
servant Documentation

Release 1.0

Jonas Friedmann

Mar 06, 2017

1	Features	3
1.1	Requirements	3
1.2	Installation	3
1.3	General	4
1.4	Virtual hosts	4
1.5	MySQL databases	5
1.6	PHP	7
1.7	Configuration file	8
2	Indices and tables	11


```
i-icons_cd0a0a_256x240.png
==> servant: [PMA][source] inflating: /var/www/phpMyAdmin-4.6.4-all-languages/themes/pmahomme/jquery/jquery-u
i-1.11.4.css
==> servant: [PMA][so
==> servant: [urce] inflating: /var/www/phpMyAdmin-4.6.4-all-languages/themes/pmahomme/layout.inc.php
==> servant: [PMA][source] extracting: /var/www/phpMyAdmin-4.6.4-all-languages/themes/pmahomme/screen.png
==> servant: [PMA][source] inflating: /var/www/phpMyAdmin-4.6.4-all-languages/themes/pmahomme/sprites.lib.php

==> servant: [PMA][source] inflating: /var/www/phpMyAdmin-4.6.4-all-languages/themes/sprites.css.php
==> servant: [PMA][source] inflating: /var/www/phpMyAdmin-4.6.4-all-languages/themes/svg_gradient.php
==> servant: [PMA][source] inflating: /var/www/phpMyAdmin-4.6.4-all-languages/transformation_overview.php
==> servant: [PMA][source] inflating: /var/www/phpMyAdmin-4.6.4-all-languages/transformation_wrapper.php
==> servant: [PMA][source] inflating: /var/www/phpMyAdmin-4.6.4-all-languages/url.php
==> servant: [PMA][source] inflating: /var/www/phpMyAdmin-4.6.4-all-languages/user_password.php
==> servant: [PMA][source] inflating: /var/www/phpMyAdmin-4.6.4-all-languages/version_check.php
==> servant: [PMA][source] inflating: /var/www/phpMyAdmin-4.6.4-all-languages/view_create.php
==> servant: [PMA][source] inflating: /var/www/phpMyAdmin-4.6.4-all-languages/view_operations.php
==> servant: [PMA][config] Enabling site 00-phpmyadmin.dev.
==> servant: [PMA][config] To activate the new configuration, you need to run:
==> servant: [PMA][config] service apache2 reload
==> servant: [PMA][service] * Restarting Apache httpd web server apache2
==> servant: [PMA][service] ...done.
==> servant: [PMA][storage] Warning: Using a password on the command line interface can be insecure.
==> servant: [PMA][storage] Warning: Using a password on the command line interface can be insecure.
==> servant: Running provisioner: shell...
servant: Running: script: vhosts_remove-stale
==> servant: Running provisioner: shell...
servant: Running: script: vhosts_add-new
==> servant: [vhost][+][newproject.com][Apache] Enabling site newproject.com.
==> servant: [vhost][+][newproject.com][Apache] To activate the new configuration, you need to run:
==> servant: [vhost][+][newproject.com][Apache] service apache2 reload
==> servant: [vhost][+][newproject.com][MySQL] Created user and database "newproject_com"
```

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.

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

Requirements

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

Installation

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

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

2. Clone this repository:

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

3. Copy the sample configuration:

```
cp config.sample.json config.json
```

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.

Start/create virtual machine

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

```
vagrant up
```

Shutdown/end VM

To power off the virtual machine:

```
vagrant suspend
```

Login via SSH

Vagrant provides a passwordless SSH login using:

```
vagrant ssh
```

Reload servant

```
vagrant provision
```

Virtual hosts

You need to work on multiple projects simultaneously? Not a problem, **servant** supports virtual hosts using a straight-forward 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.

Access document root

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

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
```

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
```

Customizations

You can override the default Apache web server configuration for your virtual host in case you need a custom DocumentRoot or an additional 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"
}
```

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

System credentials

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

Username	Password
root	The password set in <i>Configuration file</i>
phpmyadmin	phpmyadmin

phpMyAdmin



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

Adminer

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

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

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 provisioning 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
```

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
```

Destruction backups

In case you destroy 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
```

PHP

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

Default settings

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

Setting	Value
<code>error_reporting</code>	<code>root</code>
<code>display_errors</code>	<code>phpmyadmin</code>
<code>date.timezone</code>	The timezone set in <i>Configuration file</i>
<code>xdebug.show_local_vars</code>	<code>1</code>
<code>xdebug.var_display_max_depth</code>	<code>5</code>
<code>xdebug.var_display_max_children</code>	<code>256</code>
<code>xdebug.var_display_max_data</code>	<code>1024</code>

phpinfo()

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

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
```

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!

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

server.cpus

The amount of CPU cores the machine should use.

Default value 1

server.memory

And the allocated memory in MB.

Default value 1024

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`

server.timezone

The timezone PHP and the OS should use. For a list of possible timezones, [click here](#).

Default value Europe/Berlin

mysql.root_password

The MySQL root password.

Default value root

mysql.version

MySQL server version to install and use.

Possible value 5.6 or 5.5

Default value 5.6

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