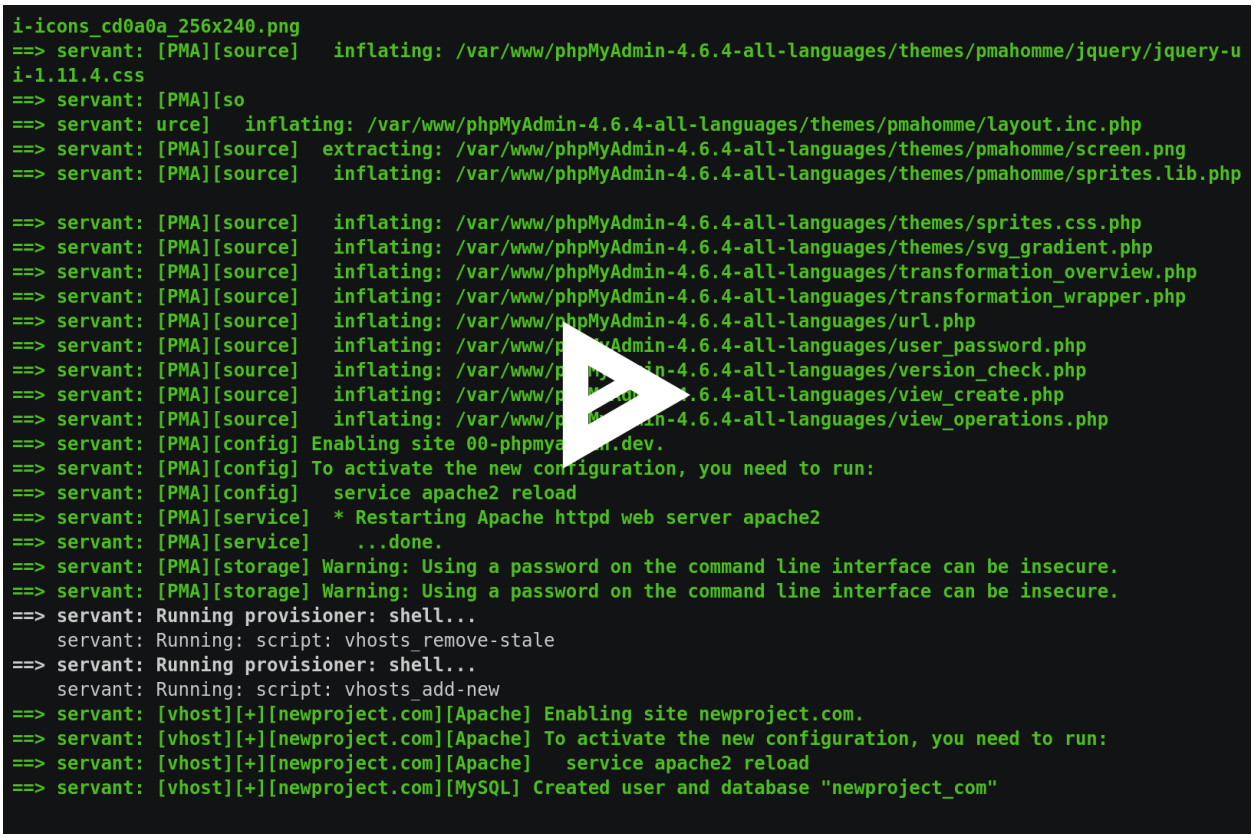
servant Documentation

Release 1.0

Jonas Friedmann

September 20, 2016



servant is a custom Vagrant virtual machine which offers a straightforward and easy to use web-development system based on services like Apache, PHP (FPM), MySQL and phpMyAdmin, but isolated from your host system. Primary goal is to provide a consistent dev environment for developers or employees of a small company/startup.

CHAPTER 1

Features

- Isolated from OS X host system (OS updates won't affect the dev services, ever)
- Performant (PHP-FPM and OPcache module enabled)
- Easily add and remove projects (virtual hosts), **servant** automatically creates the necessary web server configurations as well as a MySQL database
- Automatically write/update */etc/hosts* file on your Mac
- Supports PHP 5.6 and alternatively 5.5
- Supports MySQL 5.6 and alternatively 5.5

1.1 Requirements

- Vagrant
- **VirtualBox**
  - vagrant-bindfs plugin
  - vagrant-servant-hosts-provisioner plugin (custom fork)
  - vagrant-triggers plugin

1.2 Installation

1. Make sure you've installed the required Vagrant plugins:

```
vagrant plugin install vagrant-bindfs vagrant-servant-hosts-provisioner
```

2. Clone this repository:

```
cd
git clone https://github.com/frdmn/servant
```

3. Copy the sample configuration file into your `$HOME`:

```
cp servant/config.json ~/.servant.json
```

1.3 General

Vagrant commands expects to be executed from within the directory where the `Vagrantfile` is located. Make sure to change into the directory where you've cloned **servant** to, when running the commands below.

1.3.1 Start/create virtual machine

To initially create the **servant** machine, just run:

```
vagrant up
```

1.3.2 Shutdown/end VM

To power off the virtual machine:

```
vagrant suspend
```

1.3.3 Login via SSH

Vagrant provides a passwordless SSH login using:

```
vagrant ssh
```

1.3.4 Reload servant

```
vagrant provision
```

1.4 Virtual hosts

You need to work on multiple projects simultanously? Not a problem, **servant** supports virtual hosts using a straightforward rule: Each directory in the `public/` folder represents the hostname of the project you want to work on. A single command creates the necessary configurations within the virtual machine, reloads the services and rewrites the `/etc/hosts` file on your Mac.

1.4.1 Access document root

You can reach the root directory of the web server via http://servant.dev

1.4.2 Create new virtual hosts

1. Change into the **servant** directory:

```
cd ~/servant
```

2. Create a new folder inside *public/* named exactly the same as the hostname of your project:

```
mkdir public/testproject.io
```

3. Run the provision command to create the server configurations and update your `/etc/hosts` file locally:

```
vagrant provision
```

1.4.3 Delete virtual hosts

To remove a virtual host, simply delete the directory that represents the hostname of your project:

```
rm -r public/testproject.io
```

Don't forget to reload **servant**:

```
vagrant provision
```

1.4.4 Customizations

You can override the default Apache web server configuration for your virtual host in case you need a custom DocumentRoot or an additonal ServerAlias. To do that you need to place a JSON configuration file named `servant.json` in your project root folder. Checkout the example below:

```
{
  "document_root": "blog",
  "server_alias": "aliasdomain.com"
}
```

1.5 MySQL databases

When you setup a virtual hosts, the system also creates a MySQL database and user dedicated to that project. For simplicity's sake the database, username as well as the password represent (again) the hostname of your project. However, because of some naming restrictions within MySQL, dots (`.`) are replaced with underscores (_) and only the first 16 characters of the hostname is used. Some examples:

- If your project is using the hostname "testproject.io", the database, username and password will be `testproject_io`
- If the hostname is "prettylonghostname.com", it'll be `prettylonghostna`

1.5.1 System credentials

The following credentials are created by default in every **servant** environment:

Username	Password
root	The password set in *Configuration file*
phpmyadmin	`phpmyadmin`

1.5.2 phpMyAdmin

To manage your databases you can use the builtin phpMyAdmin: http://phpmyadmin.dev

1.5.3 Adminer

Alternatively to phpMyAdmin, you can use the super leightweight Adminer platform: http://adminer.dev

Please note: There is no quick select login functionality to authenticate, like in PMA.

1.5.4 Create MySQL backups/dumps

To create a SQL backup or dump file you just need to create an empty file named `create-mysql-backup` in the root folder of your project / virtual host (not the `htdocs` folder!) and reload **servant**:

```
cd public/testproject.io
touch create-mysql-backup

vagrant provision
```

The formula dumps all available databases that you might created with the given MySQL user of that virtual host. As soon as the privisioning process is completed, you can find your backups in the ‘‘backup/‘ folder within your project root:

```
cd public/testproject.io
ls -l backups
```

```
-rw-r--r--  1 user  staff  1283 18 Sep 13:34 201609171334_testproject_io.sql
-rw-r--r--  1 user  staff  1283 18 Sep 00:30 201609180030_testproject_io.sql
```

1.5.5 Import MySQL backups/dumps

In case you need to import an existing dump, maybe even right when the project gets bootstrapped during the creation process, rename your dump to `import.sql` and place it within your project root folder. After that reload **servant**:

```
cd public/testproject.io
mv /path/to/dump.sql import.sql

vagrant provision
```

1.5.6 Destruction backups

In case you destory the Vagrant machine, there is a hook that creates backups of every MySQL database before proceeding to destroy the machine. You can find the backups right within `public/` folder, prefix with `pre-destroy_`:

```
cd public
ls -l
```

```
drwxr-xr-x  6 user  staff      204 18 Sep 00:31 testproject.io
drwxr-xr-x  6 user  staff      204 18 Sep 00:30 testproject2.io
-rw-r--r--  1 user  staff   660906 18 Sep 00:55 pre-destroy_mysql.sql
-rw-r--r--  1 user  staff     1284 18 Sep 00:55 pre-destroy_testproject2_io.sql
-rw-r--r--  1 user  staff     1283 18 Sep 00:55 pre-destroy_testproject_io.sql
-rw-r--r--  1 user  staff  1688732 18 Sep 00:55 pre-destroy_performance_schema.sql
-rw-r--r--  1 user  staff    19082 18 Sep 00:55 pre-destroy_phpmyadmin.sql
```

1.6 PHP

To make sure your PHP applications run performant, **servant** is using FPM as handler and comes with enabled OPcache module to speed up your PHP processing.

1.6.1 Default settings

When the virtual machine is booted for the first time, some of the PHP configurations are set as listed below:

Setting	Value
`error_reporting`	`root`
`display_errors`	`phpmyadmin`
`date.timezone`	The timezone set in *Configuration file*
`xdebug.show_local_vars`	`1`
`xdebug.var_display_max_depth`	`5`
`xdebug.var_display_max_children`	`256`
`xdebug.var_display_max_data`	`1024`

1.6.2 phpinfo()

There's a built in phpinfo page if you are interested in the current loaded settings: http://phpinfo.dev

1.6.3 Customizations

Just like with the web server configuration, you can override the global PHP settings per virtual host in case you need a special PHP environment. Just create a `.user.ini` file inside your document root (`htdocs` by default) and insert your custom configuration:

```
always_populate_raw_post_data = -1
memory_limit = 512M
```

1.7 Configuration file

To provide some basic customization, you can edit your config file `config.json` within the **servant** root folder. A sample configuration file is available in the same directory as `config.sample.json`.

Caution: If you change the configuration and want to apply the changes you need to destroy and recreate the **servant** machine up from scratch. This means you loose your SQL databases so make sure to dump them before!

1.7.1 server.ip

IP address the virtual machine should use. Make sure to use a not existing subnet (eg. 192.168.1.10) to avoid IP collisions.

Default value `192.168.50.10`

1.7.2 server.cpus

The amount of CPU cores the machine should use.

Default value `1`

1.7.3 server.memory

And the allocated memory in MB.

Default value `1024`

1.7.4 server.swap

If you want to use swap, you can use this setting. Represents MB if not `false`.

Possible values `false` or `512`

Default value `false`

1.7.5 server.timezone

The timezone PHP and the OS should use. For a list of possible timezones, click here.

Default value `Europe/Berlin`

1.7.6 mysql.root_password

The MySQL root password.

Default value `root`

1.7.7 mysql.version

MySQL server version to install and use.

Possible value `5.6` or `5.5`

Default value `5.6`

1.7.8 php.version

PHP server version to install and use.

Possible value `5.6` or `5.5`

Default value `5.6`

CHAPTER 2

Indices and tables

- search