UT Southwestern
Medical Center

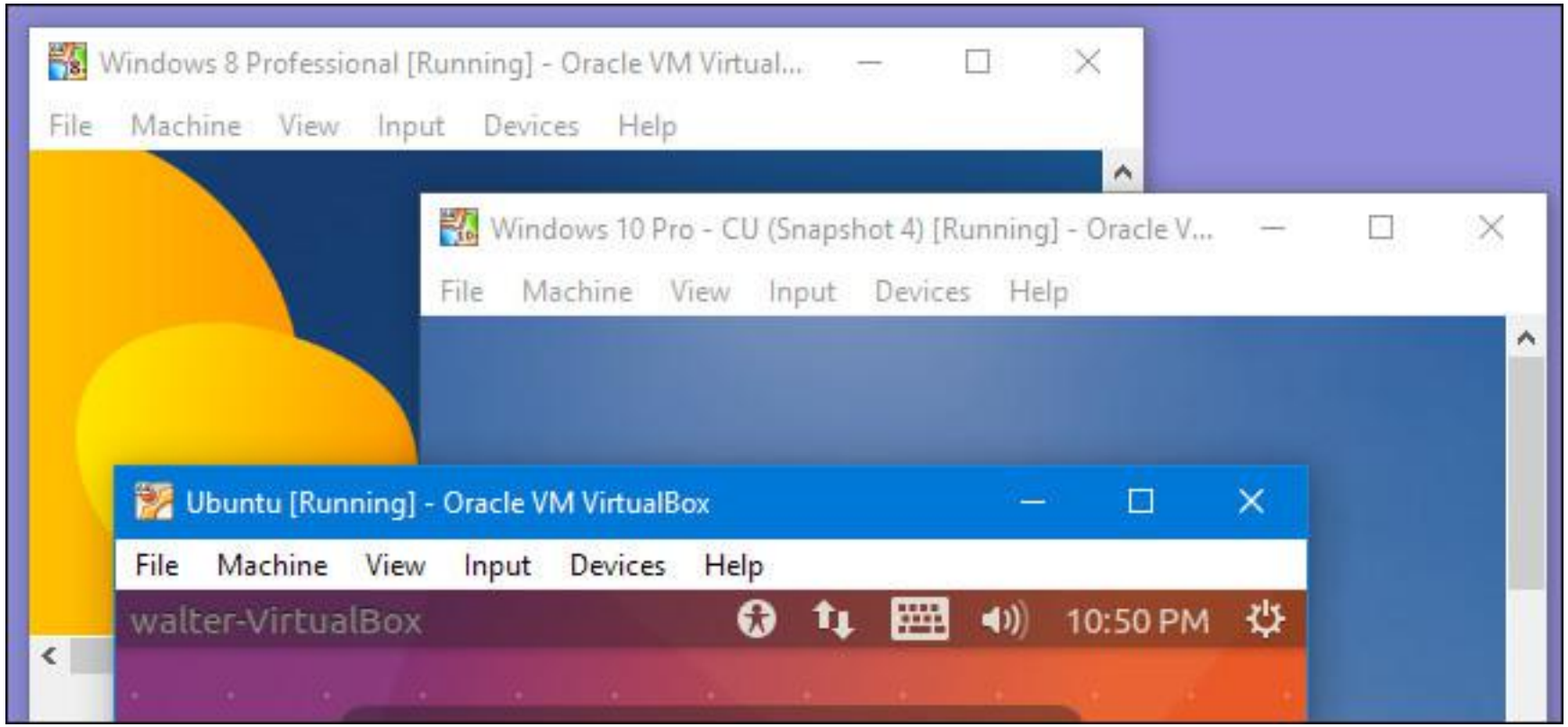
Reproducible Developmental Environment with Vagrant

Peng Lian
March 17, 2021

Virtual Machine and the Softwares

- **VirtualBox**
- **VMWare**
- **KVM**

… …



What is Vagrant?

- **A wrapper** around Virtual Machines
  - VirtualBox, VMware, KVM
- Create standard environments using provisioning scripts
  - Everyone works from the same base configuration
- Remote access to boxes
  - Help test and debug remotely

Why Vagrant?

- Create new Virtual Machines (VMs) quickly
- Reproducibility
  - Provision the environments with a script
- Portability
  - Only need to copy the initialisation and provisioning scripts
- Cross Platform
  - Linux, MacOS, Windows

Vagrant Boxes

- Previously built base VM images that are ready to run
- Create your own
- Use a publicly created one
  - **https://atlas.hashicorp.com/boxes/search**
  - Various flavours of Linux, Windows or MacOS

Getting Up and Running

- Having found a suitable image, all it needs is to be initialised and brought up

```
$ cd ~/new_project
$ vagrant init hashicorp/precise64
$ vagrant up
$ vagrant ssh
```

Vagrant init

- Check the local repository for the image and download it if required
- Creates a copy of the image in your working directory
- Creates a ***Vagrantfile*** initialisation file in your working directory

Vagrant up

- Reads the configuation from the *Vagrantfile*
- Starts the Virtual Machine
- Provisions the machine according to the configuration
  - Includes copying a SSH public key for access to the VM

Vagrant ssh

- Logs into the VM over SSH using the public key created during initialisation

Vagrantfile

- The key configuration file
- Describes the VM configuration and optional provisioning
- Written in Ruby (but simple enough to understand without detailed knowledge of Ruby)

Minimum Configuration of a Vagrantfile

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

Vagrant.configure("2") do |config|
  config.vm.box = "bento/ubuntu-16.04"
end
```

Config Options (config.vm)

- config.vm.box
  - The VM to be used
- config.vm.hostname
  - Sets the hostname of the box
- config.vm.network
  - Configures the network
- config.vm.provider
  - Configures settings specific to a provider
- config.vm.synced_folder
  - Configures the synced folders between the host

Network (config.vm.network)

- Port forwarding

```
Vagrant.configure("2") do |config|
  config.vm.network "forwarded_port", guest: 80, host:8080
end
```

- Static IP

```
Vagrant.configure("2") do |config|
  config.vm.network "private_network", ip: "192.168.1.1"
end
```

- By default, networks are private (only accessible from the host machine)
- Use the flag `"public_network"` to make the guest network accessible from the LAN

```
Vagrant.configure("2") do |config|
  config.vm.network "public_network", ip: "10.0.0.1"
end
```

Provider (config.vm.provider)

- Override options for the Virtual Machine provider.

```
config.vm.provider "virtualbox" do |vb|
  vb.memory = "2048"
end
```

Synced_folder (config.vm.synced_folder)

- By default, Vagrant shares the project directory (the one with the Vagrantfile) to the guest as `/vagrant`
- Additional folders can also be shared

```
Vagrant.configure("2") do |config|
  # other config here

  config.vm.synced_folder "src/", "/srv/website"
end
```

Provisioning

- Allows initial configuration of the VM
  - Shell scripts
  - Ansible
  - Puppet
  - etc

When provisioning happens

- On the **first time** of `vagrant up`
- When `vagrant provision` is run on an active VM
- When `vagrant reload --provision` is run

Provision Shell Commands/Scripts (config.vm.provision)

- Either **inline** or **path** (path to an external script file)

```
config.vm.provision "shell", inline: <<-SHELL
    apt-get update
    # Create the linux directory if it does not exist
    if [ ! -d /home/vagrant/src/linux ]
    then
        cd /home/vagrant
        mkdir src
        cd src
    fi
  SHELL
end
```

```
Vagrant.configure("2") do |config|
  config.vm.provision "shell", path: "script.sh"
end
```

Provision Files

- Allows uploading of files from the host to the guest VM
- Used in conjunction with shell scripts

```
Vagrant.configure("2") do |config|
  # ... other configuration

  config.vm.provision "file", source: "~/.gitconfig", destination:
".gitconfig"

end
```

Stopping the VM

- `vagrant halt`
  - Stops the VM cleanly

- `vagrant destroy`
  - Removes the VM

An Example of a Vagrantfile

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

Vagrant.configure("2") do |config|
  config.vm.box = "centos/7"


  # Forward the python dev server port on 8000 to host 8003
  config.vm.network "forwarded_port", guest: 8000, host: 8003

  # Install system things as the root user via sudo
  config.vm.provision "shell", path: "provision/vagrant_provision.sh"

end
```

An Example of a Provision Script

provision/vagrant_provision.sh

```
#
# Setup Proxy
#
if ping -c 1 proxy.swmed.edu &> /dev/null; then
    echo "SETTING UTSW PROXY"
    echo "export http_proxy=http://proxy.swmed.edu:3128" > /etc/profile.d/proxy.sh
    echo "export https_proxy=http://proxy.swmed.edu:3128" >> /etc/profile.d/proxy.sh
    echo "export all_proxy=http://proxy.swmed.edu:3128" >> /etc/profile.d/proxy.sh
    . /etc/profile.d/proxy.sh
    echo "proxy=http://proxy.swmed.edu:3128" >> /etc/yum.conf
fi


# Install libs for vbguest
yum -y install epel-release
yum -y update
```

Questions?