

Config



- [Global parameters](#)
 - [Config files](#)
 - [Remote \(SSH\) configuration](#)
 - [Context configuration](#)
 - [Environment variables](#)
-

WP-CLI has a series of global parameters (e.g. `--path=<path>` and `--user=<user>`) which work with all commands. They are called *global parameters* because they affect how WP-CLI interacts with WordPress, and have the same behavior across all commands.

```
# `--user=<user>` sets request to a specific WordPress user
$ wp --user=wpcli eval 'echo wp_get_current_user()->user_email;'
wpcli@example.com
```

For repeated usage, WP-CLI can also read options from a YAML configuration file (e.g. `wp-cli.yml`). WP-CLI automatically discovers configuration files on the filesystem based on rules defined below. These configuration files enable specifying default values for both global parameters and subcommand-specific arguments.

```
# WordPress develop includes a `wp-cli.yml` to enable easy use of WP-CLI
$ pwd
/srv/www/wordpress-develop.dev
$ cat wp-cli.yml
path: src/
```

Arguments are interpreted following an order of precedence, from highest priority to lowest:

1. Command-line arguments.
2. `wp-cli.local.yml` file inside the current working directory (or upwards).
3. `wp-cli.yml` file inside the current working directory (or upwards).
4. `~/.wp-cli/config.yml` file (path can be changed by setting the `WP_CLI_CONFIG_PATH` environment variable).
5. WP-CLI defaults.

Global parameters

The table below lists the available arguments (specified on the command-line) and options (specified in the configuration file).

Description	Argument	Option
Path to the WordPress files. Default value: <code>null</code>	<code>--path=<path></code>	<code>path: <path></code>
Perform operation against a remote server over SSH. Default value: <code>null</code>	<code>--</code> <code>ssh=[<user>@]<host>[:<port>][<path>]</code>	<code>ssh:</code> <code>[<user>@]<host>[:<port>][<path>]</code>
Perform operation against a remote WordPress install over HTTP. Default value: <code>null</code>	<code>--http=<http></code>	<code>http: <http></code>
Pretend request came from given URL. In multisite, this argument is how the target site is specified. Default value: <code>null</code>	<code>--url=<url></code>	<code>url: <url></code>
Set the WordPress user. Default value: <code>null</code>	<code>--user=<id login email></code>	<code>user: <id login email></code>
Skip loading all or some plugins. Note: mu-plugins are still loaded. Default value: <code>""</code>	<code>--skip-plugins[=<plugin>]</code>	<code>skip-plugins: <list></code>
Skip loading all or some themes. Default value: <code>""</code>	<code>--skip-themes[=<theme>]</code>	<code>skip-themes: <list></code>
Skip loading all installed packages. Default value: <code>false</code>	<code>--skip-packages</code>	<code>skip-packages: <bool></code>
Load PHP file before running the command (may be used more than once). Default value: <code>[]</code>	<code>--require=<path></code>	<code>require: <path></code>
Execute PHP code before running the command (may be used more than once). Default value: <code>[]</code>	<code>--exec=<php-code></code>	<code>exec: <php-code></code>
Load WordPress in a given context. Default value: <code>auto</code>	<code>--context[=<context>]</code>	<code>context: <context></code>
(Sub)commands to disable. Default value: <code>[]</code>	<i>Not available as a flag</i>	<code>disabled_commands: <list></code>
Whether to colorize the output. Default value: <code>"auto"</code>	<code>--[no-]color</code>	<code>color: <bool></code>

Description	Argument	Option
Show all PHP errors; add verbosity to WP-CLI bootstrap. Default value: <code>false</code>	<code>--debug[=<group>]</code>	<code>debug: <group></code>
Prompt the user to enter values for all command arguments, or a subset specified as comma-separated values. Default value: <code>false</code>	<code>--prompt[=<assoc>]</code>	<i>Not available as an option</i>
Suppress informational messages. Default value: <code>false</code>	<code>--quiet</code>	<code>quiet: <bool></code>
List of Apache Modules that are to be reported as loaded. Default value: <code>[]</code>	<i>Not available as a flag</i>	<code>apache_modules: <list></code>

Config files

WP-CLI can automatically discover and read options from a few configuration file types (when present):

1. `wp-cli.local.yml` file inside the current working directory (or upwards).
2. `wp-cli.yml` file inside the current working directory (or upwards).
3. `~/wp-cli/config.yml` file (path can be changed by setting the `WP_CLI_CONFIG_PATH` environment variable).

Besides the global parameters described above, configuration files can also contain defaults for any subcommand, as well as aliases to one or more WordPress installs.

Here's an annotated example `wp-cli.yml` file:

```
# Global parameter defaults
path: wp-core
url: http://example.com
user: admin
color: false
disabled_commands:
  - db drop
  - plugin install
require:
  - path-to/command.php

# Subcommand defaults (e.g. `wp config create`)
```

```

config create:
dbuser: root
dbpass:
extra-php: |
define( 'WP_DEBUG', true );
define( 'WP_POST_REVISIONS', 50 );

# Aliases to other WordPress installs (e.g. `wp @staging rewrite flush`)
# An alias can include 'user', 'url', 'path', 'ssh', or 'http'
@staging:
ssh: wpcli@staging.wp-cli.org
user: wpcli
path: /srv/www/staging.wp-cli.org
@production:
ssh: wpcli@wp-cli.org:2222
user: wpcli
path: /srv/www/wp-cli.org

# Aliases can reference other aliases to create alias groups
# Alias groups can be nested
@both:
- @staging
- @production

# '_' is a special value denoting configuration options for this wp-cli.yml
_:
# Merge subcommand defaults from the upstream config.yml, instead of overriding
merge: true
# Inherit configuration from an arbitrary YAML file
inherit: prod.yml

```

Remote (SSH) configuration

Using the `ssh` option, WP-CLI can be configured to run on a remote system rather than the current system. Along with the SSH protocol, WP-CLI also supports connecting to Docker containers (including docker-compose) and Vagrant VMs.

The connection type can be passed via the scheme of the `--ssh` parameter or `ssh` option.

Supported types are:

- `docker:[<user>@]<container_id>` - Runs WP-CLI in a running Docker container via `docker exec [--user <user>] <container_id> ...`
- `docker-compose:[<user>@]<container_id>` - Runs WP-CLI in a running Docker container via `docker-compose exec [--user <user>] <container_id> ...`
- `docker-compose-run:[<user>@]<container_id>` - Runs WP-CLI in a new Docker container via `docker-compose run [--user <user>] <container_id> ...`
- `vagrant` - Runs WP-CLI in a running Vagrant VM via `vagrant ssh ...`
- `[<user>@]<host>[:<port>]` (ssh) - Runs WP-CLI on a remote machine through an SSH connection via `ssh [-p <port>] [<user>@]<host> ...`

All connection types support an optional `path` suffix to specify a directory to `cd` to before running WP-CLI; `path` is a full system path starting with either `/` or `~`. (If `WP_CLI_SSH_PRE_CMD` is specified, `cd` is run after this pre-command.)

The SSH connection type also supports two advanced connection configuration options, which must be specified via an alias in the YAML configuration:

- `proxyjump` - Specifies a jumpbox connection string, which is passed to `ssh -J`
- `key` - Specifies the key (identify file) to use, which is passed to `ssh -i`

See the documentation about [running WP-CLI commands remotely](#) for more information.

Context configuration

In WP-CLI v2.6.0, a new global flag `--context=<context>` was added which allows users to select the WordPress context in which WP-CLI is supposed to execute its command(s).

One of the main goals is to allow WP-CLI to run updates on premium plugins and themes without requiring any special setup. From our initial testing, this allows a large range of popular premium extensions to *just work*™ with WP-CLI in terms of their update procedures.

Possible values for this flag:

- `cli`: The context which has been the default before introduction of this flag. This is something in-between a frontend and an admin request, to get around some of the quirks of WordPress when running on the console.
- `admin`: A context that simulates running a command as if it would be executed in the administration backend. This is meant to be used to get around issues with plugins that limit functionality behind an `is_admin()` check.
- `auto`: Switches between `cli` and `admin` depending on which command is being used. Currently, all `wp plugin *` and `wp theme *` commands use `admin`, while all other commands use `cli`.
- `frontend`: [WIP] This does nothing yet.

By default, the `--context` flag was set to `cli` in the initial release (v2.6.0). In WP-CLI v2.7.0 and later versions, the default was changed to `auto`. This gradual deployment allowed hosters and site owners to run tests on v2.6.0 by manually setting the context before the default behavior was changed.

If you are still using WP-CLI v2.6.0 but you want to use the default of `--context=auto`, you can do so by adding the necessary `context: auto` line to your global `wp-cli.yml` configuration file. Feel free to check the documentation on [WP-CLI configuration files](#) if this is new to you.

Environment variables

WP-CLI's behavior can 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, simply place the environment variable definition before the WP-CLI command you mean to run.

```
# Use vim to edit a post
$ EDITOR=vim wp post edit 1
```

To set the same environment variable value for every shell session, you'll need to include the environment variable definition in your `~/.bashrc` or `~/.zshrc` file

```
# Always use vim to edit a post
export EDITOR=vim
```

```
### #4
[] cocobook-sync [] 2026-10-28 11:01:27 UTC
[] cocobook-sync [] 2026-10-29 01:42:31 UTC
```