Introduction

to

VAGRANT

NREN

Feb 10 - 12, 2023

Introduction

Vagrant is a tool for building and managing virtual machine environments in a single workflow.

- an easy-to-use workflow and focus on automation
- lowers development environment setup time
- increases production parity
- makes the "works on my machine" excuse a relic of the past

What is Vagrant?

- Vagrant is the command line utility for managing the lifecycle of virtual machines.
- A tool to build development environments based on virtual machines
- Focused to create environments that are similar as possible or identical with production servers
- Created by Mitchell Hashimoto
- Written in Ruby
- Initially builed on top of VirtualBox API, today offers VMware, Hyper-V and Docker support

Why Vagrant?

- provides easy to configure, reproducible, and portable work environments
- built on top of industry-standard technology and controlled by a single consistent workflow to help maximize the productivity and flexibility of you and your team.

Vagrant Terms

- Boxes
    - Boxes are the package format for Vagrant environments. A box can be used by anyone on any platform that Vagrant supports to bring up an identical working environment.
- providers
    - While Vagrant ships out of the box with support for VirtualBox, Hyper-V, and Docker, Vagrant has the ability to manage other types of machines as well. This is done by using other providers with Vagrant.

Vagrant Terms (contd...)

- provisioning
    - Provisioners in Vagrant allow you to automatically install software, alter configurations, and more on the machine as part of the vagrant up process.
- networking
    - In order to access the Vagrant environment created, Vagrant exposes some high-level networking options for things such as forwarded ports, connecting to a public network, or creating a private network.

Vagrant Terms (contd...)

- synced folders
  - Synced folders enable Vagrant to sync a folder on the host machine to the guest machine, allowing you to continue working on your project's files on your host machine, but use the resources in the guest machine to compile or run your project.
- plugins
  - Vagrant comes with many great features out of the box to get your environments up and running. Sometimes, however, you want to change the way Vagrant does something or add additional functionality to Vagrant. This can be done via Vagrant plugins.

Vagrant Terms (contd...)

- triggers
    - Vagrant is capable of executing machine triggers before or after Vagrant commands.

What is a Vagrant Box?

- Is a previously builed Vagrant virtual machine image, ready-to-running
- Available in lot of platforms (Linux, Windows, BSD)
- You can create one! :)

How to add a box?

- Great box repository:
    - https://vagrantcloud.com
    - https://www.vagrantbox.es/
- Run this command:

```
$ vagrant box add <name> <url> --provider <provider>

eg:
$ vagrant box add bento/ubuntu-18.10 \
  https://vagrantcloud.com/bento/ubuntu-18.10 --provider virtualbox

or:
$ vagrant box add bento/ubuntu-18.10  --provider virtualbox

or:
$ vagrant box add bento/ubuntu-18.10 #(complete using screen options)
```

How to create a environment?

- Inside your project, create a Vagrantfile:

```
$ vagrant init <your box name>

eg:
$ vagrant init bento/ubuntu-18.10
```

How to create a environment? (contd...)

```
# 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|
  # The most common configuration options are documented and commented below.
  # For a complete reference, please see the online documentation at
  # https://docs.vagrantup.com.
  # Every Vagrant development environment requires a box. You can search for
  # boxes at https://vagrantcloud.com/search.
  config.vm.box = "bento/ubuntu-18.10"
  # ...
  # A list of options Here
  # ...
end
```

How to start using it?

- Simply run this command:

```
$ vagrant up
```

How to connect to vagrant VM?

- Easy:

```
$ vagrant ssh
```

How to stop it?

- Easy:

```
$ vagrant halt
```

How to restart it?

- Easy:

```
$ vagrant reload
```

How to customize box?

- You can change memory, CPU and other things in Vagrantfile
- Just see VBoxManage options
- Example:

```
Vagrant.configure("2") do |config|
  config.vm.box = "bento/ubuntu-18.10"
  config.vm.provider "virtualbox" do |vb|
    # Customize the amount of memory on the VM:
    vb.memory = "1024"
    # vb.customize [ 'modifyvm', :id, '--memory', '1024' ] # same as above
    vb.cpu = "2"
    # vb.customize [ 'modifyvm', :id, '--cpus', '4' ] # same as above
    vb.customize ["modifyvm", :id, "--cpuexecutioncap", "50"]
    vb.name = "Ansible-Controller"
  end
end
```

That's it?

- of course, no! :)
- it is time to configure environment using available provisioners to install required software:
    - Shell
    - Ansible
    - CFEngine
    - Chef
    - Puppet
    - Salt

Using Shell

- Create a single bash script that install all you need:

```
#!/bin/bash

apt-get update

# base
apt-get install --yes Python

# others
apt-get install --yes curl tmux screen wget

#(...)
```

Installing software

- Easy:

```
$ vagrant provision
```

Creating a custom box

- You can create custom boxes to distribute between development teams
- Requires a fresh installation of a virtual machine based on Vagrant conventions and some manual configuration
- Awesome advantage: can repackage a existing Vagrant package after updating a existing VM
- Next steps are based on Debian distro as VM with VirtualBox as provider

Creating a custom box (contd...)

- Installation steps:
    - set root password: *vagrant*
    - create a suser with username and password: *vagrant*
    - machine name: vagrant-debian-squeeze
    - machine host: vagrantup.com

Creating a custom box (contd...)

- Post-installation steps:
    - Install sudo on virtual machine
    - Add a group permission with visudo:

    ```
    %admin ALL=NOPASSWD:ALL
    ```

    - Download SSH insecure pair files:
        - https://github.com/hashicorp/vagrant/tree/master/keys
        - Save public key on GUEST in ~/.ssh/authorized_keys and all keys in HOST
    - Or generate a custom pair of SSH keys and distribute it

Creating a custom box (contd...)

- Post-installation steps:
  - Install VirtualBox Guest Additions with Ctrl-D or cmd-D
  - Remove pre-installed packages:

```
# apt-get remove --purge virtualbox-ose-*
```

Creating a custom box (contd...)

- Post-installation steps:
  - VirtualBox needs xorg drivers, kernel headers and gcc to correctly build Guest Additions kernel module
  - run:

```
# apt-get install linux-headers-$(uname -r) build-essentials xorg
```

Creating a custom box (contd...)

- Post-installation steps:
  - Run VirtualBox Guest Additions installer

```
# mount /media/cdrom
# sh /media/cdrom/VBoxLinuxAdditions.run
```

Creating a custom box (contd...)

- Post-installation steps:
  - After all steps, shutdown your VM
  - Execute in host:

```
$ vagrant package <vm-name> --base <package-name> --output <box-file>
$ vagrant box add <package-name> <box-file> --provider virtualbox
```

Creating a custom box (contd...)

- If you do not wat to build step-by-step, try veewee
  - https://github.com/jedi4ever/veewee
  - Supports VMware Fusion, VirtualBox and KVM
  - Enable boxing based on an ISO file
  - Run as a Vagrant Plugin
- Packer by HasiCorp
  - https://www.packer.io/

Performance Tips

- Slow I/O on Guest
  - enable Host I/O cache on SATA Controller
- Slow with CPU-bound tasks
  - set motherbord chipset to ICH9

More info and documentation

- Vagrant Homepage
  - https://www.vagrantup.com/
- Documentation
  - https://developer.hashicorp.com/vagrant/docs