Automated Testing

- Setup a testing environment with PHPUnit
  - 1. Install VVV
  - 2. Set up Timber for tests
  - 3. Install WordPress's Development Version
  - 4. Configure WordPress tests
  - 5. Install WordPress tests
  - 6. Run the tests!
  - Writing tests
  - Gotchas!
- Static Analysis

Setup a testing environment with PHPUnit

Follow the setup steps for Varying Vagrant Vagrants. If you're having trouble I recommend starting 100% fresh (uninstall and delete everything including VirtualBox). For this walk-through we're going to start from scratch:

1. Install VVV

Follow the setup instructions — don't forget to install VirtualBox and Vagrant first! For this tutorial I'm going to assume you've installed into `~/vagrant-local/`

2. Set up Timber for tests

Navigate into the `www` directory and clone Timber...

```
$ cd ~/vagrant-local/www/
$ git clone git@github.com:timber/timber.git
```

3. Install WordPress's Development Version

VVV no longer installs the development version of WordPress by default so we have to re-provision with it enabled. Edit `vvv-custom.yml` and find this line:

```
  wordpress-trunk:
    skip_provisioning: true # provisioning this one takes longer, so it's disabled by default
```

Set `skip_provisioning` to `false`

```
  wordpress-trunk:
    skip_provisioning: false # provisioning this one takes longer, so it's disabled by default
```

Now re-provision Vagrant

```
$ vagrant halt && vagrant up --provision
```

Warning: this one will take a while.

4. Configure WordPress tests

Copy `/wordpress-trunk/public_html/wp-tests-config-sample.php` to `/wordpress-trunk/public_html/wp-tests-config.php`. Assuming you're using VVV's defaults, we just need to specify how to access the database:

```
define('DB_NAME', 'wordpress_unit_tests');
define('DB_USER', 'root');
define('DB_PASSWORD', 'root');
```

5. Install WordPress tests

SSH into Vagrant to install Timber's tests and run them

```
$ vagrant ssh
```

Now, navigate to where you installed Timber via Git:

```
$ cd /srv/www/timber
```

And install the tests!

```
$ bin/install-wp-tests.sh wordpress_tests root root
```

All done! Now, the fun part (and the only part you have to do in the future when writing/running tests)

6. Run the tests!

Connect to your Vagrant instance trough SSH (if you're not already):

```
$ vagrant ssh
```

Now wait for it to bring you into the virtual box from the virtual environment...

```
$ cd /srv/www/timber
$ composer install
$ phpunit
```

You should see a bunch of gobbledygook across your screen (the whole process will take about 4 mins.), but we should see that WordPress is testing successfully. Hurrah! For more info, check out the Handbook on Automated Testing.

Writing tests

Now we get to the good stuff. You can add tests to the `timber/tests` directory. Any new features should be covered by tests. You can be a hero and help write tests for existing methods and functionality.

Gotchas!

- You may need to setup authorization between VVV and GitHub. Just follow the prompts to create a token if that interrupts the `composer install`.
- You may have memory problems with Composer. In case that happens, here's the script I run:

```
sudo /bin/dd if=/dev/zero of=/var/swap.1 bs=1M count=1024
sudo /sbin/mkswap /var/swap.1
sudo /sbin/swapon /var/swap.1
```

Static Analysis

Version 2.x is also covered by static analysis courtesy of PHPStan. This analyzes our code files and structure to ensure things like existence and accessibility of called methods and functions, number of passed arguments, data types in docs, etc.

To run, use:

```
vendor/bin/phpstan analyze -l 1
```

Where the last argument is the level of strictness (0-7) to apply. We're currently starting at level 1 and working our way up over time.

□□□ #2
□ □□□ □□□ 2026-08-29 16:14:01 UTC
□ □□□ □□□ 2026-08-29 16:14:01 UTC