WebDevOps Documentation Documentation

Release

Florian Tatzel

Dec 05, 2017

General

This are the Documentation pages for the WebDevOps Vagrant Docker VM.
This Vagrant VM will provide you an easy solution for virtualized development with Docker.
Please keep in mind that this documentation is still work in progress.

CHAPTER 1

Introduction

1.1 What is the Vagrant Docker VM for?

The Vagrant Docker VM is an automated environment for Docker and development with a Ubuntu 16.04 base image, development tools and an optional gui.

Also it will provide a running Reverse proxy for HTTP development and a Consul service for container management.

CHAPTER 2

Getting Started

2.1 Requirements

- Vagrant
- VagrantManager (optional)
- Virtualization Software (VirtualBox, VMware or Parallels)

Type	Minimum (just working)	Recommendation (eg. for developers)
Host CPU	2 (eg. Intel Core i5)	4 physical Cores (or more, eg. Intel Core i7)
Host RAM	8 GB	16 GB (or more)
Host Disk	60 GB free	80 GB free
VM RAM	1.5 GB	1/4 System RAM

If you want to develop in a fast way make sure to get at least the recommended values.

This VM doesn't need 16 GB RAM, but you still need your browser, IDE, mail client and other tools.

2.2 Installation

2.2.1 Install Vagrant

Download Vagrant install it.

When using VMware you also need the Vagrant VMware plugin (license needed!), for Parallels the Vagrant Parallels plugin is needed.

Virtualization Software	Command	Notes
VirtualBox (all)	*included*	Slow to medium performance
VMware Workstation (Linux/Windows)	`vagrant plugin install vagrant-vmware-workstation`	License for VMware and plugin needed
VMware Fusion (MacOS)	`vagrant plugin install vagrant-vmware-fusion`	License for VMware and plugin needed
Parallels Desktop (MacOS)	`vagrant plugin install vagrant-parallels`	License for Parallels Desktop needed

2.2.2 Clone and create Vagrant Docker VM

Just clone the Vagrant Docker VM (or download as zip), customize the *vm.yml* and create the VM by using Vagrant:

```
# Clone git repository
git clone --recursive --config core.autocrlf=false https://github.com/webdevops/
↪vagrant-development.git devvm
cd devvm

# Customize the vm.yml with your favorite editor
vim vm.yml

# Customize the Vagrantfile with your favorite editor
vim Vagrantfile

# Setup Docker environment (only linux and mac, only once)
source provision/docker-init.sh

# Start vm
vagrant up

# Enter VM
vagrant ssh
```

Setup .ssh/config:

```
Host vm vagrant 192.168.56.2
    Hostname 192.168.56.2
    User     vagrant
    ForwardAgent  yes
    Compression   no
    StrictHostKeyChecking no
    UserKnownHostsFile=/dev/null
```

-> now you can jump direclty into the VM with `ssh vm`

Under Linux and MacOS you will be asked for root rights (sudo). If you don't want to enter your password every time take a look at the vagrant manual for NFS usage: https://docs.vagrantup.com/v2/synced-folders/nfs.html

2.3 Update

If there are any updates in this repository just run `vagrant up --provision` or `vagrant provision` to update your box with the new ansible changes.

2.4 Vagrant cheatsheet

Command	Description
`vagrant up`	Create VM or startup previous created VM
`vagrant halt`	Shutdown VM
`vagrant reload`	Restart VM
`vagrant destroy`	Kill and destroy the VM
`vagrant suspend`	Suspend VM
`vagrant resume`	Resume suspended VM

If you want a GUI tool for managing Vagrant VMs you can use VagrantManager. With it you can controll your VMs from a system tray icon.

CHAPTER 3

Usage

3.1 Customization

Put your custom files for your home directory */home/vagrant* into *./home*. Files for */etc* customization must be in *customization/etc*.

After provisioning all files will be synchronized to their destination folders.

3.2 Access to VM

You can get access to this VM with SSH or SMB/CIFS, see SERVICE section for more details.

3.3 Projects storage

This VM has two disks, the main OS disk and a bigger storage disk. The `/home/vagrant/projects/` directory is stored on the bigger storage disk. Here you should put your project files if you don't want to use the automatic NFS mounts (eg. for Windows users).

You can get access to `/home/vagrant/projects/` though the Samba share `projects`.

3.4 Mounts

The directory of the Vagrantfile is mounted under /vagrant (vagrant default).

Under Linux and MacOS your home directory is mounted under the same path as your host system. eg.

Host System: /Users/foo/ VM: /Users/foo/

Hint: This handling is needed if you want to use docker-compose from your host system.

3.4.1 Windows specific

If you're working under Windows you can put your files under `/home/vagrant/projects/`

You can access these Files via Samba.

1. Map a new Network Drive
2. Select Drive Letter
3. Enter: \\192.168.56.2 (Username: `vagrant` / Password: `vagrant`)
4. Browse and select the directory **projects**

3.5 Destroy & Recreate

3.5.1 VirtualBox

With `vagrant destroy` you will destroy the VM and also the data disk so you can immediately recreate the box with `vagrant up`.

3.5.2 VMware & Parallels

If you're using VMware or Parallels your disk will be stored inside the `disk/` directory.

After `vagrant destroy` you need to destroy these disk if you want a clean reinstallation:

VMware: Change to `disks/` directory and reset the VMware disk data files with git: `git checkout data*`

Parallels: Just remove the directory `disks/parallels-disk/`.

CHAPTER 4

Services

This section is not done yet!

4.1 Docker

Docker is running on default port `2376` and is accessable from outside without SSL/TLS.

Storage is configured to use AUFS but it's also possible to use BTRFS (see `vm.yml`)

4.1.1 Environment variables

```
export DOCKER_HOST=tcp://192.168.56.2:2375
export DOCKER_TLS_VERIFY=
```

4.2 HTTP reverse proxy (dinghy)

The default reverse proxy registers automatically any Docker container with `VIRTUAL_HOST` and `VIRUTAL_PORT` to it's configuration so multiple containers are accessable from outside without using other ports.

Also containers from docker-compose are registerd by this reverse proxy: `project_app_1` -> `app.project.docker`

For more informations visit: https://github.com/codekitchen/dinghy-http-proxy

Setting	Value
Hostname	*.docker
Port	80 and 443

4.3 Consul (service discovery)

Every container (with a port) is registred inside Consul with the Registrator service. This allows lookup of the IP adresses from containers by using following scheme:

```
containername-port.service.consul
```

With Consul you can eg. connect to a MySQL database with a GUI Tool using an SSH tunnel and this address syntax without exposing ports to the VM.

4.3.1 DNS lookup examples

Container name	Port	DNS name
consul	8500	consul-8500.service.consul
consul	8600	consul-8600.service.consul
dory-http-proxy	443	dory-http-proxy-443.service.consul
typo3dockerboilerplate_app_1	80	*no supported* (underscores not allowed in domain names)
typo3dockerboilerplate_mysql_1	3306	*no supported* (underscores not allowed in domain names)

Note: docker-compose containers are currently not supported, see https://github.com/docker/compose/issues/229

4.4 SSH

Setting	Value
Server	IP or Hostname of VM (192.168.56.2)
Port	22
Username	vagrant
Password	vagrant
SSH Key	Automatically deployed from github, if account name is set (see vm.yml)

```
# connect via vagrant
vagrant ssh

# normal way
ssh vagrant@192.168.56.2
```

4.5 Samba (SMB/CIFS)

Setting	Value
Server	IP or Hostname of VM (192.168.56.2)
Username	vagrant
Password	vagrant
Share /vagrant	/home/vagrant
Share /projects	/mnt/data/projects/
Share /tmp	/tmp
Explorer URL	\\192.168.56.2code

MacOS and Linux don't need Samba, Vagrant will use shared folders.

For MacOS the /Users directory will be mounted under /Users in Vagrant VM to enable transparent external docker access.

4.6 Mail sandbox

Setting	Value
IMAP Server	IP or Hostname of VM (192.168.56.2)
IMAP Port	143 (without SSL/TLS)
SMTP Server	IP or Hostname of VM (192.168.56.2)
SMTP Port	25 (without SSL/TLS)
Username	vagrant
Password	vagrant

Any outgoing email is catched by postfix and send to mailbox of vagrant user.

4.7 File Sharing

4.7.1 Sharing types

Share type	Windows	Linux/macOS	Notes
NFS	*no*	**yes**	Fastest and reliable
CIFS	**yes**	**yes**	Fast and reliable
SMB	**yes**	**yes**	old Protocol, use CIFS
none	**yes**	**yes**	Slow on VirtualBox

4.7.2 Linux/MacOS

By default your home directory is mounted into the VM so files from outside are available inside the Vagrant Docker VM. You can customize mounting inside the `vm.yml`.

For sharing NFS is used and is a stable and fast solution for sharing data between the VM and your host system. When encounting slowdown in VirtualBox you should try to switch to VMware or Parallels.

4.7.3 Windows

On windows no directory is shared by default and sharing needs to be defined in `vm.yml`.

As alternative there is a `/home/vagrant/projects` directory which can be access by using SMB/CIFS (windows shares) from your explorer: `\\192.168.56.2`

CHAPTER 5

Troubleshooting

5.1 Startup or update errors

5.1.1 Reprovision the VM

You can safely reprovision your VM without losing data:

```
# if box is NOT started
vagrant up --provision

# if box is already started
vagrant provision
```

5.1.2 Windows 10 (Tech Preview) and VMWare

It can happen that the **Virtual Network Adapters** (in this case host only adapter) break **on every shutdown** or **disconnect** from Network.

Windows 10 has **currently issues with Virtual Networks** on different VM Softwares. With VMWare a Workaround can be resetting the Network Adapters right before starting the VM.

1. Shut Down the VM (if not down already)
2. Open VMWare
3. Open in Menu: **Edit > Virtual Network Editor...**
4. Request Admin Rights in this window (if not already started with them)
5. Click: **Restore Defaults**

This will reset all virtual Network Adapters. If you now start Vagrant with **vagrant up**, the required Adapters will be recreated.

5.2 General errors

5.2.1 Disk usage is high or disk is full

You can safely run `docker-clean` to remove old and unused images.

5.3 Networking issues

5.3.1 No IP address or no network

Remove the udev rule for network interfaces:

sudo rm /etc/udev/rules.d/70-persistent-net.rules

5.4 Docker errors

5.4.1 Error response from daemon: client and server don't have same version (client : 1.19, server: 1.18)

Your docker client has been updated, just rerun provisioning to update the Docker server.

CHAPTER 6

Project Structure

This section is not done yet!

6.1 The Vagrantfile Configuration

6.1.1 Vagrantfile

The `Vagrantfile` is the entrypoint for Vagrant and setup to use most configuration values from `vm.yml`.

There is a CUSTOMIZATION section in the top which get called after the main VM configuration if you need further configuration.

6.1.2 vm.yml

Most values are documentation in the `vm.yml`.

6.2 Provisioning

The main provisioning of the VM directly after creation is done by Ansible. With Ansible complex tasks are configured in simple yml files so the provisioing is not too complex even for new users.

Script	Description
provision/bootstrap.sh	Run for new VMs or when *vagrant provision* or *vagrant up –provison* is called
provision/maintenance.sh	Run every time the VM is started (startup tasks)