Introduction to VAGRANT

Azher Ullah Khan
DevOps
azherullahkhan@hotmail.com
Bangalore, India

What is Vagrant?

- Command line utility for managing the lifecycle of virtual machines
- Written by Mitchell Hashimoto
- Automates VM creation with
    - VirtualBox
    - VMWare
    - Hyper-V
- Integrates well with configuration management tools
    - Shell
    - Ansible
    - Puppet
    - Chef
- Runs on Linux, Windows, MacOS
- More info: http://www.vagrantup.com/

In a nutshell

- It is a tool for **developers t**o **manage VMs**
- Automate the **setup** of your d**evelopment/QA/Production environment**

Why use Vagrant?

- 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.

Why use Vagrant?

- Create new VMs quickly and easily
    - Only one command! vagrant up
- Keep the number of VMs under control
- Reproducibility
- Identical environment in development and production

(No more "works on my machine“ excuse)

- Portability
    - No more 4GB .ova files
    - git clone and vagrant up

Development
environments
made easy.

Has this happened to you?

New starter

- Someone joins your project…
- They pick up their laptop…
- Then spend the next 1-2 days following instructions on setting up their environment, tools, etc.

Instead, lets do this.

New starter

- Someone joins your project…
- They pick up their laptop…
- Then spend the next 10 minutes running a script which sets their environment up for them.

Step by step

Prerequisites for this session

- Virtualbox
- Vagrant
- Cygwin or Putty
- Options: GIT

- Both Virtualbox and Vagrant have great, simple installation instructions.

Setting up the box

Follow these steps, once Virtualbox and Vagrant are installed.

Getting set up

- **Add a Vagrant box**
- Create the VM
- Configure the VM
- Set up your project environment

Adding the box

- `vagrant box add ubuntu-precise http://cloud-images.ubuntu.com/precise/current/precise-server-cloudimg-vagrant-i386-disk1.box`

Getting set up

- Add a Vagrant box
- **Create the VM**
- Configure the VM
- Set up your project environment

Start with a working dir.

- `mkdir my_awesome_project`
- `cd my_awesome_project`

Initialise the Vagrant setup

- `vagrant init ubuntu-precise`

What happens under the hood?

- `vagrant init ubuntu-precise`

```
A `Vagrantfile` has been placed in this
directory. You are nowready 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.
```

- A Vagrantfile is created (that's all!)

Launch the Vagrant VM

- `vagrant up`

What happens under the hood?

- `vagrant up`
- The base box is downloaded and stored locally in `~/.vagrant.d/boxes/`
- A new VM is created and configured with the base box as template the VM is booted
- The box is provisioned

SSH to the VM

- `vagrant ssh`

Getting set up

- Add a Vagrant box
- Create the VM
- **Configure the VM**
- Set up your project environment

Finding base boxes

- Hosted by Hashicorp: https://atlas.hashicorp.com/
- 3rd party repository: http://vagrantbox.es/

Using another base box

From the command line (Published on Atlas):

```
$ vagrant box add centos/7
$ vagrant init centos/7
```

From the command line (Box not on Atlas):

```
$ vagrant box add --name centos71-nocm  https://tinfbo2.hogent.be/pub/vm/centos71-nocm-1.0.16.box
$ vagrant init centos71-nocm
```

In your Vagrantfile:

```
VAGRANTFILE_API_VERSION = '2'

Vagrant.configure(VAGRANTFILE_API_VERSION) do |config|
  config.vm.box = 'centos71-nocm'
  config.vm.box_url = 'https://tinfbo2.hogent.be/pub/vm/centos71-nocm-1.0.16.box'
end
```

Configuring Vagrant boxes

Minimal Vagrantfile:

```
VAGRANTFILE_API_VERSION = '2'

Vagrant.configure(VAGRANTFILE_API_VERSION) do |config|
    config.vm.box = 'ubuntu-precise‘
end
```

- Vagrantfile = Ruby

Install all the things!

-
  ```
  sudo apt-get install curl apache2 avahi-daemon
  avahi-discover avahi-utils gcc git-core
  libapache2-mod-dnssd make mysql-server samba
  git unzip vim php5 php-apc php5-cli php5-curl
  php5-dev php5-gd php5-memcache php5-memcached
  php5-mysqlnd php5-xdebug
  ```

- These aren’t all essential, and make a good base for a good development environment (for Php).

Configuring the VM

- For more info, see the docs at https://docs.vagrantup.com/

or the default Vagrantfile

Configuring the VM

```
# -*- 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.
VAGRANTFILE_API_VERSION = "2"

Vagrant.configure(VAGRANTFILE_API_VERSION) do |config|
  # Use the same key for each machine
  config.ssh.insert_key = false

  config.vm.define "vagrant1" do |vagrant1|
    vagrant1.vm.box = "ubuntu/trusty64"
    vagrant1.vm.network "private_network", ip: "192.168.50.4"
    vagrant1.vm.host_name = "serverone.example.com"
    vagrant1.vm.network "forwarded_port", guest: 80, host: 1080
    vagrant1.vm.network "forwarded_port", guest: 8080, host: 1181
```

Applying changes

- When you change the Vagrantfile, do:

```
$ vagrant reload
```

- Or, if the change is profound:

```
$ vagrant destroy -f
$ vagrant up
```

Applying the change

```
$ vagrant destroy
    default: Are you sure you want to destroy the 'default' VM? [y/N] y
==> default: Forcing shutdown of VM...
==> default: Destroying VM and associated drives...
$ vagrant up
[...]
$ vagrant ssh
```

Getting set up

- Add a Vagrant box
- Create the VM
- Configure the VM
- **Set up your project environment**

Manual setup is bad

Streamlining the setup

Manual installation is *never* efficient

Automated provisioning

=

super-quick setup

Provisioning

From Just Enough Operating System to fully functional configured box

- Shell script
- Ansible
- Puppet (Apply + Agent)
- Chef (Solo + Client)

Shell provisioning

```
Add to your Vagrantfile

      config.vm.provision 'shell', path: 'provision.sh'

Put the script into the same folder as Vagrantfile
```

Provisioning with Ansible

- **Ansible** (http://ansible.com/)
- Configuration management tool written in Python
- Simple configuration (YAML)
- No agent necessary (but recommended for large setups)
- Idempotent

```
config.vm.define vagrant1' do |node|
  [...]
  node.vm.provisioning 'ansible' do |ansible|
    ansible.playbook = 'ansible/site.yml'
  end
end
```

Quick Launch a Vagrant VM with Git+Nginx using Shell Provisioning

- git clone https://github.com/azherullahkhan/testvagrant.git
- `cd testvagrant`
- `vagrant up`
- `Vagrant ssh`

Setup with multiple VMs:

```
# -*- 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.ssh.insert_key = false


  config.vm.define "vagrant1" do |vagrant1|

    vagrant1.vm.box = "ubuntu/trusty64"

    ##vagrant1.vm.network :hostonly, "192.168.206.130"

    vagrant1.vm.network "private_network", ip: "192.168.50.4"

    #vagrant1.vm.network "private_network", type: "dhcp"

    vagrant1.vm.host_name = "server1.example.com"

    vagrant1.vm.network "forwarded_port", guest: 80, host: 1080

    vagrant1.vm.network "forwarded_port", guest: 8080, host: 1181

    vagrant1.vm.network "forwarded_port", guest: 8081, host: 1182

    vagrant1.vm.network "forwarded_port", guest: 5000, host: 1000

    vagrant1.vm.network "forwarded_port", guest: 443, host: 1443

  end

  config.vm.define "vagrant2" do |vagrant2|

    vagrant2.vm.box = "ubuntu/trusty64"

    ##vagrant2.vm.network :hostonly, "192.168.206.131"

    vagrant2.vm.network "private_network", ip: "192.168.50.5"

    #vagrant2.vm.network "private_network", type: "dhcp"

    vagrant2.vm.host_name = "desktop1.example.com"
```

Summary

```
$ vagrant init user/box    # Create Vagrantfile for specified
base box
$ vi Vagrantfile           # Customize your box
$ vagrant up [host]        # Create VM(s) if needed and boot
$ vagrant reload [host]    # After every change to
Vagrantfile
$ vagrant halt [host]      # Poweroff
$ vagrant destroy [host]   # Clean up!
$ vagrant ssh [host]       # log in
$ vagrant status [host]    # Status of your VM(s)
```

Creating your own base box

Why create a base box?

- More flexibility than ansible/puppet alone
- Trusted source
- Specific version of O/S (maybe you *really* want to run Slackware as your O/S of choice!)

How to create a base box

- Start by creating the VM in Virtualbox* as usual.

Follow community standards where possible (sizing of VM, disk, RAM, etc).

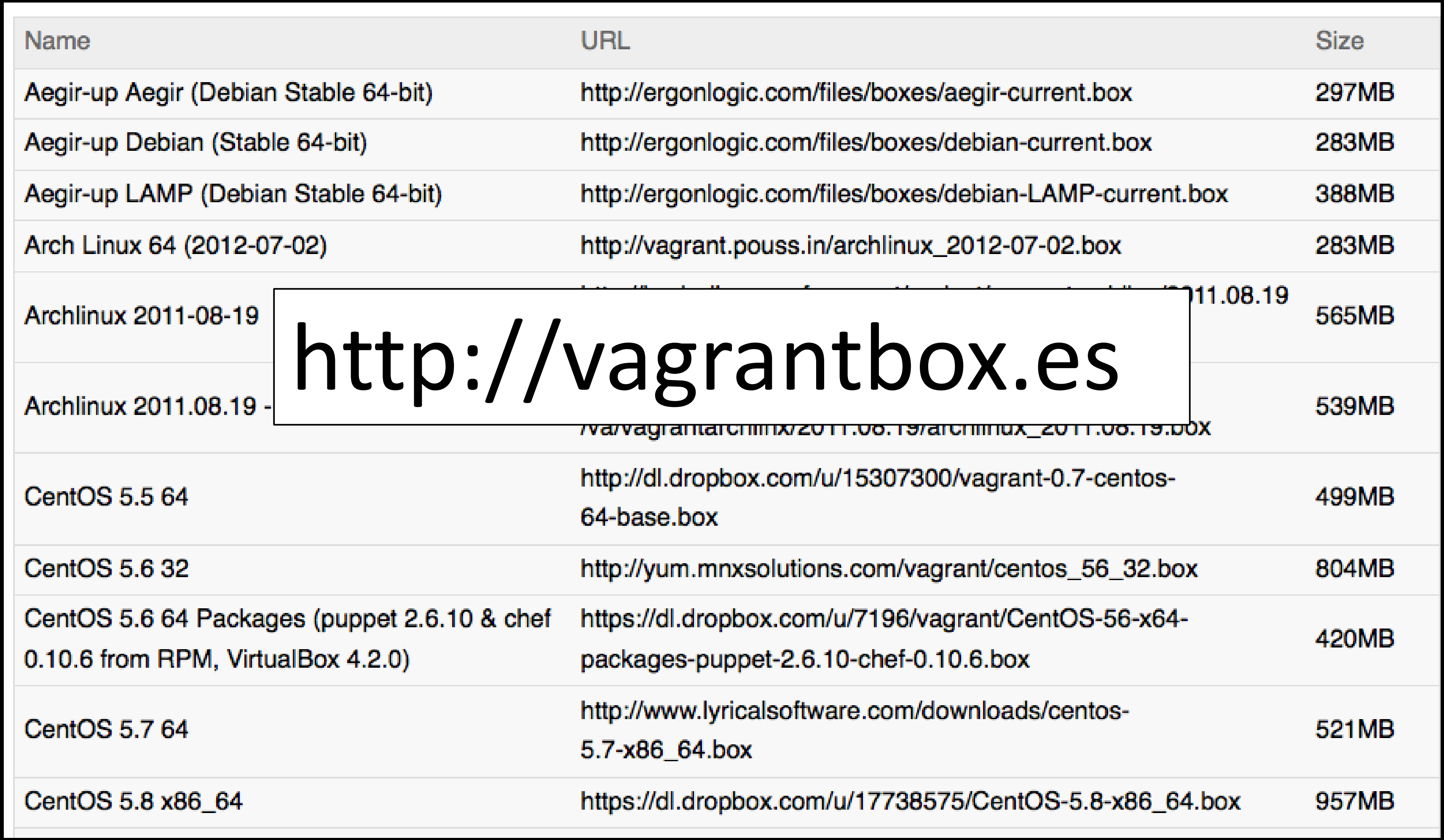
Add several Vagrant-specific tools (an SSH key, etc). Instructions on http://vagrantup.com/.

- OR: use Veewee to build it for you.

* Vagrant is becoming less Virtualbox-specific, so you may be able to use a different provider, such as VMWare.

Choosing a basebox

Choosing a base box



Name	URL	Size
Aegir-up Aegir (Debian Stable 64-bit)	http://ergonlogic.com/files/boxes/aegir-current.box	297MB
Aegir-up Debian (Stable 64-bit)	http://ergonlogic.com/files/boxes/debian-current.box	283MB
Aegir-up LAMP (Debian Stable 64-bit)	http://ergonlogic.com/files/boxes/debian-LAMP-current.box	388MB
Arch Linux 64 (2012-07-02)	http://vagrant.pouss.in/archlinux_2012-07-02.box	283MB
Archlinux 2011-08-19	[illegible]11.08.19	565MB
Archlinux 2011.08.19 - [illegible]	[illegible] /va/vagrantarchlinux/2011.08.19/archlinux_2011.08.19.box	539MB
CentOS 5.5 64	http://dl.dropbox.com/u/15307300/vagrant-0.7-centos-64-base.box	499MB
CentOS 5.6 32	http://yum.mnxsolutions.com/vagrant/centos_56_32.box	804MB
CentOS 5.6 64 Packages (puppet 2.6.10 & chef 0.10.6 from RPM, VirtualBox 4.2.0)	https://dl.dropbox.com/u/7196/vagrant/CentOS-56-x64-packages-puppet-2.6.10-chef-0.10.6.box	420MB
CentOS 5.7 64	http://www.lyricalsoftware.com/downloads/centos-5.7-x86_64.box	521MB
CentOS 5.8 x86_64	https://dl.dropbox.com/u/17738575/CentOS-5.8-x86_64.box	957MB

http://vagrantbox.es

Key resources

- **Virtualbox**
  https://www.virtualbox.org/
- **Vagrant**
  http://vagrantup.com/
- **Base-box list**
  http://www.vagrantbox.es/
- **Puppet resources**
  http://puppetlabs.com/
- **Chef resources**
  http://www.opscode.com/chef/

Beyond Vagrant & dev VMs

Beyond Vagrant & dev VMs

- Vagrant is expanding to cover other provisioning tools:
  - VMWare Fusion
  - ESXi
  - Amazon
  - ???
- Puppet, Ansible and Chef can manage your test/stage/CI/production environments too.
- Tools like Cobbler and Satellite can fully-automate the build of new VMs

THANK YOU
FOR YOUR ATTENTION
DIYLOL.COM