WARNING You're browsing the documentation for an old version of Laravel. Consider upgrading your project to Laravel 13.x.

Laravel Valet

Introduction

Laravel Valet is a development environment for macOS minimalists. Laravel Valet configures your Mac to always run Nginx in the background when your machine starts. Then, using DnsMasq, Valet proxies all requests on the `*.test` domain to point to sites installed on your local machine.

In other words, Valet is a blazing fast Laravel development environment that uses roughly 7 MB of RAM. Valet isn't a complete replacement for Sail or Homestead, but provides a great alternative if you want flexible basics, prefer extreme speed, or are working on a machine with a limited amount of RAM.

Out of the box, Valet support includes, but is not limited to:

Laravel	ExpressionEngine	Slim
Lumen	Jigsaw	Statamic
Bedrock	Joomla	Static HTML
CakePHP 3	Katana	Symfony
Concrete5	Kirby	WordPress
Contao	Magento	Zend
Craft	OctoberCMS	
Drupal	Sculpin	

However, you may extend Valet with your own custom drivers.

Installation

!

Valet requires macOS and Homebrew. Before installation, you should make sure that no other programs such as Apache or Nginx are binding to your local machine's port 80.

To get started, you first need to ensure that Homebrew is up to date using the `update` command:

```
brew update
```

Next, you should use Homebrew to install PHP:

```
brew install php
```

After installing PHP, you are ready to install the Composer package manager. In addition, you should make sure the `~/.composer/vendor/bin` directory is in your system's "PATH". After Composer has been installed, you may install Laravel Valet as a global Composer package:

```
composer global require laravel/valet
```

Finally, you may execute Valet's `install` command. This will configure and install Valet and DnsMasq. In addition, the daemons Valet depends on will be configured to launch when your system starts:

```
valet install
```

Once Valet is installed, try pinging any `*.test` domain on your terminal using a command such as `ping foobar.test`. If Valet is installed correctly you should see this domain responding on `127.0.0.1`.

Valet will automatically start its required services each time your machine boots.

PHP Versions

Valet allows you to switch PHP versions using the `valet use php@version` command. Valet will install the specified PHP version via Homebrew if it is not already installed:

```
valet use php@7.2

valet use php
```

You may also create a `.valetphprc` file in the root of your project. The `.valetphprc` file should contain the PHP version the site should use:

```
php@7.2
```

Once this file has been created, you may simply execute the `valet use` command and the command will determine the site's preferred PHP version by reading the file.

> [!] Valet only serves one PHP version at a time, even if you have multiple PHP versions installed.

Database

If your application needs a database, check out DBngin. DBngin provides a free, all-in-one database management tool that includes MySQL, PostgreSQL, and Redis. After DBngin

has been installed, you can connect to your database at `127.0.0.1` using the `root` username and an empty string for the password.

Resetting Your Installation

If you are having trouble getting your Valet installation to run properly, executing the `composer global update` command followed by `valet install` will reset your installation and can solve a variety of problems. In rare cases, it may be necessary to "hard reset" Valet by executing `valet uninstall --force` followed by `valet install`.

Upgrading Valet

You may update your Valet installation by executing the `composer global update` command in your terminal. After upgrading, it is good practice to run the `valet install` command so Valet can make additional upgrades to your configuration files if necessary.

Serving Sites

Once Valet is installed, you're ready to start serving your Laravel applications. Valet provides two commands to help you serve your applications: `park` and `link`.

The `park` Command

The `park` command registers a directory on your machine that contains your applications. Once the directory has been "parked" with Valet, all of the directories within that directory will be accessible in your web browser at `http://<directory-name>.test`:

```
cd ~/Sites

valet park
```

That's all there is to it. Now, any application you create within your "parked" directory will automatically be served using the `http://<directory-name>.test` convention. So, if your

parked directory contains a directory named "laravel", the application within that directory will be accessible at `http://laravel.test`. In addition, Valet automatically allows you to access the site using wildcard subdomains (`http://foo.laravel.test`).

The `link` Command

The `link` command can also be used to serve your Laravel applications. This command is useful if you want to serve a single site in a directory and not the entire directory:

```
cd ~/Sites/laravel

valet link
```

Once an application has been linked to Valet using the `link` command, you may access the application using its directory name. So, the site that was linked in the example above may be accessed at `http://laravel.test`. In addition, Valet automatically allows you to access the site using wildcard sub-domains (`http://foo.laravel.test`).

If you would like to serve the application at a different hostname, you may pass the hostname to the `link` command. For example, you may run the following command to make an application available at `http://application.test`:

```
cd ~/Sites/laravel

valet link application
```

You may execute the `links` command to display a list of all of your linked directories:

```
valet links
```

The `unlink` command may be used to destroy the symbolic link for a site:

```
cd ~/Sites/laravel

valet unlink
```

Securing Sites With TLS

By default, Valet serves sites over HTTP. However, if you would like to serve a site over encrypted TLS using HTTP/2, you may use the `secure` command. For example, if your site is being served by Valet on the `laravel.test` domain, you should run the following command to secure it:

```
valet secure laravel
```

To "unsecure" a site and revert back to serving its traffic over plain HTTP, use the `unsecure` command. Like the `secure` command, this command accepts the hostname that you wish to unsecure:

```
valet unsecure laravel
```

Serving A Default Site

Sometimes, you may wish to configure Valet to serve a "default" site instead of a `404` when visiting an unknown `test` domain. To accomplish this, you may add a `default` option to your `~/.config/valet/config.json` configuration file containing the path to the site that should serve as your default site:

```
    "default": "/Users/Sally/Sites/foo",
```

Sharing Sites

Valet even includes a command to share your local sites with the world, providing an easy way to test your site on mobile devices or share it with team members and clients.

Sharing Sites Via Ngrok

To share a site, navigate to the site's directory in your terminal and run Valet's `share` command. A publicly accessible URL will be inserted into your clipboard and is ready to paste directly into your browser or share with your team:

```
cd ~/Sites/laravel

valet share
```

To stop sharing your site, you may press `Control + C`. Sharing your site using Ngrok requires you to create an Ngrok account and setup an authentication token.

You may pass additional Ngrok parameters to the share command, such as `valet share --region=eu`. For more information, consult the ngrok documentation.

Sharing Sites Via Expose

If you have Expose installed, you can share your site by navigating to the site's directory in your terminal and running the `expose` command. Consult the Expose documentation for information regarding the additional command-line parameters it supports. After sharing

the site, Expose will display the sharable URL that you may use on your other devices or amongst team members:

```
cd ~/Sites/laravel

expose
```

To stop sharing your site, you may press `Control + C`.

Sharing Sites On Your Local Network

Valet restricts incoming traffic to the internal `127.0.0.1` interface by default so that your development machine isn't exposed to security risks from the Internet.

If you wish to allow other devices on your local network to access the Valet sites on your machine via your machine's IP address (eg: `192.168.1.10/application.test`), you will need to manually edit the appropriate Nginx configuration file for that site to remove the restriction on the `listen` directive. You should remove the `127.0.0.1:` prefix on the `listen` directive for ports 80 and 443.

If you have not run `valet secure` on the project, you can open up network access for all non-HTTPS sites by editing the `/usr/local/etc/nginx/valet/valet.conf` file. However, if you're serving the project site over HTTPS (you have run `valet secure` for the site) then you should edit the `~/.config/valet/Nginx/app-name.test` file.

Once you have updated your Nginx configuration, run the `valet restart` command to apply the configuration changes.

Site Specific Environment Variables

Some applications using other frameworks may depend on server environment variables but do not provide a way for those variables to be configured within your project. Valet

allows you to configure site specific environment variables by adding a `.valet-env.php` file within the root of your project. This file should return an array of site / environment variable pairs which will be added to the global `$_SERVER` array for each site specified in the array:

```php
<?php

return [
    // Set $_SERVER['key'] to "value" for the laravel.test site...
    'laravel' => [
        'key' => 'value',
    ],

    // Set $_SERVER['key'] to "value" for all sites...
    '*' => [
        'key' => 'value',
    ],
];
```

Proxying Services

Sometimes you may wish to proxy a Valet domain to another service on your local machine. For example, you may occasionally need to run Valet while also running a separate site in Docker; however, Valet and Docker can't both bind to port 80 at the same time.

To solve this, you may use the `proxy` command to generate a proxy. For example, you may proxy all traffic from `http://elasticsearch.test` to `http://127.0.0.1:9200`:

```
// Proxy over HTTP...
valet proxy elasticsearch http://127.0.0.1:9200

// Proxy over TLS + HTTP/2...
```

```
valet proxy elasticsearch http://127.0.0.1:9200 --secure
```

You may remove a proxy using the `unproxy` command:

```
valet unproxy elasticsearch
```

You may use the `proxies` command to list all site configurations that are proxied:

```
valet proxies
```

Custom Valet Drivers

You can write your own Valet "driver" to serve PHP applications running on a framework or CMS that is not natively supported by Valet. When you install Valet, a `~/.config/valet/Drivers` directory is created which contains a `SampleValetDriver.php` file. This file contains a sample driver implementation to demonstrate how to write a custom driver. Writing a driver only requires you to implement three methods: `serves`, `isStaticFile`, and `frontControllerPath`.

All three methods receive the `$sitePath`, `$siteName`, and `$uri` values as their arguments. The `$sitePath` is the fully qualified path to the site being served on your machine, such as `/Users/Lisa/Sites/my-project`. The `$siteName` is the "host" / "site name" portion of the domain (`my-project`). The `$uri` is the incoming request URI (`/foo/bar`).

Once you have completed your custom Valet driver, place it in the `~/.config/valet/Drivers` directory using the `FrameworkValetDriver.php` naming convention. For example, if you are writing a custom valet driver for WordPress, your filename should be `WordPressValetDriver.php`.

Let's take a look at a sample implementation of each method your custom Valet driver should implement.

The `serves` Method

The `serves` method should return `true` if your driver should handle the incoming request. Otherwise, the method should return `false`. So, within this method, you should attempt to determine if the given `$sitePath` contains a project of the type you are trying to serve.

For example, let's imagine we are writing a `WordPressValetDriver`. Our `serves` method might look something like this:

```
/**
 * Determine if the driver serves the request.
 *
 * @param  string  $sitePath
 * @param  string  $siteName
 * @param  string  $uri
 * @return bool
 */
public function serves($sitePath, $siteName, $uri)
{
    return is_dir($sitePath.'/wp-admin');
}
```

The `isStaticFile` Method

The `isStaticFile` should determine if the incoming request is for a file that is "static", such as an image or a stylesheet. If the file is static, the method should return the fully qualified path to the static file on disk. If the incoming request is not for a static file, the method should return `false`:

```
/**
 * Determine if the incoming request is for a static file.
 *
```

```
     * @param  string  $sitePath
     * @param  string  $siteName
     * @param  string  $uri
     * @return string|false
     */
    public function isStaticFile($sitePath, $siteName, $uri)
    {
        if (file_exists($staticFilePath = $sitePath.'/public/'.$uri)) {
            return $staticFilePath;
        }

        return false;
    }
```

!

The `isStaticFile` method will only be called if the `serves` method returns `true` for the incoming request and the request URI is not `/`.

The `frontControllerPath` Method

The `frontControllerPath` method should return the fully qualified path to your application's "front controller", which is typically an "index.php" file or equivalent:

```
    /**
     * Get the fully resolved path to the application's front controller.
     *
     * @param  string  $sitePath
     * @param  string  $siteName
     * @param  string  $uri
     * @return string
     */
    public function frontControllerPath($sitePath, $siteName, $uri)
    {
        return $sitePath.'/public/index.php';
```

```
}
```

Local Drivers

If you would like to define a custom Valet driver for a single application, create a `LocalValetDriver.php` file in the application's root directory. Your custom driver may extend the base `ValetDriver` class or extend an existing application specific driver such as the `LaravelValetDriver`:

```
class LocalValetDriver extends LaravelValetDriver
{
    /**
     * Determine if the driver serves the request.
     *
     * @param  string  $sitePath
     * @param  string  $siteName
     * @param  string  $uri
     * @return bool
     */
    public function serves($sitePath, $siteName, $uri)
    {
        return true;
    }

    /**
     * Get the fully resolved path to the application's front controller.
     *
     * @param  string  $sitePath
     * @param  string  $siteName
     * @param  string  $uri
     * @return string
     */
    public function frontControllerPath($sitePath, $siteName, $uri)
    {
        return $sitePath.'/public_html/index.php';
    }
```

```
28     }
```

Other Valet Commands

Command	Description
`valet forget`	Run this command from a "parked" directory to remove it from the parked directory list.
`valet log`	View a list of logs which are written by Valet's services.
`valet paths`	View all of your "parked" paths.
`valet restart`	Restart the Valet daemons.
`valet start`	Start the Valet daemons.
`valet stop`	Stop the Valet daemons.
`valet trust`	Add sudoers files for Brew and Valet to allow Valet commands to be run without prompting for your password.
`valet uninstall`	Uninstall Valet: shows instructions for manual uninstall. Pass the `--force` option to aggressively delete all of Valet's resources.

Valet Directories & Files

You may find the following directory and file information helpful while troubleshooting issues with your Valet environment:

`~/.config/valet`

Contains all of Valet's configuration. You may wish to maintain a backup of this directory.

`~/.config/valet/dnsmasq.d/`

This directory contains DNSMasq's configuration.

`~/.config/valet/Drivers/`

This directory contains Valet's drivers. Drivers determine how a particular framework / CMS is served.

`~/.config/valet/Extensions/`

This directory contains custom Valet extensions / commands.

`~/.config/valet/Nginx/`

This directory contains all of Valet's Nginx site configurations. These files are rebuilt when running the `install`, `secure`, and `tld` commands.

`~/.config/valet/Sites/`

This directory contains all of the symbolic links for your linked projects.

`~/.config/valet/config.json`

This file is Valet's master configuration file.

`~/.config/valet/valet.sock`

This file is the PHP-FPM socket used by Valet's Nginx installation. This will only exist if PHP is running properly.

`~/.config/valet/Log/fpm-php.www.log`

This file is the user log for PHP errors.

`~/.config/valet/Log/nginx-error.log`

This file is the user log for Nginx errors.

`/usr/local/var/log/php-fpm.log`

This file is the system log for PHP-FPM errors.

`/usr/local/var/log/nginx`

This directory contains the Nginx access and error logs.

`/usr/local/etc/php/X.X/conf.d`

This directory contains the `*.ini` files for various PHP configuration settings.

`/usr/local/etc/php/X.X/php-fpm.d/valet-fpm.conf`

This file is the PHP-FPM pool configuration file.

`~/.composer/vendor/laravel/valet/cli/stubs/secure.valet.conf`

This file is the default Nginx configuration used for building SSL certificates for your sites.

Laravel is the most productive way to build, deploy, and monitor software.

First name

Last name

Email address

Stay updated

By submitting this form, you agree to our terms. You can opt-out anytime.




© 2026 Laravel Legal Status

Products

Cloud

Forge

Packages

Cashier

Dusk

Nightwatch

Vapor

Nova

Horizon

Octane

Scout

Pennant

Pint

Sail

Sanctum

Socialite

Telescope

Pulse

Reverb

Echo

Resources

Documentation

Starter Kits

Release Notes

Blog

News

Community

Partners

Curotec

byte5

Tighten

Jump24

UCodeSoft

Redberry

Larabelles

Learn

Jobs

Careers

Trust

Steadfast Collective

CACI Limited

Vehikl

Threadable

See All