DevOps with Vagrant and KVM/qemu

Hiroshi Miura (@miurahr)
NTT DATA Corporation.
Disclaimer

- Any product name, service name, software name and other marks are trade mark or registered mark of corresponding companies.

- This presentation is in a purpose of providing current information on emerging technologies and there is no grantee of correctness and/or persistence of features in any future.

- A presenter and NTT Data Corporation provide information in as-is basis and have no responsiveness for results that you got according to information in this presentation material.
Who am I?

- Production work
  - OpenStack SI team
  - Swift object storage

- OSS devel:
  - Vagrant-KVM
  - Linux Kernel
  - etc...

Github.com/nttdata-osscloud
Github.com/miurahr
Agenda

1. What is DevOps
2. Introduction to Vagrant
3. Vagrant-KVM
4. Infrastructure changes
5. Future
What is DevOps?

Mike Loukides, 2012, @Velocity conference

“If you're going to do operations reliably,
you need to make it reproducible and programmatic.”
What is DevOps?

• The nature of “operations”

  Cloud changes

• Growing distributed systems

• Software development practice

• “cooperation and collaboration” between dev and ops
DevOps for developer

• Gap between Development and Operations
  • Goals
  • Process and approach
  • Tools

• Infrastructure engineering becomes like a development.
  • Infrastructure as a code
  • Adoption of Vagrant
Introduction
To Vagrant
What is Vagrant

- development environment on VM, container or cloud
- same
  - among team members.
  - among production and development.
  - VirtualBox, AWS EC2
What is Vagrant

“Mature, stable, proven.”

by author,
Mr. Mitchell Hashimoto
Targeted Vagrant Users

• Web application developers
New Vagrant Users

• Infrastructure engineer programmer
• Deploy on VMwarevCenter, OpenStack
• Test Puppet, Chef or Ansible
Vagrant work flow

• Just run

$ vagrant up

It download “vagrant box” and start and provision guest OS.

• Config file: “Vagrantfile”
What is Vagrant box

- Preconfigured VM images
- Base for environment
- Community shared
- VagrantCloud.com: shares boxes
Brief history

• Launched at 2010
• 2013, March, Version 1.1+
  • Plugins: 3rd party projects
  • Vagrant-KVM plugin start

• 2014, March
  • Vagrant Cloud
Vagrant provisioner

- Shell, Puppet, Ansible, Docker and Chef
- 3rd party: Salt etc.
Vagrant Plugins

- Supported plugins from 1.1+ (2013.2~)
  - Provider: Driver for VM monitor
  - Provisioner: Configure guest images
  - Synced_folder: folder sync between guest/host

Now we can make Vagrant work on my Linux and KVM!
Supported Providers

- Many provider plugins

LXC
HPCS
Sakura Cloud
VMWare

KVM
Google Compute Engine
DigitalOcean
NIFTY Cloud
IIJ GIO

cloudstack
Parallels
openstack

SOFTLAYER
VMware
VMWare

Copyright © 2014 NTT DATA Corporation
Current status of Vagrant-KVM
Vagrant-KVM

Usage: Vagrant provider plugin to utilize KVM

Author: Alexandre Drahon, UK

License: MIT

Language: Ruby

Libraries: Ruby-libvirt, libvirt, qemu and kvm
Current Status of Vagrant-KVM

• V0.1.8 March, 2014
  • Basic Vagrant features(private nw, NFS, provisioning, GUI, customize)
  • Plan9 host file share
  • QEMU 1.1 – 1.7

• V0.2.0 will come May, 2014
  • Multiple guests and networks
  • Bridged network
  • Address conflict resolver
Vagrant-KVM vs. -libvirt

- **kvm plugin:** simple, single host
- **libvirt plugin:** multi-host, multi-vmm

<table>
<thead>
<tr>
<th>Features</th>
<th>Vagrant-KVM</th>
<th>Vagrant-libvirt</th>
</tr>
</thead>
<tbody>
<tr>
<td>KVM</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Xen</td>
<td>N.A.</td>
<td>Plan</td>
</tr>
<tr>
<td>Remote</td>
<td>N.A.</td>
<td>Experimental</td>
</tr>
<tr>
<td>Multiple guests</td>
<td>Yes</td>
<td>Plan</td>
</tr>
<tr>
<td>File share</td>
<td>Plan9, NFS</td>
<td>NFS, rsync</td>
</tr>
<tr>
<td>Snapshot</td>
<td>Yes(sahara)</td>
<td>Yes(sahara)</td>
</tr>
<tr>
<td>Image convert</td>
<td>Yes(mutate)</td>
<td>Yes(mutate)</td>
</tr>
<tr>
<td>Multiple arch</td>
<td>Plan</td>
<td>N.A.</td>
</tr>
<tr>
<td>Current Version</td>
<td>0.1.8</td>
<td>0.0.16</td>
</tr>
</tbody>
</table>
## Dependencies

- There are small amount of dependency.

<table>
<thead>
<tr>
<th>Name</th>
<th>Usage</th>
<th>Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>libvirt</td>
<td>Virtualization abstraction library</td>
<td>Ruby, C</td>
</tr>
<tr>
<td>Ruby-libvirt</td>
<td>Ruby bridge to libvirt</td>
<td>C, C++</td>
</tr>
<tr>
<td>Qemu</td>
<td>Virtual Machine emulator</td>
<td>C</td>
</tr>
<tr>
<td>rspec/mock</td>
<td>Test framework</td>
<td>Ruby</td>
</tr>
</tbody>
</table>
Quality and Test

- Rspec to test the plugins
- Test vagrant-kvm in guest OS on KVM prepared by vagrant-kvm.
Quality and Test

- CI: Github/Travis-CI combination
  - Github.com: development platform
  - Travis-CI: test automation platform
  - RVM: Ruby Virtual Machines prepare versions of ruby
OSS and CI

• Every commit/patch and PRs are tested with Travis-CI and Rspec.

Github.com
Pull-Request example: Green
OSS and CI

- Every commit/patch and PRs are tested with Travis-CI and Rspec.
## Issues fixed

There are many issues to be fixed in related projects.

<table>
<thead>
<tr>
<th>No</th>
<th>Component</th>
<th>Problem</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>libvirt</td>
<td>security treatment for plan9 file share</td>
<td>Dynamic AppArmor policy update</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>External SELinux label control work around</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Dynamic permission control not to restore permission</td>
<td>Lack of restore function in libvirt. Work around in a plugin.</td>
</tr>
<tr>
<td>2</td>
<td>Ruby-libvirt</td>
<td>Not updated 2 yrs. Unsupported new API</td>
<td>Push developer to update. Release 0.5.x in Dec. 2013</td>
</tr>
<tr>
<td>3</td>
<td>Linux kernel</td>
<td>Fail to configure AppArmor rule</td>
<td>Wait kernel update</td>
</tr>
<tr>
<td>4</td>
<td>CentOS6</td>
<td>Unsupported plan9fs (not configured in kernel)</td>
<td>Provide special VM image.</td>
</tr>
</tbody>
</table>
Vagrant-kvm: Future plan

- Support full features of Vagrant
- Multiple architecture (ARM)
- Linux kernel debug support
Change the way: infrastructure
New trend on DevOps

- Infrastructure development is Changing

- New technology comes on Infrastructure Engineering / development
Blue Green Deployment

- No change to production environment
- New/Change for environment => create new instance and switch it

Before deploy

Router

Active

Application Server Ver 1

Application Server Ver 1

Deployment

Application Server Ver2

Database Ver2

Switch back when fail

Router

Direct Green

Active

Application Server Ver2

Application Server Ver 1

Application Server Ver 1

Database Ver2

http://www.nttdata.com/jp/ja/insights/trend_keyword/2013122601.html
Packaging for deploy

• Application container help agile deployment of Application

Fresh deployment everyday

Temporaty configuration

difference

Update master

Same environment deployment

Bare Metal
Virtual Machines

Production

Instance To run

Linux Container

Development

Master Image

Application container(Packer) technology
Packer

- [http://www.packer.io/](http://www.packer.io/)
- Create virtual machine image for several kind of cloud / virtualization infrastructures.
- Use JSON as profile description language.
- Infra can be managed as same as source code.
Packer

Usage: Multi platform VM image build tool

Origin: HashiCorp

License: Mozilla Public License Version 2.0

Language: Go

Single definition made multiple images for platforms

Definition File (JSON)

packer

Provisioning tool

Virtual BOX

- Amazon Web Services
- Docker
- OpenStack
- DigitalOcean
- VMware
- Google Compute Engine
Docker

• Portable container for Application

Agility:
Low overhead and quick setup

Portability:
dependency packed into container image.
Vagrant and Docker

• **Prepare Docker environment on Vagrant**
  • Auto detect guest OS and install proper packages.
  • **Start Docker environment only one line command.**

Vagrant 1.5+ Provisioner

Vagrant 1.6+ Provisioner/Provider
Stack of DevOps infrastructure

Vagrant enables control over infrastructure tool sets

- Container images
- Provisioner (Puppet, Chef, Docker)
- Box Images (Fedora, Ubuntu, etc)
- Provider (VirtualBox, KVM, LXC, Libvirt, AWS)
- Command (Snapshot mutate)
- Host OS/Hypervisor (VB, VMware, KVM)
Work flow example (1/2)

Creation of Guest Images

- OS package repository
- RubyGems
- Pypi, etc

Recipe

Puppet

Shell

PACKER

LXC image

Docker template

VM image

docker repository

Vagrant box repository
Work flow example (2/2)

Test on local Virtual Machines

Then run on production
Future
Future of Vagrant/vagrant-kvm

- Embedded development with ARM support
- Various cloud and private platform support
- Packer/Docker integration
- Test automation
Contact

miurahr at nttdata co jp
@miurahr (Twitter/LinkedIn/Github)