How to Setup a Development Environment for ONAP

Victor Morales

December 12th, 2017

Agenda

- Problem statement
- Proposed solution
- ONAP on Vagrant
  - Installation process
- Key features
  - Why Vagrant?
  - Synced Folders
  - Plugins
  - Shell Provisioning
- Architecture
- Usage
- Benefits

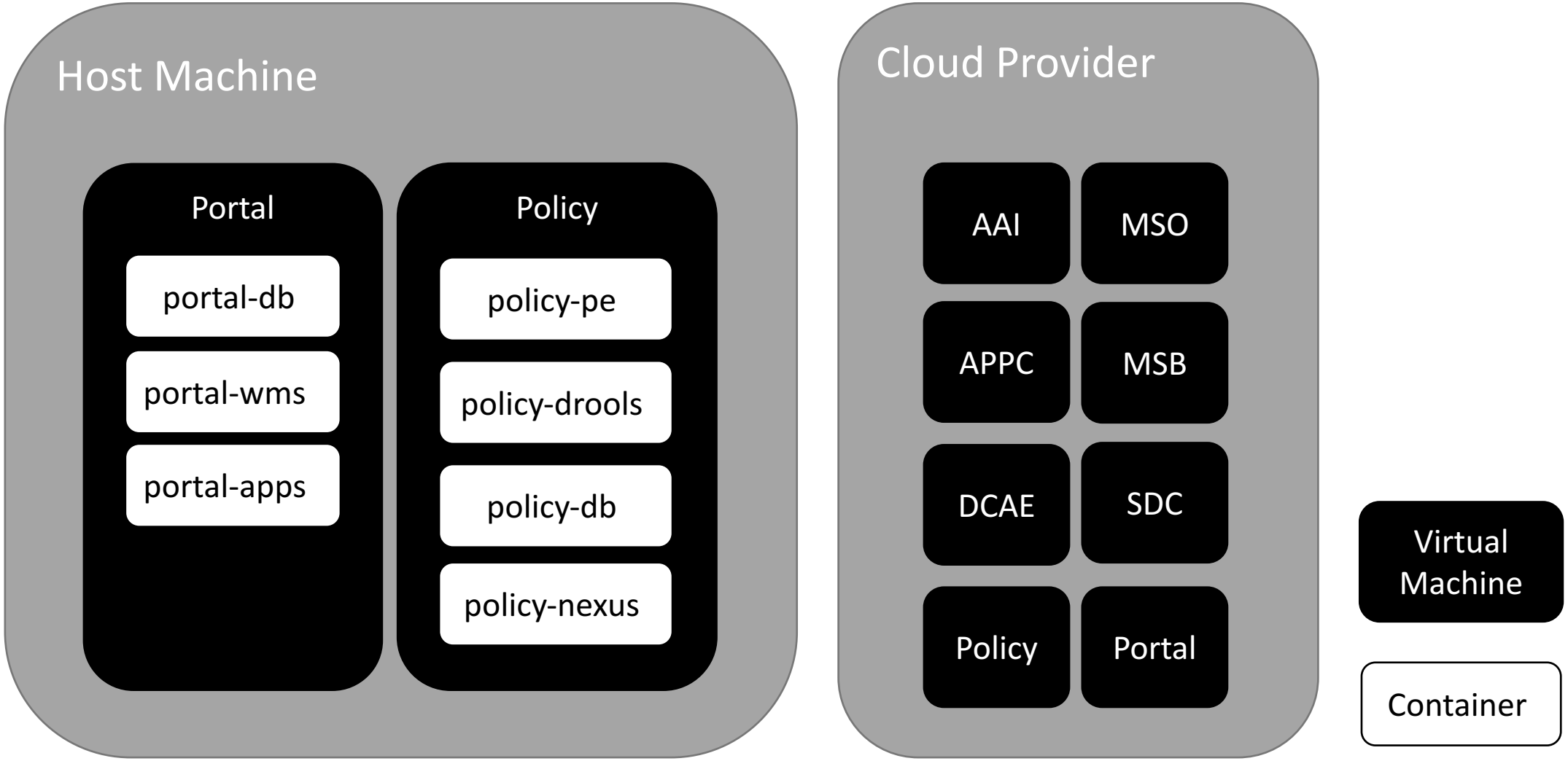
Problem Statement – Multiple Developer Guides

- Every project has their own methods and/or guides for creation a Development Environment
  - A&AI Developer Environment Setup (https://wiki.onap.org/pages/viewpage.action?pageId=10782088)
  - Building & Testing APP-C Component Locally (https://wiki.onap.org/pages/viewpage.action?pageId=6590586)
  - DCAE Controller Development Guide (https://wiki.onap.org/display/DW/DCAE+Controller+Development+Guide)
  - Development Environment for SO (https://wiki.onap.org/display/DW/Development+Environment)
  - Portal Build Instructions (https://wiki.onap.org/display/DW/Portal+Build+Instructions)
  - OpenECOMP SDC Developer Guide (https://wiki.onap.org/download/attachments/1015849/OpenECOMP_SDC_Developer_Guide.pdf?version=1&modificationDate=1499061898000&api=v2)

Problem Statement – Heterogeneous deployments

- Dependencies and instructions omitted or assumed in development guides
- Unable to quickly replicate a fresh environment for validation of latest changes on the project
- Documentation requires a separate task to be synchronized with any latest addition on the project

Proposed solution



ONAP on Vagrant

Automated provisioning tool for ONAP development environments, through common development tasks such as:

- Clones a group of repositories associated to specific component.
- Compilation of java artifacts per component.
- Builds Docker images of specific component.
- Deals with networking configuration behind corporate proxy.
- Manage dependencies required by component.

Minimal Requirements

Component	Requirement
Vagrant	>= 1.8.6
Provider	VirtualBox, Libvirt or OpenStack
Operating System	Linux, Mac OS or Windows
Hard Disk	> 8 GB of free disk
Memory	> 12 GB

Installation process - VirtualBox

http://onap.readthedocs.io/en/latest/submodules/integration.git/bootstrap/vagrant-onap/doc/source/install/index.html

ONAP

Table Of Contents

This Page

Show Source

Quick search

Go

Installation Guide

This project collects instructions related to the automatic creation of a development environment. However, this require platform (VirtualBox, Libvirt and OpenStack). This section explains how to install the most common set of configuration

Ubuntu 14.04 (“Trusty”)

```
$ wget -q https://releases.hashicorp.com/vagrant/2.0.1/vagrant_2.0.1_x86_64.deb
$ sudo dpkg -i vagrant_2.0.1_x86_64.deb
$ echo "deb http://download.virtualbox.org/virtualbox/debian trusty contrib" >> /etc/apt/sources.list
$ wget -q https://www.virtualbox.org/download/oracle_vbox_2016.asc -O- | sudo apt-key add -
$ wget -q https://www.virtualbox.org/download/oracle_vbox.asc -O- | sudo apt-key add -
$ sudo apt-get update -y
$ sudo apt-get install -y virtualbox-5.1 dkms
```

CentOS

```
$ wget -q https://releases.hashicorp.com/vagrant/2.0.1/vagrant_2.0.1_x86_64.rpm
$ sudo yum install vagrant_2.0.1_x86_64.rpm
$ wget -q http://download.virtualbox.org/virtualbox/rpm/rhel/virtualbox.repo -P /etc/yum.repos.d
$ sudo yum --enablerepo=epel install dkms
$ wget -q https://www.virtualbox.org/download/oracle_vbox.asc -O- | rpm --import -
$ sudo yum install VirtualBox-5.1
```

Mac OS

```
$ /usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"
$ brew cask install vagrant
$ brew cask install virtualbox
```

Installation process – Libvirt & OpenStack


```
export OS_AUTH_URL=http://<keystone_ip>:5000/v3
export OS_TENANT_NAME=<project_or_tenant_name>
export OS_PROJECT_NAME=<project_or_tenant_name>
export OS_USERNAME=<openstack_username>
export OS_PASSWORD=<openstack_password>
export OS_REGION_NAME=<openstack_region_name>
export OS_IDENTITY_API_VERSION=<keystone_version_number>
export OS_PROJECT_DOMAIN_ID=<openstack_domain_name>
```

```
export OS_IMAGE=<ubuntu_cloud_image_name>
export OS_NETWORK=<neutron_private_network>
export OS_FLOATING_IP_POOL=<neutron_floating_ip_pool>
export OS_SEC_GROUP=<onap-ssh-secgroup>
```

```
# ONAP Bootstrap variables
export VAGRANT_DEFAULT_PROVIDER=libvirt
```

http://onap.readthedocs.io/en/latest/submodules/integration.git/bootstrap/vagrant-onap/doc/source/features/openstack.html

Key Features

Why Vagrant?

It’s a tool for building and managing virtual machine environments in a single workflow. With an easy-to-use workflow and focus on automation, Vagrant lowers development environment setup time, increases production parity, and makes the "works on my machine" excuse a relic of the past.

Synced Folders

Enables Vagrant to sync a folder on the host machine to the guest machine, allowing you to continue working on your project's files on your host machine, but use the resources in the guest machine to compile or run your project.

```
config.vm.synced_folder './opt', '/opt/', create: true
config.vm.synced_folder './lib', '/var/onap/', create: true
config.vm.synced_folder '~/.m2', '/root/.m2/', create: true
```

Shell Provisioning

Provisioners in Vagrant allow you to automatically install software, alter configurations, and more on the machine as part of the vagrant up process. Shell provisioning is ideal for users new to Vagrant who want to get up and running quickly and provides a strong alternative for users who are not comfortable with a full configuration management system such as Chef or Puppet.

```
all_in_one.vm.provision 'shell' do |s|
  s.path = 'postinstall.sh'
  s.args = ['mr', 'sdc', 'aai', 'mso', 'robot', 'vid',
  s.env = conf
end
```

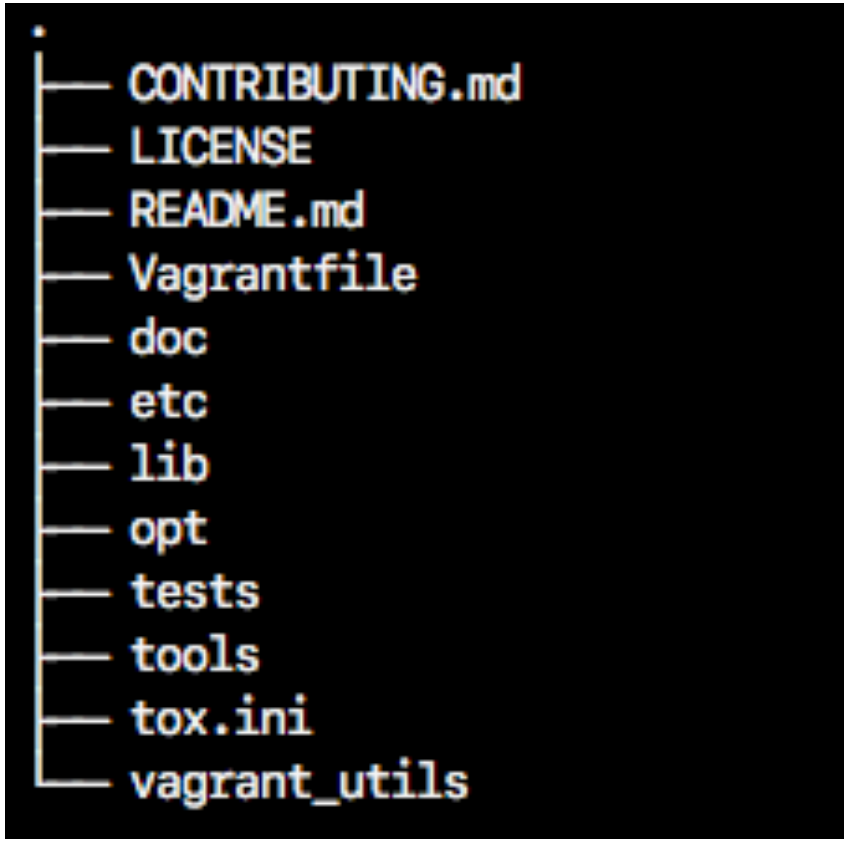
Plugins

They are powerful, first-class citizens that extend Vagrant using a well-documented, stable API that can withstand major version upgrades.

```
if ENV['http_proxy'] != nil and ENV['https_proxy'] != nil and ENV['no_proxy'] != nil
  if not Vagrant.has_plugin?('vagrant-proxyconf')
    system 'vagrant plugin install vagrant-proxyconf'
    raise 'vagrant-proxyconf was installed but it requires to execute again'
  end
  config.proxy.http     = ENV['http_proxy']
  config.proxy.https    = ENV['https_proxy']
  config.proxy.no_proxy = ENV['no_proxy']
end
```

Architecture

- **doc** – Provides documentation about the installation and usage of the tool.
- **etc** – Allows to persist configuration changes
- **lib** – Contains common development functions to setup and work with ONAP components.
- **opt** – Shared folder that synchronizes host and virtual machine source code changes.
- **tests** – Used to ensure correct functionality of the scripts located into *lib* folder.
- **tools** – Helper scripts to facilitate some daily tasks.
- **vagrant_utils** – Script connectors between *Vagrantfile* and *lib* folder.



Usage - Demo

$ ./tools/run.sh <app_name>

Conclusion

- This can be used as vehicle to standardize process and dependencies through an automated provisioning mechanism.
- Setup a development environment using only a single instruction.
- Adding this tool into a CI/CD pipeline can prevent any compilation failure in the future and guarantee building image process works any time.