**Vagrant**

Azure Provider

Draft A

13th October 2015

****

**Actian Europe Limited**

16/17 Pavilion Business Park

Royds Hall Road

Leeds

LS12 6AJ

www.actian.com

© Actian Europe Limited, 2015

**Contents**

1 Introduction 3

2 Azure 4

2.1 Gather Essential Information 4

2.2 Create Certificates 7

3 Vagrant 10

3.1 Prerequisites for Azure 10

3.2 Vagrantfile Changes 10

3.3 Gotchas 12

4 Azure Linux VM Access 15

4.1 Putty Private Key 15

4.2 Putty Access 16

Appendix A Example Vagrant file 18

# Introduction

This document is intended for the less able, such as this author, to aid in getting Vagrant to create a VM using in the Microsoft Azure cloud service.

The author makes no promises that this is the best approach as other more enlightened individuals may be able to streamline or suggest better alternatives and any valid contribution is gladly accepted.

This document is necessary because, where as creating a local VM using Vagrant against say the Oracle Virtual Box (VB) provider is fairly straight forward, no so for Azure. There are quite a few ‘nuances’ with Azure as well as components,such as Chef, that just don’t seem to work properly in the Azure envieonmentand hence need to be circumvented.

Additionally, unlike VB you can’t have a Vagrant file for the Azure provider that works for everyone. There are details specific to you and your Azure Account Subscription.

This document assumes that you are running Vagrant pre-installed on a Windows machine.

A Microsoft Azure free trial was used to develop and test this approach.

# Azure

First thing to do is sign up for Azure. At the time of writing a 30 day free trial with £125 credit was available. This should be ample to get you off the ground as usage can be kept very low by simply destroying your VMs before you go home.

Next you need the image name for the box. Unlike the other providers, the image names for the boxes lacks sensibility because Microsoft never make anything easy. i.e. a Virtual Box CentOS images for version is 6.7 is box-cutter/centos67. The following may seem convoluted but seems to be the documented approach to getting the Azure image name.

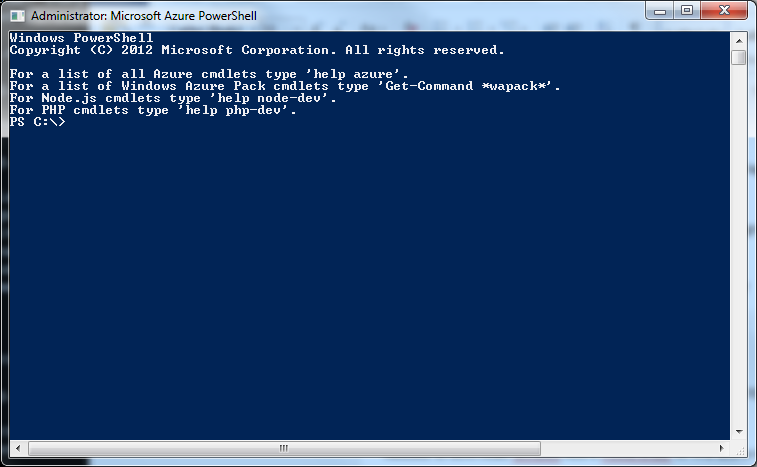
## Gather Essential Information

For Azure it does not seem possible to get a list of box image names from the ‘Management Interface’ so you have to proceed as follows:

1. Download Azure PowerShell

<http://go.microsoft.com/fwlink/p/?LinkId=320376>

Once you have installed it run up the PowerShell:



1. Configure Access to Azure

In the PowerShell download the settings files for your subscriptions. In this example the author has used the Free Trial. In the PowerShell enter:

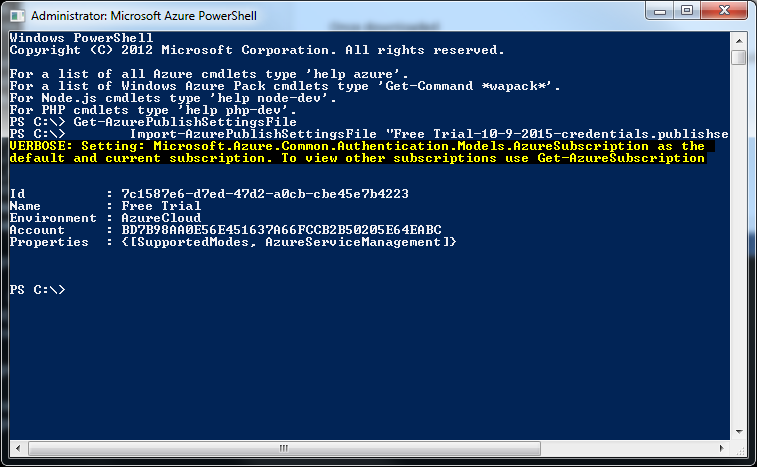
Get-AzurePublishSettingsFile

If you are not currently logged on to your Azure account you will be prompted for your user and password.

In your downloads folder you will now have a file something like ‘Free Trial-10-13-2015-credentials.publishsettings’. I find it easiest to copy to C: which is the default directory for the PowerShell

Now import these settings by typing the following in the PowerShell:

Import-AzurePublishSettingsFile "Free Trial-10-9-2015-credentials.publishsettings"



1. Subscription ID

Note the ‘Id’ from above. This is your Free Trial Subscription ID and you will need this later when updating your Vagrant file to use the Azure provider.

1. OS Name in Azure

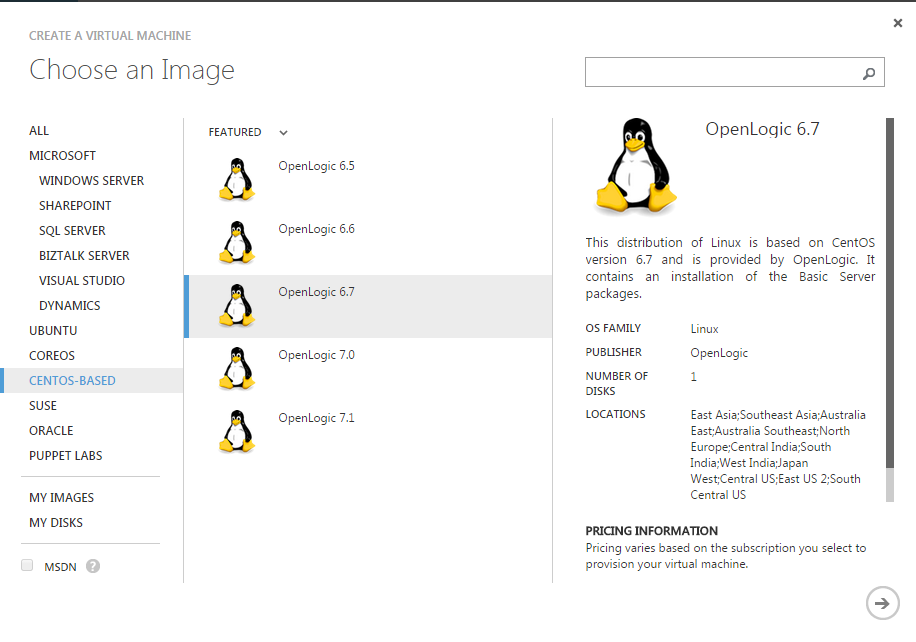
Before you can get the actual OS image name we use in the Vagrant file you need to get the OS name as Microsoft see it. Microsoft can’t just call it Centos 6.7!

Logon to your Microsoft Azure account.

Select ‘Virtual Machines’ 🡪 ‘CREATE A VIRTUAL MACHINE’

Select ‘FROM GALLERY’

Select the appropriate image. In this example ‘CentOS 6.7’ which Microsoft refer to as ‘OpenLogic 6.7’.

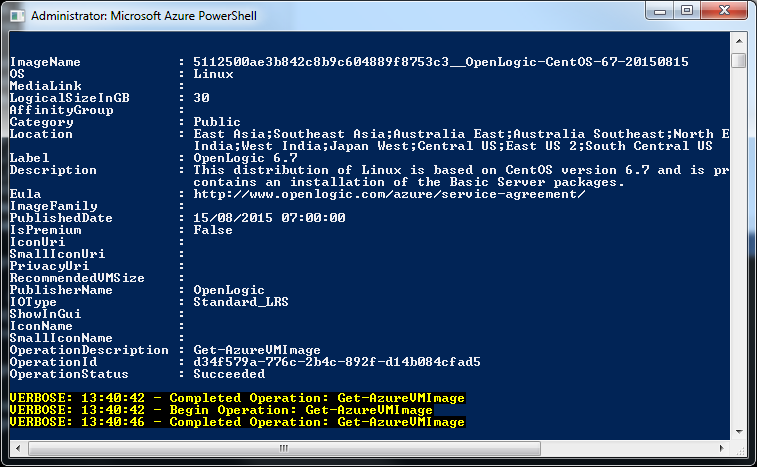


Note the OS name e.g. ‘OpenLogic 6.7’

1. OS Image Name

Now return the PowerShell and run the following command with the OS name substituted:

Get-AzureVMImage | where-object { $\_.Label -like "OS Name" }



Note the long image name at the top as you will need this later for your Vagrant file. Told you wasn’t as straight forward as VB!

## Create Certificates

In order for Vagrant to have access to your Azure account you are going to need to create a certificate to upload to your Azure Account.

The author ran these command from a DOS prompt in the directory where the Vagrantfile was located e.g. C:\Azure\_Example. This just makes life easier for the purpose of this document.

The certificates were created by the author using openssl on Windows:

openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout azurevagrant.pem -out azurevagrant.pem

Note

The author is unsure how and when ‘opensll’ was installed but would hazard a guess it came with GitHub client. However, if you don’t have it the following command from your Azure PowerShell should install it:

choco install openssl.light

You’ll need to answer a few questions here but most can be left blank. For the purposes of this examples the only ones completed are highlighted.

Generating a 2048 bit RSA private key

........................+++

............+++

writing new private key to 'azurevagrant.pem'

-----

You are about to be asked to enter information that will be incorporated

into your certificate request.

What you are about to enter is what is called a Distinguished Name or a DN.

There are quite a few fields but you can leave some blank

For some fields there will be a default value,

If you enter '.', the field will be left blank.

-----

Country Name (2 letter code) [AU]:**UK**

State or Province Name (full name) [Some-State]:

Locality Name (eg, city) []:

Organization Name (eg, company) [Internet Widgits Pty Ltd]:**Actian**

Organizational Unit Name (eg, section) []:

Common Name (e.g. server FQDN or YOUR name) []:

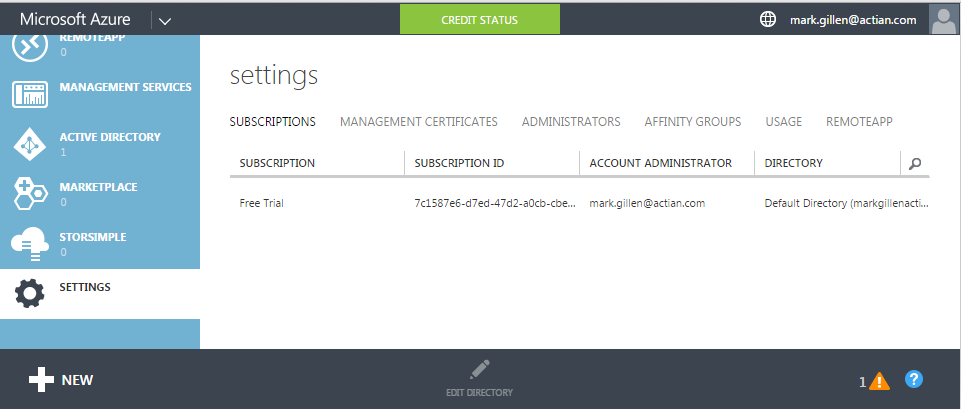
Email Address []:**my.name@actian.com**

Then run this command to create a certificate:

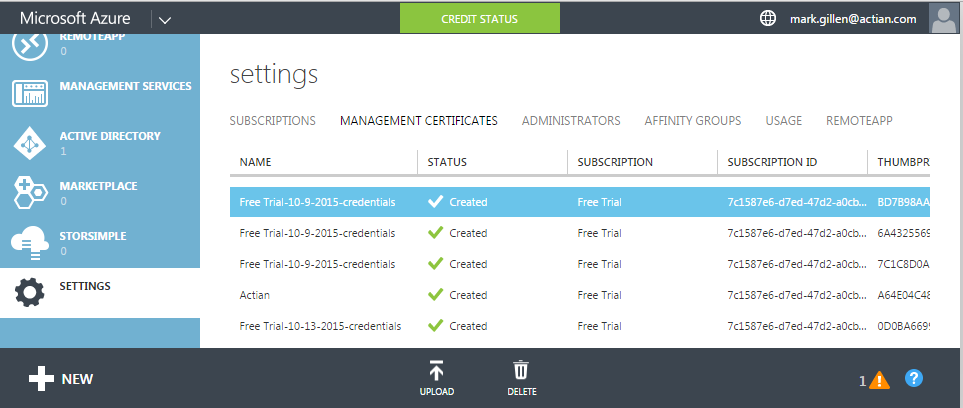
openssl x509 -inform pem -in azurevagrant.pem -outform der -out azurevagrant.cer

The .cer file is the one you are going to import into your Azure account and the .pem and will be used in conjunction with it in the Vagrant provider configuration. Additionally, later you will use the .pem file to generate a private key file to give Putty access to the Azure CentOS VM, if that is your preferred terminal access method.

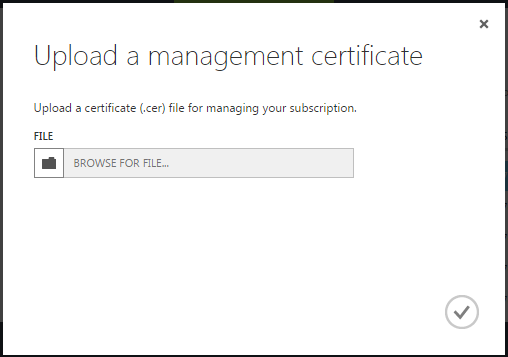
In your Microsoft Azure account select ‘SETTINGS’ at the bottom of the left scrollable panel:



Select ‘MANAGEMENT CERTIFICATES’



Select the ‘UPLOAD’ icon at the bottom



Select the .cer file you created earlier and select the tick to upload.

# Vagrant

## Prerequisites for Azure

You should already have installed vagrant. There are a couple of additional essential installs to create a VM against Azure:

1. Install the Vagrant Azure plugin:

vagrant plugin install vagrant-azure

1. Install a basic Vagrant Azure Box. I can’t explain the reason for this but rest assured it won’t work without:

vagrant box add azure https://github.com/msopentech/vagrant-azure/raw/master/dummy.box

There was reference to the need for a plug-in to manage the version download but I don’t believe this is required.

vagrant plugin install vagrant-omnibus

## Vagrantfile Changes

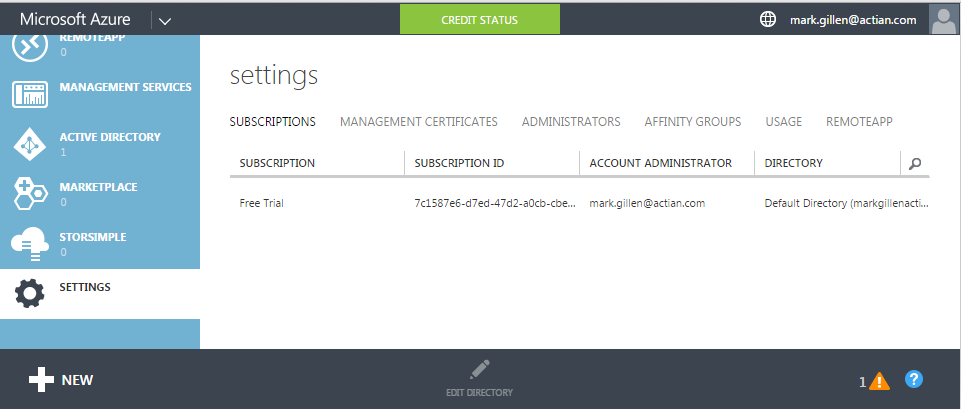
First thing to know is that unlike VB you can’t have a Vagrant file for the Azure provider that works for everyone. There are details specific to you and you only these being:

1. Your Azure Subscription ID;
2. Your certificate:

* The .pem file.

You obtained these earlier in Sections 2.1 and 2.2. It should be noted however that most of section 2.1 is related to getting the VM image name.

If you just want to stick with CentOS 6.7 then Section 2.1 is not necessary and the essential Subscription ID can be obtained from the subscription details in your Azure Account. It is in your Microsoft Azure account select ‘SETTINGS’ at the bottom of the left scrollable panel:



In your chosen folder e.g. c:\Azure\_Example copy in the files from GIT hub for the Vagrant package to which this document is associated.

In the ‘vagrantfile’ the settings that you will need to change related to configuring an Azure provider are documented in the code snipper below though the full file is available in Appendix A.

A colour scheme has been used to denote the key settings to change:

**RED** – Settings specific to your Azure Account and must be changed.

**AMBER** – Settings that can remain unchanged if you choose to follow the author’s naming standard when completing Section 2.2.

**GREEN** – Settings that affect the naming of Azure components created which you may wish to reflect your own naming standards.

config.vm.provider :azure do |azure, override|

override.vm.box = 'azure'

override.ssh.private\_key\_path = **'azurevagrant.pem'**

override.ssh.pty = true

override.vm.boot\_timeout = 1500

# Mandatory Settings

azure.mgmt\_certificate = **'azurevagrant.pem'**

azure.mgmt\_endpoint = 'https://management.core.windows.net'

azure.subscription\_id = **'#########################'**

azure.vm\_image = '5112500ae3b842c8b9c604889f8753c3\_\_OpenLogic-CentOS-67-20150815'

azure.vm\_name = **'VectorEvaluationVM'**

azure.ssh\_private\_key\_file = **'azurevagrant.pem'**

# Optional Settings

azure.cloud\_service\_name = **'VectorEvaluationVM'**

azure.vm\_location = 'North Europe'

azure.ssh\_port = '22'

# Need larger than default Standard A1 to run Vector

azure.vm\_size = 'Basic\_A2'

end

## Gotchas

1. ResourceNotFound

I did find that the following message is frequently displayed on vagrant up:

ResourceNotFound : The hosted service does not exist.

According to some reports I have read this caused the VM creation to abort. This causes no problems and I can’t suppress the message. According to the vagrant documentation the cloud service is auto generated if you leave azure.cloud\_service\_name blank but I still got this message and I prefer to give things a proper name.

2. Time Out

The default timeout for vagrant when waiting for the VM to start is 360 seconds. Nobody mentions this at all but this is just not sufficient for an Azure VM initially being created as this is a minimum of 5 minutes. Although I have found at times even 10 minutes is not a long enough time out value. It does make me wonder whether anyone has really used this provider properly but to overcome this:

override.vm.boot\_timeout = 1500

Note – ‘ovverride’ rather than config as this is an example from the dual provider Vagrantfile in Appendix A. This allows the default for the none Azure providers to remain at the default.

3. pty

The following setting seems to be mandatory to get things working on Azure:

override.ssh.pty = true

It was found that, despite the dire Vagrant documentation warning below, this setting had to be used otherwise the following error was found

sorry, you must have a tty to run sudo

The SSH command responded with a non-zero exit status.

Vagrant assumes that this means the command failed.

The output for this command should be in the log above.

Please read the output to determine what went wrong.

Be aware, from Vagrant online documentation:

config.ssh.pty - If true, pty will be used for provisioning. Defaults to false.

This setting is an advanced feature that should not be enabled unless absolutely necessary. It breaks some other features of Vagrant, and is really only exposed for cases where it is absolutely necessary. If you can find a way to not use a pty, that is recommended instead.

I don’t know what features it actually breaks and as is my opinion with much OpenSource software why don’t they document what it breaks!

4. Chef

For some as yet undiagnosed reason Chef will not auto install on an Azure CentOS VM. When you perform a “chef\_apply” in a Vagrantfile it checks and installs Chef if it not present. Whatever, method is employed by Vagrant appears not to be acceptable to the Azure VM and results in the message:

Vagrant attempted to execute the capability 'chef\_install'

on the detect guest OS 'linux', but the guest doesn't

support that capability.

This capability is required for your configuration of Vagrant. Please either reconfigure Vagrant to avoid this capability or fix the issue by creating the capability.

This problem was circumvented by pre-installing Vagrant from a shell provision as follows:

sudo su - -c 'curl -L https://www.opscode.com/chef/install.sh | bash'

Additionally Vagrant does not appear to be able to apply Chef Recipes using the expected:

config.vm.provision ‘chef\_apply’ do ….

This appears to work but the results in strange failures? As a result this was circumvented by running it via a shell after uploading the required Chef Recipe script:

config.vm.provision 'file', source: 'chef\_script.rb', destination: '/tmp/chef\_script.rb'

config.vm.provision 'shell', name: Run Chef Recipe', privileged: true, inline: <<-SHELL

sudo su - -c 'chef-apply /tmp/chef\_script.rb'

SHELL

# Azure Linux VM Access

The author has always used Putty for Unix/Linux terminal access so this section covers that approach. There may be better approaches to accessing your VM created in Azure but this works and was fairly simple to achieve.

## Putty Private Key

In order to access the Azure VM using Putty you will need a private key to match the certificate uploaded to your Azure account. You generated one, the .pem file, earlier. However, Putty does not understand this and requires a .ppk.

First create an rsa file using openssl used previously. At your DOS command prompt:

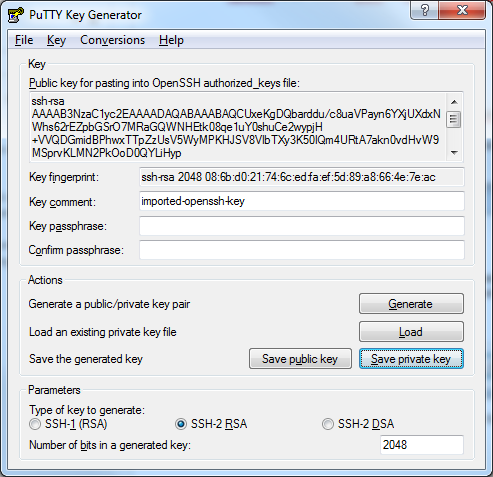
openssl rsa -in azurevagrant.pem -out azurevagrant\_rsa

Only problem is that Putty still does not understand this format and it needs to be converted. This is very easy to achieve because if you have Putty you should have PuttyGen. Start PuttyGen:

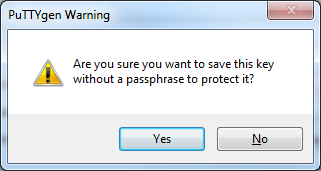


Select ‘Conversions’ -> ‘Import key’ from the menu bar.

From the file browser displayed select the file you created above.



Select the ‘Save private key’ button and respond ‘Yes’ as you don’t want a pass phrase.

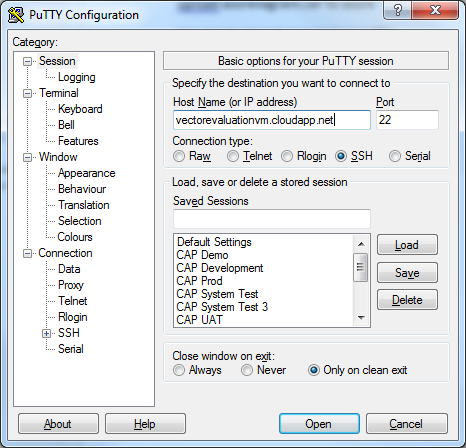


This will create you a .ppk file that Putty will understand. The private key obviously already matches the certificate we uploaded to the Azure account in Section 2.2.

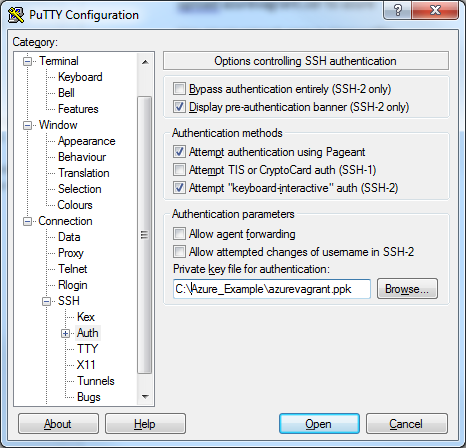
## Putty Access

Now that you have the private key, access is fairly straight forward.

Start Putty and set the hostname. You can get this from your Azure Account but the author is fairly sure it is nearly always your azure.vm\_name set in the Vagrant file + ‘cloudapp.net’ e.g. vectorevaluationvm.cloudapp.net.



Expand ‘SSH’ and select ‘Auth’ and ‘Browse’ for the .ppk file you created earlier:



Select ‘Open’ to connect to your running Azure VM.

1. Example Vagrant file

The following ‘vagrantfile’ installs a vector evaluation edition, ingbuild or rpm, on a CentOS 6.7 environment. The provider is Oracle Virtual Box by default with the option to use a Microsoft Azure subscription.

#-------------------------------------------------------------------------------

# Copyright 2015 Actian Corporation

# Licensed under the Apache License, Version 2.0 (the "License");

# you may not use this file except in compliance with the License.

# You may obtain a copy of the License at

# http://www.apache.org/licenses/LICENSE-2.0

# Unless required by applicable law or agreed to in writing, software

# distributed under the License is distributed on an "AS IS" BASIS,

# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

# See the License for the specific language governing permissions and

# limitations under the License.

#-------------------------------------------------------------------------------

# Pre-requisites required before running this script:

# 1. Install Vagrant (Version 1.7.4 used constructing the above)

# 2. Install Oracle Virtual Box (5.0.4 or later)

# 3. Enable hardware virtulaisation in the BIOS if it is disabled.

# This Vagrant script will perfom the following operations:

# 1. Create a Cento 6.7 Linux environment that is fully up to date.

# 2. Install, via Chef, Actian Vector previously downloaded.

# - Requires an authstring

# - Will also require a Public Key for RPM install.

# 3. Run the Actian DBT3 tests.

# The approach to using 'Chef' in this script may seem strange as the installation

# and chef-apply are performed via the "config.vm.provision 'shell' ...."

# This was intentional to create a generic script that would work for providers

# Oracle Virtual Box and Azure.

# Using Azure 'chef\_apply' will fail installing Chef. Even when Chef is manually

# installed to circumvent this, it will then fail applying a Recipe even

# though it appears to complete successfully.

#-------------------------------------------------------------------------------

# -\*- mode: ruby -\*-

# vi: set ft=ruby :

Vagrant.configure(2) do |config|

config.vm.box = 'box-cutter/centos67'

config.vm.synced\_folder '.', '/vagrant', disabled: true

# Provider - Virtual Box VM (Default)

config.vm.provider :virtualbox do |vb, override|

# Display the VirtualBox GUI when booting the machine

vb.gui = true

# Give the VM an appropriate name

vb.name = 'VectorEvaluationVM'

# Customize the amount of memory on the VM

vb.memory = "4096"

end

# Provider - Microsoft Azure VM

# Documented below are the settings that need to be changed as they are specific

# to youe Azure subscription.

config.vm.provider :azure do |azure, override|

override.vm.box = 'azure'

override.ssh.private\_key\_path = 'azurevagrant.pem'

# You can stick with the naming of this file but you must generate

# your own.

override.ssh.pty = true

override.vm.boot\_timeout = 1500

# Mandatory Settings

azure.mgmt\_certificate = 'azurevagrant.pem'

# See above.

azure.mgmt\_endpoint = 'https://management.core.windows.net'

azure.subscription\_id = ‘########################'

# Your Azure Account Subscription ID.

azure.vm\_image = '5112500ae3b842c8b9c604889f8753c3\_\_OpenLogic-CentOS-67-20150815'

azure.vm\_name = 'VectorEvaluationVM'

azure.ssh\_private\_key\_file = 'azurevagrant.pem'

# See above.

# Optional Settings

azure.cloud\_service\_name = 'VectorEvaluationVM'

azure.vm\_location = 'North Europe'

# You may wish to set this to something appropriate to your location.

azure.ssh\_port = '22'

# Need larger than default Standard A1 Azuure VM to install and run Actian Vector

azure.vm\_size = 'Basic\_A2'

end

# Common code from here.

if Vagrant.has\_plugin?("vagrant-cachier")

# Configure cached packages to be shared between instances of the same base box.

# More info on http://fgrehm.viewdocs.io/vagrant-cachier/usage

config.cache.scope = :box

end

config.vm.provision 'shell', name: 'OS Updates', privileged: true, inline: <<-SHELL

echo never > /sys/kernel/mm/transparent\_hugepage/enabled

sed -i \'s/^SELINUX=.\*$/SELINUX=disabled/\' /etc/selinux/config

yum -y update

# Required for DBT3 Scripts

yum -y install git gcc time

# Required for Vector

yum -y install libaio

# Required for Vector rpm

yum -y install libX11 libXext libXi libXrender libXtst alsa-lib

SHELL

# Upload the required files for the Vector install

# This approach taken as Azure does not allow access to /vagrant share

config.vm.provision 'file', source: 'authstring', destination: '/tmp/authstring'

Dir['actian-vector\*.tgz'].each do |file\_name|

config.vm.provision :file do |file|

file.source = file\_name

file.destination = '/tmp/' + File.basename(file\_name)

end

end

Dir['actian-vector\*.asc'].each do |file\_name|

config.vm.provision :file do |file|

file.source = file\_name

file.destination = '/tmp/' + File.basename(file\_name)

end

end

# Upload Chef files (Run locally to circumvent Azure problem)

config.vm.provision 'file', source: 'actian-user.rb', destination: '/tmp/actian-user.rb'

config.vm.provision 'file', source: 'vector-installer.rb', destination: '/tmp/vector-installer.rb'

# Install Chef (Circumvent auto install as problematic for Azure)

config.vm.provision 'shell', name: 'Install Chef', privileged: true, inline: <<-SHELL

sudo su - -c 'curl -L https://www.opscode.com/chef/install.sh | bash'

SHELL

# Create the 'actian' user ('chef\_apply' fails for Azure)

# Separate from Vector install so user can be given sudo access to uploaded files

config.vm.provision 'shell', name: 'Create Actian User', privileged: true, inline: <<-SHELL

sudo su - -c 'chef-apply /tmp/actian-user.rb'

SHELL

# Set the 'actian' passwd

config.vm.provision 'shell', name: 'Set Actian Password', privileged: true, inline: <<-SHELL

sudo su - -c 'echo -e "actian\nactian" | passwd actian > /tmp/passwd.log 2>&1'

SHELL

# Give actian sudo access with NOPASSWD (Required for DBT3 Test Suite)

config.vm.provision 'shell', name: 'Grant Actian sudo', privileged: true, inline: <<-SHELL

echo 'actian ALL=(ALL) NOPASSWD: ALL' > /etc/sudoers.d/actian

SHELL

# Install Vector ('chef\_apply' fails for Azure)

config.vm.provision 'shell', name: 'Install Vector', privileged: true, inline: <<-SHELL

sudo su - -c 'chef-apply /tmp/vector-installer.rb'

SHELL

# Always Start Vector. Doesn't matter if already started on initial install

# Required for restart.

config.vm.provision 'shell', name: 'Start Vector', run: 'always', privileged: true, inline: <<-SHELL

sudo su - actian -c 'ingstart > /tmp/ingstart.log 2>&1; echo "Done"'

SHELL

# Download and Run the DBT3 Test Suite

config.vm.provision 'shell', name: 'DBT3 Test Suite', privileged: true, inline: <<-SHELL

cd /home/actian

if [ ! -d VectorH-DBT3-Scripts ]; then

su actian -c 'git clone -q https://github.com/ActianCorp/VectorH-DBT3-Scripts'

su - actian -c 'cd VectorH-DBT3-Scripts;chmod 755 \*.sh;./load-run-dbt3-benchmark.sh > /tmp/load-run-dbt3-benchmark.log 2>&1'

fi

SHELL

end

#-------------------------------------------------------------------------------

# End of Vagrant script

#-------------------------------------------------------------------------------