

Troubleshooting Guide



- [How do I get more verbose information about my WP-CLI installation?](#)
 - [What should I do, before I start debugging issues?](#)
 - [What should I do if the WP-CLI output is different than expected?](#)
 - [Environment Variables](#)
 - [WP-CLI Configuration Files](#)
 - [WP-CLI Packages](#)
 - [WordPress Configuration File \(wp-config.php\)](#)
 - [WordPress Extensions](#)
 - [What should I do if WP-CLI reports an error?](#)
 - [I have checked all above, but still have an issue. Where can I report issues?](#)
-

Before you start to report a new issue on the GitHub repository, make sure to check your local installation, as some settings may typically result in an unexpected outcome of your WP-CLI commands.

How do I get more verbose information about my WP-CLI installation?

WP-CLI offers the command `wp --info`, which provides you with a lot of information about your WP-CLI install environment. The output will tell you,

- what operating system and shell you work on
- which PHP binary is used to run WP-CLI
- what version of PHP is used
- where to find WP-CLI's root directory
- where to find the vendor directory of WP-CLI
- which WP-CLI binary you are currently working with (phar path)
- where WP-CLI packages are stored
- where to find global and project configuration files for WP-CLI and
- which version of WP-CLI you use.

What should I do, before I start debugging issues?

Before you start to debug issues, make sure you are using the latest version of WP-CLI. The latest version may already have solved an issue you experience. The command `wp cli update` will upgrade your WP-CLI version or confirm you already use the latest version. If the installation hangs, please ensure that you are allowed to connect to GitHub using SSL (port 443) and git (port 9418) for outbound connections.

What should I do if the WP-CLI output is different than expected?

Before starting to investigate a bug, you should be aware of the factors that can change the default behavior of WP-CLI and how you can check whether they might be at the root of the issue. There are five main subsystems for modifying this default behavior: environment variables, configuration files, WP-CLI packages, `wp-config.php` file and WordPress extensions (plugins, themes, must-use plugins, drop-ins).

Environment Variables

The setup of your working environment is a prerequisite for running WP-CLI. You can check your current environment settings with the shell command `env`.

If you want to run WP-CLI remotely using SSH, it is required that the command `wp` is accessible on the path of the remote server. WP-CLI's behavior can also be changed at runtime through the use of environment variables:

- `WP_CLI_CACHE_DIR` - Directory to store the WP-CLI file cache. Default is `~/.wp-cli/cache/`.
- `WP_CLI_CONFIG_PATH` - Path to the global `config.yml` file. Default is `~/.wp-cli/config.yml`.
- `WP_CLI_CUSTOM_SHELL` - Allows the user to override the default `/bin/bash` shell used.
- `WP_CLI_DISABLE_AUTO_CHECK_UPDATE` - Disable WP-CLI automatic checks for updates.
- `WP_CLI_DOCKER_NO_TTY` - Ignore detected tty status and remove the `--tty` flag from `docker exec` when running WP-CLI against a docker container via ssh.
- `WP_CLI_DOCKER_NO_INTERACTIVE` - Ignore detected tty status and remove the `--interactive` flag from `docker exec` when running WP-CLI against a docker container via ssh.
- `WP_CLI_PACKAGES_DIR` - Directory to store packages installed through WP-CLI's package management. Default is `~/.wp-cli/packages/`.
- `WP_CLI_PHP` - PHP binary path to use when overriding the system default (only works for non-Phar installation).
- `WP_CLI_PHP_ARGS` - Arguments to pass to the PHP binary when invoking WP-CLI (only works for non-Phar installation).

- `WP_CLI_SSH_PRE_CMD` - When using `--ssh=<ssh>`, perform a command before WP-CLI calls WP-CLI on the remote server.
- `WP_CLI_SSH_BINARY` - When using `--ssh=<ssh>`, this is the command that will be called on the remote machine in place of `wp`.
- `WP_CLI_STRICT_ARGS_MODE` - Avoid ambiguity by telling WP-CLI to treat any arguments before the command as global, and after the command as local.
- `WP_CLI_SUPPRESS_GLOBAL_PARAMS` - Set to `true` to skip showing the global parameters at the end of the help screen. This saves screen estate for advanced users.
- `WP_CLI_FORCE_USER_LOGIN` - Set to `1` to force the value provided to the `--user` flag to be interpreted as a login instead of an ID, to get around ambiguous types.
- `WP_CLI_EARLY_REQUIRE` - Load one or more custom PHP files (comma-separated) early on in the bootstrap process.
- `WP_CLI_REQUIRE` - Load one or more custom PHP files (comma-separated) after the bootstrap process. This has the same effect as `--require` on the command line or a `require` mapping in a `wp-cli.yml` config file.
- `WP_CLI_USER_AGENT` - Sets the `$_SERVER['HTTP_USER_AGENT']` value that some plugins and tools rely on.

To set an environment variable on demand, you can place the environment variable definition before the WP-CLI command you mean to run (e.g. `EDITOR=vim wp post edit 1`); to overwrite environment variables, use `export VARIABLE=value` in your `~/.bashrc` or `~/.zshrc`.

WP-CLI Configuration Files

Configuration files let you customize the behavior of WP-CLI to adapt it to your personal needs or those of your project. If the configuration file is incorrect or includes unwanted modifications to the default behavior, the output of WP-CLI will most likely reflect the error.

Note that a project configuration file can override settings in a global configuration file.

Rename or delete your configuration file(s) and compare the result, to find out if a configuration setting might have caused the issue.

WP-CLI Packages

WP-CLI packages are community-maintained projects built on WP-CLI. They can contain WP-CLI commands, but they can also just extend WP-CLI in some way. While WP-CLI might be perfectly working, a package with errors can cause unexpected results, too. To skip loading all installed packages, use `wp --skip-packages`.

WordPress Configuration File (`wp-config.php`)

Errors may result from moving or editing `wp-config.php` beyond what WP-CLI supports. If you get a parse error, check the file encoding of your `wp-config.php` (UTF-8 without BOM).

Make sure that the line `require_once(ABSPATH . 'wp-settings.php');` remains in the `wp-config.php` file and don't modify `wp-config.php` beyond constant definitions. If you call WordPress functions within `wp-config.php`, PHP will fail with a fatal error.

If you want to use `$_SERVER['HTTP_HOST']` in your `wp-config.php`, you'll need to set a default value in WP-CLI context:

```
if ( defined( 'WP_CLI' ) && WP_CLI && ! isset( $_SERVER['HTTP_HOST'] ) ) {  
    $_SERVER['HTTP_HOST'] = 'example.com';  
}
```

Instead of `$_SERVER['document_root']` use `dirname( __FILE__ )` or similar.

WordPress Extensions

WordPress Themes and Plugins might conflict with the loading process of WP-CLI or interfere e.g. by redirecting users. They often make assumptions that are not true in the WP-CLI context, like using the hostname of the current request (which is a concept that does not exist within command-line context).

You can bypass single plugins and themes (e.g. `--skip-plugins=akismet`) or skip them entirely (`wp --skip-plugins --skip-themes`).

What should I do if WP-CLI reports an error?

Common problems are the result of changes in your `wp-config.php` or web server configuration.

The section "[Common issues](#)" of the handbook lists the most common error messages and explains how to handle these errors.

Try to reproduce the issue in a fresh installation of WordPress with a default Theme (Twenty ...) and no plugins installed. If the issue only reproduces in a custom environment, then the issue is a bug in your environment, not WP-CLI. Also make sure to enable WordPress' debugging mode by setting the constant `define( 'WP_DEBUG', true );` in your `wp-config.php`. It may reveal problems with your existing WordPress installation, not caused by WP-CLI.

Additionally you can use WP-CLI's global parameter `--debug` to show all PHP errors and add verbosity to WP-CLI bootstrap.

I have checked all above, but still have an issue. Where can I report issues?

If you think you've found a bug, we'd love to hear from you to get it fixed.

Bug reporting for WP-CLI is handled on GitHub. Before you create a new issue, please [search existing issues](#) to see if there's an existing resolution to it. If there isn't an open or fixed issue for your bug, please [follow our guidelines for submitting a bug report](#) to make sure it gets addressed in a timely manner. Providing the summary, steps to reproduce, environmental details, and other specifics identified below will help guarantee you are submitting a full bug report.

Please provide us with:

- a summary of the issue in narrative form,
- a detailed and specific list of steps to reproduce the issue,
- details of the environment you're working on,
- a description, how the bug affects your usage (i.e. expected results compared with actual results),
- severity level,
- possible workarounds and
- relevant diagnostics, such as crash reports, stack traces or debug output.

You can find a verbose description of these details of an issue report in the [guidelines for submitting a bug report](#).

🔖 #4

📄 cocobook-sync 🔖 2026-10-28 10:59:56 UTC

📄 cocobook-sync 🔖 2026-10-29 01:42:30 UTC