Sharing WP-CLI Packages

- WP_CLI_PACKAGES_DIR environment variable
  - Composer project in a shared directory

By default, WP-CLI places installed packages in `~/.wp-cli/packages/`, a hidden subdirectory for the user's home directory.

Because the home directory is different for each user, this naturally means each system user will have a separate directory of installed packages. If you have multiple active shell users on a server, and want to share installed WP-CLI packages between them, there are a couple of supported ways to do this.

`WP_CLI_PACKAGES_DIR` environment variable

To override the directory WP-CLI uses for installed packages, provide a `WP_CLI_PACKAGES_DIR` environment variable. If you wish for multiple users to share the same packages directory, you can simply provide the same `WP_CLI_PACKAGES_DIR` environment variable for each user.

```
vim /etc/environment
export WP_CLI_PACKAGES_DIR=/usr/local/lib/wp-cli-packages
```

Similarly, you can make sure the directory is only writable by a specific user to make packages available to all users, but only installable by the specific user.

It's worth noting the `WP_CLI_PACKAGES_DIR` environment variable *overrides* WP-CLI's default behavior of loading packages installed in the user's home directory. If you want to support both, you'll need to take the second approach.

Composer project in a shared directory

WP-CLI’s installed packages directory is simply a Composer project under the hood. Given this architecture, you can create your own Composer project in an arbitrary directory, and load it into scope using WP-CLI’s `--require=<path>` flag.

First, create your Composer project.

```
$ mkdir /usr/local/lib/wp-cli-packages
$ cd /usr/local/lib/wp-cli-packages
$ composer init -n --name=runcommand/wp-cli-packages -s=dev --repository=https://wp-cli.org/package-index/
$ composer require runcommand/hook
Using version dev-master for runcommand/hook
./composer.json has been updated
Loading composer repositories with package information
Updating dependencies (including require-dev)
  - Installing runcommand/hook (dev-master 7a7beae)
    Cloning 7a7beae2013eeea243cc44524a7c5c21da11979e

Writing lock file
Generating autoload files
```

Now, once your Composer project has a dependency or two, you can use `wp --require=<path/to/autoload>` (or the equivalent `config.yml` statement) to load the packages into WP-CLI.

□□□ #2
□ □□□ □□□ 2026-03-26 17:08:46 UTC
□ □□□ □□□ 2026-03-26 17:08:46 UTC