Customizing dependency updates

In this article

You can customize how Dependabot maintains your dependencies.

> **Who can use this feature**
> People with write permissions to a repository can configure Dependabot for the repository.

About customizing dependency updates

After you've enabled version updates, you can customize how Dependabot maintains your dependencies by adding further options to the `dependabot.yml` file. For example, you could:

- Specify which day of the week to open pull requests for version updates: `schedule.day`
- Set reviewers, assignees, and labels for each package manager: `reviewers`, `assignees`, and `labels`
- Create groups of dependencies (per package ecosystem), so that Dependabot updates the group of dependencies in a single pull request: `groups`
- Define a versioning strategy for changes to each manifest file: `versioning-strategy`
- Change the maximum number of open pull requests for version updates from the default of 5: `open-pull-requests-limit`
- Open pull requests for version updates to target a specific branch, instead of the default branch: `target-branch`

For more information about the configuration options, see "Configuration options for the dependabot.yml file."

When you update the `dependabot.yml` file in your repository, Dependabot runs an immediate check with the new configuration. Within minutes you will see an updated list of dependencies on the **Dependabot** tab, this may take longer if the repository has many dependencies. You may also see new pull requests for version updates. For more information, see "Listing dependencies configured for version updates."

Impact of configuration changes on security updates

If you customize the `dependabot.yml` file, you may notice some changes to the pull requests raised for security updates. These pull requests are always triggered by a security advisory for a dependency, rather than by the Dependabot schedule. However, they inherit relevant configuration settings from the `dependabot.yml` file unless you specify a different target branch for version updates.

For an example, see "Setting custom labels" below.

Modifying scheduling

When you set a `daily` update schedule, by default, Dependabot checks for new versions at 05:00 UTC. You can use `schedule.time` to specify an alternative time of day to check for updates (format: `hh:mm`).

The example `dependabot.yml` file below expands the npm configuration to specify when Dependabot should check for version updates to dependencies.

```
# `dependabot.yml` file with
# customized schedule for version updates

version: 2
updates:
  # Keep npm dependencies up to date
  - package-ecosystem: "npm"
    directory: "/"
    # Check the npm registry for updates at 2am UTC
    schedule:
      interval: "daily"
      time: "02:00"
```

Setting reviewers and assignees

By default, Dependabot raises pull requests without any reviewers or assignees.

You can use `reviewers` and `assignees` to specify reviewers and assignees for all pull requests raised for a package manager. When you specify a team, you must use the full team name, as if you were @mentioning the team (including the organization).

The example `dependabot.yml` file below changes the npm configuration so that all pull requests opened with version and security updates for npm will have two reviewers and one assignee.

```
# `dependabot.yml` file with
# reviews and an assignee for all npm pull requests

version: 2
updates:
  # Keep npm dependencies up to date
  - package-ecosystem: "npm"
    directory: "/"
    schedule:
      interval: "weekly"
    # Raise all npm pull requests with reviewers
    reviewers:
      - "my-org/team-name"
      - "octocat"
    # Raise all npm pull requests with an assignee
    assignees:
      - "user-name"
```

Setting custom labels

By default, Dependabot raises all pull requests with the `dependencies` label. If more than one package manager is defined, Dependabot includes an additional label on each pull request. This indicates which language or ecosystem the pull request will update, for example: `java` for Gradle updates and `submodules` for git submodule updates. Dependabot creates these default labels automatically, as necessary in your repository.

You can use `labels` to override the default labels and specify alternative labels for all pull requests raised for a package manager. You can't create new labels in the `dependabot.yml` file, so the alternative labels must already exist in the repository.

The example `dependabot.yml` file below changes the npm configuration so that all pull requests opened with version and security updates for npm will have custom labels. It also changes the Docker configuration to check for version updates against a custom branch and to raise pull requests with custom labels against that custom branch. The changes to Docker will not affect security update pull requests because security updates are always made against the default branch.

Note: The new `target-branch` must contain a Dockerfile to update, otherwise this change will have the effect of disabling version updates for Docker.

```
# `dependabot.yml` file with
# customized npm configuration

version: 2
updates:
  # Keep npm dependencies up to date
  - package-ecosystem: "npm"
    directory: "/"
    schedule:
      interval: "weekly"
    # Raise all npm pull requests with custom labels
    labels:
      - "npm dependencies"
      - "triage-board"

    # Keep Docker dependencies up to date
  - package-ecosystem: "docker"
    directory: "/"
    schedule:
      interval: "weekly"
    # Raise pull requests for Docker version updates
    # against the "develop" branch. The Docker configuration
    # no longer affects security update pull requests.
    target-branch: "develop"
    # Use custom labels on pull requests for Docker version updates
    labels:
      - "Docker dependencies"
      - "triage-board"
```

Grouping Dependabot version updates into one pull request

By default, Dependabot raises a single pull request for each dependency that needs to be updated to a newer version. You can use `groups` to create sets of dependencies (per package manager), so that Dependabot opens a single pull request to update multiple dependencies at the same time.

You can also specify grouping settings based on how updates affect a specific ecosystem and follow semantic versioning (SemVer). This means you can, for example, group all

patch updates together. This approach helps Dependabot create as few pull requests as possible, while also reducing the chances of accidentally accepting changes that could cause issues. If a package follows SemVer, there's a higher chance (but not a guarantee) that minor and patch updates will be backwards compatible.

> **Note:** SemVer is an accepted standard for defining versions of software packages, in the form `x.y.z`. Dependabot assumes that versions in this form are always `major.minor.patch`.

Dependabot creates groups in the order they appear in your `dependabot.yml` file. If a dependency update could belong to more than one group, it is only assigned to the first group it matches with.

You can only create groups for Dependabot version updates. Dependabot security updates do not support grouped updates. In addition, if there is a grouped pull request for a vulnerable package, Dependabot security updates will always attempt to create a separate pull request, even if the existing group pull request is an update to the same, or a later, version.

You must configure groups per package ecosystem.

Example configurations for `groups`

Example 1

The `dependabot.yml` file configuration uses `patterns` and `dependency-type` options to include specific dependencies in the group, and `exclude-patterns` to exclude a dependency (or multiple dependencies) from the group.

```yaml
# `dependabot.yml` file using the `dependency-type` option to group updates
# in conjunction with `patterns` and `exclude-patterns`.

groups:
  production-dependencies:
    dependency-type: "production"
  development-dependencies:
    dependency-type: "development"
    exclude-patterns:
    - "rubocop*"
  rubocop:
    patterns:
    - "rubocop*"
```

Example 2

A `dependabot.yml` file with a customized Bundler configuration, which has been modified to create a group of dependencies. The configuration specifies `patterns` (strings of characters) that match with the name of a dependency (or multiple dependencies) in order to include the dependencies in the group.

```yaml
# `dependabot.yml` file with customized Bundler configuration
# In this example, the name of the group is `dev-dependencies`, and
# only the `patterns` and `exclude-patterns` options are used.
version: 2
updates:
  # Keep bundler dependencies up to date
  - package-ecosystem: "bundler"
    directory: "/"
    schedule:
      interval: "weekly"
    # Create a group of dependencies to be updated together in one pull request
    groups:
```

```
      # Specify a name for the group, which will be used in pull request titles
      # and branch names
      dev-dependencies:
         # Define patterns to include dependencies in the group (based on
         # dependency name)
         patterns:
           - "rubocop" # A single dependency name
           - "rspec*"  # A wildcard string that matches multiple dependency
names
           - "*"       # A wildcard that matches all dependencies in the package
                       # ecosystem. Note: using "*" may open a large pull
request
         # Define patterns to exclude dependencies from the group (based on
         # dependency name)
         exclude-patterns:
           - "gc_ruboconfig"
           - "gocardless-*"
```

Example 3

The `dependabot.yml` file is configured so that any packages matching the pattern `@angular*` where the highest resolvable version is `minor` or `patch` will be grouped together. Dependabot will create a separate pull request for any package that doesn't match the pattern, or that doesn't update to a `minor` or `patch` version.

```
# `dependabot.yml` file using the `update-types` option to group updates.
# Any packages matching the pattern @angular* where the highest resolvable
# version is minor or patch will be grouped together.
version: 2
updates:
  - package-ecosystem: "npm"
    directory: "/"
    schedule:
      interval: "weekly"
    groups:
      angular:
        patterns:
        - "@angular*"
        update-types:
        - "minor"
        - "patch"
```

Example 4

The `dependabot.yml` file uses an `ignore` condition to exclude updates to `major` versions of `@angular*` packages.

```
# `dependabot.yml` file using the `update-types` option to group updates
# in conjunction with an `ignore` condition.
# If you do not want updates to `major` versions of `@angular*` packages, you can
specify an `ignore` condition
groups:
  angular:
    patterns:
    - "@angular*"
    update-types:
    - "minor"
    - "patch"
ignore:
  - dependency-name: "@angular*"
    update-types: ["version-update:semver-major"]
```

For more information about configuring dependency groups in the `dependabot.yml` file, see "Configuration options for the dependabot.yml file."

Ignoring specific dependencies for Dependabot version updates

If you are not ready to adopt changes from dependencies in your project, you can configure Dependabot to ignore those dependencies when it opens pull requests for version updates. You can do this using one of the following methods.

- Configure the `ignore` option for the dependency in your `dependabot.yml` file. You can use this to ignore updates for specific dependencies, versions, and types of updates. For more information, see "Configuration options for the dependabot.yml file."
- Use `@dependabot ignore` comment commands on a Dependabot pull request for version updates. You can use comment commands to ignore updates for specific dependencies and versions. For more information, see "Managing pull requests for dependency updates."

If you would like to un-ignore a dependency or ignore condition, you can delete the ignore conditions from the `dependabot.yml` file or reopen the pull request.

For pull requests for grouped version updates, you can also use `@dependabot unignore` comment commands. The `@dependabot unignore` comment commands enable you to do the following by commenting on a Dependabot pull request:

- Un-ignore a specific ignore condition
- Un-ignore a specific dependency
- Un-ignore all ignore conditions for all dependencies in a Dependabot pull request

> **Note:** The `@dependabot unignore` comment commands only work on pull requests for grouped version updates.

For more information, see "Managing pull requests for dependency updates."

More examples

For more examples, see "Configuration options for the dependabot.yml file."