Laravel Homestead

Introduction

Laravel strives to make the entire PHP development experience delightful, including your local development environment. Vagrant provides a simple, elegant way to manage and provision Virtual Machines.

Laravel Homestead is an official, pre-packaged Vagrant box that provides you a wonderful development environment without requiring you to install PHP, a web server, and any other server software on your local machine. No more worrying about messing up your operating system! Vagrant boxes are completely disposable. If something goes wrong, you can destroy and re-create the box in minutes!

Homestead runs on any Windows, Mac, or Linux system, and includes Nginx, PHP, MySQL, PostgreSQL, Redis, Memcached, Node, and all of the other goodies you need to develop amazing Laravel applications.

If you are using Windows, you may need to enable hardware virtualization (VT-x). It can usually be enabled via your BIOS. If you are using Hyper-V on a UEFI system you may additionally need to disable Hyper-V in order to access VT-x.

Included Software

Ubuntu 18.04	MariaDB database	Beanstalkd
Git	snapshots	Mailhog
PHP 7.3	Sqlite3	avahi
PHP 7.2	PostgreSQL	ngrok
PHP 7.1	Composer	Xdebug
PHP 7.0	Node (With Yarn,	XHProf / Tideways /
PHP 5.6	Bower, Grunt, and	XHGui
Nginx	Gulp)	wp-cli
MySQL	Redis	
lmm for MySQL or	Memcached	

Optional Software

Apache	Gearman	Oh My Zsh
Blackfire	Go	Open Resty
Cassandra	Grafana	PM2
Chronograf	InfluxDB	Python
CouchDB	MariaDB	RabbitMQ
Crystal & Lucky Framework	MinIO	Solr
Docker	MongoDB	Webdriver & Laravel Dusk Utilities
Elasticsearch	MySQL 8	
	Neo4j	

Installation & Setup

First Steps

Before launching your Homestead environment, you must install VirtualBox 6.x, VMWare, Parallels or Hyper-V as well as Vagrant. All of these software packages provide easy-to-use visual installers for all popular operating systems.

To use the VMware provider, you will need to purchase both VMware Fusion / Workstation and the VMware Vagrant plug-in. Though it is not free, VMware can provide faster shared folder performance out of the box.

To use the Parallels provider, you will need to install Parallels Vagrant plug-in. It is free of charge.

Because of Vagrant limitations, The Hyper-V provider ignores all networking settings.

Installing The Homestead Vagrant Box

Once VirtualBox / VMware and Vagrant have been installed, you should add the `laravel/homestead` box to your Vagrant installation using the following command in your terminal. It will take a few minutes to download the box, depending on your Internet connection speed:

```
vagrant box add laravel/homestead
```

If this command fails, make sure your Vagrant installation is up to date.

Homestead periodically issues "alpha" / "beta" boxes for testing, which may interfere with the `vagrant box add` command. If you are having issues running `vagrant box add`, you may run the `vagrant up` command and the correct box will be downloaded when Vagrant attempts to start the virtual machine.

Installing Homestead

You may install Homestead by cloning the repository onto your host machine. Consider cloning the repository into a `Homestead` folder within your "home" directory, as the Homestead box will serve as the host to all of your Laravel projects:

```
git clone https://github.com/laravel/homestead.git ~/Homestead
```

You should check out a tagged version of Homestead since the `master` branch may not always be stable. You can find the latest stable version on the GitHub Release Page. Alternatively, you may checkout the `release` branch which always contains the latest stable release:

```
cd ~/Homestead

git checkout release
```

Once you have cloned the Homestead repository, run the `bash init.sh` command from the Homestead directory to create the `Homestead.yaml` configuration file. The `Homestead.yaml` file will be placed in the Homestead directory:

```
// Mac / Linux...
bash init.sh

// Windows...
init.bat
```

Configuring Homestead

Setting Your Provider

The `provider` key in your `Homestead.yaml` file indicates which Vagrant provider should be used: `virtualbox`, `vmware_fusion`, `vmware_workstation`, `parallels` or `hyperv`. You may set this to the provider you prefer:

```
provider: virtualbox
```

Configuring Shared Folders

The `folders` property of the `Homestead.yaml` file lists all of the folders you wish to share with your Homestead environment. As files within these folders are changed, they will be kept in sync between your local machine and the Homestead environment. You may configure as many shared folders as necessary:

```
folders:
    - map: ~/code/project1
      to: /home/vagrant/project1
```

Windows users should not use the `~/` path syntax and instead should use the full path to their project, such as `C:\Users\user\Code\project1`.

You should always map individual projects to their own folder mapping instead of mapping your entire `~/code` folder. When you map a folder the virtual machine must keep track of all disk IO for *every* file in the folder. This leads to performance issues if you have a large number of files in a folder.

```
folders:
    - map: ~/code/project1
      to: /home/vagrant/project1

    - map: ~/code/project2
      to: /home/vagrant/project2
```

You should never mount `.` (the current directory) when using Homestead. This causes Vagrant to not map the current folder to `/vagrant` and will break optional features and cause unexpected results while provisioning.

To enable NFS, you only need to add a simple flag to your synced folder configuration:

```
folders:
    - map: ~/code/project1
      to: /home/vagrant/project1
      type: "nfs"
```

When using NFS on Windows, you should consider installing the vagrant-winnfsd plug-in. This plug-in will maintain the correct user / group permissions for files and directories within the Homestead box.

You may also pass any options supported by Vagrant's Synced Folders by listing them under the `options` key:

```
folders:
    - map: ~/code/project1
      to: /home/vagrant/project1
      type: "rsync"
      options:
          rsync__args: ["--verbose", "--archive", "--delete", "-zz"]
          rsync__exclude: ["node_modules"]
```

Configuring Nginx Sites

Not familiar with Nginx? No problem. The `sites` property allows you to easily map a "domain" to a folder on your Homestead environment. A sample site configuration is

included in the `Homestead.yaml` file. Again, you may add as many sites to your Homestead environment as necessary. Homestead can serve as a convenient, virtualized environment for every Laravel project you are working on:

```
sites:
    - map: homestead.test
      to: /home/vagrant/project1/public
```

If you change the `sites` property after provisioning the Homestead box, you should re-run `vagrant reload --provision` to update the Nginx configuration on the virtual machine.

Homestead scripts are built to be as idempotent as possible. However, if you are experiencing issues while provisioning you should destroy and rebuild the machine via `vagrant destroy && vagrant up`.

Hostname Resolution

Homestead publishes hostnames over `mDNS` for automatic host resolution. If you set `hostname: homestead` in your `Homestead.yaml` file, the host will be available at `homestead.local`. MacOS, iOS, and Linux desktop distributions include `mDNS` support by default. Windows requires installing Bonjour Print Services for Windows.

Using automatic hostnames works best for "per project" installations of Homestead. If you host multiple sites on a single Homestead instance, you may add the "domains" for your web sites to the `hosts` file on your machine. The `hosts` file will redirect requests for your Homestead sites into your Homestead machine. On Mac and Linux, this file is located at `/etc/hosts`. On Windows, it is located at `C:\Windows\System32\drivers\etc\hosts`. The lines you add to this file will look like the following:

```
192.168.10.10  homestead.test
```

Make sure the IP address listed is the one set in your `Homestead.yaml` file. Once you have added the domain to your `hosts` file and launched the Vagrant box you will be able to access the site via your web browser:

```
http://homestead.test
```

Launching The Vagrant Box

Once you have edited the `Homestead.yaml` to your liking, run the `vagrant up` command from your Homestead directory. Vagrant will boot the virtual machine and automatically configure your shared folders and Nginx sites.

To destroy the machine, you may use the `vagrant destroy --force` command.

Per Project Installation

Instead of installing Homestead globally and sharing the same Homestead box across all of your projects, you may instead configure a Homestead instance for each project you manage. Installing Homestead per project may be beneficial if you wish to ship a `Vagrantfile` with your project, allowing others working on the project to `vagrant up`.

To install Homestead directly into your project, require it using Composer:

```
composer require laravel/homestead --dev
```

Once Homestead has been installed, use the `make` command to generate the `Vagrantfile` and `Homestead.yaml` file in your project root. The `make` command will automatically configure the `sites` and `folders` directives in the `Homestead.yaml` file.

Mac / Linux:

```
php vendor/bin/homestead make
```

Windows:

```
vendor\\bin\\homestead make
```

Next, run the `vagrant up` command in your terminal and access your project at `http://homestead.test` in your browser. Remember, you will still need to add an `/etc/hosts` file entry for `homestead.test` or the domain of your choice if you are not using automatic hostname resolution.

Installing Optional Features

Optional software is installed using the "features" setting in your Homestead configuration file. Most features can be enabled or disabled with a boolean value, while some features allow multiple configuration options:

```
features:
    - blackfire:
        server_id: "server_id"
        server_token: "server_value"
        client_id: "client_id"
        client_token: "client_value"
    - cassandra: true
    - chronograf: true
    - couchdb: true
    - crystal: true
    - docker: true
    - elasticsearch:
        version: 7
    - gearman: true
    - golang: true
    - grafana: true
    - influxdb: true
    - mariadb: true
    - minio: true
    - mongodb: true
    - mysql8: true
    - neo4j: true
    - ohmyzsh: true
    - openresty: true
    - pm2: true
    - python: true
    - rabbitmq: true
    - solr: true
    - webdriver: true
```

MariaDB

Enabling MariaDB will remove MySQL and install MariaDB. MariaDB serves as a drop-in replacement for MySQL, so you should still use the `mysql` database driver in your application's database configuration.

MongoDB

The default MongoDB installation will set the database username to `homestead` and the corresponding password to `secret`.

Elasticsearch

You may specify a supported version of Elasticsearch, which may be a major version or an exact version number (major.minor.patch). The default installation will create a cluster named 'homestead'. You should never give Elasticsearch more than half of the operating system's memory, so make sure your Homestead machine has at least twice the Elasticsearch allocation.

Check out the Elasticsearch documentation to learn how to customize your configuration.

Neo4j

The default Neo4j installation will set the database username to `homestead` and corresponding password to `secret`. To access the Neo4j browser, visit `http://homestead.test:7474` via your web browser. The ports `7687` (Bolt), `7474` (HTTP), and `7473` (HTTPS) are ready to serve requests from the Neo4j client.

Aliases

You may add Bash aliases to your Homestead machine by modifying the `aliases` file within your Homestead directory:

```
alias c='clear'
alias ..='cd ..'
```

After you have updated the `aliases` file, you should re-provision the Homestead machine using the `vagrant reload --provision` command. This will ensure that your new aliases are available on the machine.

Daily Usage

Accessing Homestead Globally

Sometimes you may want to `vagrant up` your Homestead machine from anywhere on your filesystem. You can do this on Mac / Linux systems by adding a Bash function to your Bash profile. On Windows, you may accomplish this by adding a "batch" file to your `PATH`. These scripts will allow you to run any Vagrant command from anywhere on your system and will automatically point that command to your Homestead installation:

Mac / Linux

```
function homestead() {
    ( cd ~/Homestead && vagrant $* )
}
```

Make sure to tweak the `~/Homestead` path in the function to the location of your actual Homestead installation. Once the function is installed, you may run commands like `homestead up` or `homestead ssh` from anywhere on your system.

Windows

Create a `homestead.bat` batch file anywhere on your machine with the following contents:

```
@echo off

set cwd=%cd%
set homesteadVagrant=C:\Homestead

cd /d %homesteadVagrant% && vagrant %*
cd /d %cwd%

set cwd=
set homesteadVagrant=
```

Make sure to tweak the example `C:\Homestead` path in the script to the actual location of your Homestead installation. After creating the file, add the file location to your `PATH`. You may then run commands like `homestead up` or `homestead ssh` from anywhere on your system.

Connecting Via SSH

You can SSH into your virtual machine by issuing the `vagrant ssh` terminal command from your Homestead directory.

But, since you will probably need to SSH into your Homestead machine frequently, consider adding the "function" described above to your host machine to quickly SSH into the Homestead box.

Connecting To Databases

A `homestead` database is configured for both MySQL and PostgreSQL out of the box. To connect to your MySQL or PostgreSQL database from your host machine's database client, you should connect to `127.0.0.1` and port `33060` (MySQL) or `54320` (PostgreSQL). The username and password for both databases is `homestead` / `secret`.

You should only use these non-standard ports when connecting to the databases from your host machine. You will use the default 3306 and 5432 ports in your Laravel database configuration file since Laravel is running *within* the virtual machine.

Database Backups

Homestead can automatically backup your database when your Vagrant box is destroyed. To utilize this feature, you must be using Vagrant 2.1.0 or greater. Or, if you are using an older version of Vagrant, you must install the `vagrant-triggers` plug-in. To enable automatic database backups, add the following line to your `Homestead.yaml` file:

```
backup: true
```

Once configured, Homestead will export your databases to `mysql_backup` and `postgres_backup` directories when the `vagrant destroy` command is executed. These

directories can be found in the folder where you cloned Homestead or in the root of your project if you are using the per project installation method.

Database Snapshots

Homestead supports freezing the state of MySQL and MariaDB databases and branching between them using Logical MySQL Manager. For example, imagine working on a site with a multi-gigabyte database. You can import the database and take a snapshot. After doing some work and creating some test content locally, you may quickly restore back to the original state.

Under the hood, LMM uses LVM's thin snapshot functionality with copy-on-write support. In practice, this means that changing a single row in a table will only cause the changes you made to be written to disk, saving significant time and disk space during restores.

Since `lmm` interacts with LVM, it must be run as `root`. To see all available commands, run `sudo lmm` inside your Vagrant box. A common workflow looks like the following:

1. Import a database into the default `master` lmm branch.
2. Save a snapshot of the unchanged database using `sudo lmm branch prod-YYYY-MM-DD`.
3. Modify the database.
4. Run `sudo lmm merge prod-YYYY-MM-DD` to undo all changes.
5. Run `sudo lmm delete <branch>` to delete unneeded branches.

Adding Additional Sites

Once your Homestead environment is provisioned and running, you may want to add additional Nginx sites for your Laravel applications. You can run as many Laravel installations as you wish on a single Homestead environment. To add an additional site, add the site to your `Homestead.yaml` file:

```
sites:
    - map: homestead.test
      to: /home/vagrant/project1/public
    - map: another.test
      to: /home/vagrant/project2/public
```

If Vagrant is not automatically managing your "hosts" file, you may need to add the new site to that file as well:

```
192.168.10.10  homestead.test
192.168.10.10  another.test
```

Once the site has been added, run the `vagrant reload --provision` command from your Homestead directory.

Site Types

Homestead supports several types of sites which allow you to easily run projects that are not based on Laravel. For example, we may easily add a Symfony application to Homestead using the `symfony2` site type:

```
sites:
    - map: symfony2.test
      to: /home/vagrant/my-symfony-project/web
      type: "symfony2"
```

The available site types are: `apache`, `apigility`, `expressive`, `laravel` (the default), `proxy`, `silverstripe`, `statamic`, `symfony2`, `symfony4`, and `zf`.

Site Parameters

You may add additional Nginx `fastcgi_param` values to your site via the `params` site directive. For example, we'll add a `FOO` parameter with a value of `BAR`:

```
sites:
    - map: homestead.test
      to: /home/vagrant/project1/public
      params:
          - key: FOO
            value: BAR
```

Environment Variables

You can set global environment variables by adding them to your `Homestead.yaml` file:

```
variables:
```

```
    - key: APP_ENV
      value: local
    - key: FOO
      value: bar
```

After updating the `Homestead.yaml`, be sure to re-provision the machine by running `vagrant reload --provision`. This will update the PHP-FPM configuration for all of the installed PHP versions and also update the environment for the `vagrant` user.

Configuring Cron Schedules

Laravel provides a convenient way to schedule Cron jobs by scheduling a single `schedule:run` Artisan command to be run every minute. The `schedule:run` command will examine the job schedule defined in your `App\Console\Kernel` class to determine which jobs should be run.

If you would like the `schedule:run` command to be run for a Homestead site, you may set the `schedule` option to `true` when defining the site:

```
sites:
    - map: homestead.test
      to: /home/vagrant/project1/public
      schedule: true
```

The Cron job for the site will be defined in the `/etc/cron.d` folder of the virtual machine.

Configuring Mailhog

Mailhog allows you to easily catch your outgoing email and examine it without actually sending the mail to its recipients. To get started, update your `.env` file to use the following mail settings:

```
MAIL_DRIVER=smtp
MAIL_HOST=localhost
MAIL_PORT=1025
MAIL_USERNAME=null
MAIL_PASSWORD=null
MAIL_ENCRYPTION=null
```

Once Mailhog has been configured, you may access the Mailhog dashboard at `http://localhost:8025`.

Configuring Minio

Minio is an open source object storage server with an Amazon S3 compatible API. To install Minio, update your `Homestead.yaml` file with the following configuration option in the features section:

```
minio: true
```

By default, Minio is available on port 9600. You may access the Minio control panel by visiting `http://localhost:9600/`. The default access key is `homestead`, while the default secret key is `secretkey`. When accessing Minio, you should always use region `us-east-1`.

In order to use Minio you will need to adjust the S3 disk configuration in your `config/filesystems.php` configuration file. You will need to add the `use_path_style_endpoint` option to the disk configuration, as well as change the `url` key to `endpoint`:

```
's3' => [
    'driver' => 's3',
    'key' => env('AWS_ACCESS_KEY_ID'),
    'secret' => env('AWS_SECRET_ACCESS_KEY'),
    'region' => env('AWS_DEFAULT_REGION'),
    'bucket' => env('AWS_BUCKET'),
    'endpoint' => env('AWS_URL'),
    'use_path_style_endpoint' => true
]
```

Finally, ensure your `.env` file has the following options:

```
AWS_ACCESS_KEY_ID=homestead
AWS_SECRET_ACCESS_KEY=secretkey
AWS_DEFAULT_REGION=us-east-1
AWS_URL=http://localhost:9600
```

To provision buckets, add a `buckets` directive to your Homestead configuration file:

```
buckets:
    - name: your-bucket
      policy: public
    - name: your-private-bucket
      policy: none
```

Supported `policy` values include: `none`, `download`, `upload`, and `public`.

Ports

By default, the following ports are forwarded to your Homestead environment:

- **SSH:** 2222 → Forwards To 22
- **ngrok UI:** 4040 → Forwards To 4040
- **HTTP:** 8000 → Forwards To 80
- **HTTPS:** 44300 → Forwards To 443
- **MySQL:** 33060 → Forwards To 3306
- **PostgreSQL:** 54320 → Forwards To 5432
- **MongoDB:** 27017 → Forwards To 27017
- **Mailhog:** 8025 → Forwards To 8025
- **Minio:** 9600 → Forwards To 9600

Forwarding Additional Ports

If you wish, you may forward additional ports to the Vagrant box, as well as specify their protocol:

```
ports:
    - send: 50000
      to: 5000
    - send: 7777
      to: 777
      protocol: udp
```

Sharing Your Environment

Sometimes you may wish to share what you're currently working on with coworkers or a client. Vagrant has a built-in way to support this via `vagrant share`; however, this will not work if you have multiple sites configured in your `Homestead.yaml` file.

To solve this problem, Homestead includes its own `share` command. To get started, SSH into your Homestead machine via `vagrant ssh` and run `share homestead.test`. This will share the `homestead.test` site from your `Homestead.yaml` configuration file. You may substitute any of your other configured sites for `homestead.test`:

```
share homestead.test
```

After running the command, you will see an Ngrok screen appear which contains the activity log and the publicly accessible URLs for the shared site. If you would like to specify a custom region, subdomain, or other Ngrok runtime option, you may add them to your `share` command:

```
share homestead.test -region=eu -subdomain=laravel
```

> Remember, Vagrant is inherently insecure and you are exposing your virtual machine to the Internet when running the `share` command.

Multiple PHP Versions

Homestead 6 introduced support for multiple versions of PHP on the same virtual machine. You may specify which version of PHP to use for a given site within your `Homestead.yaml` file. The available PHP versions are: "5.6", "7.0", "7.1", "7.2" and "7.3" (the default):

```
sites:
    - map: homestead.test
      to: /home/vagrant/project1/public
      php: "7.1"
```

In addition, you may use any of the supported PHP versions via the CLI:

```
php5.6 artisan list
php7.0 artisan list
php7.1 artisan list
php7.2 artisan list
php7.3 artisan list
```

You may also update the default CLI version by issuing the following commands from within your Homestead virtual machine:

```
php56
php70
php71
php72
php73
```

Web Servers

Homestead uses the Nginx web server by default. However, it can install Apache if `apache` is specified as a site type. While both web servers can be installed at the same time, they cannot both be *running* at the same time. The `flip` shell command is available to ease the process of switching between web servers. The `flip` command automatically determines which web server is running, shuts it off, and then starts the other server. To use this command, SSH into your Homestead machine and run the command in your terminal:

```
flip
```

Mail

Homestead includes the Postfix mail transfer agent, which is listening on port `1025` by default. So, you may instruct your application to use the `smtp` mail driver on `localhost` port `1025`. Then, all sent mail will be handled by Postfix and caught by Mailhog. To view your sent emails, open http://localhost:8025 in your web browser.

Debugging & Profiling

Debugging Web Requests With Xdebug

Homestead includes support for step debugging using Xdebug. For example, you can load a web page from a browser, and PHP will connect to your IDE to allow inspection and modification of the running code.

By default Xdebug is already running and ready to accept connections. If you need to enable Xdebug on the CLI run the `sudo phpenmod xdebug` command within your Vagrant box. Next, follow your IDE's instructions to enable debugging. Finally, configure your browser to trigger Xdebug with an extension or bookmarklet.

Xdebug causes PHP to run significantly slower. To disable Xdebug, run `sudo phpdismod xdebug` within your Vagrant box and restart the FPM service.

Debugging CLI Applications

To debug a PHP CLI application, use the `xphp` shell alias inside your Vagrant box:

```
xphp path/to/script
```

Autostarting Xdebug

When debugging functional tests that make requests to the web server, it is easier to autostart debugging rather than modifying tests to pass through a custom header or cookie to trigger debugging. To force Xdebug to start automatically, modify `/etc/php/7.#/fpm/conf.d/20-xdebug.ini` inside your Vagrant box and add the following configuration:

```
; If Homestead.yml contains a different subnet for the IP address, this address ma
xdebug.remote_host = 192.168.10.1
xdebug.remote_autostart = 1
```

Profiling Applications with Blackfire

Blackfire is a SaaS service for profiling web requests and CLI applications and writing performance assertions. It offers an interactive user interface which displays profile data in call-graphs and timelines. It is built for use in development, staging, and production, with no overhead for end users. It provides performance, quality, and security checks on code and `php.ini` configuration settings.

The Blackfire Player is an open-source Web Crawling, Web Testing and Web Scraping application which can work jointly with Blackfire in order to script profiling scenarios.

To enable Blackfire, use the "features" setting in your Homestead configuration file:

```
features:
    - blackfire:
        server_id: "server_id"
        server_token: "server_value"
        client_id: "client_id"
        client_token: "client_value"
```

Blackfire server credentials and client credentials require a user account. Blackfire offers various options to profile an application, including a CLI tool and browser extension. Please review the Blackfire documentation for more details.

Profiling PHP Performance Using XHGui

XHGui is a user interface for exploring the performance of your PHP applications. To enable XHGui, add `xhgui: 'true'` to your site configuration:

```
sites:
    -
        map: your-site.test
        to: /home/vagrant/your-site/public
        type: "apache"
        xhgui: 'true'
```

If the site already exists, make sure to run `vagrant provision` after updating your configuration.

To profile a web request, add `xhgui=on` as a query parameter to a request. XHGui will automatically attach a cookie to the response so that subsequent requests do not need the query string value. You may view your application profile results by browsing to `http://your-site.test/xhgui`.

To profile a CLI request using XHGui, prefix the command with `XHGUI=on`:

```
XHGUI=on path/to/script
```

CLI profile results may be viewed in the same way as web profile results.

Note that the act of profiling slows down script execution, and absolute times may be as much as twice as real-world requests. Therefore, always compare percentage improvements and not absolute numbers. Also, be aware the execution time includes any time spent paused in a debugger.

Since performance profiles take up significant disk space, they are deleted automatically after a few days.

Network Interfaces

The `networks` property of the `Homestead.yaml` configures network interfaces for your Homestead environment. You may configure as many interfaces as necessary:

```
networks:
    - type: "private_network"
      ip: "192.168.10.20"
```

To enable a bridged interface, configure a `bridge` setting and change the network type to `public_network`:

```
networks:
    - type: "public_network"
      ip: "192.168.10.20"
      bridge: "en1: Wi-Fi (AirPort)"
```

To enable DHCP, just remove the `ip` option from your configuration:

```
networks:
    - type: "public_network"
      bridge: "en1: Wi-Fi (AirPort)"
```

Extending Homestead

You may extend Homestead using the `after.sh` script in the root of your Homestead directory. Within this file, you may add any shell commands that are necessary to properly configure and customize your virtual machine.

When customizing Homestead, Ubuntu may ask you if you would like to keep a package's original configuration or overwrite it with a new configuration file. To avoid this, you should use the following command when installing packages to avoid overwriting any configuration previously written by Homestead:

```
sudo apt-get -y \
    -o Dpkg::Options::="--force-confdef" \
    -o Dpkg::Options::="--force-confold" \
    install your-package
```

User Customizations

When using Homestead in a team setting, you may want to tweak Homestead to better fit your personal development style. You may create a `user-customizations.sh` file in the root of your Homestead directory (The same directory containing your `Homestead.yaml`). Within this file, you may make any customization you would like; however, the `user-customizations.sh` should not be version controlled.

Updating Homestead

Before you begin updating Homestead ensure you have removed your current virtual machine by running the following command in your Homestead directory:

```
vagrant destroy
```

Next, you need to update the Homestead source code. If you cloned the repository you can run the following commands at the location you originally cloned the repository:

```
git fetch

git pull origin release
```

These commands pull the latest Homestead code from the GitHub repository, fetches the latest tags, and then checks out the latest tagged release. You can find the latest stable release version on the GitHub releases page.

If you have installed Homestead via your project's `composer.json` file, you should ensure your `composer.json` file contains `"laravel/homestead": "^9"` and update your dependencies:

```
composer update
```

Then, you should update the Vagrant box using the `vagrant box update` command:

```
vagrant box update
```

Finally, you will need to regenerate your Homestead box to utilize the latest Vagrant installation:

```
vagrant up
```

Provider Specific Settings

VirtualBox

`natdnshostresolver`

By default, Homestead configures the `natdnshostresolver` setting to `on`. This allows Homestead to use your host operating system's DNS settings. If you would like to override this behavior, add the following lines to your `Homestead.yaml` file:

```
provider: virtualbox
natdnshostresolver: 'off'
```

Symbolic Links On Windows

If symbolic links are not working properly on your Windows machine, you may need to add the following block to your `Vagrantfile`:

```
config.vm.provider "virtualbox" do |v|
    v.customize ["setextradata", :id, "VBoxInternal2/SharedFoldersEnableSymlinksCr
end
```

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.