

Configuring Dependabot version updates

In this article

About version updates for dependencies

Enabling Dependabot version updates

Checking the status of version updates

Disabling Dependabot version updates

You can configure your repository so that Dependabot automatically updates the packages you use.

Who can use this feature

People with write permissions to a repository can enable or disable Dependabot version updates for the repository.

Note: Your site administrator must set up Dependabot updates for your GitHub Enterprise Server instance before you can use this feature. For more information, see "[Enabling Dependabot for your enterprise](#)."

You may not be able to enable or disable Dependabot updates if an enterprise owner has set a policy at the enterprise level. For more information, see "[Enforcing policies for code security and analysis for your enterprise](#)."

About version updates for dependencies

You enable Dependabot version updates by checking a `dependabot.yml` configuration file in to your repository's `.github` directory. Dependabot then raises pull requests to keep the dependencies you configure up-to-date. For each package manager's dependencies that you want to update, you must specify the location of the package manifest files and how often to check for updates to the dependencies listed in those files. For information about enabling security updates, see "[Configuring Dependabot security updates](#)."

When you first enable version updates, you may have many dependencies that are outdated and some may be many versions behind the latest version. Dependabot checks for outdated dependencies as soon as it's enabled. You may see new pull requests for version updates within minutes of adding the configuration file, depending on the number of manifest files for which you configure updates. Dependabot will also run an update on subsequent changes to the configuration file.

Dependabot may also create pull requests when you change a manifest file after an update has failed. This is because changes to a manifest, such as removing the dependency that caused the update to fail, may cause the newly triggered update to succeed.

To keep pull requests manageable and easy to review, Dependabot raises a maximum of five pull requests to start bringing dependencies up to the latest version. If you merge some of these first pull requests before the next scheduled update, remaining pull requests will be opened on the next update, up to that maximum. You can change the

maximum number of open pull requests by setting the [open-pull-requests-limit configuration option](#).

For more information, see "[Customizing dependency updates](#)."

By default only direct dependencies that are explicitly defined in a manifest are kept up to date by Dependabot version updates. You can choose to receive updates for indirect dependencies defined in lock files. For more information, see "[Configuration options for the dependabot.yml file](#)."

When running security or version updates, some ecosystems must be able to resolve all dependencies from their source to verify that updates have been successful. If your manifest or lock files contain any private dependencies, Dependabot must be able to access the location at which those dependencies are hosted. Organization owners can grant Dependabot access to private repositories containing dependencies for a project within the same organization. For more information, see "[Managing security and analysis settings for your organization](#)." You can configure access to private registries in a repository's `dependabot.yml` configuration file. For more information, see "[Configuration options for the dependabot.yml file](#)." Additionally, Dependabot doesn't support private GitHub dependencies for all package managers. For more information, see "[About Dependabot version updates](#)" and "[GitHub language support](#)."

Enabling Dependabot version updates

You enable Dependabot version updates by committing a `dependabot.yml` configuration file to your repository.

- 1 Create a `dependabot.yml` configuration file in the `.github` directory of your repository.
- 2 Add a `version` .
- 3 Optionally, if you have dependencies in a private registry, add a `registries` section containing authentication details.
- 4 Add an `updates` section, with an entry for each package manager you want Dependabot to monitor.
- 5 For each package manager, use:
 - `package-ecosystem` to specify the package manager.
 - `directory` to specify the location of the manifest or other definition files.
 - `schedule.interval` to specify how often to check for new versions.
- 6 Check the `dependabot.yml` configuration file in to the `.github` directory of the repository.

For information about all the configuration options, see "[Configuration options for the dependabot.yml file](#)."

Example `dependabot.yml` file

The example `dependabot.yml` file below configures version updates for two package managers: npm and Docker. When this file is checked in, Dependabot checks the manifest files on the default branch for outdated dependencies. If it finds outdated dependencies, it will raise pull requests against the default branch to update the dependencies.

```
# Basic `dependabot.yml` file with
# minimum configuration for two package managers

version: 2
updates:
  # Enable version updates for npm
  - package-ecosystem: "npm"
    # Look for `package.json` and `lock` files in the `root` directory
    directory: "/"
    # Check the npm registry for updates every day (weekdays)
    schedule:
      interval: "daily"


  # Enable version updates for Docker
  - package-ecosystem: "docker"
    # Look for a `Dockerfile` in the `root` directory
    directory: "/"
    # Check for updates once a week
    schedule:
      interval: "weekly"
```

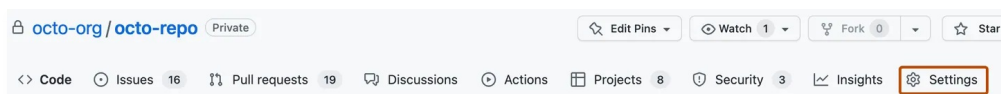
In the example above, if the Docker dependencies were very outdated, you might want to start with a `daily` schedule until the dependencies are up-to-date, and then drop back to a weekly schedule.


Enabling version updates on forks [↗](#)

If you want to enable version updates on forks, there's an extra step. Version updates are not automatically enabled on forks when a `dependabot.yml` configuration file is present. This ensures that fork owners don't unintentionally enable version updates when they pull changes including a `dependabot.yml` configuration file from the original repository.

On a fork, you also need to explicitly enable Dependabot.

- 1 On your GitHub Enterprise Server instance, navigate to the main page of the repository.
- 2 Under your repository name, click  **Settings**. If you cannot see the "Settings" tab, select the `...` dropdown menu, then click **Settings**.



- 3 In the "Security" section of the sidebar, click  **Code security and analysis**.
- 4 Under "Code security and analysis", to the right of "Dependabot version updates", click **Enable** to allow Dependabot to initiate version updates.

Checking the status of version updates [↗](#)

After you enable version updates, the **Dependabot** tab in the dependency graph for the repository is populated. This tab shows which package managers Dependabot is configured to monitor and when Dependabot last checked for new versions.

<> CodeIssuesPull requests55ActionsProjectsWikiSecurity93InsightsSettings

Pulse

Contributors

Community

Traffic

Commits

Code frequency

Dependency graph

Dependency graph

DependenciesDependentsDependabot

go.mod ...Last checked 5 days ago

Cargo.toml ...Last checked 2 days ago

For information, see "[Listing dependencies configured for version updates](#)."

Disabling Dependabot version updates

You can disable version updates entirely by deleting the `dependabot.yml` file from your repository. More usually, you want to disable updates temporarily for one or more dependencies, or package managers.

- Package managers: disable by setting `open-pull-requests-limit: 0` or by commenting out the relevant `package-ecosystem` in the configuration file.
- Specific dependencies: disable by adding `ignore` attributes for packages or applications that you want to exclude from updates.

When you disable dependencies, you can use wild cards to match a set of related libraries. You can also specify which versions to exclude. This is particularly useful if you need to block updates to a library, pending work to support a breaking change to its API, but want to get any security fixes to the version you use.

Example disabling version updates for some dependencies

The example `dependabot.yml` file below includes examples of the different ways to disable updates to some dependencies, while allowing other updates to continue.

```
# `dependabot.yml` file with updates
# disabled for Docker and limited for npm

version: 2
updates:
  # Configuration for Dockerfile
  - package-ecosystem: "docker"
    directory: "/"
    schedule:
      interval: "weekly"
      # Disable all pull requests for Docker dependencies
      open-pull-requests-limit: 0

  # Configuration for npm
  - package-ecosystem: "npm"
    directory: "/"
    schedule:
      interval: "weekly"
    ignore:
      # Ignore updates to packages that start with 'aws'
      # Wildcards match zero or more arbitrary characters
      - dependency-name: "aws*"
      # Ignore some updates to the 'express' package
      - dependency-name: "express"
        # Ignore only new versions for 4.x and 5.x
        versions: ["4.x", "5.x"]
      # For all packages, ignore all patch updates
      - dependency-name: "*"
        update-types: ["version-update:semver-patch"]
```

For more information about checking for existing ignore preferences, see "[Configuration options for the dependabot.yml file](#)."

Legal

© 2023 GitHub, Inc. [Terms](#) [Privacy](#) [Status](#) [Pricing](#) [Expert services](#) [Blog](#)