ON THIS PAGE

Build Plugins

Netlify Build Plugins are integrations that extend the functionality of the Netlify Build process. You can install plugins made by others, or write your own. You can save them locally in your repository, or share them with others using npm.

Build Plugins expand what your Netlify builds are capable of. For example, you can use plugins to:

- speed up builds by optimizing and debugging your build cache

analyze and optimize site asset handling for better runtime performance

generate content like sitemaps, RSS feeds, and search indexes

Install a plugin

To get a sampling of what plugins can do, visit the Integrations Hub . Plugins listed there can be installed directly from the Netlify UI. They can also be installed using the Netlify configuration file, which allows more configuration options. Netlify automatically installs plugins or runtimes recommended for certain frontend frameworks when you link a repository for a new site.

UI installation

You can install Build Plugins through the Integrations Hub.

1. In the Netlify UI, select the **Integrations** tab from any team-level page. Or go directly to the Integrations Hub .
2. Search for the integration you want or select a category to discover new integration options.
3. On the integration page, select **Add Build Plugin**.
4. Follow the installation guidance in the Netlify UI.
5. To use your new plugin, visit the **Deploys** tab for your site and select **Trigger deploy.**

Consider the context

Required environment variables

Though most plugins that can be installed using the Netlify UI require no configuration for default operation, some may require you to set one or more build environment variables. Refer to the plugin's documentation, linked from the **Options** menu for the plugin listing in the Netlify UI, for details.

File-based installation

File-based plugin installation allows advanced plugin configuration.

You can use file-based installation for either of the following:

- installing local plugins that you write and store in your repository
- accessing a wide selection of plugins published by the community on npm

In both cases, you configure settings in `netlify.toml`. For a plugin published to npm, you also add it as a dependency. Then you can test or run the plugin as part of a build.

Configure settings

To run a plugin during your build, add it to a Netlify configuration file stored in your site's base directory. A plugin configured globally with `[[plugins]]` runs in all deploy contexts, but you can also configure a plugin by deploy context.

toml

```toml
# Configuration for a plugin published to npm
[[plugins]]
package = "netlify-plugin-lighthouse"

  [plugins.inputs]
  output_path = "reports/lighthouse.html"

# Configuration for a local plugin
[[plugins]]
package = "/plugins/netlify-plugin-hello-world"
```

Each `[[plugins]]` entry accepts two keys:

`package` (required):

for a plugin installed from npm, the npm package name of the plugin.

for a local plugin, the path to a directory containing the plugin's `index.js` and `manifest.yml` files. The `package` value for a local plugin must start with `.` or `/`.

`inputs`: custom settings that the plugin author may specify as required or available for configuring the plugin. To specify `inputs` per deploy context, refer to configure by deploy context.

For npm-published plugins, you can find these details in each plugin's package documentation on the npm Public Registry .

Sometimes order matters

> published plugin's README should indicate if order is important to that plugin's functionality.

Configure by deploy context

Using specific settings in your Netlify configuration file, you can limit a build plugin to run in a certain deploy context only, or you can configure a plugin's `inputs` settings differently per context.

Here's an example configuration that runs the Sitemap plugin in the context of production deploys only.

toml

```
# Use double brackets since `plugins` is an array of tables.
[[context.production.plugins]]
package = "@netlify/plugin-sitemap"
```

And here's an example configuration that runs the Cypress plugin differently based on deploy contexts.

toml

```
# Use Cypress plugin for this site.
# This section, by itself, configures the plugin
# for all deploy contexts (production, branch deploys, Deploy P
[[plugins]]
package = "netlify-plugin-cypress"
  [plugins.inputs]
  record = true

# Don't record Cypress tests in Deploy Previews.
# Since this entry is more specific, it overrides the entry abo
```

```
# Use single brackets since `inputs` is an object property
[context.deploy-preview.plugins.inputs]
record = false
```

This configuration records test results and artifacts on the Cypress Dashboard for production and branch deploys only, not Deploy Previews.

> **UI-installed plugins run on all contexts**
>
> To limit a plugin to certain deploy contexts, ensure that you've configured the plugin for your site using file-based installation only and not UI installation.

Next steps

If you're installing a local plugin, you can run and test it after configuration. Otherwise, you'll add a dependency to `package.json`.

Add dependency

For a plugin from npm, there's an additional step beyond editing the Netlify configuration file. You must use npm, yarn, or another Node.js package manager to add the plugin to `devDependencies` in your site's `package.json`.

> **Evaluate the plugin code**
>
> Plugins available on npm but not yet listed in the Integrations Hub have not been reviewed or approved by Netlify staff. Review plugin code for security concerns before installing.

dependency.

npm Yarn

sh

```
# Replace `BUILD_PLUGIN_NAME` with a real plugin name,
# like `netlify-plugin-lighthouse`
npm install -D BUILD_PLUGIN_NAME
```

Run and test

When you save your changes to your repository and push them to your Git provider, the build that's triggered on Netlify will run with plugins installed for that deploy context. If you would like to test a plugin before running it in a production build, you can use a branch deploy or Deploy Preview, or you can run the build locally with Netlify CLI.

Automatic installation

When you link a repository for a new site, Netlify runs a framework detection utility to determine whether your site uses a particular frontend framework. Certain frameworks have recommended Build Plugins or runtimes. These help extend the functionality of the Netlify Build process to support key framework-specific features. Recommended plugins and runtimes may have site conditions requirements, such as a minimum Node.js version.

If your new site uses a framework with recommended plugins or runtimes, Netlify checks whether these are already installed in a Netlify configuration file. If not, Netlify automatically installs them. These automatically installed plugins run in all deploy contexts.

Manage plugin versions

Netlify encourages plugin authors to regularly update functionality and release new versions using semantic versioning . Minor plugin version updates introduce only backward compatible new features, while major plugin version updates can introduce breaking changes. Refer to the plugin's changelog, linked from the **Options** menu for the plugin listing in the Netlify UI, for version details.

The steps for managing plugin versions for your site depend on the plugin installation method.

For plugins installed in the UI or installed automatically, Netlify updates your site for minor plugin version releases automatically. To manage major plugin updates for a site, go to **Site settings > Plugins** and take one of these steps:

> To upgrade to a new major version for an installed plugin, select **Change version**.
>
> To roll back to a previous major version for an installed plugin, select **Options > Change version**.

Subsequent builds will use the plugin version that you've chosen and confirmed.

For plugins installed through file-based installation, you can manage versions in your site's `package.json` file under `devDependencies`.

Remove a plugin

1. For your selected site, go to **Site settings > Plugins**.
2. Find the plugin you want to remove.
3. In the plugin's **Options** menu, select **Uninstall**.

   Subsequent builds will not use the uninstalled plugin.

For plugins installed through file-based installation:

1. Open your site's `netlify.toml` .
2. Delete or comment out the plugin's configuration fields.

   When you push your committed changes, the resulting build will run without the plugin. If you're removing an npm-published plugin and want to avoid installing code you won't use, you can uninstall the plugin package using npm .

Create a plugin

Once you've had a chance to try out plugins, you may want to make one of your own. To learn how, visit the create plugins doc.

Get help

Netlify Build Plugins are created by our partners and developers at Netlify and in the community. If you need help with a plugin, contact the plugin author by submitting an issue on the plugin repository. For plugins in the Netlify UI, you can find a link to the plugin issues under the **Options** menu

For more general questions, or to discuss Build Plugins with other members of the community, visit the Netlify Support Forums .

More Build Plugins resources

Integrations Hub

Create Build Plugins using build events

Share Build Plugins

Create Build Plugins →

Did you find this doc useful?

Your feedback helps us improve our docs.

What else would you like to tell us about this doc?

Send