Virtualization using VirtualBox and Vagrant

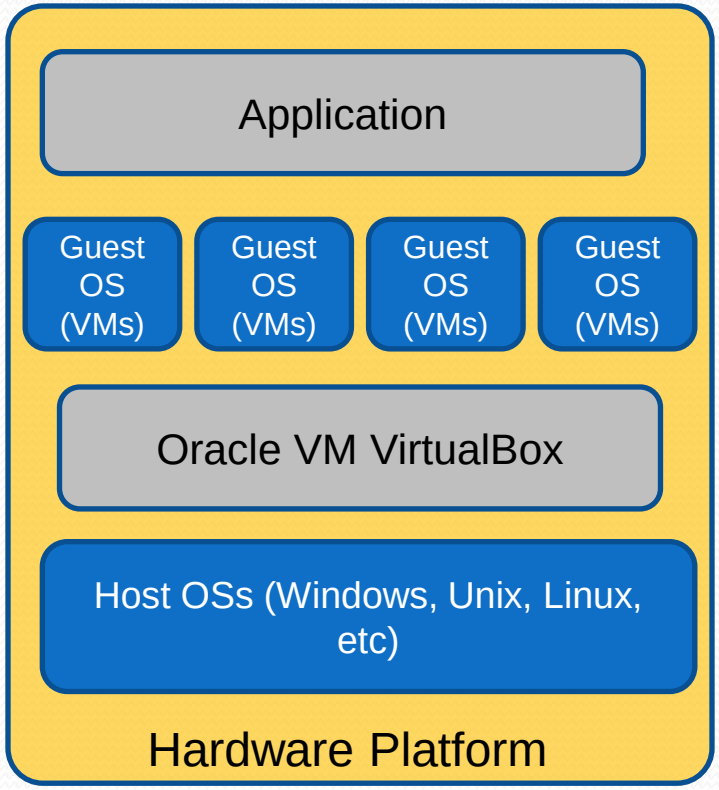
By, Ganesh Palnitkar

Agenda

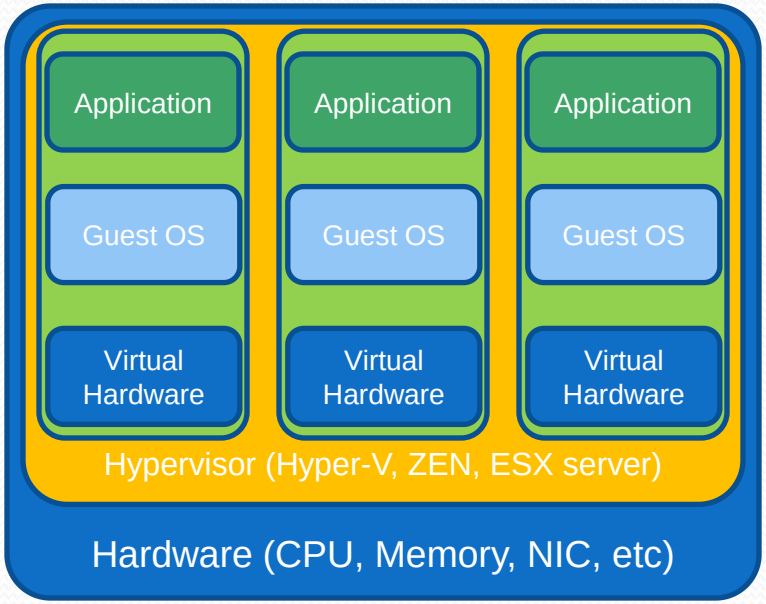
- What is Virtualization
- How to automate virtualization
- Vagrant as Automation tool
- Using ‘Vagrantfile’
- Vagrant commands

What is Virtualization?

Virtualization refers to the technology in which virtual machines are created on a platform that enables sharing the hardware resources. This way the virtual machine can be build with any operating system, e.g. Linux, Windows, Unix, etc.



VirtualBox Arch.



Virtualization Architecture

Virtualization Automation

- **Vagrant** is a way to use Oracle's **Virtual Box** or **VMWare Fusion** on a developer workstation for the purpose of creating disposable and shareable development environments.
  - Vagrant does not compare to Puppet, Chef or Docker, Vagrant is meant to be used with them.
  - Vagrant takes the entire description of your development environment and couches it in a Ruby file. This means your development environment configuration is code and can be shared, rolled back and rolled forward with ease.
  - The developers are free to try new and innovative things such as, Java package, Python or the latest version of PHP without worrying that a failure might take you days to set up or unravel.
  - Vagrant is a killer development application, and should be considered by nearly every development team.
- **Vagrantfile** is to describe, how to configure and provision virtual machines.
  - Syntax of Vagrantfile is in Ruby.
  - Supports VirtualBox, Vmware, AWS and other providers.
  - One can choose from publicly available boxes at, https://atlas.hashicorp.com/boxes/search or, http://www.vagrantbox.es/.

Vagrant - practical session

- Deploying vagrant.
- Deploying provider, VirtualBox.
- Browse through vagrant box repository.
- Vagrantfile walk through.
- Download vagrant box and spin up an instance.
- Connect to instance using ssh.
- Automate vagrant instance creation.

Vagrant Commands

Common commands:

- box manages boxes: installation, removal, etc.
- connect connect to a remotely shared Vagrant env.
- destroy stops and deletes all traces of the vagrant machine
- global-status outputs status Vagrant environments for this user
- halt stops the vagrant machine
- help shows the help for a subcommand
- init initializes a new Vagrant environment by creating a Vagrantfile
- login log in to Vagrant Cloud
- package packages a running vagrant environment into a box
- plugin manages plugins: install, uninstall, update, etc.
- provision provisions the vagrant machine
- rdp connects to machine via RDP
- reload restarts vagrant machine, loads new Vagrantfile configuration
- resume resume a suspended vagrant machine
- share share your Vagrant environment with anyone in the world
- ssh connects to machine via SSH
- ssh-config outputs OpenSSH valid configuration to connect to the machine
- status outputs status of the vagrant machine
- suspend suspends the machine
- up starts and provisions the vagrant environment
- version prints current and latest Vagrant version

Primary commands you want to focus on at first:

- init – Create Vagrantfile in current directory.
- up – If environment never been created before, create it and run provision on machines. If machines are stopped (halt), just start them without provision
- halt – Shutdown environment. Equivalent to powering off machines.
- provision – (re)Run provision scripts as defined in Vagrantfile.
- destroy – Stop and delete environment.
- reload – Will restart environment and apply new setting from Vagrantfile without destroying.
- ssh – SSH into machine.
- status – Get environment status to see whether machines are running or not.

- vagrant box add my-box file:///d:/path/to/file.box

Vagrantfile structure

Vagrant + Configuration Management tool = Docker !!!!!

```
Vagrant.configure "2" do |c|
    c.vm.hostname = "chefserver"
    c.vm.box = "ubuntu/trusty64"
    c.vm.network = "private_network", ip:"172.28.128.28"
    c.vm.provider "virtualbox" do |v|
        vb.memory = "1500"
    end
    c.vm.provision "shell", inline: <<-SHELL
        cd ~
        wget https://opscode-omnibus-packages.s3.amazonaws.com/ubuntu/12.04/x86_64/chef-ser
        sudo dpkg -i chef-server
        sudo chef-server-ctl reconfigure
    SHELL
end
```

Thanks You