

Error List

You may encounter a variety of errors when you interact with the Vercel platform. This section focuses on errors that can happen when you interact with the Vercel Dashboard.


Table of Contents ▾

Missing public directory

The **build step** will result in an error if the output directory is missing, empty, or invalid (for example, it is not a directory). To resolve this error, you can try the following steps:

- Make sure the **output directory** is specified correctly in project settings
- If the output directory is correct, check the build command (**documentation**) or the **root directory**
- Try running the build command locally and make sure that the files are correctly generated in the specified output directory

Missing build script

 This is only relevant if you're using Vercel CLI 16.7.3 or older.

Suppose your project contains a `package.json` file, no `api` directory, and no `vercel.json` configuration. In that case, it is expected to provide a **build script** that performs a static build of your frontend and outputs it to a `public` directory at the root of your project.

When properly configured, your `package.json` file would look similar to this:

package.json

```
1 {
2   "scripts": {
```

On this page

- Missing public directory
- Missing build script
- Maximum team member requests
- Inviting users to team who request access
- Request access with the required C account
- Blocked scopes
- Unused build and development settings
- Unused Serverless Function region setting
- Invalid route source pattern
- Invalid route destination segment
- Failed to install builder dependencies
- Mixed routing properties
- Conflicting configuration files
- Conflicting functions and builds configuration
- Unsupported functions configuration with Nextjs
- Deploying Serverless Functions to multiple regions
- Unmatched function pattern
- Cannot load project settings
- Project name validation
- Repository connection limitation
- Domain verification through CLI
- Leaving the team
- Git Default ignore list
- GitHub app installation not found
- Preview branch used as production branch
- Lost Git repository access
- Production deployment cannot be redeployed
- SSL certificate deletion denied
- Production branch used as preview branch

Ask 

```
3     "build": "[my-framework] build --(
4   }
5 }
```

stating that a different Git account must be used to request access to the team.

Once the Git account is connected to the Hobby team on Vercel, it is possible to request access to the team. A **Team Owner** can then approve or decline this access request. If the request was approved, the failed commit would be retried, and the following commits would not fail due to missing team access.

Blocked scopes

A Hobby or team on Vercel can be blocked if it violates our [fair use guidelines](#) or [Terms of Service](#).

Blocked Hobby teams or teams cannot create new deployments, or be invited to new teams. Please contact [Vercel Support](#) if you need help.

Unused build and development settings

A Project has several settings that can be found in the dashboard. One of those sections, **Build & Development Settings**, is used to change the way a Project is built.

However, the **Build & Development Settings** are only applied to **zero-configuration** deployments.

If a deployment defines the `builds` configuration property, the **Build & Development Settings** are ignored.

Unused Serverless Function region setting

A Project has several settings that can be found in the dashboard. One of those settings, **Serverless Function Region**, is used to select the **region** where your Serverless Functions execute.

If a deployment defines the `regions` configuration property in `vercel.json`, the **Serverless Function Region** setting is ignored.

If a CLI Deployment defines the `--regions` option, the **Serverless Function Region** setting is ignored.

Invalid route source pattern

The `source` property follows the syntax from `path-to-regexp`, not the `RegExp` syntax.

For example, negative lookaheads must be wrapped in a group.

Before

```
vercel.json
1 {
2   "source": "/feedback/(?!general)",
3   "destination": "/api/feedback/general",
4 }
```

After

```
vercel.json
1 {
2   "source": "/feedback/(?!(general)).*",
3   "destination": "/api/feedback/general",
4 }
```

Invalid route destination segment

The `source` property follows the syntax from `path-to-regexp`.

A colon (`:`) defines the start of a named segment parameter.

A named segment parameter defined in the `destination` property must also be defined in the `source` property.

Before

```
vercel.json
1 {
2   "source": "/feedback/:type",
3   "destination": "/api/feedback/:id",
4 }
```

After

```
vercel.json
1 {
2   "source": "/feedback/:id",
3   "destination": "/api/feedback/:id",
4 }
```

```
1 {  
2   "source": "/feedback/:id",  
3   "destination": "/api/feedback/:id"  
4 }
```

Failed to install builder dependencies

When running the `vercel build` or `vercel dev` commands, `npm install` errors can be encountered if `npm` was invoked to install Builders that are defined in your `vercel.json` file.

`npm install` may fail if:

- `npm` is not installed
- Your internet connection is unavailable
- The Builder that is defined in your configuration is not published to the npm registry

Double-check that the name and version of the Builder you are requesting is correct.

Mixed routing properties

If you have `rewrites`, `redirects`, `headers`, `cleanUrls` or `trailingSlash` defined in your configuration file, then `routes` cannot be defined.

This is a necessary limitation because `routes` is a lower-level primitive that contains all of the other types. Therefore, it cannot be merged safely with the new properties.

See the [Upgrading Routes](#) section for examples of `routes` compared to the new properties.

Conflicting configuration files

For backward compatibility purposes, there are two naming conventions for configuration files used by Vercel CLI (for example `vercel.json` and `now.json`). Both naming conventions are supported, however only *one* may be defined at a time. Vercel CLI will output an error message if both naming conventions are used at the same time.

These conflicting configuration errors occur if:

- Both `vercel.json` and `now.json` exist in your project.
Solution: Delete the `now.json` file
- Both `.vercel` and `.now` directories exist in your project.
Solution: Delete the `.now` directory
- Both `.vercelignore` and `.nowignore` files exist in your project.
Solution: Delete the `.nowignore` file
- Environment Variables that begin with `VERCEL_` have a conflicting Environment Variable that begins with `NOW_`.
Solution: Only define the `VERCEL_` prefixed Environment Variable

Conflicting functions and builds configuration

There are two ways to configure Serverless Functions in your project: `functions` or `builds`. However, only one of them may be used at a time - they cannot be used in conjunction.

For most cases, it is recommended to use the `functions` property because it supports more features, such as:

- Allows configuration of the amount of memory that the Serverless Function is provided with
- More reliable because it requires a specific npm package version for the `runtime` property
- Supports "clean URLs" by default, which means that the Serverless Functions are automatically accessible without their file extension in the URL

However, the `builds` property will remain supported for backward compatibility purposes.

Unsupported functions configuration with Nextjs

When using Next.js, only `memory` and `maxDuration` can be configured within the `functions` property. Next.js automatically handles the other configuration values for you.

Deploying Serverless Functions to multiple regions

It's possible to deploy Serverless Functions to multiple regions. This functionality is only available to Enterprise teams.

On the Pro plan, the limitation has existed since the launch of the current pricing model but was applied on **July 10, 2020**. For Projects created on or after the date, it's no longer possible to deploy to multiple regions.

To select the region closest to you, read our [guide](#) on choosing deployment regions for Serverless Functions.

Unmatched function pattern

The `functions` property uses a glob pattern for each key. This pattern must match Serverless Function source files within the `api` directory.

If you are using Next.js, Serverless Functions source files can be created in the following:

- `pages/api` directory
- `src/pages/api` directory
- `pages` directory when the module exports `getServerSideProps`
- `src/pages` directory when the module exports `getServerSideProps`

Additionally, if you'd like to use a Serverless Function that isn't written with Node.js, and in combination with Next.js, you can place it in the `api` directory (provided by the platform), since `pages/api` (provided by Next.js) only supports JavaScript.

Not Allowed

vercel.json

```
1 {
2   "functions": {
3     "users/**/*.js": {
4       "maxDuration": 30
5     }
6   }
7 }
```

Allowed

```
vercel.json

1 {
2   "functions": {
3     "api/users/**/*.js": {
4       "maxDuration": 30
5     }
6   }
7 }
```

Allowed (Next.js)

```
vercel.json

1 {
2   "functions": {
3     "pages/api/users/**/*.js": {
4       "maxDuration": 30
5     }
6   }
7 }
```

Cannot load project settings

If the Project configuration in `.vercel` belongs to a team you are not a member of, attempting to deploy the project will result in an error.

This can occur if you clone a Git repository that includes the `.vercel` directory, or you are logged in to the wrong Vercel account.

To fix, remove the `.vercel` directory and redeploy to link the project again by running these commands.

On macOS and Linux:

```
$ rm -rf .vercel
$ vercel
```

On Windows:

```
$ rmdir /s /q .vercel
$ vercel
```

Project name validation

Project names can only consist of up to one hundred alphanumeric lowercase characters. Hyphens can be used in between words in the name, but never at the start or end.

Repository connection limitation

The amount of Vercel Projects that can be connected with the same Git repository is limited depending on your plan.

If you have reached the limitation and would like to connect a new project to the repository, you will need to disconnect an existing project from the same Git repository.

To increase this limit, please [contact our Sales Team](#).

Domain verification through CLI

To verify your domain, point the domain to Vercel by configuring our nameservers or a DNS Record. You can learn more about what to do for your Domain by running `vercel domains inspect <domain>`, where `<domain>` is the domain you're interested in.

Alternatively, if you already added the domain to a project, read [the configuring a domain](#) section of the custom domain documentation.

Leaving the team

You cannot leave a team if you are the last remaining **Owner** or the last confirmed **Member**. In order to leave the Team, first designate a different confirmed **Member** to be an **Team Owner**.

If you are the only remaining **Member**, you should instead delete the Team.

Git Default ignore list

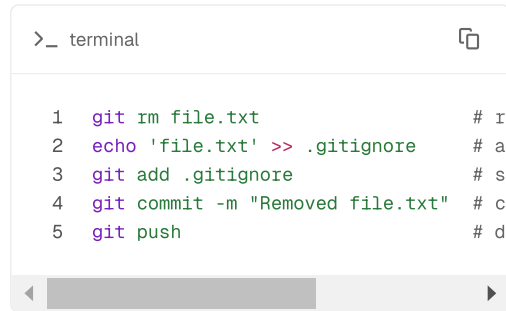
Deployments created using Vercel CLI will automatically ignore **several** files for security and performance reasons.

However, these files are *not* ignored for deployments created using Git and a warning is printed instead. This is because `.gitignore` determines which files should be ignored.

If the file was intentionally committed to Git, you can ignore the warning.

If the file was accidentally committed to Git, you can remove it using the following

commands:



```
>_ terminal

1  git rm file.txt           # I
2  echo 'file.txt' >> .gitignore # a
3  git add .gitignore        # s
4  git commit -m "Removed file.txt" # c
5  git push                  # d
```

GitHub app installation not found

In some cases, signing up with GitHub fails due to GitHub's database inconsistencies.

When you connected your Hobby team with your GitHub account, the **Vercel GitHub App** was installed on your GitHub account and then GitHub notified Vercel that the app was successfully installed.

However, Vercel was unable to retrieve the app installation from GitHub, which made it appear as if the **Vercel GitHub App** was never installed.

In order to solve this issue, wait a couple of minutes and try connecting to GitHub again. If you are still unable to connect, please contact **GitHub Support** to determine why the **Vercel GitHub App** was not able to be installed.

Preview branch used as production branch

If you have configured a custom Git branch for a domain or an environment variable, it is considered a preview domain and a preview environment variable. Because of this, the Git branch configured for it is considered a **preview branch**.

When configuring the production branch in the project settings, it is not possible to use a preview branch.

If you still want to use this particular Git branch as a production branch, please follow these steps:

1. Assign your affected domains to the production environment (clear out the Git branch you've defined for them)
2. Assign your affected environment variables to the production environment (clear out the Git branch you've defined for them)

Afterwards, you can use the Git branch you originally wanted to use as a production branch.

Lost Git repository access

In order for Vercel to be able to deploy commits to your Git repository, a Project on Vercel has to be connected to it.

This connection is interrupted if the Git repository is deleted, archived, or if the Vercel App was uninstalled from the corresponding Git account or Git organization. Make sure none of these things apply.

Additionally, when using GitHub, the connection is also interrupted if you or a **Team Member** modifies the access permissions of the Vercel GitHub App installed on the respective personal GitHub account or GitHub organization.

To verify the access permissions of the Vercel GitHub App installed on your personal GitHub account, navigate to the **Applications page** and select **Vercel** under **Installed GitHub Apps**. You will see a list of Git repositories that the GitHub App has access to. Make sure that the Git repository you're looking to connect to a Vercel Project is listed there.

To verify the access permissions of the Vercel GitHub App installed on your GitHub organization, select **Vercel** under **Installed GitHub Apps** in the organization settings. You will see a list of Git repositories that the GitHub App has access to. Make sure that the Git repository you're looking to connect to a Vercel Project is listed there.

Production deployment cannot be redeployed

You cannot redeploy a production deployment if a more recent one exists.

The reason is that redeploying an old production deployment would result in overwriting the most recent source code you have deployed to production.

To force an explicit overwrite of the current production deployment, select **Promote** instead.

SSL certificate deletion denied

Certain SSL Certificates associated with your Hobby team or team (i.e. Wildcard SSL Certificates for your Vercel Project's staging domains) are automatically generated by the Vercel platform.

Because these SSL Certificates are managed by the Vercel platform, they cannot be manually deleted on the Vercel Dashboard – nor through Vercel CLI.

Custom SSL Certificates may be uploaded to teams on the **Enterprise plan**, which are allowed to be manually deleted.

Production branch used as preview branch

The Git branch that is configured using the **production branch** field in the project settings, is considered the branch that contains the code served to your visitors.

If you'd like to assign a domain or environment variable to that particular Git branch, there's no need to manually fill it in.

By default, if no custom Git branch is defined for them, domains are already assigned to the production branch. The same is true for environment variables: If no custom Git branch is defined for them and **Production** is selected as an environment, they're already assigned to the production branch.

If you still want to enter a specific Git branch for a domain or an environment variable, it has to be a **preview branch**.

Command not found in vercel dev

The *"Command not found"* error message happens when a sub-process that `vercel dev` is attempting to create is not installed on your local machine. You need to install the particular program onto your operating system before `vercel dev` will work correctly.

For example, you may see the error *"Command not found: go"* if you are writing a Serverless Function in Go, but do not have the `go` binary installed. In this case you need to install `go`

first, and then try invoking your Serverless Function again.

Recursive invocation of commands

Why this error occurred

You have configured one of the following for your Project:

- The **Build Command** defined in the Project Settings invokes `vercel build`
- The **Development Command** defined in the Project Settings invokes `vercel dev`

Because the Build Command is invoked by `vercel build` when deploying, it cannot invoke `vercel build` itself, as that would cause an infinite recursion.

The same applies to the Development Command: When developing locally, `vercel dev` invokes the Development Command, so it cannot invoke `vercel dev` itself.

Possible ways to fix it

Adjust the Build and Development Commands defined for your Project to not invoke `vercel build` or `vercel dev`.

Instead, they should invoke the Build Command provided by your framework.

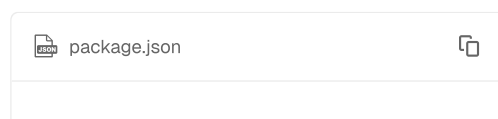
If you are unsure about which value to provide, disable the **Override** option in order to default to the preferred settings for the **Framework Preset** you have selected.

Pnpm engine unsupported

`ERR_PNPM_UNSUPPORTED_ENGINE` occurs when `package.json#engines.pnpm` does not match the currently running version of `pnpm`.

To fix, do one of the following:

- Set the env var `ENABLE_EXPERIMENTAL_COREPACK` to 1, and make sure the `packageManager` value is set correctly in your `package.json`



```

1  {
2    "engines": {
3      "pnpm": "^7.5.1"
4    },
5    "packageManager": "pnpm@7.5.1"
6  }

```

- Remove the `engines.pnpm` value from your `package.json`

You cannot use `engine-strict` to solve this error. `engine-strict` only handles dependencies.

Invalid Edge Config connection string

This error occurs when attempting to create a deployment where at least one of its environment variables contains an outdated Edge Config connection string. A connection string can be outdated if either the Edge Config itself was deleted or if the token used in the connection string is invalid or has been deleted.

To resolve this error, delete or update the environment variable that contains the connection string. In most cases, the environment variable is named `EDGE_CONFIG`.

Globally installed `@vercel/speed-insights` or `@vercel/analytics` packages

You must reference `@vercel/speed-insights` or `@vercel/analytics` packages in your application's `package.json` file. This error occurs when you deploy your application with these packages globally available, but not referenced in your `package.json` file, like in a monorepo.

To fix this error, add the packages to your `package.json` file as a dependency.

Oversized Incremental Static Regeneration page

Incremental Static Regeneration (ISR) responses that are greater than 20 MB result in pages not rendering in production with a `FALLBACK_BODY_TOO_LARGE` error. This affects all Next.js build time pre-rendering, and other frameworks that use Prerender Functions.

Last updated on September 26, 2024

Was this helpful?



Products

- AI
- Enterprise
- Next.js
- Observability
- Previews
- Rendering
- Security
- Turbo
- v0

Company

- About
- Blog
- Careers
- Changelog
- Contact Us
- Customers
- Partners
- Privacy Policy
- Legal

Resources

- Community
- Docs
- Experts
- Guides
- Help
- Integrations
- Pricing
- Resources
- Templates

Social

- GitHub
- LinkedIn
- Twitter
- YouTube