Installation Deployment

Installation Deployment

- How do I install Aero on Mac?
- How do I install Aero on Windows?
- What are the system requirements?

How do I install Aero on Mac?

Before continuing, please ensure your Mac is running macOS Mojave (10.14) or newer.

There are a couple of ways to develop locally on the Mac:

- Valet – manually install the services directly on your Mac. - Homestead – a virtual machine that runs the required services.

Valet

Laravel Valet is a development environment for Mac minimalists. It provides a blazing fast local development environment with minimal resource consumption, and points all requests on the `*.test` domain to sites installed on the local machine.

Step 1: Install Homebrew

Visit the Homebrew website and follow the instructions to install Homebrew.

Make sure everything is up-to-date by running:

```
brew update
```

Step 2: Install the services

Run the following command to install the services needed:

```
brew install php@7.4 mysql openjdk@8 elasticsearch@6 node composer
```

The services can then be started with the following commands.

Start MySQL:

```
brew services start mysql
```

Start Elasticsearch:

```
brew services start elasticsearch@6
```

Make sure to place Composer's system-wide vendor bin directory in your `$PATH`:

```
export PATH="$PATH:$HOME/.composer/vendor/bin"
```

Step 3: Install Valet

You can then pull in the Valet package using Composer and install it using the following commands.

First require the Valet package:

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

Then install the package:

```
valet install
```

Step 4: Create a database

To connect to the local MySQL server, run the following in your terminal application:

```
mysql -u root
```

Once connected, create a new database for your project. It should be named something that relates to your project to ease managing multiple local projects in the future. Run the following SQL, **replacing `[project-name]` with the name of your project**:

```
create database [project-name];
```

If your database name contains hyphens, you must wrap the name in back-ticks, e.g. `my-first-store`.

You can then disconnect from the MySQL server by typing:

```
exit;
```

Step 5: Install the Aero Commerce CLI tool

If this is your first project, you'll need to download the command-line tool using Composer:

```
composer global require aerocommerce/cli
```

Once installed, you'll have access to the `aero new` command. This will create a fresh project installation in the directory you specify.

For instance, `aero new myfirststore` will create a directory named `myfirststore` containing an Aero project with all dependencies installed:

```
aero new [project-name]
```

During installation, you'll need to enter the package repository credentials and database connection details for the project. The installer will setup the database tables and seed the project with essential information.

Step 6: Link the project to Valet

The final thing to do is configure the project with Valet. This is achieved by running the `valet link` command from within the project's directory. You can provide a name to use as the domain, for example, to proxy all requests made to `http://myfirststore.test` to the project, the command would be `valet link myfirststore`:

```
valet link [project-name]
```

You can now open your web browser and visit your newly created ecommerce store!

Homestead

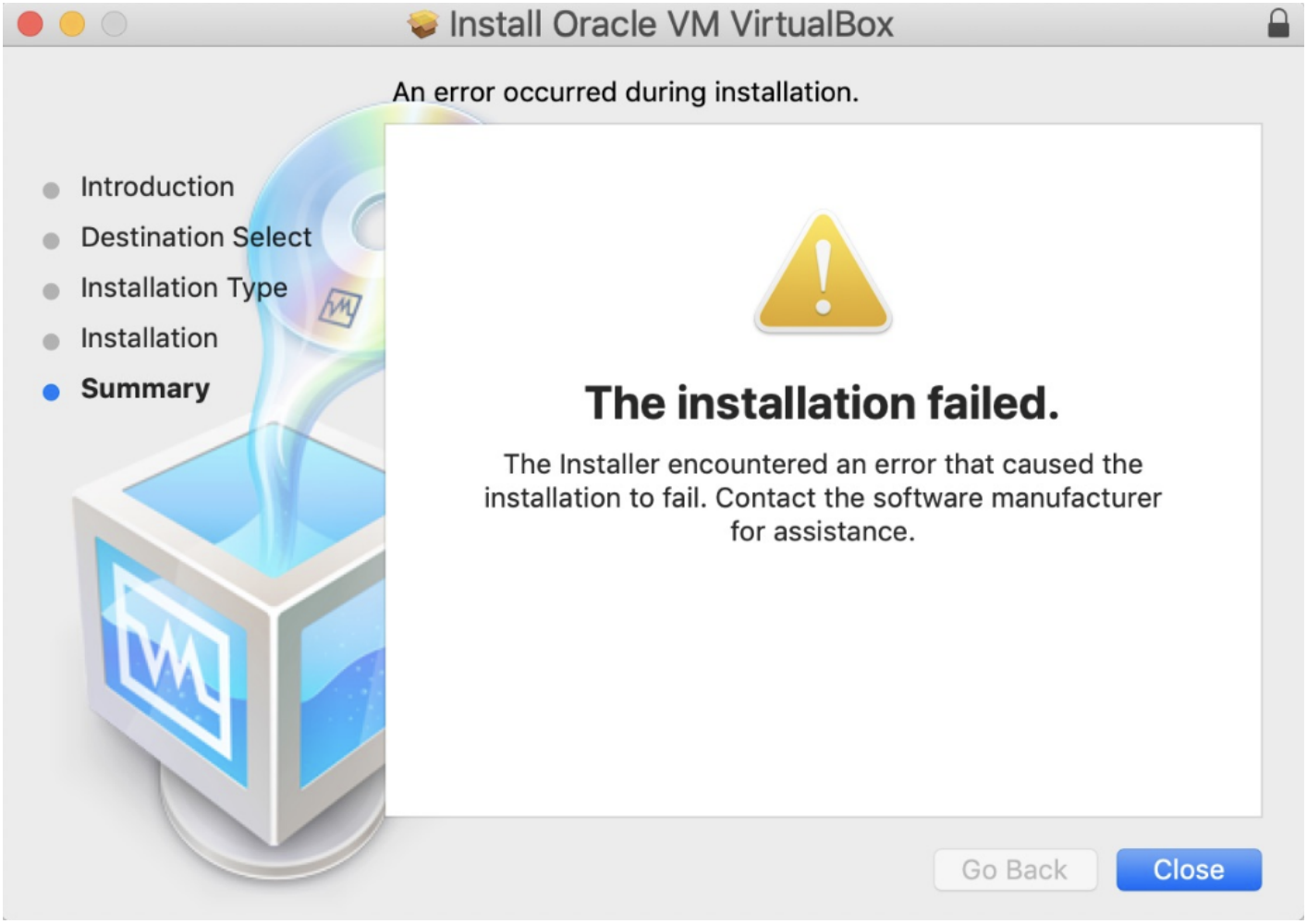
We are currently unable to offer support for the configuration of Homestead, or any issues associated during the installation process.

Laravel Homestead is a pre-packaged Vagrant box that provides a development environment without requiring you to install any server software on your local machine.

Step 1: Install VirtualBox

Firstly, you'll need to download VirtualBox 6.x. Select the OS X hosts binary and run the installer.

During installation, you may encounter the following error screen:



This is due to security restrictions in macOS. To resolve this, open your terminal application and run the command:

```
sudo spctl --master-disable
```

Retry the installation. If the problem still persists, try restarting your Mac and running the installer again.

Once you've successfully completed the installation wizard, you should re-enable Gatekeeper by running the command:

```
sudo spctl --master-enable
```

Step 2: Install Vagrant

Head over to the Vagrant download page and select the macOS 64-bit link. Follow the prompts on the installation wizard.

Once complete, you can confirm Vagrant is on your machine by typing the following command into your terminal application to display the Vagrant version:

```
vagrant -v
```

Step 3: Install the Homestead Vagrant box

Run the following command in your terminal application:

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

The download may take several minutes, depending on your internet connection.

Step 4: Install Homestead

The Homestead GitHub repository needs to be cloned to your local machine. To do this, run the command:

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

This will create a copy of the repository in a new folder named `Homestead` within your home directory.

Step 5: Configure Homestead

Next, you'll need to create a default `Homestead.yaml` file which will serve as the instructions to configure your virtual machine. This can be done using the following commands:

```
$ cd ~/Homestead
$ git checkout release
$ bash init.sh
```

If you open the newly created Homestead.yaml file, you should see a format similar to this:

```
---
ip: "192.168.10.10"
memory: 2048
```

```
cpus: 2
provider: virtualbox

authorize: ~/.ssh/id_rsa.pub

keys:
    - ~/.ssh/id_rsa

folders:
    - map: ~/myfirststore
      to: /home/vagrant/myfirststore

sites:
    - map: myfirststore.test
      to: /home/vagrant/myfirststore/public
      php: "7.4"

databases:
    - homestead

features:
    - elasticsearch:
        version: "6.8.6"
```

The `authorize` and `keys` properties specify the SSH key pair that will be used to connect to the virtual machine. You can generate the `id_rsa` files using the command below. There is no need to enter any information when prompted, so just hit enter to use the default values.

```
ssh-keygen
```

The `folders` property lists all of the folders to be shared between your Mac and the virtual machine. This allows you to develop using your local IDE (code editor), whilst also providing the web server and other services on the virtual machine access to the project files.

In the example `Homestead.yaml` shown above, we've specified to share the `~/myfirststore` folder. This is going to be the local directory of our new Aero project, however the folder does not yet exist. To create this directory, run the following command, **replacing `[project-name]` with the name of your project**:

```
mkdir ~/[project-name]
```

This empty directory will be populated later on, from within the virtual machine. The `to` value of this folder mapping is the project's path on the virtual machine, e.g. `/home/vagrant/[project-name]`.

The `sites` property allows you to define which domains will provide access to the projects. Using the example above, you can see the `myfirststore.test` domain is configured to point to the project's `public` directory, which is the web server entry point to the project. Ensure the `to` value is the virtual machine's project path and not the local one, i.e. `/home/vagrant/[project-name]/public`. Remember to replace `[project-name]` with the name of your project.

Homestead will automatically create the databases listed in the `database` property of the configuration file. By default, the database name is `homestead`. The installer will automatically detect this database later on, however, if you plan on developing multiple projects, you should name the databases in this list to match your projects.

Finally, you need to ensure that your `Homestead.yaml` configuration file contains `elasticsearch` in the `features` property, along with the `version: 6` declaration.

Step 6: Add the domain to the hosts file

As the project is hosted locally, we need to add an entry to the `/etc/hosts` file, which will route all requests made to the development domain to your project:

```
sudo vi /etc/hosts
```

Open the `/etc/hosts` file in `vi` and add a record to point the domain to the IP address of the virtual machine (the IP listed in the `Hometead.yaml` configuration file). Press i to enter insert mode, and use the arrow keys on the keyboard to move the cursor to the bottom of the file. Add the following line, **replacing `[project-name]` with the domain name entered in the sites property of your `Homestead.yaml` configuration file**:

```
192.168.10.10   [project-name].test
```

To save the changes and exit the file, press Control + c to leave insert mode, and then type :wq! followed by pressing the Enter key.

Step 7: Provision the virtual machine

You are now ready to boot up the virtual machine. To do so, open your terminal application and navigate to the directory that Homestead was installed in earlier. You can then run the `vagrant up` command, which will read from the `Homestead.yaml` file and configure the virtual machine.

```
$ cd ~/Homestead
$ vagrant up
```

Step 8: Install the Aero Commerce CLI tool

The CLI tool needs to be installed on the virtual machine. To do this, SSH into the virtual machine using the command:

```
vagrant ssh
```

From within the virtual machine you will need to set the default PHP version by running:

```
php74
```

Next, download the command-line tool using Composer:

```
composer global require aerocommerce/cli
```

Once installed, you'll have access to the `aero new` command, which needs to be ran from within the project directory on the virtual machine. For instance, running `aero new .` from within the directory `myfirststore` will scaffold an Aero project with all dependencies installed:

```
$ cd ~/myfirststore
$ aero new .
```

During installation, you'll need to enter the package repository credentials and database connection details for the project. The installer will setup the database tables and seed the project with essential information.

You can now open your web browser and visit your newly created ecommerce store!

How do I install Aero on Windows?

Homestead

We are currently unable to offer support for the configuration of Homestead, or any issues associated during the installation process.

Laravel Homestead is a pre-packaged Vagrant box that provides a development environment without requiring you to install any server software on your local machine.

Step 1: Enable hardware virtualization

Modern CPUs include hardware virtualization features that help accelerate virtual machines created in VirtualBox. We recommend consulting your computer's user guide on how to enable VT-x/AMD-V in your BIOS or UEFI firmware.

Step 2: Install VirtualBox and Vagrant

Firstly, you'll need to download VirtualBox 6.x. Select the Windows hosts binary and run the installer.

Next we'll head over to the Vagrant download page and select the Windows link. Follow the prompts on the installation wizard.

Once complete, you can confirm Vagrant is on your machine by typing the following command into a PowerShell window:

```
vagrant -v
```

Step 3: Install the Homestead Vagrant box

Open up a PowerShell window and run the following command:

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

The download may take several minutes, depending on your internet connection.

Step 4: Install Homestead

The Homestead GitHub repository needs to be cloned to your local machine.

If you have not yet installed Git on your machine, head over to gitforwindows.org to download.

With Git available on your machine, run the following command from within a PowerShell window:

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

This will create a copy of the repository in a new folder named `Homestead` within your home directory.

Next, you'll need to create a default `Homestead.yaml` file which will serve as the instructions to configure your virtual machine. This can be done using the following commands:

```
cd ~/Homestead
git checkout release
./init.bat
```

If you open the newly created Homestead.yaml file, you should see a format similar to this:

```
---
ip: "192.168.10.10"
memory: 2048
cpus: 2
provider: virtualbox

authorize: C:\Users\user\.ssh\id_rsa.pub

keys:
    - C:\Users\user\.ssh\id_rsa

folders:
    - map: C:\Users\user\Code\myfirststore
      to: /home/vagrant/myfirststore
      type: "nfs"
```

```
sites:
    - map: myfirststore.test
      to: /home/vagrant/myfirststore/public
      php: "7.4"

databases:
    - homestead

features:
    - elasticsearch:
        version: "6.8.6"
```

The `authorize` and `keys` properties specify the SSH key pair that will be used to connect to the virtual machine. You can generate the `id_rsa` files using the command below in a PowerShell window. There is no need to enter any information when prompted, so just hit enter to use the default values.

```
ssh-keygen
```

The `folders` property lists all of the folders to be shared between your Windows host and the virtual machine. This allows you to develop using your local IDE (code editor), whilst also providing the web server and other services on the virtual machine access to the project files.

The `~/` syntax is not valid when running on Windows. Instead use the full path, e.g. `C:\Users\user\myfirststore`

In the example `Homestead.yaml` shown above, we've specified to share the `C:\Users\user\Code\myfirststore` folder. This is going to be the local directory of our new Aero project, however the folder does not yet exist. Create a directory using *Windows Explorer* and update the `Homestead.yaml` file.

This empty directory will be populated later on, from within the virtual machine. The `to` value of this folder mapping is the project's path on the virtual machine, e.g. `/home/vagrant/[project-name]`.

The `sites` property allows you to define which domains will provide access to the projects. Using the example above, you can see the `myfirststore.test` domain is configured to point to the project's `public` directory, which is the web server entry point to the project. Ensure the `to` value is the virtual machine's project path and not the local one, i.e. `/home/vagrant/[project-name]/public`. **Remember to replace `[project-name]` with the name of your project.**

Homestead will automatically create the databases listed in the `database` property of the configuration file. By default, the database name is `homestead`. The installer will automatically detect this database later on, however, if you plan on developing multiple projects, you should

name the databases in this list to match your projects.

Finally, you need to ensure that your `Homestead.yaml` configuration file contains `elasticsearch` in the `features` property, along with the `version: 6` declaration.

Step 5: Add the domain to the hosts file

As the project is hosted locally, we need to add an entry to the `C:\Windows\System32\drivers\etc\hosts` file, which will route all requests made to the development domain to your project.

Press the *Windows* key and type *notepad* in the search field. Right-click on *Notepad* in the search results and select *Run as administrator*. Open the `C:\Windows\System32\Drivers\etc\hosts` file and add the following line, **replacing `[project-name]` with the domain name entered in the sites property of your `Homestead.yaml` configuration file**:

```
192.168.10.10   [project-name].test
```

Save the file and close *Notepad*.

Step 6: Add the NFS for Vagrant plugin

A plugin is required when using NFS to sync folders on Windows and will maintain the correct user / group permissions. To install the plugin, run the command in a PowerShell window:

```
vagrant plugin install vagrant-winnfsd
```

You'll then need to open up the `Vagrantfile` found in your Homestead directory and add the following line to the end of the file (within the config area):

```
config.vm.network "private_network", type: "dhcp"
```

Step 7: Provision the virtual machine

You are now ready to boot up the virtual machine. To do so, open a PowerShell window and navigate to the directory that Homestead was installed in earlier. You can then run the `vagrant up` command, which will read from the `Homestead.yaml` file and configure the virtual machine. **The virtual machine must be ran with administrator privileges to ensure files are synced correctly**.

```
cd ~/Homestead
Start-Process powershell -Verb runAs
vagrant up
```

Step 8: Install the Aero Commerce CLI tool

The CLI tool needs to be installed on the virtual machine. To do this, SSH into the virtual machine using the following command from a PowerShell window:

```
vagrant ssh
```

From within the virtual machine you will need to set the default PHP version by running:

```
php74
```

Next, download the command-line tool using Composer:

```
composer global require aerocommerce/cli
```

Once installed, you'll have access to the `aero new` command, which needs to be ran from within the project directory on the virtual machine. For instance, running `aero new .` from within the directory `myfirststore` will scaffold an Aero project with all dependencies installed:

```
cd ~/myfirststore
aero new .
```

During installation, you'll need to enter the package repository credentials and database connection details for the project. The installer will setup the database tables and seed the project with essential information.

You can now open your web browser and visit your newly created ecommerce store!

What are the system requirements?

Your system must satisfy the following requirements:

- PHP (>= 7.2) with BCMath, Ctype, JSON, Mbstring, OpenSSL, PDO, Tokenizer, XML, cURL,GD/ImageMagick
- Composer
- Nginx or Apache
- Node.js (>= 8.10)
- MySQL (>= 5.7)
- Elasticsearch (6.*)