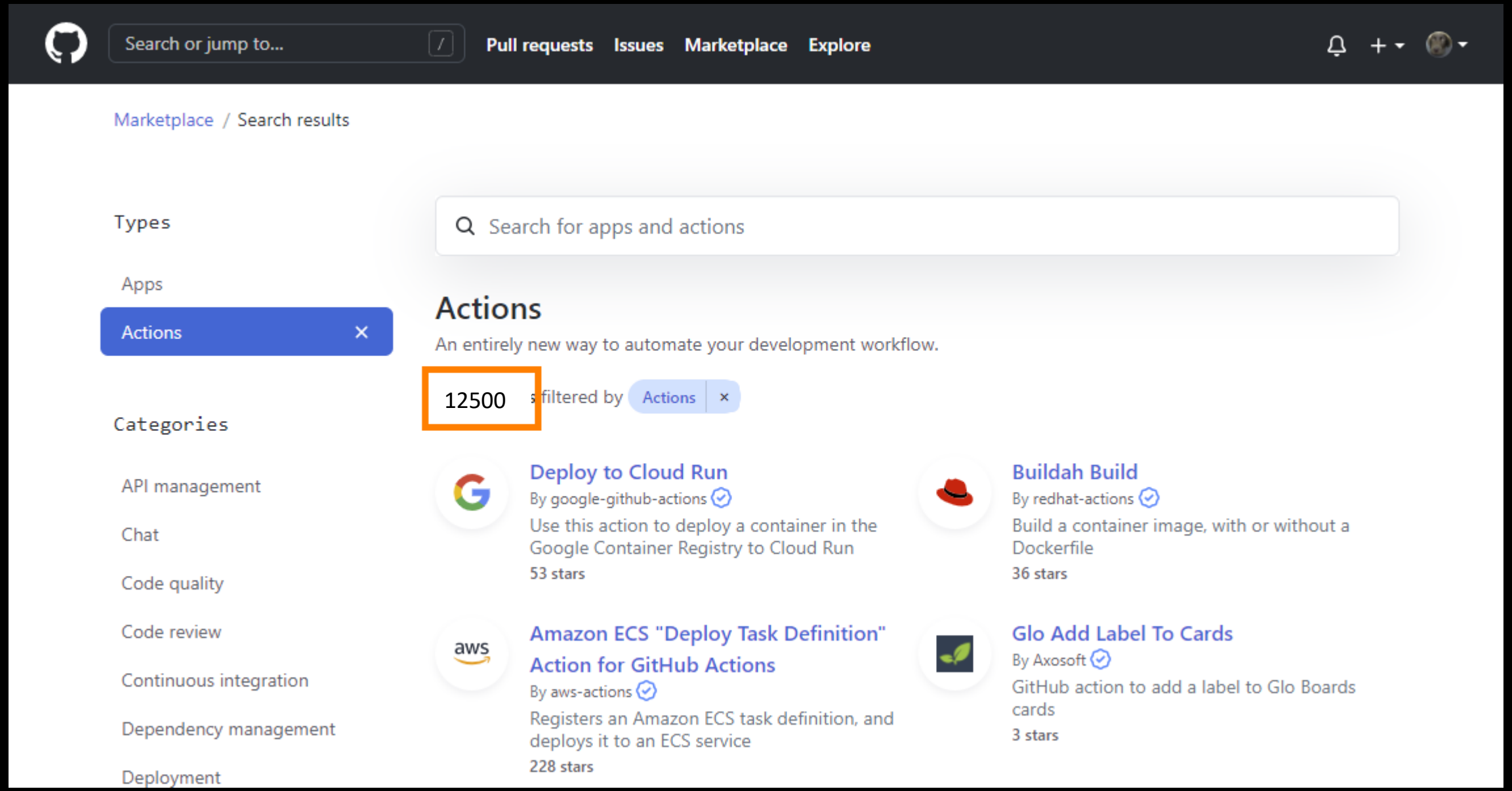
Execute one or more Actions

Workflows triggered by events:

- Push
- Comment
- Creating an Issue
- Release
- Etc.

What are GitHub Actions?

- Steps in the workflows
- Basis: Run a shell script
- Create your own
- Use an existing one from the marketplace


Search or jump to...
/
Pull requests
Issues
Marketplace
Explore
Marketplace / Search results
Types
Apps
Actions
Categories
API management
Chat
Code quality
Code review
Continuous integration
Dependency management
Deployment
Search for apps and actions
Actions
An entirely new way to automate your development workflow.
12500
filtered by
Actions
Deploy to Cloud Run
By google-github-actions
Use this action to deploy a container in the Google Container Registry to Cloud Run
53 stars
Buildah Build
By redhat-actions
Build a container image, with or without a Dockerfile
36 stars
Amazon ECS "Deploy Task Definition" Action for GitHub Actions
By aws-actions
Registers an Amazon ECS task definition, and deploys it to an ECS service
228 stars
Glo Add Label To Cards
By Axosoft
GitHub action to add a label to Glo Boards cards
3 stars

Workflow example

main

dotnetcore-webapp / .github / workflows / dotnetcore.yml

```yaml
name: .NET Core

on: [push]

jobs:
  build-and-deploy:
    environment: Production

    runs-on: ubuntu-latest

    steps:
    - uses: actions/checkout@v1
    - name: Setup .NET Core
      uses: actions/setup-dotnet@v1
      with:
        dotnet-version: 3.0.100

    # dotnet build
    - name: Build with dotnet
      run: |
        dotnet build --configuration Release ./dotnet-core-webapp/dotnetcore-webapp.csproj
```

GitHub Actions Security

- Repository security
- Runners and security
- Actions and security

- Forking actions
- Keeping up to date

Repository security

- Access to code
- Workflow secrets
- Your code

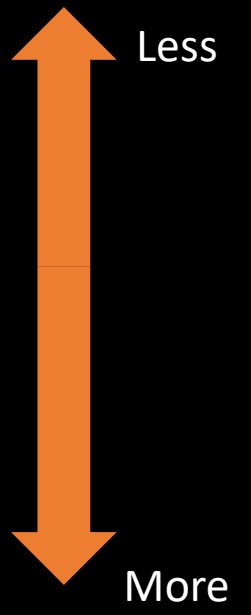
Code – Who has access?

Access levels can be set at:

- Repository
- Organization
- Enterprise

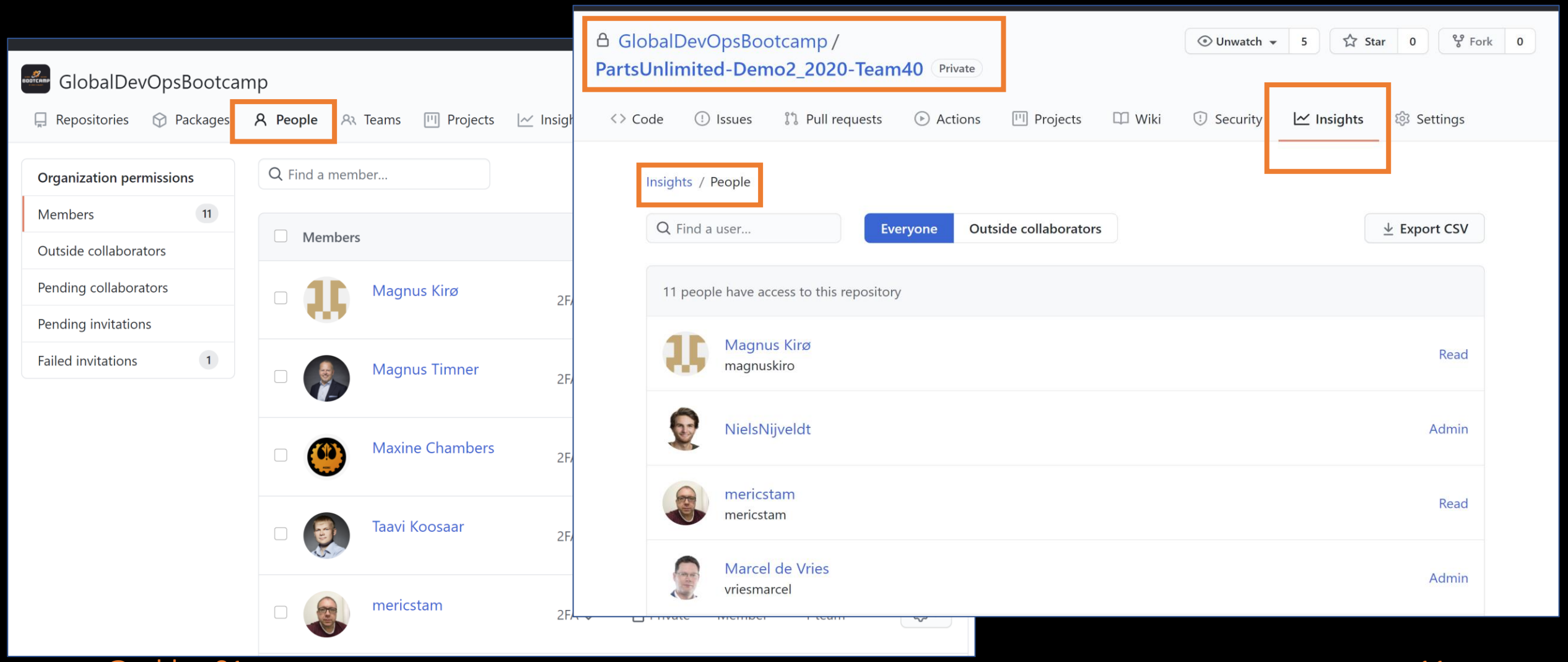
Code – Who has access?

Permission levels

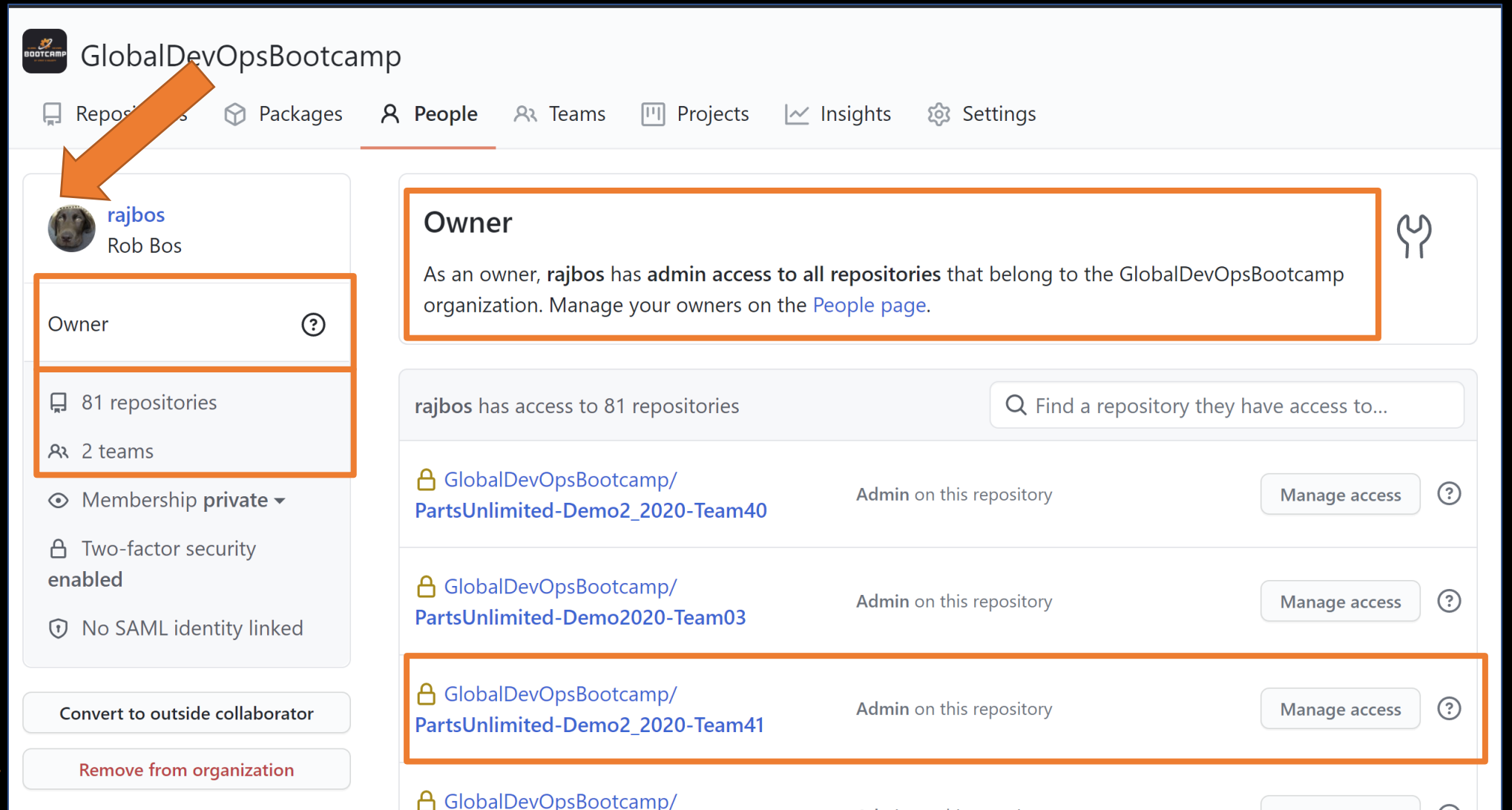


No access
Read only access
Triage: manage issues & pull requests
Write access
Maintain: No sensitive or destructive actions
Admin: full access

Configuring access



From the user

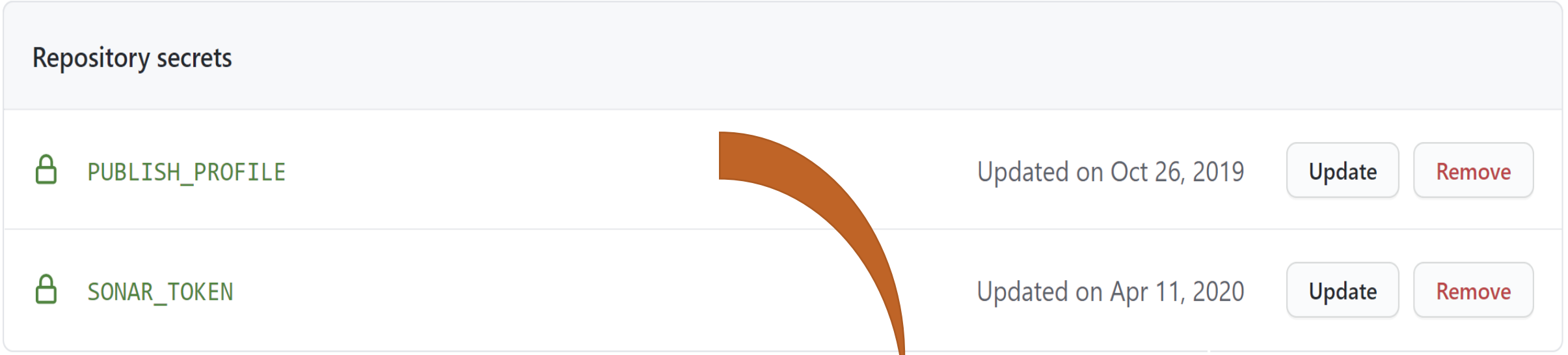


Repository security

- Access to code
- Workflow secrets
- Your code

Workflow secrets

@robbos81




```
41
42      # publish to Azure App Service
43      - name: 'Run Azure webapp deploy action using publish profile cred ntials'
44        uses: azure/webapps-deploy@v2
45        with:
46          app-name: dotnetcorewebapp19 # Replace with your a
47          publish-profile: ${{ secrets.publish_profile }} #  ine secret variable in repository settings as per action documentation
48          package: './dotnetcorewebapp'
49
```

Workflow secrets

Encrypted client side before reaching GitHub:

- Encrypted with the public key for your org or repo (created and stored by GitHub)
- Used when using the UI
- Encrypt yourself before posting to the REST API

Secrets are not shared to forked repositories

Who has access to your secrets?

For creating at repo level: Repository Owner access

For creating at org level: Admin access to the org

Set an access policy for the secrets:

- All repositories
- Private repositories
- Only selected repositories

Who has access to your secrets?

Encrypted until used, then injected as:

- An environment variable
- Direct input

Will be redacted in logs

Don’t use structured data (like json): hard to redact

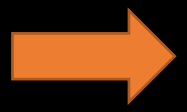
Who has access to your secrets?

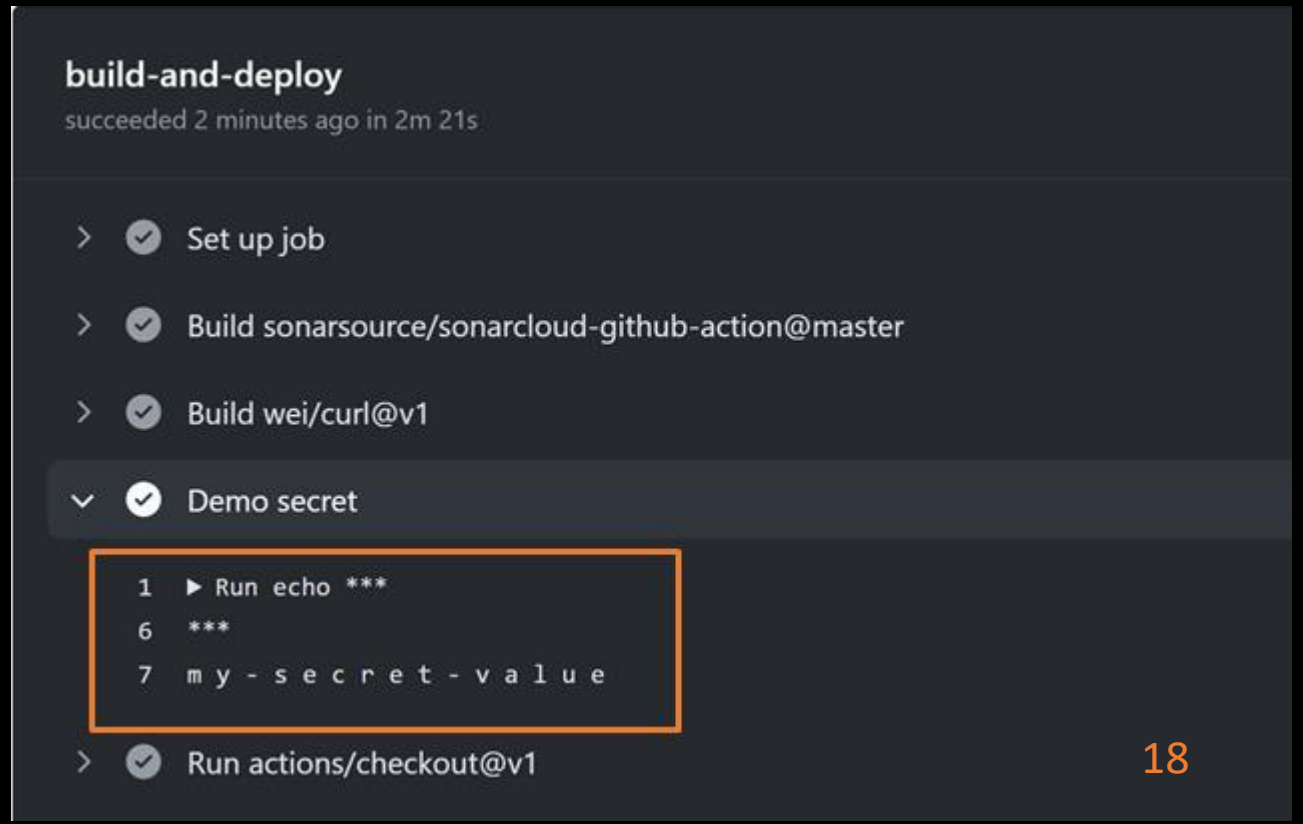
- Actions can do anything with them!
- Anyone with access to the Action Logs should be considered to have access to your secrets

```yaml
jobs:
  build-and-deploy:

    runs-on: ubuntu-latest

    steps:
    - name: Demo secret
      run: |
        echo ${{ secrets.DEMO_LOG }}
        echo ${{ secrets.DEMO_LOG }} | sed 's/./& /g'
```





Repository security

- Access to code
- Workflow secrets
- Your code/repo

Your code

Anything in your repository:

- Workflow files
- Shell scripts
- Your own code
- Dependencies:
  - Packages
  - Containers

Best practices:

- Static code analysis
  - Check your own code!
- Third party dependency scanning
  - 99% of your code, is not yours:
    - Scan for known vulnerabilities
  - Keep your dependencies up to date!

Your code – Best practices

Preventing changes by a single person → Pull Request

Protect your main branch

Branch protection rule:

- 1 approver (4 eyes / 2 person principle)
- Build validation

Your code/repo – trace changes

Who made changes:

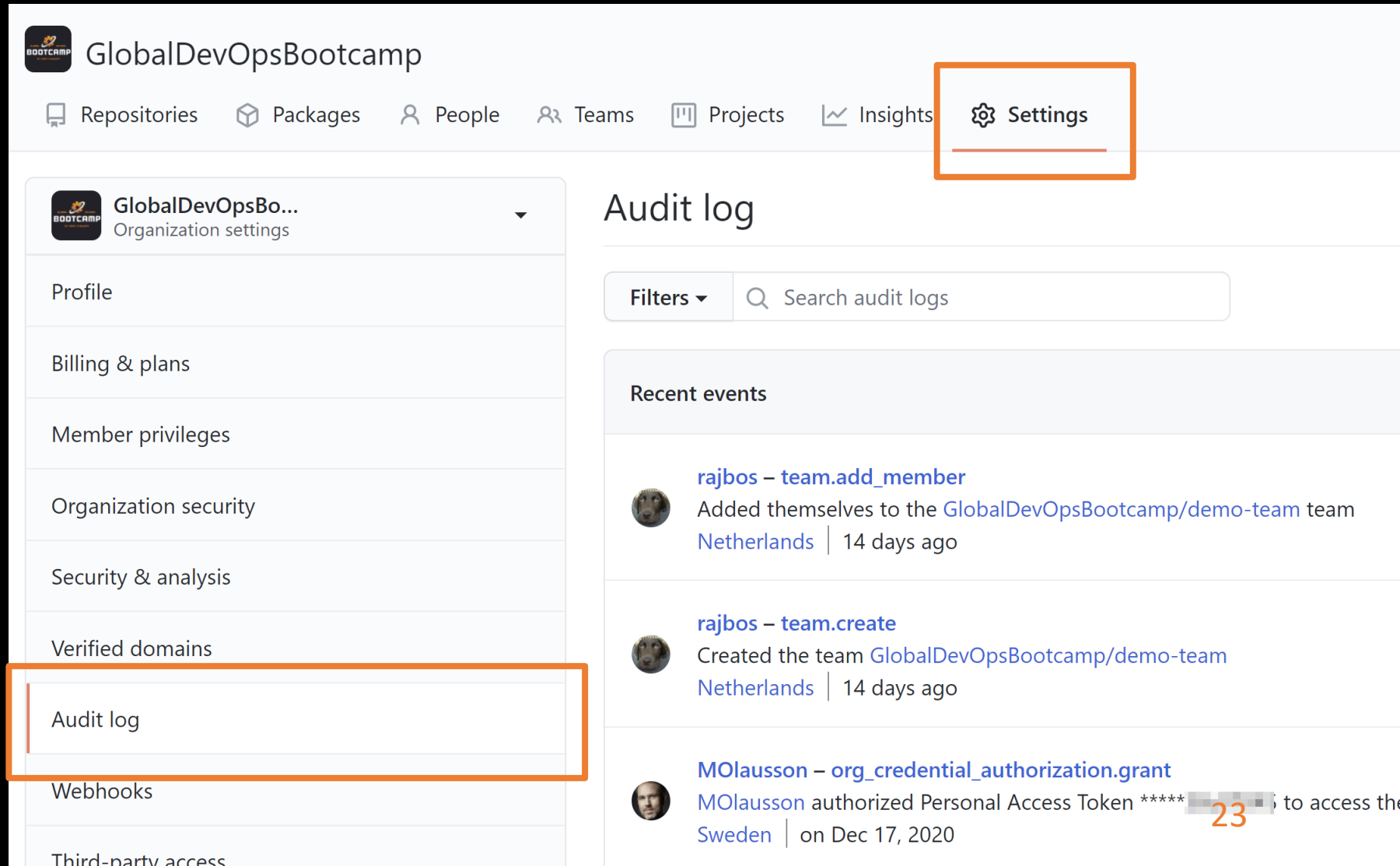
- Code: Git commit history

- Everything around your code is in the audit log

Your code/repo – trace changes (org level)

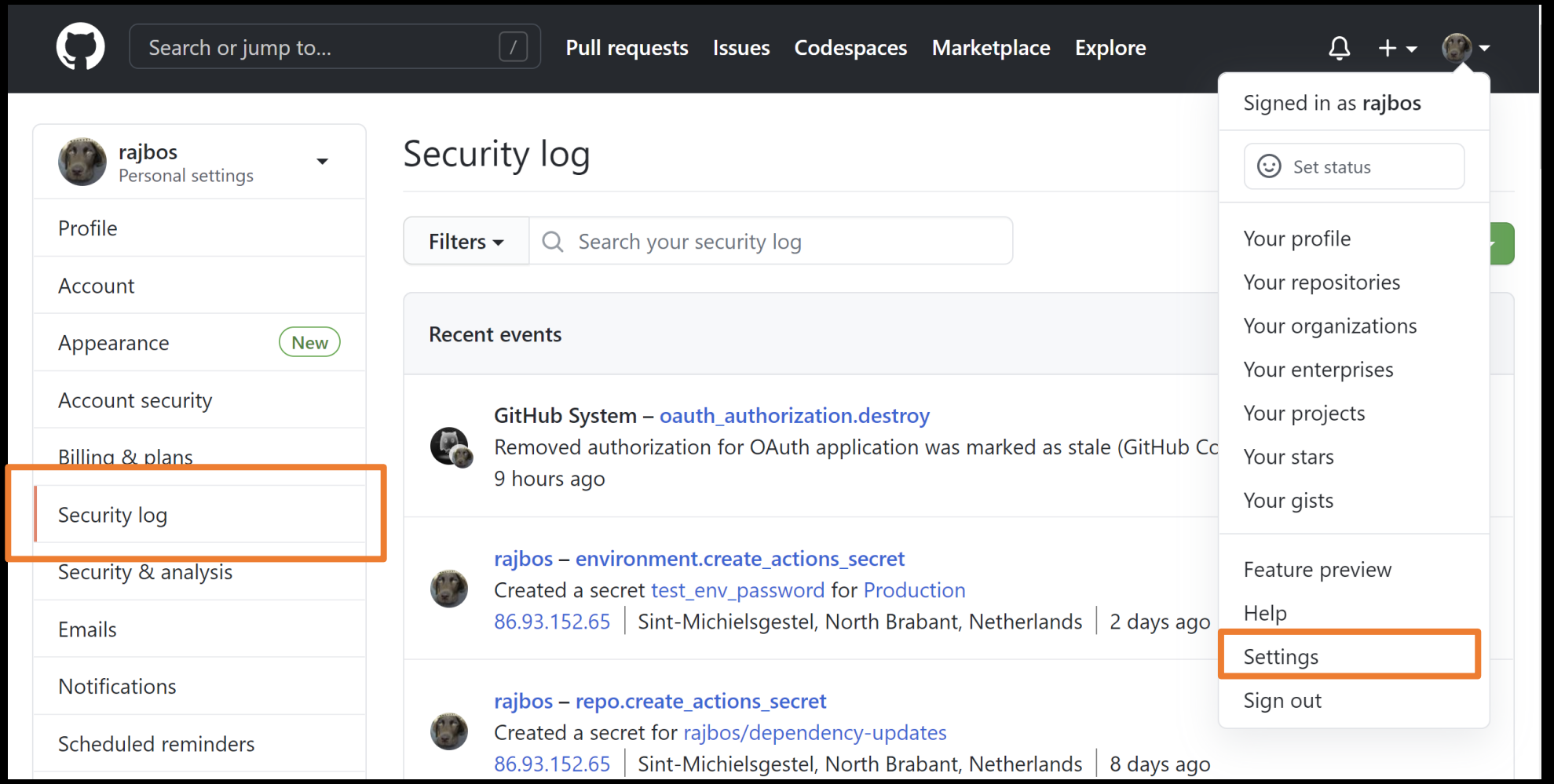
Audit log:

- Access
- Secrets
- Access Tokens
- OAuth grants
- Enabling features
- Etc.



Your code/repo – trace changes

Account level:



GitHub Actions Security

- Repository security
- Runners and security
- Actions and security

- Forking actions
- Keeping up to date

Workflow Runners

Actions execute on runners

Self hosted

- Cloud / On premises hosted by yourself
- OS + Tools update = YOUR responsibility
- Enables specific environment setup
- No usage limits

GitHub hosted

- OS + Tools update = GitHub's responsibility
- Per minute rating applies after the free minutes
- Clean execution environment with every run

```yaml
name: .NET Core Deploy to IIS

on:
  push:
    branches:
      - "self-hosted"

jobs:
  build-and-deploy:

    runs-on: self-hosted

    steps:
    - uses: actions/checkout@v1
    - name: Setup .NET Core
      uses: actions/setup-dotnet@v1
      with:
        dotnet-version: 3.0.100
```

```yaml
name: .NET Core

on: [push]

jobs:
  build-and-deploy:

    runs-on: ubuntu-latest

    steps:
    - uses: actions/checkout@v1
    - name: Setup .NET Core
      uses: actions/setup-dotnet@v1
      with:
        dotnet-version: 3.0.100
```

Workflow Runners

Security

- Environment scope
  - Network
  - Shared state between runs
- User: limit its access!

Best practice: Run the action inside of a container

```yaml
jobs:
  my_first_job:
    steps:
      - name: My first step
        uses: docker://gcr.io/cloud-builders/gradle
```

```yaml
jobs:
  test-box:
    runs-on: ubuntu-latest
    container:
      image: azul/zulu-openjdk-alpine:8-jre
    steps:
    - uses: actions/checkout@v2
    - name: What OS is running
      run: uname -a
    - name: What java version do we have
      run: java -version
```

Workflow runners

Best practice: DO NOT EVER use self hosted runners for public repositories

Example:

- Your repo
- New fork
- Adds malicious code
- Create pull request to your repo
- Workflow is executed on your self hosted runner?

Persisting data between runs

Run 1:

- Download dependencies
- Build the code
- Somehow overwrite the dependency cache

Run 2:

- Use cached dependencies
- Build the code
- Malicious dependency in build artefact

Solarwind attack:
https://xpir.it/Solorigate

Workflow runners – Best practice

Don't share runners (and machines!) between repositories:

- Run 1 can influence Run 2

Risks:

- Malicious programs
- Escaping the runner sandbox
- Exposing access to the (network) environment
- Persisting unwanted or dangerous data

GitHub Actions Security

Repository security

Runners and security

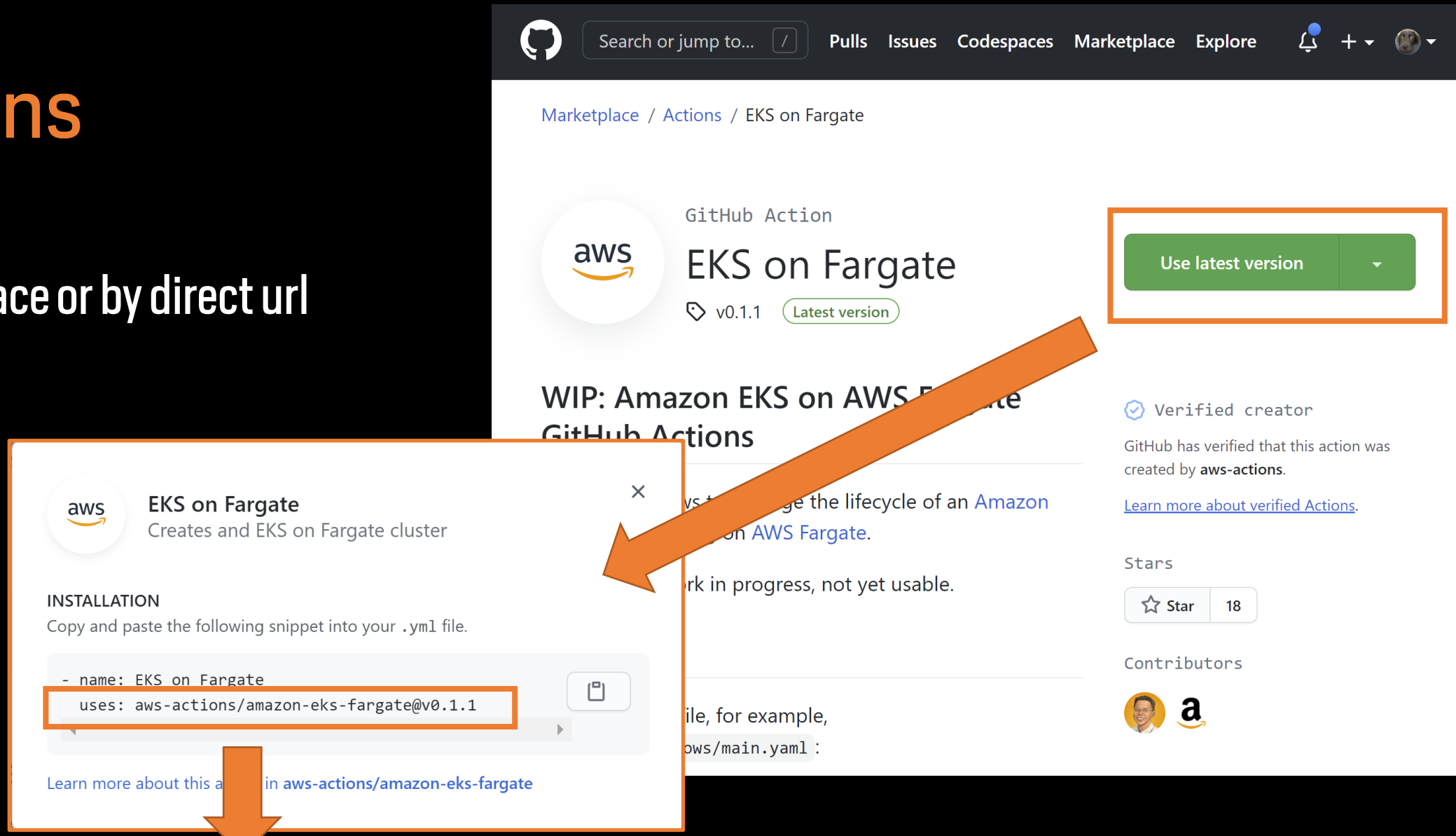
Actions and security

Forking actions

Keeping up to date

Actions

Marketplace or by direct url



https://github.com/aws-actions/amazon-eks-fargate

Actions and security

Are you running just any action from the internet?

Scary! Especially in an enterprise or on local runners

Attack vectors

1. **Data Theft**
2. **Data Integrity Breaches**
3. **Availability**

Protective measures

Manually:

1. Check the action repo code before use
2. Check its container images and dependencies before use

Protective measures

```yaml
uses: shprink/nonharmful-and-must-have-actions@v1
with:
  my-secret: ${{ secrets.MY_SECRET }}
```

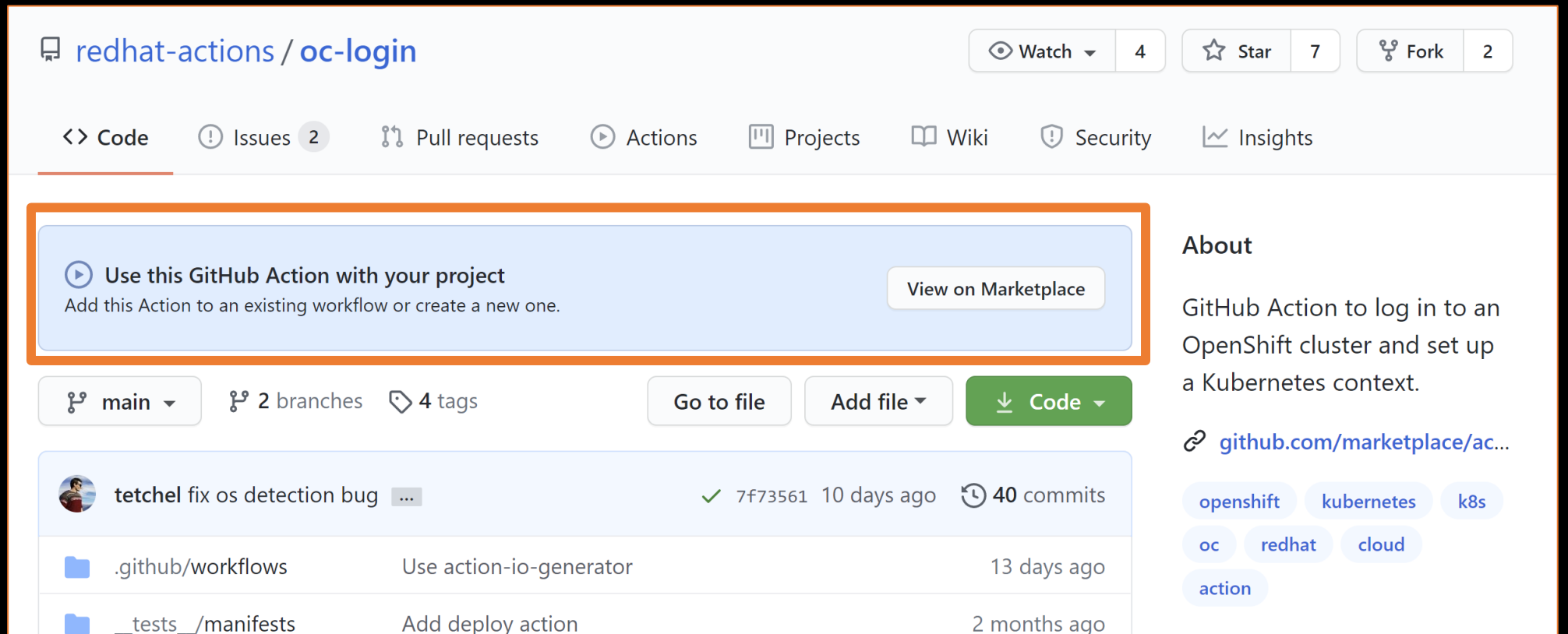
https://github.com/shprink/nonharmful-and-must-have-actions

If the repo has an action.yml, you can use it in your workflow

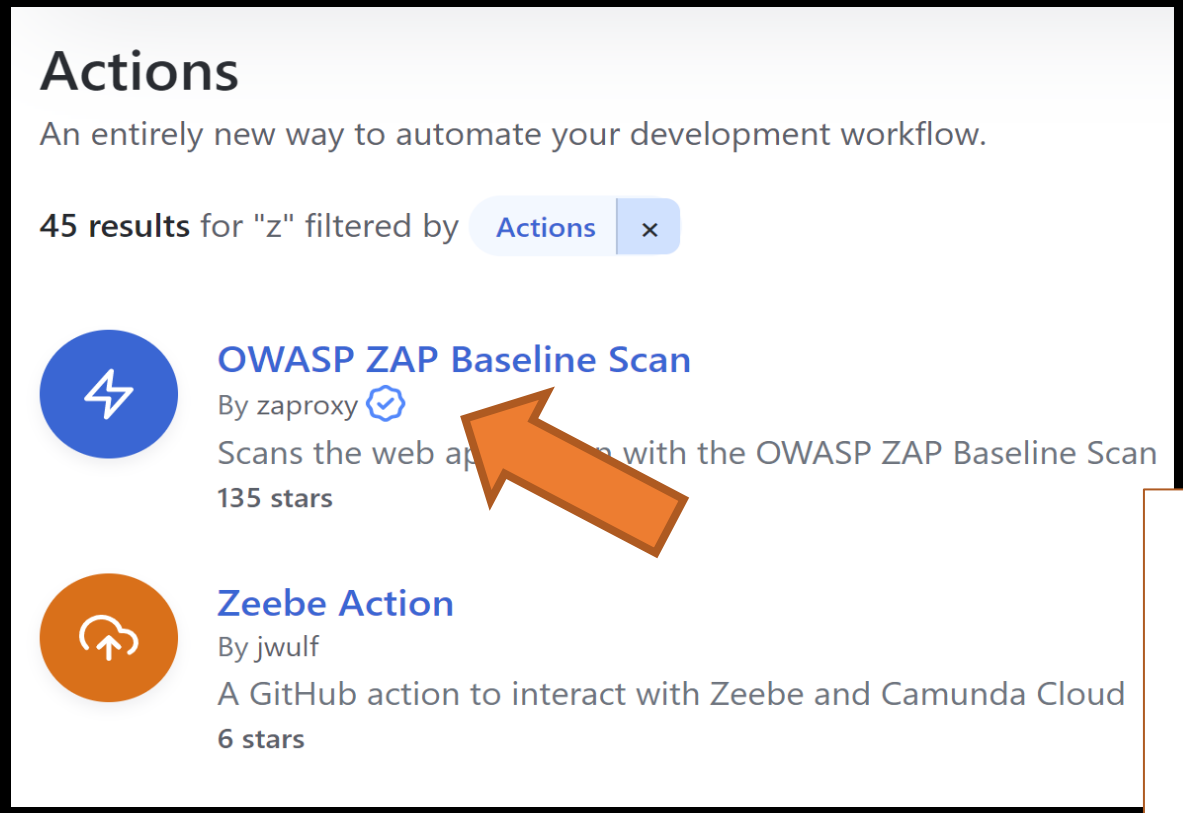
Protective measures

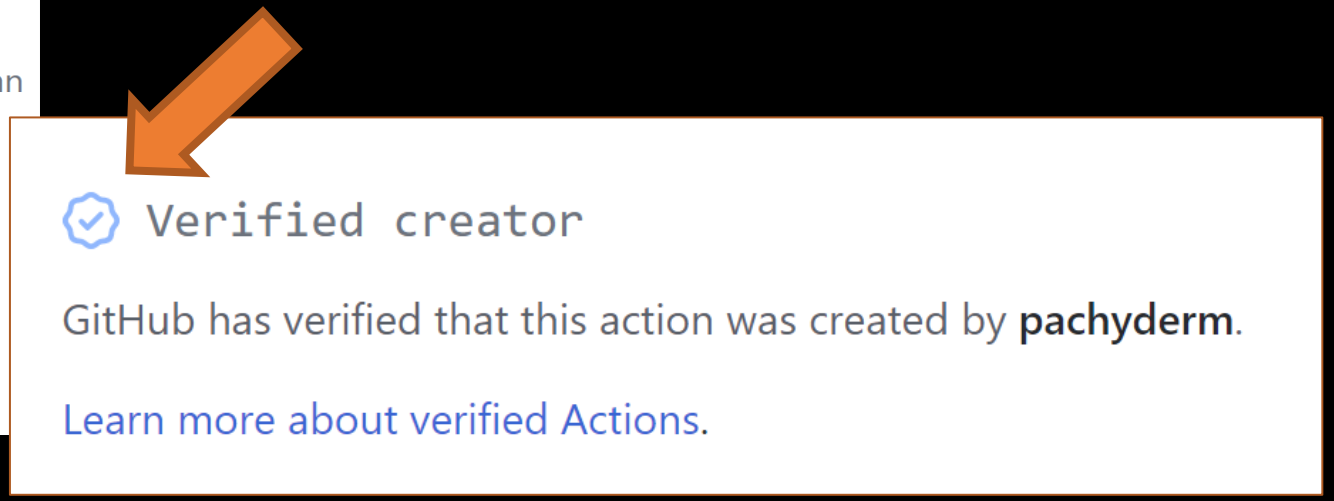
Only use actions listed in the marketplace?

- There is no real verification process for it ☹



Protective measures





Verified Creator

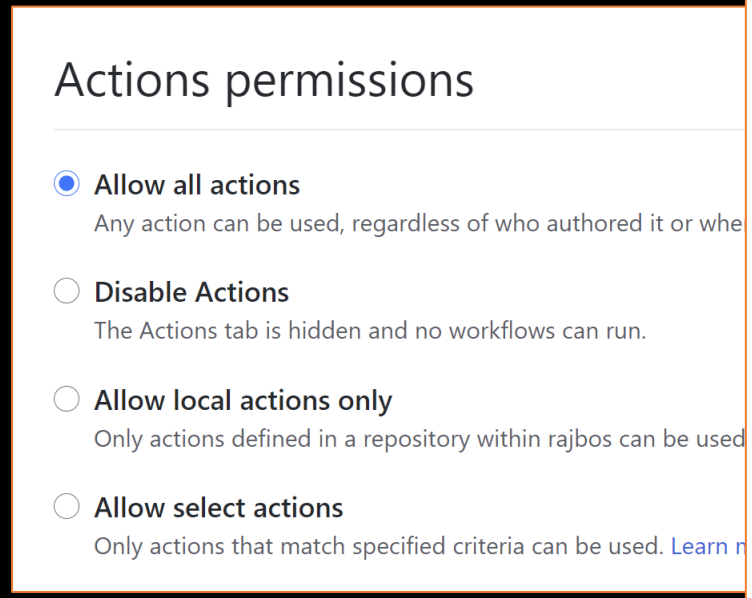
Verification process:

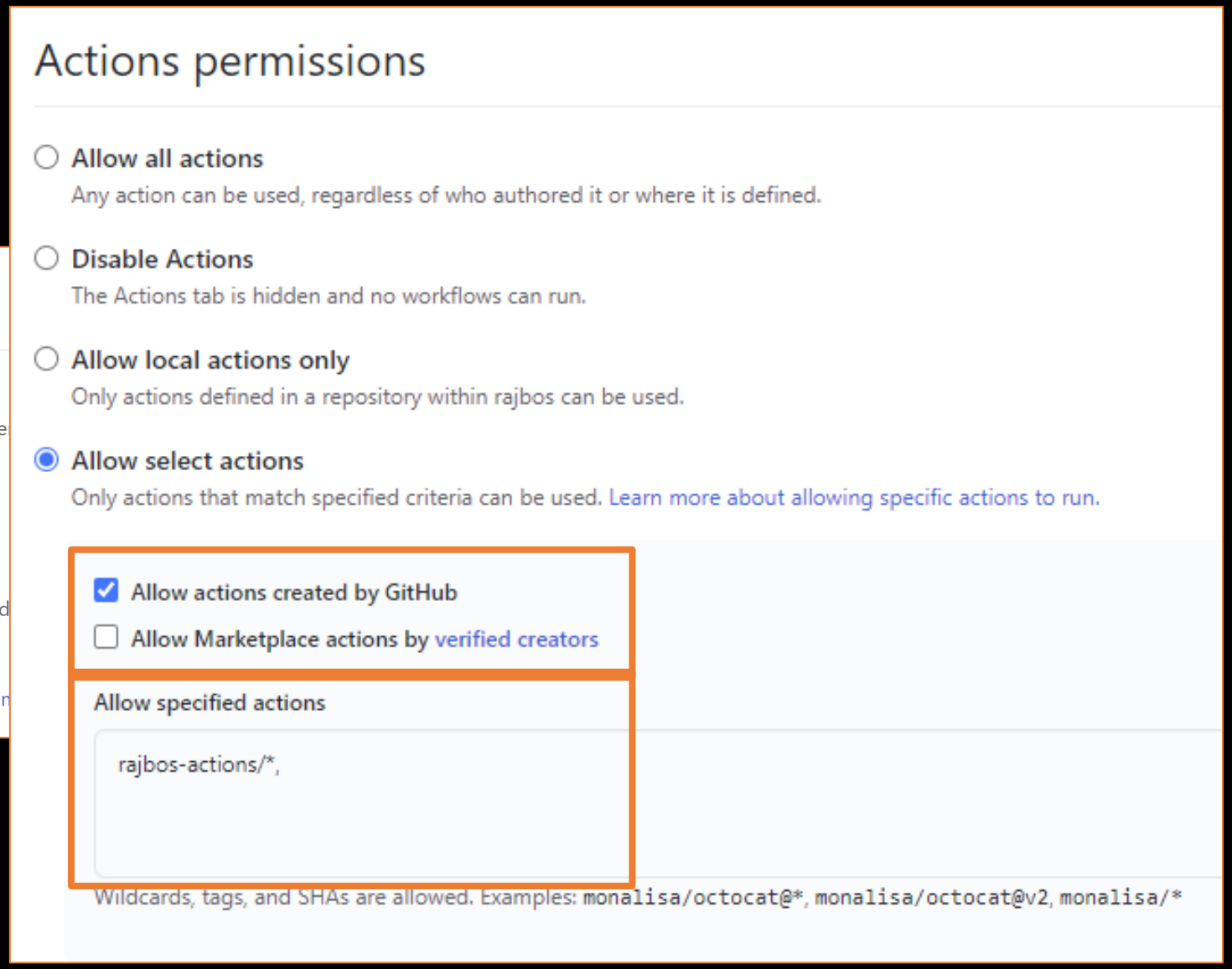
- GitHub Profile information is present and accurate
- Two factor authentication is on for the organization
- Domain verification through a txt record

See: https://xpir.it/verified-publisher

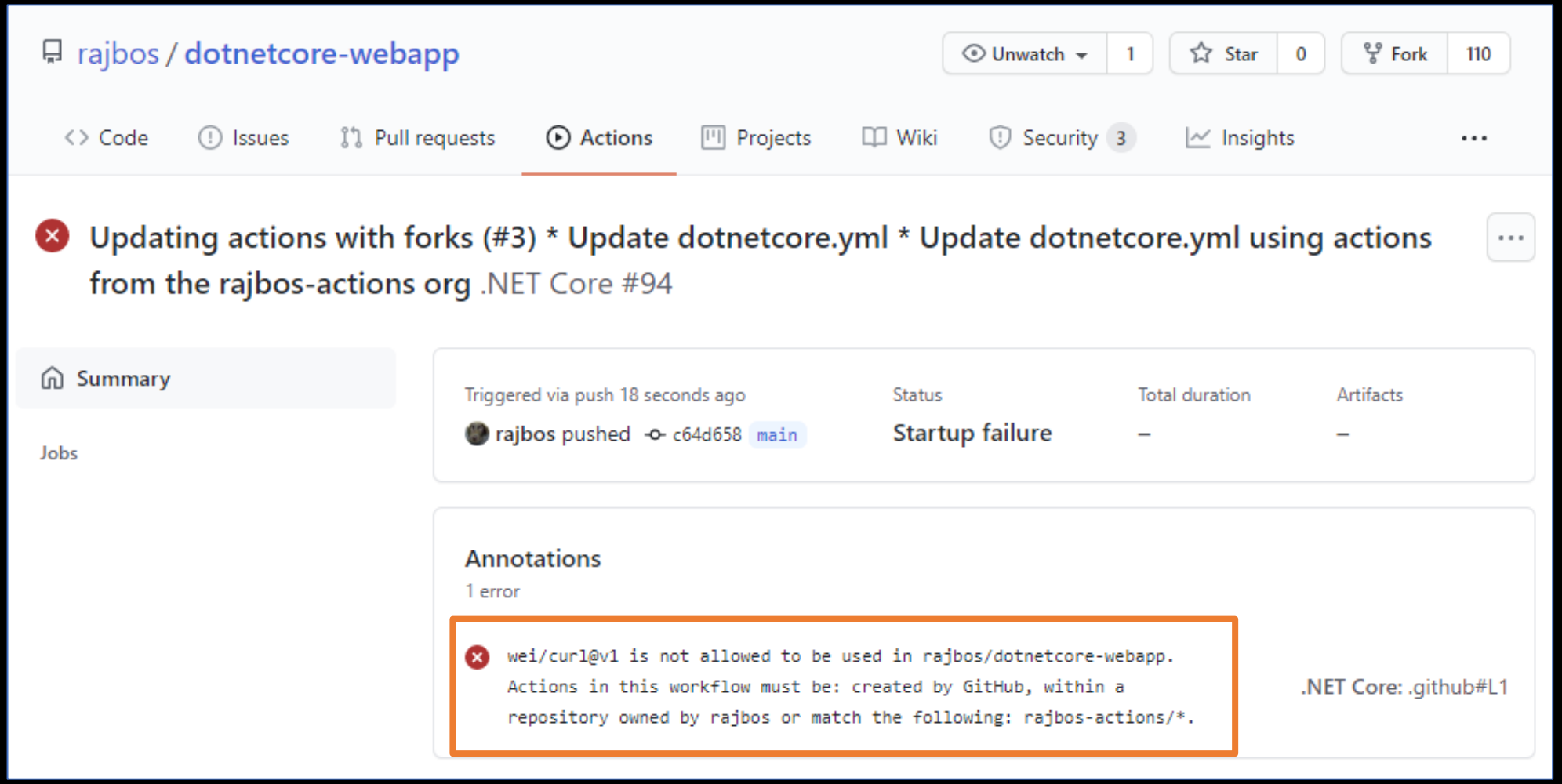
Protective measures

Limiting actions altogether





Protective measures



Protective measures

Pin the action version:

uses: gaurav-nelson/github-action-markdown-link-check@v1
uses: gaurav-nelson/github-action-markdown-link-check@v1.0.1

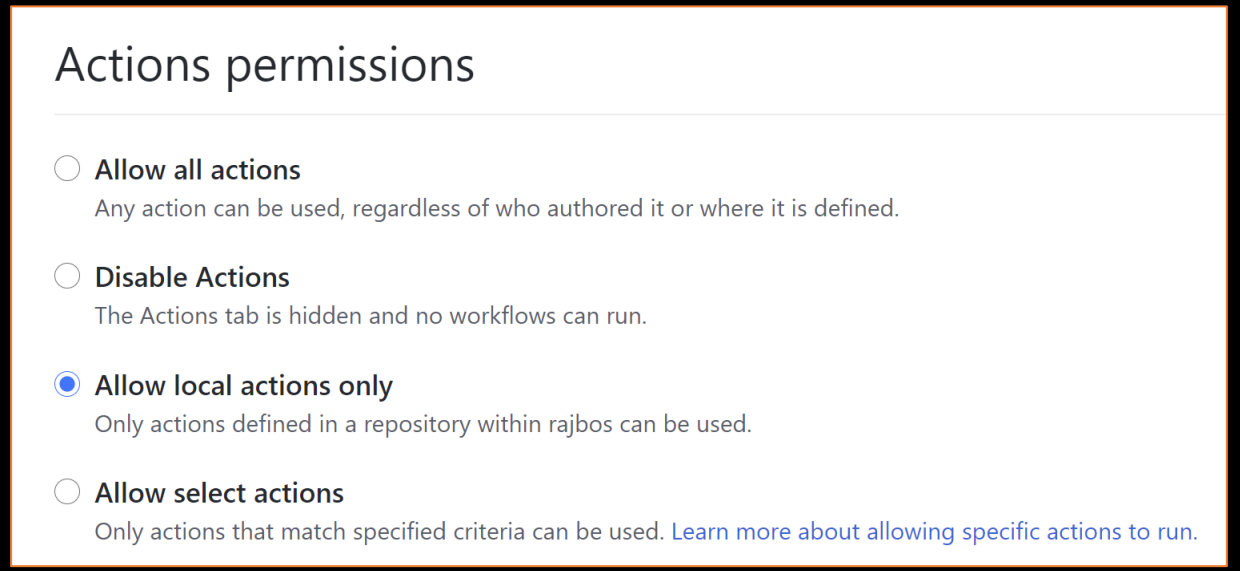
Best practice: Pin the Action's commit SHA:

uses: gaurav-nelson/github-action-markdown-link-check@44a942b2f7ed0dc101d556f281e906fb79f1f478

Recommendation

- Best practice: Limit to local actions and fork action repositories

- Create a separate org to test actions in
  - Enable DevOps teams to own the actions



Workflow attack vectors

- Forks of public repos
- Common fields

Forks of public repos

```
3   on:
4     - push
5     - pull_request
6     - pull_request_target
7
8   jobs:
9     build-and-deploy:
10      environment: PullRequestEnvironment
11
12      runs-on: ubuntu-latest
13
14      steps:
15      - uses: actions/checkout@v1
```

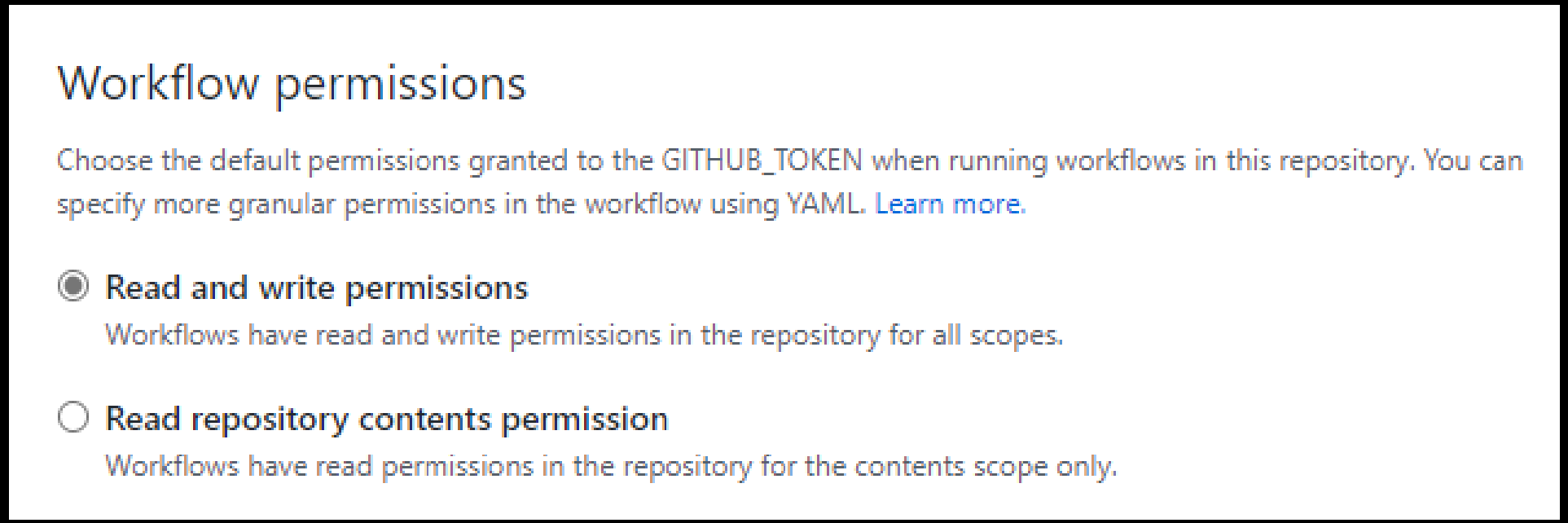
Safe, runs on merge commit, read only access

High risks! Runs on the target, has read + write access and can access secrets

https://xpir.it/gh-pwn-request

Pull Requests

${{ secrets.GITHUB_TOKEN }}



Pull Requests

${{ secrets.GITHUB_TOKEN }}

```yaml
name: Pull request labeler

on: [ pull_request_target ]

permissions:
  contents: read
  pull-requests: write

jobs:
  triage:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/labeler@v2
        with:
          repo-token: ${{ secrets.GITHUB_TOKEN }}
```

Common fields

```
github.event.issue.title
github.event.issue.body
github.event.pull_request.title
github.event.pull_request.body
github.event.comment.body
github.event.review.body
github.event.review_comment.body
github.event.pages.*.page_name
github.event.commits.*.message
github.event.head_commit.message
github.event.head_commit.author.email
github.event.head_commit.author.name
github.event.commits.*.author.email
github.event.commits.*.author.name
github.event.pull_request.head.ref
github.event.pull_request.head.label
github.event.pull_request.head.repo.default_branch
github.head_ref
```

Common fields

```
- name: Check title
  run: |
    title="${{ github.event.issue.title }}"
    if [[ ! $title =~ ^.*:\ .*$ ]]; then
      echo "Bad issue title"
      exit 1
    fi
```

Payload: `a"; echo test`

Remediation

```
- name: print title
  env:
    TITLE: ${{ github.event.issue.title }}
  run: echo ‘$TITLE’
```

https://xpir.it/actions-untrusted-input

GitHub Actions Security

Repository security

Runners and security

Actions and security

Forking actions

Keeping up to date

Forking actions

Pros:

- More secure
- Backup of actions that can be deleted or moved to a different org/repo

Cons:

- More maintenance work
  - Fork needs to be created
  - Kept up to date
- Limits the usage of new actions in your org, as someone create the new action (and by that take responsibility for enabling its use)

GitHub Actions Security

Repository security

Runners and security

Actions and security

Forking actions

Keeping up to date

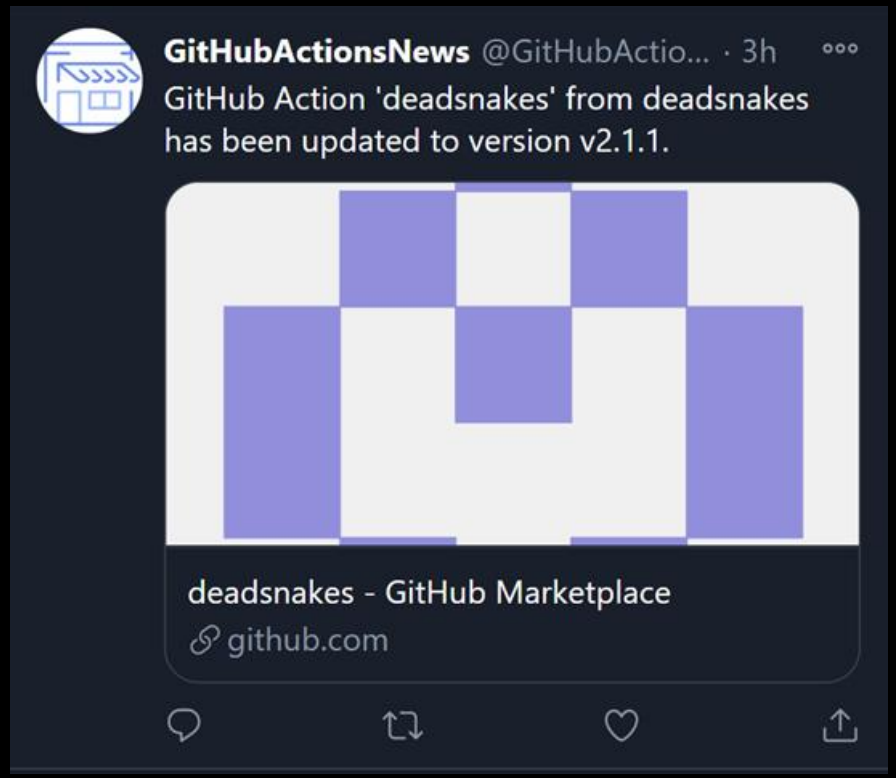
Updates

- Actions are updated regularly
- Wait for a deprecation message?
- How do you stay up to date?

- Auto update with a PR?
- Read the changes in the source repo

Staying up to date

Follow @githubactions on Twitter!



Update action versions

1. Review the Action
   Use Actions + Commit SHA + Dependabot

---

2. Review the Action
   Fork the Actions repo, update your forks and use Dependabot

Option 1: Use SHA + Dependabot

Best practice: Pin the Action's commit SHA:

uses: gaurav-nelson/github-action-markdown-link-check@44a942b2f7ed0dc101d556f281e906fb79f1f478

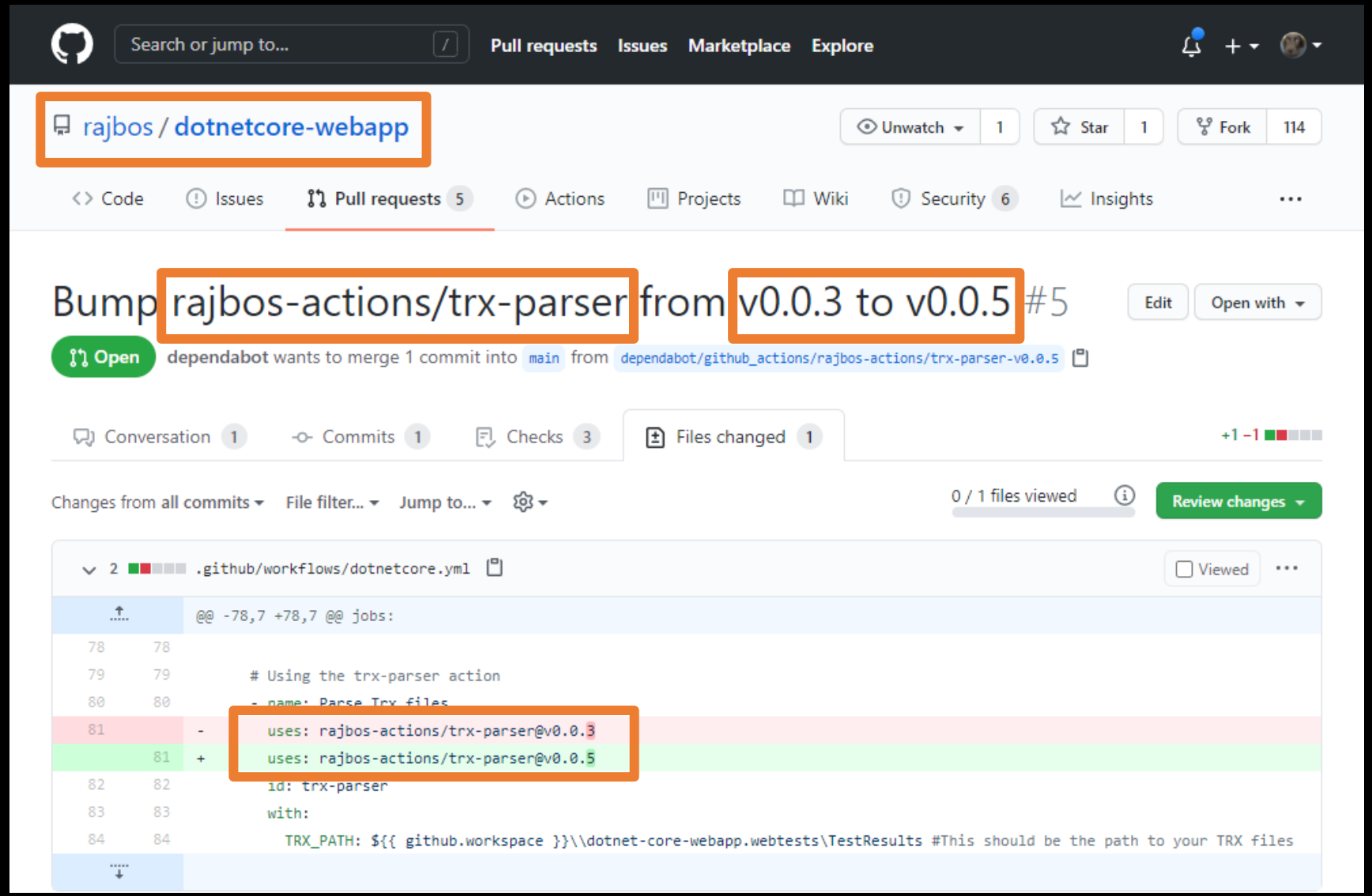
Add .github/dependabot.yml to the repo

```yaml
#Dependabot will check the dependencies in this repo for updates

version: 2
updates:
  - package-ecosystem: "github-actions"
    directory: "/"
    schedule:
      # Check for updates to GitHub Actions every weekday
      interval: "daily"


  - package-ecosystem: "nuget"
    directory: "/"
    schedule:
      # Check for updates to on nuget packages every weekday
      interval: "daily"
```

Use Dependabot



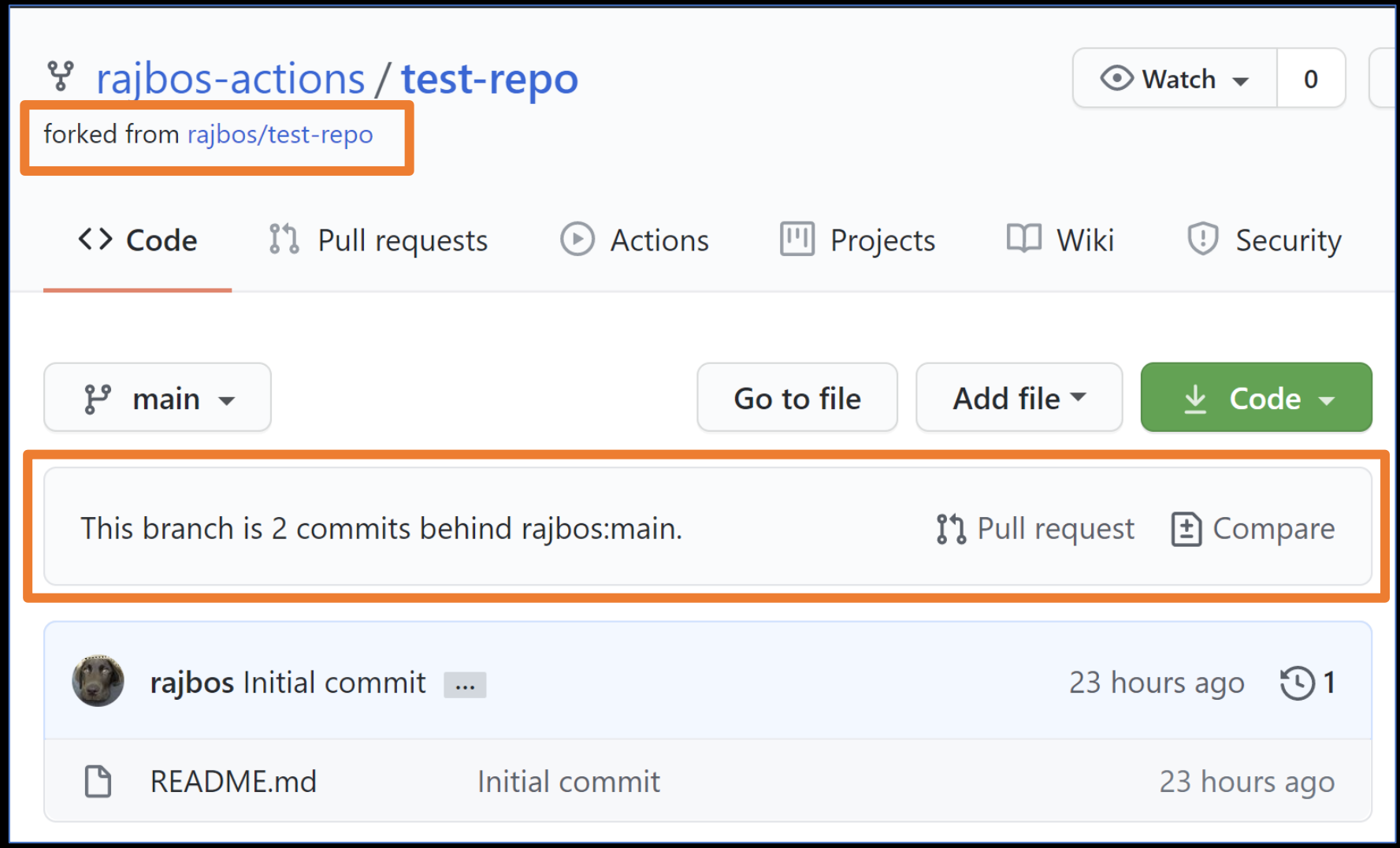
Update action versions

1. Review the Action
   Use Actions + Commit SHA + Dependabot

---

2. Review the Action
   Fork the Actions repo, update your forks and use Dependabot

Keep you forked action up to date



Keep your forked action up to date

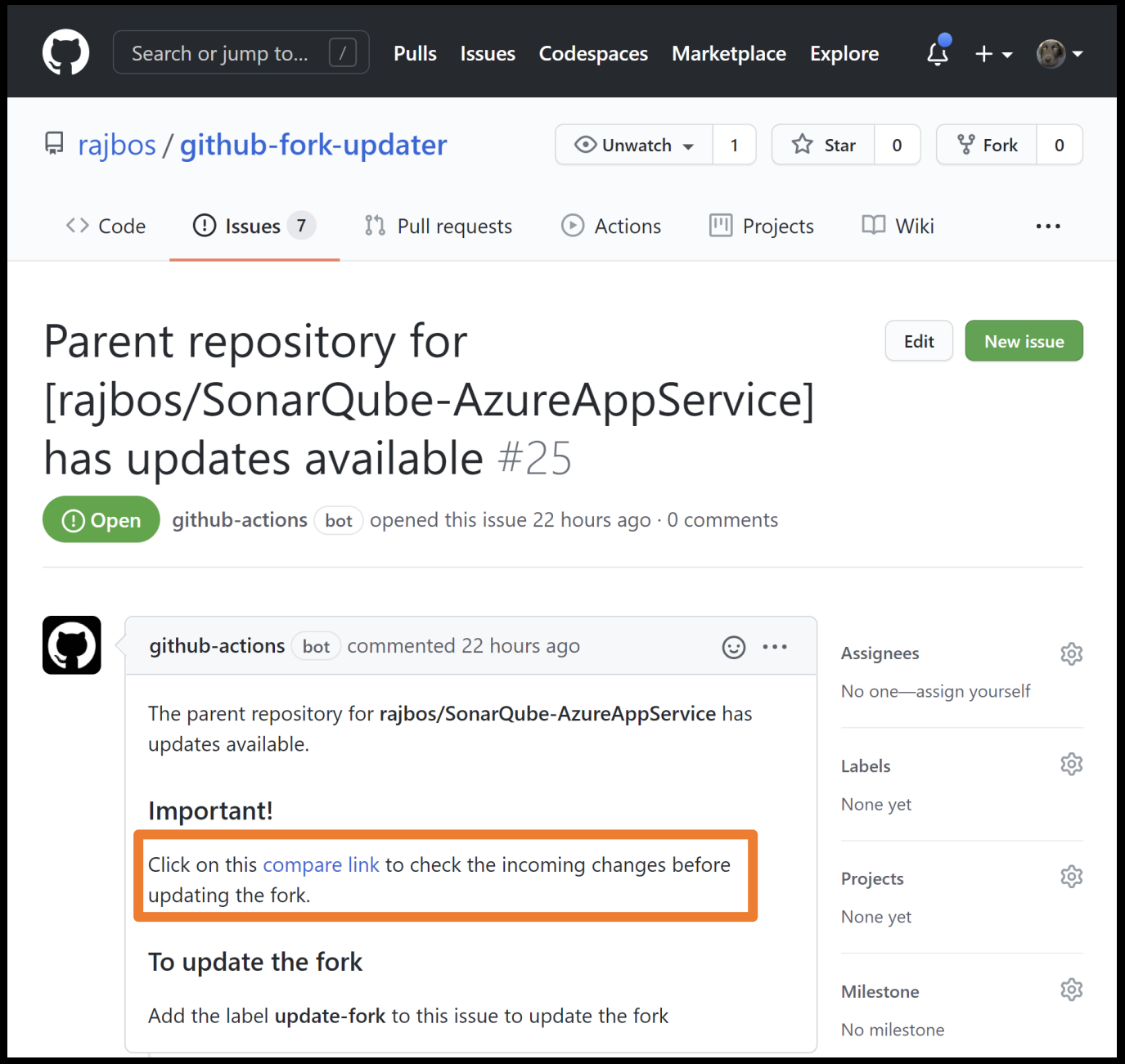
Fork a repo and automate it!

https://github.com/rajbos/github-fork-updater

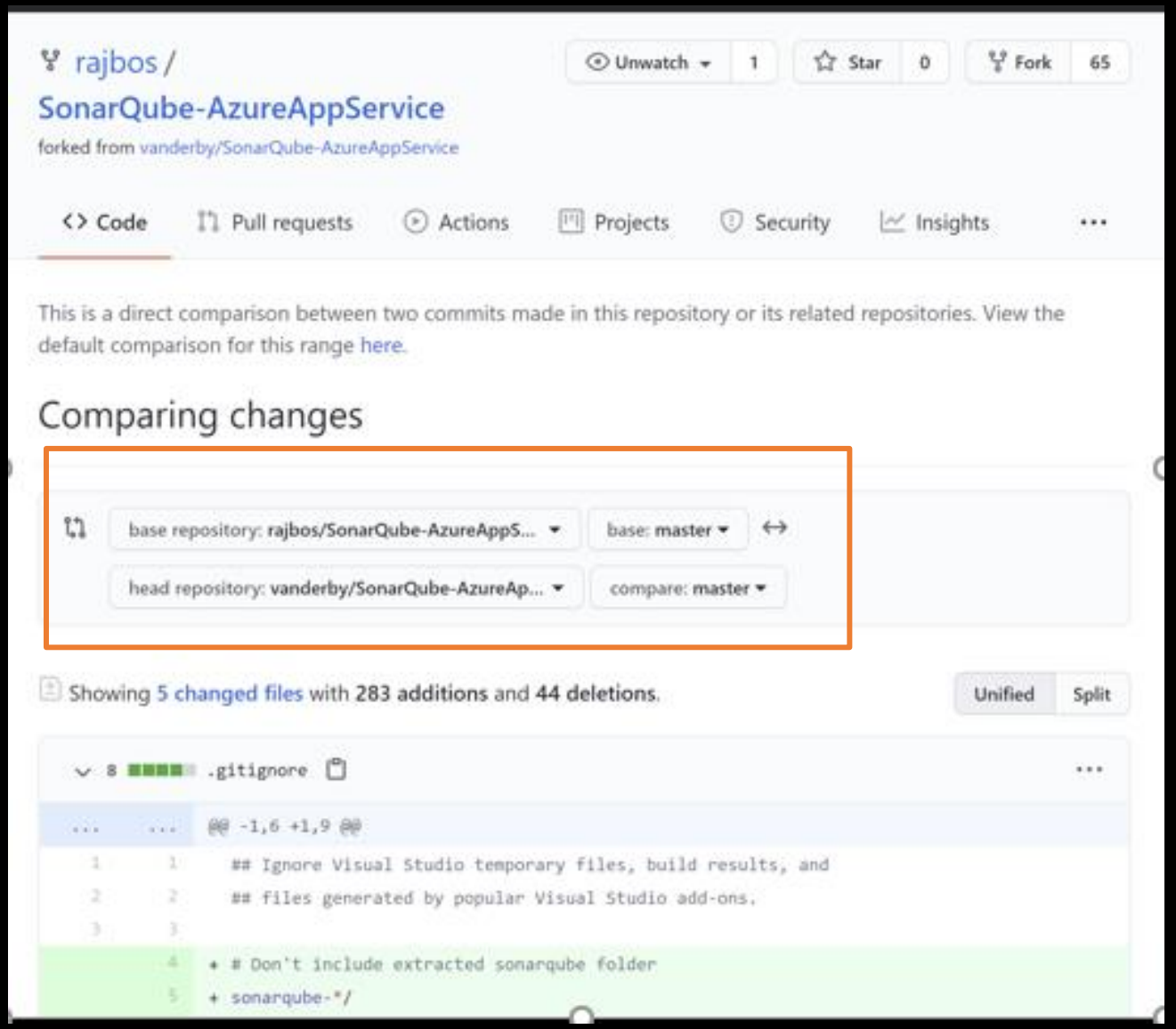
Contains:

- Scheduled workflow
- Creates an issue
- Review the changes
- Label the issue
- Pull in changes

Creates issues

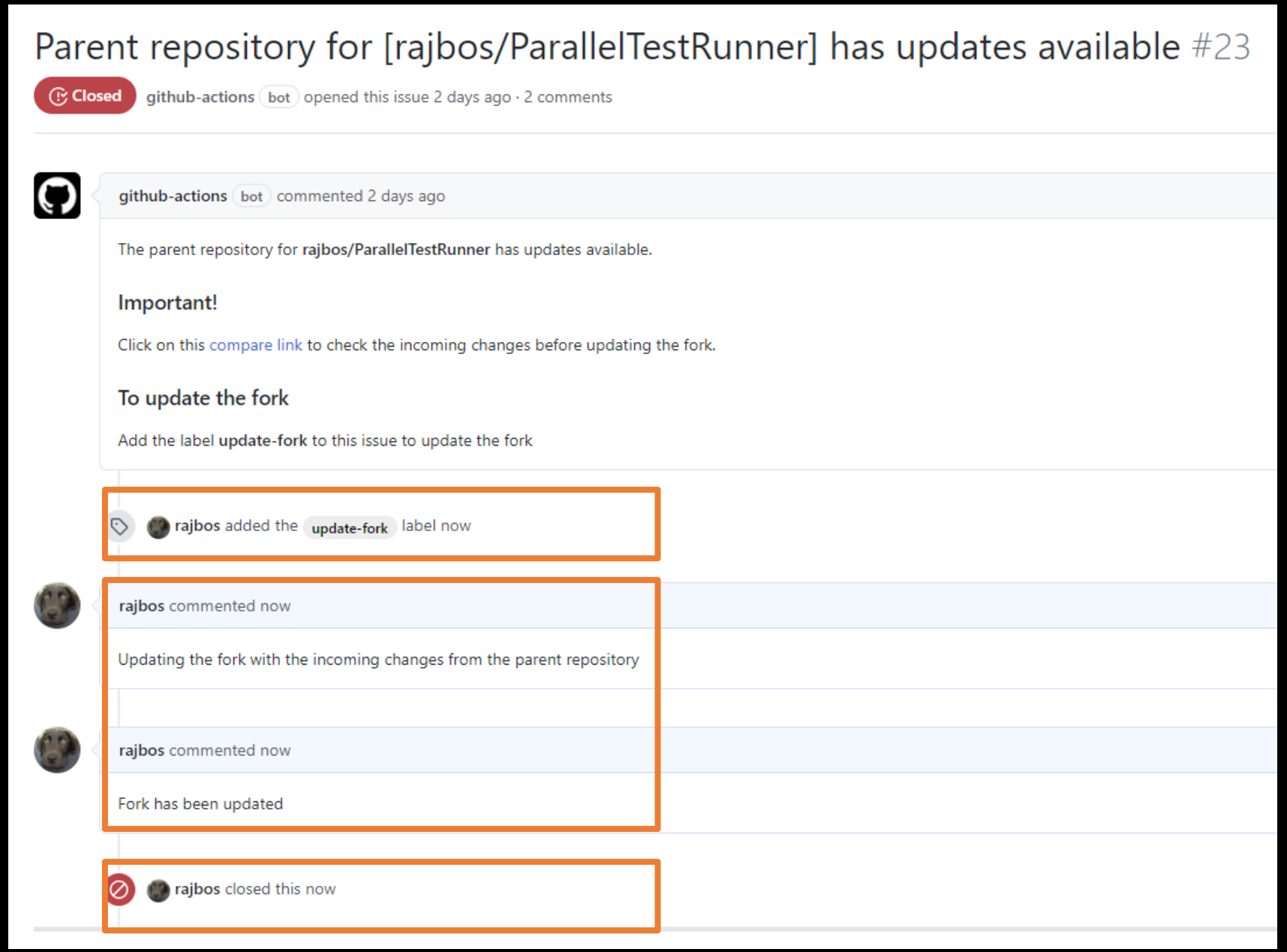


Review before merging



Automation

- Add a label
- Fork gets updated
- Issue gets closed



Pros of forking

- Backup of the action
- Full control over updates
- Pull in updates with validation centrally
- Only allow actions from your actions organization

- Skip commit SHA lookup and updating in every workflow
- Skip adding Dependabot in every repository

https://xpir.it/actions-best-practices

GitHub Actions Security

Repository security
Runners and security
Actions and security

Forking actions
Keeping up to date

Best practices summarized

- Treat workflow secrets very carefully: best to think of them as public
- Review actions' source code
- Pin actions to commit SHA
- Don't trust incoming Pull Requests on public repos
- Fork the action repo and limit actions to local actions only
- Have an organization setup to test with
- Keep your forked actions up to date

How to use GitHub Actions with security in mind

NDC { Security }

Rob Bos
DevOps Consultant – Xpirit
The Netherlands
https://devopsjournal.io

@robbos81

https://myoctocat.com