Automating Dependabot with GitHub Actions

In this article

Examples of how you can use GitHub Actions to automate common Dependabot related tasks.

> **Who can use this feature**
> People with write permissions to a repository can configure GitHub Actions to respond to Dependabot-created pull requests.

About Dependabot and GitHub Actions

Dependabot creates pull requests to keep your dependencies up to date, and you can use GitHub Actions to perform automated tasks when these pull requests are created. For example, fetch additional artifacts, add labels, run tests, or otherwise modifying the pull request.

Responding to events

Dependabot is able to trigger GitHub Actions workflows on its pull requests and comments; however, certain events are treated differently.

For workflows initiated by Dependabot (`github.actor == 'dependabot[bot]'`) using the `pull_request`, `pull_request_review`, `pull_request_review_comment`, `push`, `create`, `deployment`, and `deployment_status` events, the following restrictions apply:

- `GITHUB_TOKEN` has read-only permissions by default.
- Secrets are populated from Dependabot secrets. GitHub Actions secrets are not available.

For workflows initiated by Dependabot (`github.actor == 'dependabot[bot]'`) using the `pull_request_target` event, if the base ref of the pull request was created by Dependabot (`github.actor == 'dependabot[bot]'`), the `GITHUB_TOKEN` will be read-only and secrets are not available.

These restrictions apply even if the workflow is re-run by a different actor.

For more information, see "Keeping your GitHub Actions and workflows secure: Preventing pwn requests."

Changing `GITHUB_TOKEN` permissions

By default, GitHub Actions workflows triggered by Dependabot get a `GITHUB_TOKEN` with

read-only permissions. You can use the `permissions` key in your workflow to increase the access for the token:

```yaml
name: CI
on: pull_request

# Set the access for individual scopes, or use permissions: write-all
permissions:
  pull-requests: write
  issues: write
  repository-projects: write
  ...

jobs:
  ...
```

For more information, see "Automatic token authentication."

Accessing secrets

When a Dependabot event triggers a workflow, the only secrets available to the workflow are Dependabot secrets. GitHub Actions secrets are not available. Consequently, you must store any secrets that are used by a workflow triggered by Dependabot events as Dependabot secrets. For more information, see "Configuring access to private registries for Dependabot."

Dependabot secrets are added to the `secrets` context and referenced using exactly the same syntax as secrets for GitHub Actions. For more information, see "Using secrets in GitHub Actions."

If you have a workflow that will be triggered by Dependabot and also by other actors, the simplest solution is to store the token with the permissions required in an action and in a Dependabot secret with identical names. Then the workflow can include a single call to these secrets. If the secret for Dependabot has a different name, use conditions to specify the correct secrets for different actors to use. For examples that use conditions, see "Common automations" below.

To access a private container registry on AWS with a user name and password, a workflow must include a secret for `username` and `password`. In the example below, when Dependabot triggers the workflow, the Dependabot secrets with the names `READONLY_AWS_ACCESS_KEY_ID` and `READONLY_AWS_ACCESS_KEY` are used. If another actor triggers the workflow, the actions secrets with those names are used.

```yaml
name: CI
on:
  pull_request:
    branches: [ main ]

jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout repository
        uses: actions/checkout@v4

      - name: Login to private container registry for dependencies
        uses: docker/login-action@v2
        with:
          registry: https://1234567890.dkr.ecr.us-east-1.amazonaws.com
          username: ${{ secrets.READONLY_AWS_ACCESS_KEY_ID }}
          password: ${{ secrets.READONLY_AWS_ACCESS_KEY }}

      - name: Build the Docker image
        run: docker build . --file Dockerfile --tag my-image-name:$(date +%s)
```

Manually re-running a workflow

When you manually re-run a Dependabot workflow, it will run with the same privileges as before even if the user who initiated the rerun has different privileges. For more information, see "Re-running workflows and jobs."

Common Dependabot automations

Here are several common scenarios that can be automated using GitHub Actions.

Fetch metadata about a pull request

A large amount of automation requires knowing information about the contents of the pull request: what the dependency name was, if it's a production dependency, and if it's a major, minor, or patch update.

The `dependabot/fetch-metadata` action provides all that information for you:

```yaml
name: Dependabot fetch metadata
on: pull_request

permissions:
  pull-requests: write
  issues: write
  repository-projects: write

jobs:
  dependabot:
    runs-on: ubuntu-latest
    if: ${{ github.actor == 'dependabot[bot]' }}
    steps:
      - name: Dependabot metadata
        id: metadata
        uses: dependabot/fetch-metadata@v1
        with:
          github-token: "${{ secrets.GITHUB_TOKEN }}"
      # The following properties are now available:
      #  - steps.metadata.outputs.dependency-names
      #  - steps.metadata.outputs.dependency-type
      #  - steps.metadata.outputs.update-type
```

For more information, see the `dependabot/fetch-metadata` repository.

Label a pull request

If you have other automation or triage workflows based on GitHub labels, you can configure an action to assign labels based on the metadata provided.

For example, if you want to flag all production dependency updates with a label:

```yaml
name: Dependabot auto-label
on: pull_request

permissions:
  pull-requests: write
  issues: write
  repository-projects: write

jobs:
  dependabot:
    runs-on: ubuntu-latest
```

```
    if: ${{ github.actor == 'dependabot[bot]' }}
    steps:
      - name: Dependabot metadata
        id: metadata
        uses: dependabot/fetch-metadata@v1
        with:
          github-token: "${{ secrets.GITHUB_TOKEN }}"
      - name: Add a label for all production dependencies
        if: ${{ steps.metadata.outputs.dependency-type == 'direct:production' }}
        run: gh pr edit "$PR_URL" --add-label "production"
        env:
          PR_URL: ${{github.event.pull_request.html_url}}
```

Approve a pull request

If you want to automatically approve Dependabot pull requests, you can use the GitHub CLI in a workflow:

```
name: Dependabot auto-approve
on: pull_request

permissions:
  pull-requests: write

jobs:
  dependabot:
    runs-on: ubuntu-latest
    if: ${{ github.actor == 'dependabot[bot]' }}
    steps:
      - name: Dependabot metadata
        id: metadata
        uses: dependabot/fetch-metadata@v1
        with:
          github-token: "${{ secrets.GITHUB_TOKEN }}"
      - name: Approve a PR
        run: gh pr review --approve "$PR_URL"
        env:
          PR_URL: ${{github.event.pull_request.html_url}}
          GITHUB_TOKEN: ${{secrets.GITHUB_TOKEN}}
```

Enable auto-merge on a pull request

If you want to allow maintainers to mark certain pull requests for auto-merge, you can use GitHub's auto-merge functionality. This enables the pull request to be merged when any tests and approvals required by the branch protection rules are successfully met. For more information, see "Automatically merging a pull request" and "Managing a branch protection rule."

> **Note:** If you use status checks to test pull requests, you should enable **Require status checks to pass before merging** for the target branch for Dependabot pull requests. This branch protection rule ensures that pull requests are not merged unless all the required status checks pass. For more information, see "Managing a branch protection rule."

You can instead use GitHub Actions and the GitHub CLI. Here is an example that auto merges all patch updates to `my-dependency`:

```
name: Dependabot auto-merge
on: pull_request

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

```yaml
jobs:
  dependabot:
    runs-on: ubuntu-latest
    if: ${{ github.actor == 'dependabot[bot]' }}
    steps:
      - name: Dependabot metadata
        id: metadata
        uses: dependabot/fetch-metadata@v1
        with:
          github-token: "${{ secrets.GITHUB_TOKEN }}"
      - name: Enable auto-merge for Dependabot PRs
        if: ${{contains(steps.metadata.outputs.dependency-names, 'my-dependency')
&& steps.metadata.outputs.update-type == 'version-update:semver-patch'}}
        run: gh pr merge --auto --merge "$PR_URL"
        env:
          PR_URL: ${{github.event.pull_request.html_url}}
          GITHUB_TOKEN: ${{secrets.GITHUB_TOKEN}}
```

Troubleshooting failed workflow runs

If your workflow run fails, check the following:

- You are running the workflow only when the correct actor triggers it.
- You are checking out the correct `ref` for your `pull_request`.
- Your secrets are available in Dependabot secrets rather than as GitHub Actions secrets.
- You have a `GITHUB_TOKEN` with the correct permissions.

For information on writing and debugging GitHub Actions, see "Learn GitHub Actions."