VAGRANT

Virtualisation with Vagrant

By Noorvir Aulakh

James Lambert

Thomas Rickard

Learning Outcomes

- Understand different types of virtual machines
- Be able to run, provision and stop a vagrant virtual machine.

Virtual Machine Recap

- Software implementation of a machine that executes as if it *were* physical machine
- Emulates a particular computer system
- Two main types:
  - System virtual machines
  - Process virtual machines

System Virtual machines

A system virtual machine allows the execution of a complete operating system

- Multiple virtual machines can co-exist on the same primary hard drive.
- Can provide emulated hardware environments, different from the host instruction set.
- Less efficient that actual machine.

Process Virtual Machines

Process virtual machines are designed to run a single program and therefore support a single process.

- Platform independent programming environment
- A common example is the Java Virtual Machine
- Another example is the .NET framework which runs on Common Language Runtime

Virtualisation - Hypervisors

- Can use type 1 or type 2 hypervisor
- Type 1
  - Runs directly on the hardware
- Type 2
  - Runs on top of the operating system

Virtualisation - Raw Hardware

- Also known as native or embedded.
- Provides full virtualisation
  - Multiple different systems can be run
  - Runs directly on the hardware
- Some common hypervisors:
  - Xen, KVM, Vmware, Virtualbox

Virtualisation – Operating System Level

- Takes place on the operating system (kernel) layer
- Slices a single server in multiple smaller partitions called Vi*rtual Environments (VEs)*
- Has very little overhead
- Limited to same kernel
- Can run much a much higher density of VEs than fully virtual hardware
- Docker is an example of this type of virtualisation

Vagrant

- Software for easily creating and configuring virtual environments
- Wrapper around virtualisation software (providers)
  - Virtualbox, Vmware
- Wraps around configuration management software (provisioners)
  - Ansible, Puppet, Chef, salt

Vagrant

Today we will be using Vagrant commands and puppet for setting up or virtual environment.

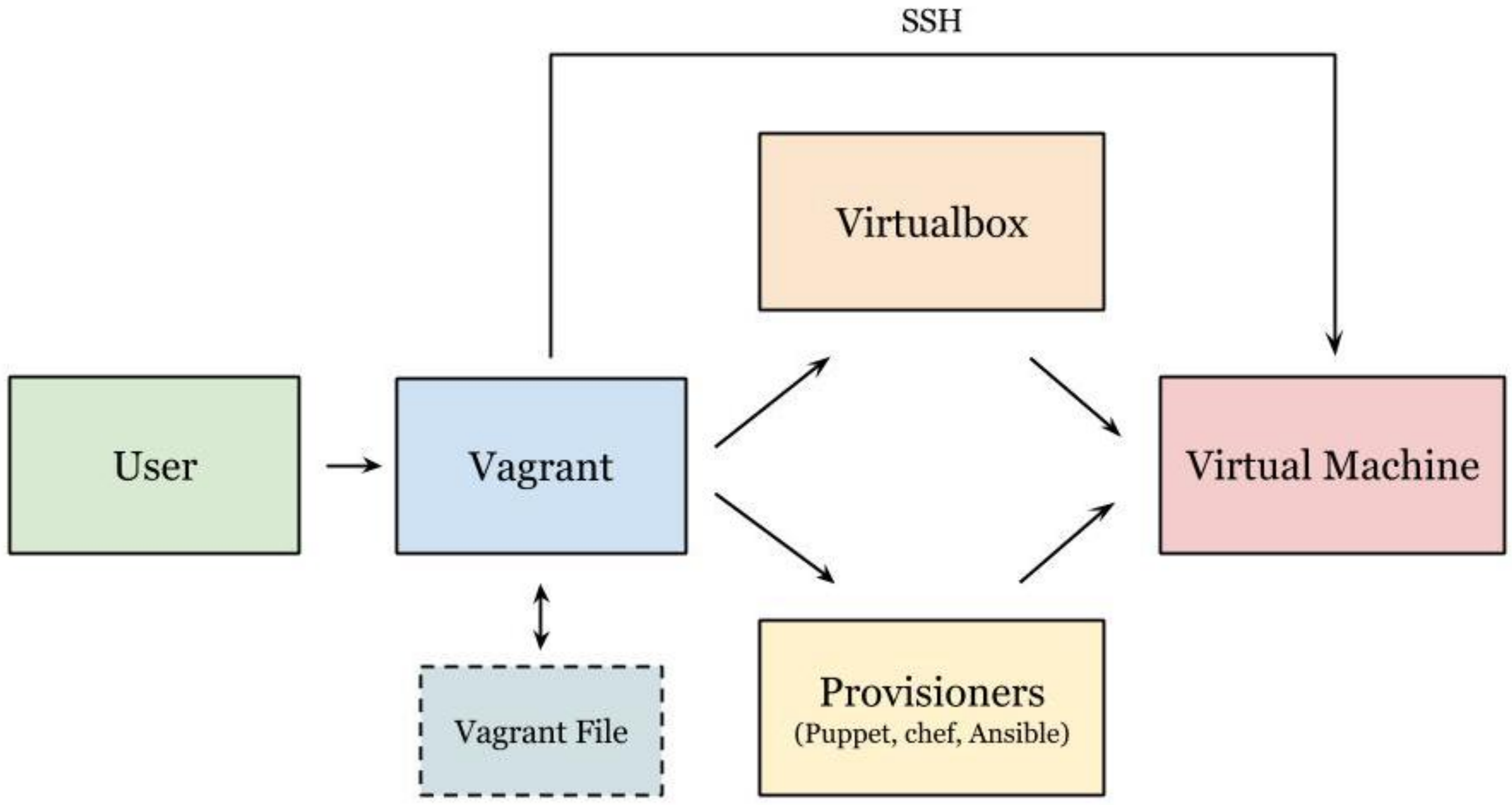
Vagrant is configured using the Vagrantfile

- Stored in plain text
- Located in Vagrant current directory
- There must only be **one** Vagrantfile in the Vagrant current directory.

Vagrant – why?

- Allows the set up multiple virtual machines with ease
- Highly portable
- Can use source control on setup files
- Can try a large number of various platforms quickly

Workflow


SSH
User
Vagrant
Vagrant File
Virtualbox
Provisioners
(Puppet, chef, Ansible)
Virtual Machine

Puppet

- A configuration management tool for Unix-like and Windows systems
- Configuration is placed in a **manifest** file
- Uses puppet's declarative language
- Configuration is converted into resources and dependencies used to install software

Puppet – why?

- Makes it easy to install and setup software in an automated way
- Can be included in the Vagrantfile

Vagrant Cheat Sheet

Initialising

`vagrant **init** *optional_box_address*`

Boxes

vagrant **add** *box*	Add a specified box
vagrant **package**	Saves modified box

General

vagrant **status**	Vagrant machine state
vagrant **global-status**	State of all active vagrant environments

Running, SSH and Teardown

vagrant **up**	Starts VM
vagrant **ssh**	Opens SSH connection
vagrant **suspend**	Saves current running state
vagrant **halt**	Shuts down VM
vagrant **destroy**	Removes all traces of VM

Vagrant File Basics

(can be done using the command line interface)

`Vagrant.configure("2") do |config|`	
`config.vm.box = "hashicorp/precise32"`	Adds the box
`config.vm.provision :shell, path: "bootstrap.sh"`	Provisioning using shell
`config.vm.network :forwarded_port, host: 4567, guest: 80`	Networking
`end`	