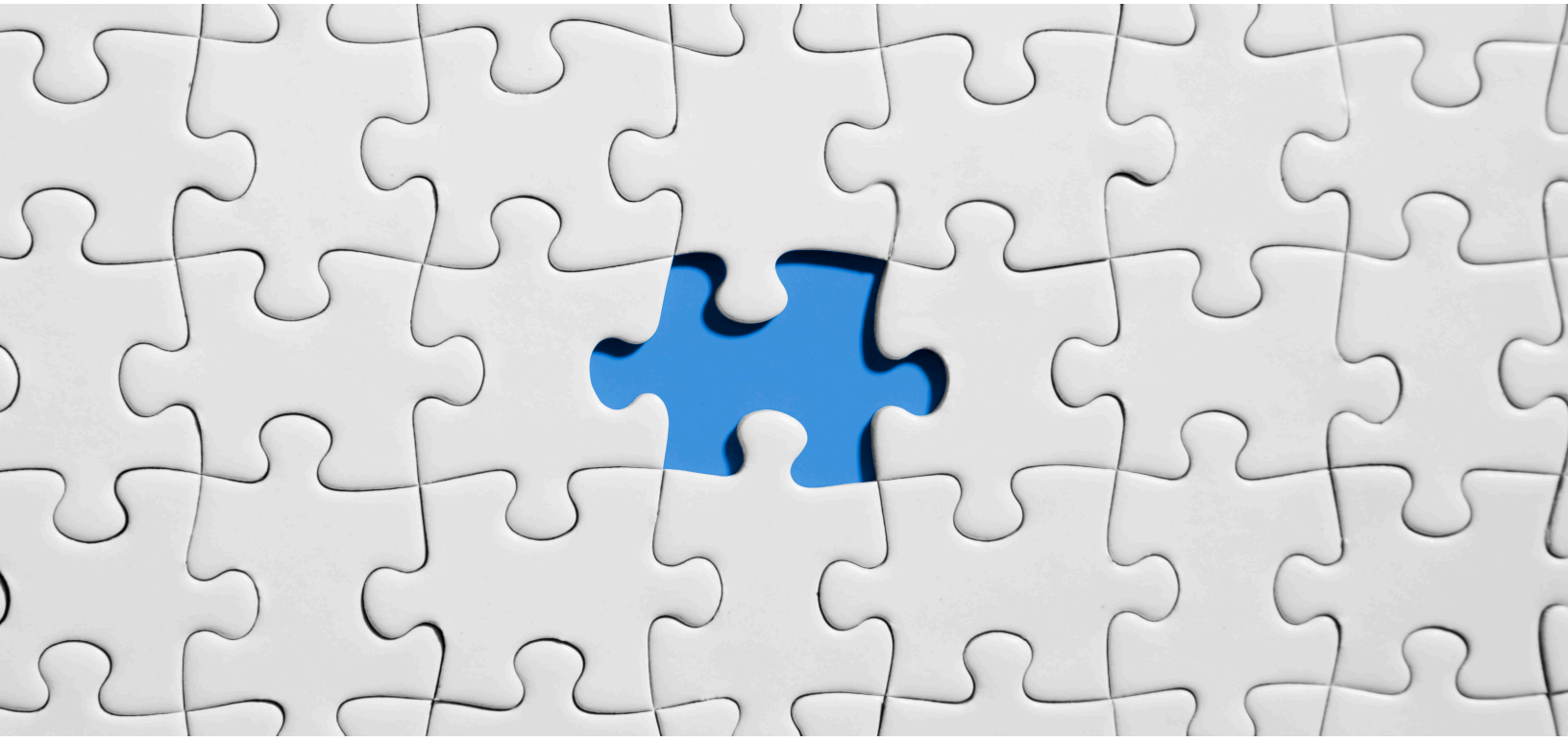
DELL Technologies

MOVING TOWARDS DEVOPS WITH VAGRANT



Malathy Ramani

Consultant, IT Infrastructure

Dell Technologies

Malathybr@yahoo.com

Malathy.ramani@dell.com

Knowledge Sharing Article

DELL Technologies Proven Professional

Table of Contents

Preface

Have you ever wondered what it would be like not to set up your environment manually? What if we can deploy it at one go without going through the intricacies of working out the dependencies? This is precisely what Vagrant does.

Vagrant is a smart piece of software written in Ruby by HashiCorp Inc. which lets you create a virtualized environment quickly. Vagrant provides the ability to create portable and reproducible environments on top of virtual machine providers like Oracle VirtualBox, VMware workstation, AWS and others. Vagrant functions as a virtual machine manager enabling custom configurations on virtual machines. This is achieved by defining the configuration of the virtual machines in the "Vagrantfile" and getting the configuration up and running using "vagrant up". The main benefit is that all the members of a project can get a uniform configuration without any variation and independent of if they are using Mac, Linux, or Windows. Vagrant is a powerful tool which can help in an organization's transition to DevOps and Infrastructure as code solutions.

In this article, we will discuss the process of setting up a simple Vagrant environment on top of Oracle VirtualBox on our laptop. We will also set up Ansible, a provisioning tool on top of a Vagrant host. Ansible is an automation tool for IT delivery and management. No license is needed for Vagrant as it is free.

While it is easy to create the Vagrant environment, deleting it is equally easy. The ability to stand up a full stack locally in a matter of minutes and then destroy it equally fast when work is complete is very exciting and useful, indeed.

The need for DevOps and why it came into existence

DevOps seeks to promote better collaboration between development and operations teams and deliver software in a shorter lifecycle. The goal of DevOps is to reduce software release cycles while delivering new features, fixes for bugs, and new updates regularly, as well as automating processes from development to deployment.

Traditionally, developers and IT professionals have functioned in their own domains with separate responsibilities, objectives, and processes. Many challenges exist in such an environment, for example, development servers on which code was developed may have a different configuration than the production servers and consequently, code at the time of deployment may run into issues. The handshake between development and deployment needs to be finely tuned for a seamless launch of the code from development to deployment. Many of these problems could be circumvented by the adoption of DevOps.

What are the factors which will help in the adoption of DevOps in organizations?

Cultural Change

DevOps culture requires developers and IT professionals to think together as a single team instead of in silos in any code release. This is based on developing a climate of collaboration and trust among them.

Work on Resolution time

Keep the communications open and flowing between the teams, work on getting feedback and get it fast. Customer satisfaction rests completely on getting issues resolved quickly and effectively.

Automate

Automating repeatable tasks and processes eliminates human errors, and is faster and reliable. Continuous delivery can be achieved by developing code, running through automated tests, and build and deploy to production.

Automated deployment will ensure that the production is identical to the development environment, to avoid scenarios such as when code works in development but fails in production.

Infrastructure which is reliable and resilient can be built by considering configuration as code. If a server encounters corruption, it becomes easier to reinstate the server with the same configuration by using code for configuration.

Automate Testing

Test-driven development with a focus on automated test is crucial to faster code releases. Organizations need to standardize on tools and processes for doing testing and other processes so that time is not spent in the handshake process between different disparate tools and processes.

Agile development

Focusing on eliminating activities which are low in value and stressing on getting the code to production or market faster will bring agility. Being agile sometimes could mean the code is not perfect and will be going through continuous improvement and the team should be agile to recover from any setback, learn from it and improve it.

DevOps with Vagrant

Vagrant is a tool from Hashicorp Inc. for managing virtual machines in an easy workflow. It is a wrapper on top of virtualization solutions like Oracle VirtualBox, VMware workstation, etc. and helps in setting up configurations on virtual machines quickly. Vagrant can set up the same configurations on any machine by using a simple text file called “Vagrantfile”. This enables the environments to be ported and reproduced at will.

Vagrant helps in the development of a strong DevOps culture in organizations because developers can set up their environments on their own and work more closely with operations. Vagrant provides a proven way to easily deploy development environments in minutes. Developers can spin up and destroy their sandboxes as and when they want, and do not need to rely on system admins. By creating their own environments, developers can better understand the challenges faced by operations and their processes. The development environments created with Vagrant can be exactly reproduced in production environments without any discrepancy in the configurations between them, making “works on my machine” excuses a thing of the past.

From the DevOps point of view, Vagrant gives a consistent workflow to develop and test infrastructure management scripts. After testing it locally on your system, the same configuration can be ported to remote public clouds like AWS or Rackspace with the same workflow.

Advantages

Vagrant’s Cross-Platform feature works on MAC, Linux, Windows, and other operating systems. Vagrant allows us to code using our favorite editor, and debug using our tool of choice, all from the local laptop. There is no need to give up one’s favorite tools when working with Vagrant.

Vagrant uses a simple and powerful declarative configuration file called vagrantfile which describes all the software, OS type, configurations, users, and other details.

When using Vagrant, there is no need to use the individual CLI of virtualization providers like VirtualBox, Hyper-V, VMware fusion, and VMware workstation. Vagrant uses many of these utilities internally and it is enough to know and use Vagrant CLI.

Compatible across different versions of the virtualization software, Vagrant will automatically detect the version of VirtualBox being used and use the flags appropriately.

Vagrant Terminology

Providers

Providers enable Vagrant software to work with various virtualization solutions like Oracle VirtualBox, Hyper-V, and VMware.

Provisioners

Provisioners enable Vagrant to get systems into desired state through configuration management tools like Ansible, Chef, and Puppet or simple shell scripts.

Vagrant Box

Vagrant boxes are vagrant packages that can be full development environments or just plain operating system images. These packages can be imported on any machine that runs Vagrant.

Shared Folders

Vagrant allows sharing of folders between the host machine and guest. This means that we can edit files on the local host and see the changes on the guest machine.

Port forwarding

Virtualization software such as Oracle VirtualBox run virtual machines in NAT network mode; thus, the virtual machine has its own virtual address which cannot accessed from the host. Port forwarding creates rules to forward traffic from local host port to virtual machine port.

Vagrant Plugins

Vagrant Plugins allow support for adding new provisioners.

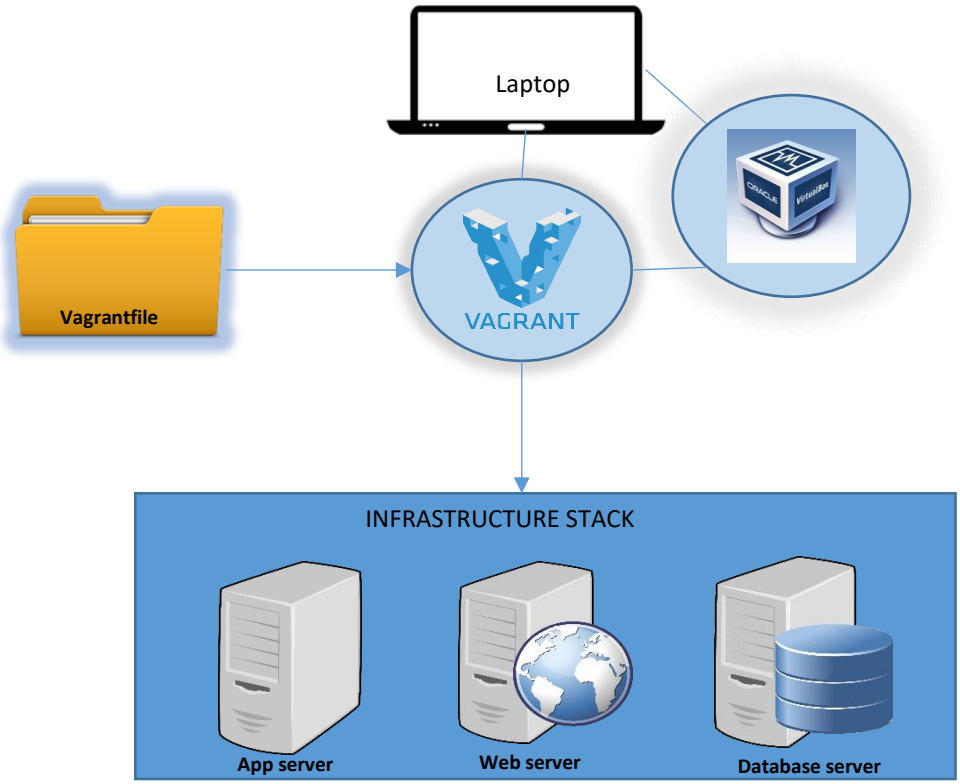
An Infrastructure stack in miniature with Vagrant

Let's go through the process of building a simple infrastructure stack using Vagrant on a local laptop.

The software and versions used are:

Software	Version
Laptop O/S	Windows 10 64-bit
Virtualization solution	Oracle VirtualBox version 5.2.20r125813
Vagrant	Vagrant 2.2.0
Ansible	Ansible 2.7.0
Server Type	Operating System
Application Server	Red Hat Enterprise Linux Server release 7.5 (Maipo)
Web Server	CentOS Linux release 7.5.1804 (Core)
Database Server	CentOS Linux release 7.5.1804 (Core)

Architecture



Setting up Vagrant/Oracle Virtual box

1. Install Vagrant from https://www.vagrantup.com/downloads.html
   Choose the defaults and install. Reboot.
   Go to cmd on Windows laptop and check vagrant.

```
C:\>vagrant
WARNING: This command has been deprecated in favor of `vagrant cloud auth
login`
Usage: vagrant [options] <command> [<args>]

    -v, --version                    Print the version and exit.
    -h, --help                       Print this help.
```

2. Install Oracle Virtual Box from https://www.virtualbox.org/
   Choose the defaults and install.
   Go to cmd and check vboxmanage.

```
C:\>vboxmanage
Oracle VM VirtualBox Command Line Management Interface Version 5.2.20
(C) 2005-2018 Oracle Corporation
All rights reserved.

Usage:

  VBoxManage [<general option>] <command>
```

Set the Path variable to include c:\program files\oracle\virtual box.

Create a directory in which you want to create the Vagrant environment and change to it.
Vagrant init creates the Vagrantfile in the current directory in which it is run. The vagrantfile contains the instructions of how we want to set up the environment and we can modify it according to our needs.

```
C:\MyVagrantProject>vagrant init
A `Vagrantfile` has been placed in this directory. You are now
ready to `vagrant up` your first virtual environment! Please read
the comments in the Vagrantfile as well as documentation on
`vagrantup.com` for more information on using Vagrant.
```

Make edits to the vagrantfile:
Remove anything other than –

```
# -*- mode: ruby -*-
# vi: set ft=ruby :

# All Vagrant configuration is done below. The "2" in Vagrant.configure
# configures the configuration version (we support older styles for
# backwards compatibility). Please don't change it unless you know what
# you're doing.
Vagrant.configure("2") do |config|

End
```

Before adding anything more to the Vagrantfile, note down the names of the OS images from vagrantcloud.com. They will be needed for editing the Vagrantfile.

Modify the vagrantfile as follows:

```
# -*- mode: ruby -*-

# vi: set ft=ruby :


# All Vagrant configuration is done below. The "2" in Vagrant.configure

# configures the configuration version (we support older styles for

# backwards compatibility). Please don't change it unless you know what

# you're doing.

Vagrant.configure("2") do |config|

config.vm.define "app" do |app|

app.vm.box = "generic/oracle7"

app.vm.hostname = "app"

app.vm.network "private_network", ip: "192.168.10.4"

end


config.vm.define "web" do |web|

web.vm.box = "centos/7"

web.vm.hostname = "web"
```

```
    web.vm.network "private_network", ip: "192.168.10.5"
    web.vm.network "forwarded_port", guest: 80, host:8080
    end

    config.vm.define "db" do |db|
    db.vm.box = "centos/7"
    db.vm.hostname = "db"
    db.vm.network "private_network", ip: "192.168.10.6"
    end

end
```

We are running Vagrant version 2.2. In configure("2"), the “2” in brackets refers to the Vagrant version. The outer block creates the object “config” which holds all the settings. The inner do loop defines the VM, its name and IP. The web server also has a port forwarding set up. The forwarded port configuration expects two parameters, the port on the guest and the port on the host. This will allow accessing port 80 on the guest via port 8080 on the host.

Run the command “vagrant up” to pull the images and set up the configuration, and also to boot up the machines:

```
C:\MyVagrantProject>vagrant up
Bringing machine 'app' up with 'virtualbox' provider...
Bringing machine 'web' up with 'virtualbox' provider...
Bringing machine 'db' up with 'virtualbox' provider...
```

Run “vagrant ssh” to ssh into each of the servers and set them up.

APP Server/ Ansible Setup

Vagrant ssh to the App Server.

```
C:\MyVagrantProject>vagrant ssh app
Last login: Mon Jan 14 12:27:54 2019 from gateway
[vagrant@app ~]$
```

Before installing Ansible, the appropriate EPEL release package needs to be installed.

```
[vagrant@app ~]$ sudo yum install https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm
```

Enable the Ansible engine repository. Then, install Ansible.

```
[vagrant@app ~]$ sudo yum subscription-manager repos --enable "rhel-*-optional-rpms" --enable "rhel-*-extras-rpms"

[vagrant@app ~]$ sudo yum update
[vagrant@app ~]$ sudo yum install ansible
[vagrant@app ~]$ ansible --version
ansible 2.7.0
  config file = /etc/ansible/ansible.cfg
  configured module search path = [u'/home/vagrant/.ansible/plugins/modules', u'/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python2.7/site-packages/ansible
  executable location = /usr/bin/ansible
  python version = 2.7.5 (default, Jul  3 2018, 06:28:28) [GCC 4.8.5 20150623 (Red Hat 4.8.5-28.0.1)]
```

Setting up Web Server

Vagrant ssh into the server.

Before installing Apache, update the package repository. After install of Apache, start the httpd daemon and check the status.

```
C:\MyVagrantProject>vagrant ssh web
Last login: Mon Jan 14 20:25:05 2019 from 10.0.2.2
[vagrant@web ~]$
[vagrant@web ~]$ sudo yum update
[vagrant@web ~]$ sudo yum install httpd
[vagrant@web ~]$ sudo systemctl start httpd
[vagrant@web ~]$ service httpd status
Redirecting to /bin/systemctl status httpd.service
● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; vendor preset: disabled)
   Active: active (running) since Mon 2019-01-07 18:52:55 UTC; 55s ago
     Docs: man:httpd(8)
           man:apachectl(8)
```

```
Main PID: 3943 (httpd)
  Status: "Total requests: 0; Current requests/sec: 0; Current traffic:   0 B/sec"
  CGroup: /system.slice/httpd.service
          ├─3943 /usr/sbin/httpd -DFOREGROUND
          ├─3944 /usr/sbin/httpd -DFOREGROUND
          ├─3945 /usr/sbin/httpd -DFOREGROUND
          ├─3946 /usr/sbin/httpd -DFOREGROUND
          ├─3947 /usr/sbin/httpd -DFOREGROUND
          ├─3948 /usr/sbin/httpd -DFOREGROUND
          └─4023 /usr/sbin/httpd –DFOREGROUND
```

Testing the Web Server

Create a test page by modifying /var/www/html/index.html.

Change to superuser before modifying the file; the default password is “vagrant”

[vagrant@web html]$ su -

Password:

[root@web ~]# cd /var/www/html

Restart httpd service.

```
<html>

<head>
<title> favorites / bookmark title goes here </title>
</head>

<body bgcolor="yellow" text="blue">

<h1> This is hosted on a web server on Vagrant guest </h1>


</body>

</html>
```

The webserver can be tested on both the ports 80 using web VM IP and 8080 using localhost IP. Both should work due to port forwarding handled by Vagrant.

Webserver using port 80 of Guest Virtual Machine

Webserver using port 8080 of Host Machine

Port 80 on the guest is accessed via port 8080 on the host due to Port forwarding.

Setting up Database Server

Vagrant ssh into the db server.

C:\ MyVagrantProject >vagrant ssh db

We will be doing a base install of MariaDB on Centos Linux on the vagrant guest db.

1. Install MariaDB

   Install MariaDB using Centos package manager.

   [vagrant@db ~]$ sudo yum install mariadb-server

2. Start the MariaDB service and enable chkconfig utility to ensure that the database server launches after a reboot.

   [vagrant@db ~]$ sudo systemctl start mariadb.service
   [vagrant@db ~]$ sudo systemctl enable mariadb.service
   Created symlink from /etc/systemd/system/multi-user.target.wants/mariadb.service to /usr/lib/systemd/system/mariadb.service.
   [vagrant@db ~]$ /usr/bin/mysql -u root -p
   Enter password:
   Welcome to the MariaDB monitor. Commands end with ; or \g.
   Your MariaDB connection id is 3
   Server version: 5.5.60-MariaDB MariaDB Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

```
[vagrant@db ~]$ sudo systemctl start mariadb.service
[vagrant@db ~]$ sudo systemctl enable mariadb.service
Created symlink from /etc/systemd/system/multi-user.target.wants/mariadb.service to /usr/lib/systemd/system/mariadb.service.
```

3. Start the MariaDB shell and use a select statement on the database mysql.

```
[vagrant@db ~]$ /usr/bin/mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 3
Server version: 5.5.60-MariaDB MariaDB Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
```

```
MariaDB [(none)]> SELECT User, Host, Password FROM mysql.user;
+------+-----------+----------+
| User | Host      | Password |
+------+-----------+----------+
| root | localhost |          |
| root | db        |          |
| root | 127.0.0.1 |          |
| root | ::1       |          |
|      | localhost |          |
|      | db        |          |
+------+-----------+----------+
6 rows in set (0.00 sec)
```

4. Create a test database and verify that it is created.

```
MariaDB [(none)]> CREATE DATABASE TestDB;
Query OK, 1 row affected (0.00 sec)

MariaDB [(none)]> Show Databases
    -> ;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| TestDB             |
| mysql              |
| performance_schema |
| test               |
+--------------------+
5 rows in set (0.00 sec)
```

```
MariaDB [(none)]> quit
Bye
```

Deleting the environment

The environment can be deleted quickly when no longer needed. It just takes a single command, “vagrant destroy”.

```
C:\MyVagrantProject>vagrant destroy
    app: Are you sure you want to destroy the 'app' VM? [y/N] y
==> app: Forcing shutdown of VM...
==> app: Destroying VM and associated drives...
```

Conclusion

Vagrant provides a great tool for developers to set up the environment quickly and easily and then replicate it to multiple machines through creating Vagrant box. The advantage is that the same setup can be used by a whole team and the team can have uniform configuration across the board. One can also start with a configured box in the beginning from Vagrant cloud instead of setting it up. A word of caution though. Since each virtual machine added to the local laptop will consume resources, host system resources should be sufficient and should be planned.

Appendix

1. https://docs.ansible.com/ansible/latest/installation_guide/intro_installation.html
2. https://www.vagrantup.com/
3. https://www.atlassian.com/