

Plugin Integration Tests



- [Running tests in continuous integration](#)
 - [Running tests locally](#)
 - [1. Install PHPUnit](#)
 - [2. Generate the plugin test files](#)
 - [3. Initialize the testing environment locally](#)
 - [4. Run the plugin tests](#)
 - [Integrating WP Unit Testing in Windows](#)
-

This guide demonstrates how to run integration tests both in continuous integration and locally. The scaffolding uses the WordPress Core "unit tests" that serve to unit-test WordPress Core. Unit tests are useful for testing plugins and themes as well, but if they are used in that way, it turns those tests into "integration tests" - they test the integration between a plugin/theme and WordPress Core. Proper unit tests for a plugin or theme would not load WordPress.

Running tests in continuous integration

The `wp scaffold plugin-tests` command generates a CircleCI configuration by default. Use `--ci=<provider>` to generate a configuration for a different supported provider, such as GitHub Actions:

```
wp scaffold plugin-tests my-plugin --ci=github
```

All you need to do to enable this is copy and then edit the following files from the [WP-CLI sample plugin](#):

- `.circleci/config.yml`, `phpunit.xml.dist` and `.phpcs.xml.dist` files
- `tests` folder

See the docs for an [explanation of what each file does](#). You will then need to specify your unit tests in the `tests/` folder.

Running tests locally

Running tests locally can be beneficial during development as it is quicker than committing changes and waiting for CI to run the tests.

We're going to assume that:

- You already have a plugin called `my-plugin`
- You have installed `git`, `svn`, `php`, `apache` ([How To Install Linux, Apache, MySQL, PHP \(LAMP\) stack on Ubuntu 16.04](#))

So, let's get started:

1. [Install PHPUnit](#)

The version of PHPUnit to install depends on both PHP and WordPress versions. See: [PHPUnit Compatibility and WordPress Versions – Make WordPress Core](#)

2. Generate the plugin test files

```
wp scaffold plugin-tests my-plugin
```

This command will generate all the files needed for running tests, including a `.circleci/config.yml` file by default. Use `--ci=<provider>` to generate a configuration for a different supported CI provider.

3. Initialize the testing environment locally

`cd` into the plugin directory and run the install script (you will need to have `wget` installed).

```
bash bin/install-wp-tests.sh wordpress_test root '' localhost latest
```

The install script first installs a copy of WordPress in the `/tmp` directory (by default) as well as the WordPress unit testing tools. Then it creates a database to be used while running tests. The

parameters that are passed to `install-wp-tests.sh` setup the test database.

- `wordpress_test` is the name of the test database (**all data will be deleted!**)
- `root` is the MySQL user name
- `''` is the MySQL user password
- `localhost` is the MySQL server host
- `latest` is the WordPress version; could also be `3.7`, `3.6.2` etc.

NOTE: This script can be run multiple times without errors, but it will *not* overwrite previously existing files. So if your DB credentials change, or you want to switch to a different instance of mysql, simply re-running the script won't be enough. You'll need to manually edit the `wp-config.php` that's installed in `/tmp`.

4. Run the plugin tests

```
phpunit
```

If you have trouble running the install script or PHPUnit, check the [common issues guide](#) for help and answers to common issues.

Integrating WP Unit Testing in Windows

Tried and gotten stuck setting up unit tests for your project in Windows?

First, know that some WP-CLI commands don't work in Windows, so you'll want to use Cygwin. Cygwin is also [preferred by the WordPress core project](#).

Second, the `bin/install-wp-tests.sh` script doesn't work directly in Windows. Windows 10 [introduced a Windows Subsystem for Linux](#) but older versions require extra effort. Cygwin is preferred because it runs bash scripts by default.

Third, the bash installation script uses `which`, a Linux command not available by default in Windows. `curl`, `svn` and `wget` all can be installed in Windows as separate packages.

Lastly, sometimes the bash script fails to build WordPress because of how it uses tmp folders. If it fails, then you can manually install WordPress to a writable directory, and then manually create `wp-tests-config.php`.

