About Dependabot version updates

In this article

You can use Dependabot to keep the packages you use updated to the latest versions.

Dependabot version updates are free to use for repositories (user-owned and organization-owned) on GitHub Enterprise Server, provided enterprise administrators enable the feature for your enterprise.

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 Dependabot version updates

Dependabot takes the effort out of maintaining your dependencies. You can use it to ensure that your repository automatically keeps up with the latest releases of the packages and applications it depends on.

You enable Dependabot version updates by checking a `dependabot.yml` configuration file into your repository. The configuration file specifies the location of the manifest, or of other package definition files, stored in your repository. Dependabot uses this information to check for outdated packages and applications. Dependabot determines if there is a new version of a dependency by looking at the semantic versioning (semver) of the dependency to decide whether it should update to that version. For certain package managers, Dependabot version updates also supports vendoring. Vendored (or cached) dependencies are dependencies that are checked in to a specific directory in a repository rather than referenced in a manifest. Vendored dependencies are available at build time even if package servers are unavailable. Dependabot version updates can be configured to check vendored dependencies for new versions and update them if necessary.

When Dependabot identifies an outdated dependency, it raises a pull request to update the manifest to the latest version of the dependency. For vendored dependencies, Dependabot raises a pull request to replace the outdated dependency with the new version directly. You check that your tests pass, review the changelog and release notes included in the pull request summary, and then merge it. For more information, see

"Configuring Dependabot version updates."

If you enable *security updates*, Dependabot also raises pull requests to update vulnerable dependencies. For more information, see "About Dependabot security updates."

When Dependabot raises pull requests, these pull requests could be for *security* or *version* updates:

- *Dependabot security updates* are automated pull requests that help you update dependencies with known vulnerabilities.
- *Dependabot version updates* are automated pull requests that keep your dependencies updated, even when they don't have any vulnerabilities. To check the status of version updates, navigate to the Insights tab of your repository, then Dependency Graph, and Dependabot.

GitHub Actions is required for Dependabot version updates and Dependabot security updates to run on GitHub Enterprise Server. Before you enable Dependabot updates, you must configure your GitHub Enterprise Server instance to use GitHub Actions with self-hosted runners. For more information, see "Enabling Dependabot for your enterprise."

Frequency of Dependabot pull requests

You specify how often to check each ecosystem for new versions in the configuration file: daily, weekly, or monthly.

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."

If you've enabled security updates, you'll sometimes see extra pull requests for security updates. These are triggered by a Dependabot alert for a dependency on your default branch. Dependabot automatically raises a pull request to update the vulnerable dependency.

Supported repositories and ecosystems

You can configure version updates for repositories that contain a dependency manifest or lock file for one of the supported package managers. For some package managers, you can also configure vendoring for dependencies. For more information, see "Configuration options for the dependabot.yml file."

Note: To ensure that GitHub Enterprise Server supports Dependabot updates for the latest

supported ecosystem versions, your enterprise owner must download the most recent version of the Dependabot action. For more information about the action, and for instructions about how to download the most recent version, see "Using the latest version of the official bundled actions."

Note: 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."

Dependabot doesn't support private GitHub dependencies for all package managers. See the details in the table below.

The following table shows, for each package manager:

- The YAML value to use in the `dependabot.yml` file
- The supported versions of the package manager
- Whether dependencies in private GitHub repositories or registries are supported
- Whether vendored dependencies are supported

Package manager	YAML value	Supported versions	Private repositories	Private registries	Vendoring
Bundler	`bundler`	v1, v2	✗	✓	✓
Cargo	`cargo`	v1	✓	✓ (git only)	✗
Composer	`composer`	v1, v2	✓	✓	✗
Docker	`docker`	v1	✓	✓	Not applicable
Hex	`mix`	v1	✗	✓	✗
elm-package	`elm`	v0.19	✓	✓	✗
git submodule	`gitsubmodule`	Not applicable	✓	✓	Not applicable
GitHub Actions	`github-actions`	Not applicable	✓	✓	Not applicable
Go modules	`gomod`	v1	✓	✓	✓
Gradle	`gradle`	Not applicable	✓	✓	✗
Maven	`maven`	Not applicable	✓	✓	✗
npm	`npm`	v6, v7, v8, v9	✓	✓	✗
NuGet	`nuget`	<= 4.8	✓	✓	✗
pip	`pip`	v21.1.2	✗	✓	✗
pipenv	`pip`	<= 2021-05-29	✗	✓	✗
pip-compile	`pip`	6.1.0	✗	✓	✗
pnpm	`npm`	v7, v8	✓	✓	✗

poetry	`pip`	v1	✕	✓	✕
pub	`pub`	v2	✕	✕	✕
Terraform	`terraform`	>= 0.13, <= 1.5.x	✓	✓	Not applicable
yarn	`npm`	v1, v2, v3	✓	✓	✓

Tip: For package managers such as `pipenv` and `poetry`, you need to use the `pip` YAML value. For example, if you use `poetry` to manage your Python dependencies and want Dependabot to monitor your dependency manifest file for new versions, use `package-ecosystem: "pip"` in your `dependabot.yml` file.

Cargo

Private registry support applies to git registries, and doesn't include cargo registries.

Docker

Dependabot can add metadata from Docker images to pull requests for version updates. The metadata includes release notes, changelogs and the commit history. Repository administrators can use the metadata to quickly evaluate the stability risk of the dependency update.

In order for Dependabot to fetch Docker metadata, maintainers of Docker images must add the `org.opencontainers.image.source` label to their Dockerfile, and include the URL of the source repository. Additionally, maintainers must tag the repository with the same tags as the published Docker images. For an example, see the `dependabot-fixtures/docker-with-source` repository. For more information on Docker labels, see Extension image labels and BUILDX_GIT_LABELS in the Docker documentation.

Dependabot can update Docker image tags in Kubernetes manifests. Add an entry to the Docker `package-ecosystem` element of your `dependabot.yml` file for each directory containing a Kubernetes manifest which references Docker image tags. Kubernetes manifests can be Kubernetes Deployment YAML files or Helm charts. For information about configuring your `dependabot.yml` file for `docker`, see "`package-ecosystem`" in "Configuration options for the dependabot.yml file."

Dependabot supports both public and private Docker registries. For a list of the supported registries, see "`docker-registry`" in "Configuration options for the dependabot.yml file."

GitHub Actions

Dependabot only supports updates to GitHub Actions using the GitHub repository syntax, such as actions/checkout@v4. Docker Hub and GitHub Packages Container registry URLs are currently not supported.

Dependabot supports both public and private repositories for GitHub Actions. For private registry configuration options, see "`git`" in "Configuration options for the dependabot.yml file."

Gradle

Gradle is supported for Dependabot version updates only.

Dependabot doesn't run Gradle but supports updates to the following files:

- `build.gradle`, `build.gradle.kts` (for Kotlin projects)
- `gradle/libs.versions.toml` (for projects using a standard Gradle version catalog)
- Files included via the `apply` declaration that have `dependencies` in the filename. Note that `apply` does not support `apply to`, recursion, or advanced syntaxes (for example, Kotlin's `apply` with `mapOf`, filenames defined by property).

Maven

Dependabot doesn't run Maven but supports updates to `pom.xml` files.

NuGet CLI

Dependabot doesn't run the NuGet CLI but does support most features up until version 4.8.

pip and pip-compile

In addition to supporting updates to `requirements.txt` files, Dependabot supports updates to `pyproject.toml` files if they follow the PEP 621 standard.

pnpm

pnpm is supported for Dependabot version updates only. Dependabot security updates are not currently supported.

pub

Dependabot won't perform an update for `pub` when the version that it tries to update to is ignored, even if an earlier version is available.

yarn

Dependabot supports vendored dependencies for v2 onwards.

If your repository already uses an integration for dependency management, you will need to disable this before enabling Dependabot.

About automatic deactivation of Dependabot updates

When maintainers of a repository stop interacting with Dependabot pull requests, Dependabot temporarily pauses its updates and lets you know. This automatic opt-out behavior reduces noise because Dependabot doesn't create pull requests for version and security updates, and doesn't rebase Dependabot pull requests for inactive repositories.

The automatic deactivation of Dependabot updates only applies to repositories where Dependabot has opened pull requests but the pull requests remain untouched. If Dependabot hasn't opened any pull requests, Dependabot will never become paused.

An active repository is a repository for which a user (not Dependabot) has carried out *any* of the actions below in the last 90 days:

- Merge or close a Dependabot pull request on the repository.
- Make a change to the `dependabot.yml` file for the repository.
- Manually trigger a security update or a version update.
- Enable Dependabot security updates for the repository.
- Use `@dependabot` commands on pull requests.

An inactive repository is a repository that has at least one Dependabot pull request open for more than 90 days, has been enabled for the full period, and where none of the actions listed above has been taken by a user.

When Dependabot is paused, GitHub adds a notice to the body of all open Dependabot pull requests, and assigns a `dependabot-paused` label to these pull requests. You'll also see a banner notice in the UI of the **Settings** tab of the repository (under **Code security and analysis**, then **Dependabot**), as well in the list of Dependabot alerts (if Dependabot security updates are affected). Additionally, you will be able to see whether Dependabot is paused at the organization-level in the security overview. The `paused` status will also be visible via the API. For more information, see "Repositories" in the REST API documentation.

As soon as a maintainer interacts with a Dependabot pull request again, Dependabot will unpause itself:

- Security updates are automatically resumed for Dependabot alerts.
- Version updates are automatically resumed with the schedule specified in the `dependabot.yml` file.

Dependabot also stops rebasing pull requests for version and security updates after 30 days, reducing notifications for inactive Dependabot pull requests.

About notifications for Dependabot version updates

You can filter your notifications on GitHub to show notifications for pull requests created by Dependabot. For more information, see "Managing notifications from your inbox."