Laravel Valet

Introduction

Valet is a Laravel development environment for Mac minimalists. No Vagrant, no `/etc/hosts` file. You can even share your sites publicly using local tunnels. *Yeah, we like it too.*

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, a blazing fast Laravel development environment that uses roughly 7 MB of RAM. Valet isn't a complete replacement for Vagrant 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	Drupal	Sculpin
Lumen	Jigsaw	Slim
Bedrock	Joomla	Statamic
CakePHP 3	Katana	Static HTML
Concrete5	Kirby	Symfony
Contao	Magento	WordPress
Craft	OctoberCMS	Zend

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

Valet Or Homestead

As you may know, Laravel offers Homestead, another local Laravel development environment. Homestead and Valet differ in regards to their intended audience and their approach to local development. Homestead offers an entire Ubuntu virtual machine with automated Nginx configuration. Homestead is a wonderful choice if you want a fully virtualized Linux development environment or are on Windows / Linux.

Valet only supports Mac, and requires you to install PHP and a database server directly onto your local machine. This is easily achieved by using Homebrew with commands like `brew install php` and `brew install mysql`. Valet provides a blazing fast local development environment with minimal resource consumption, so it's great for developers who only require PHP / MySQL and do not need a fully virtualized development environment.

Both Valet and Homestead are great choices for configuring your Laravel development environment. Which one you choose will depend on your personal taste and your team's needs.

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.

- Install or update Homebrew to the latest version using `brew update`.
- Install PHP 7.3 using Homebrew via `brew install php`.
- Install Composer.

- Install Valet with Composer via `composer global require laravel/valet`. Make sure the `~/.composer/vendor/bin` directory is in your system's "PATH".
- Run the `valet install` command. This will configure and install Valet and DnsMasq, and register Valet's daemon to launch when your system starts.

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 daemon each time your machine boots. There is no need to run `valet start` or `valet install` ever again once the initial Valet installation is complete.

Using Another Domain

By default, Valet serves your projects using the `.test` TLD. If you'd like to use another domain, you can do so using the `valet tld tld-name` command.

For example, if you'd like to use `.app` instead of `.test`, run `valet tld app` and Valet will start serving your projects at `*.app` automatically.

Database

If you need a database, try MySQL by running `brew install mysql@5.7` on your command line. Once MySQL has been installed, you may start it using the `brew services start mysql@5.7` command. You can then connect to the database at `127.0.0.1` using the `root` username and an empty string for the password.

PHP Versions

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

```
valet use php@7.2

valet use php
```

Upgrading

You may update your Valet installation using 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.

Upgrading To Valet 2.0

Valet 2.0 transitions Valet's underlying web server from Caddy to Nginx. Before upgrading to this version you should run the following commands to stop and uninstall the existing Caddy daemon:

```
valet stop
valet uninstall
```

Next, you should upgrade to the latest version of Valet. Depending on how you installed Valet, this is typically done through Git or Composer. If you installed Valet via Composer, you should use the following command to update to the latest major version:

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

Once the fresh Valet source code has been downloaded, you should run the `install` command:

```
valet install
valet restart
```

After upgrading, it may be necessary to re-park or re-link your sites.

Serving Sites

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

The `park` Command

- Create a new directory on your Mac by running something like `mkdir ~/Sites`. Next, `cd ~/Sites` and run `valet park`. This command will register your current working directory as a

path that Valet should search for sites.

- Next, create a new Laravel site within this directory: `laravel new blog`.
- Open `http://blog.test` in your browser.

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

The `link` Command

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

- To use the command, navigate to one of your projects and run `valet link app-name` in your terminal. Valet will create a symbolic link in `~/.config/valet/Sites` which points to your current working directory.
- After running the `link` command, you can access the site in your browser at `http://app-name.test`.

To see a listing of all of your linked directories, run the `valet links` command. You may use `valet unlink app-name` to destroy the symbolic link.

You can use `valet link` to serve the same project from multiple (sub)domains. To add a subdomain or another domain to your project run `valet link subdomain.app-name` from the project folder.

Securing Sites With TLS

By default, Valet serves sites over plain HTTP. However, if you would like to serve a site over encrypted TLS using HTTP/2, 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 host name that you wish to unsecure:

```
valet unsecure laravel
```

Sharing Sites

Valet even includes a command to share your local sites with the world. No additional software installation is required once Valet is installed.

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

To stop sharing your site, hit `Control + C` to cancel the process.

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. These variables will be added to the `$_SERVER` global array:

```
<?php

return [
    'WEBSITE_NAME' => 'My Blog',
];
```

Custom Valet Drivers

You can write your own Valet "driver" to serve PHP applications running on another 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 file name 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 pretend 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 your "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` 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';
    }
}
```

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 daemon.
`valet start`	Start the Valet daemon.
`valet stop`	Stop the Valet daemon.
`valet trust`	Add sudoers files for Brew and Valet to allow Valet commands to be run without prompting for passwords.
`valet uninstall`	Uninstall the Valet daemon.

Become a Laravel Partner

Laravel Partners are elite shops providing top-notch Laravel development and consulting. Each of our partners can help you craft a beautiful, well-architected project.

Our Partners

Laravel

Highlights

- Release Notes
- Getting Started
- Routing
- Blade Templates
- Authentication
- Authorization
- Artisan Console
- Database
- Eloquent ORM
- Testing

Resources

- Laracasts
- Laravel News
- Laracon
- Laracon EU
- Laracon AU
- Jobs
- Certification
- Forums

Partners

- Vehikl
- Tighten Co.
- Kirschbaum
- Byte 5
- 64Robots
- Cubet
- DevSquad
- Ideil
- Cyber-Duck
- ABOUT YOU
- Become A Partner

Ecosystem

- Vapor
- Forge
- Envoyer
- Horizon
- Lumen
- Nova
- Echo
- Valet
- Mix
- Spark
- Cashier
- Homestead
- Dusk
- Passport
- Scout
- Socialite

Laravel is a web application framework with expressive, elegant syntax. We believe development must be an enjoyable and creative experience to be truly fulfilling. Laravel attempts to take the pain out of development by easing common tasks used in most web projects.