Deploying MySQL HA

with Ansible and Vagrant

Daniel Guzman Burgos (Percona)
Robert Barabas (Percona)
2015-04-13

Agenda

- Introductions
- Environment Setup
  - Virtual Machines
  - Git
  - Ansible
- Ansible Insights
- Build an Ansible repo

Introductions

- Daniel Guzman Burgos
    - daniel.guzman.burgos(at)percona.com
    - longest email address in percona!
- Robert Barabas
    - robert.barabas(at)percona.com

Link to the Tutorial

- https://github.com/robertbarabas/ansible-tutorial
  - Homepage for this session
  - git clone http://bit.ly/1CvbJ9H
- tutorial/
  - Markdown pages with instructions
- demo/
  - Final Ansible repository

Before we begin...

- Create directory for your new project
  - mkdir demo/
- If you get lost or cannot see something:
  - check out our repo!
    - *detailed instructions (tutorial/)*
    - *complete files (demo/)*

Virtual Machine Setup

- $REPO/tutorial/$TREE/vm_setup.md
- Install
  - VirtualBox
  - Vagrant
- Configure
  - Vagrantfile

Vagrantfile

```ruby
# -*- mode: ruby -*-
# vi: set ft=ruby :
Vagrant.configure("2") do |config|
    config.vm.box = "perconajayj/centos-x86_64"
    # Master
    config.vm.define "master" do |master|
        master.vm.network "private_network", ip: "192.168.10.100"
        master.vm.hostname = "master"
    end
    # Slave
    config.vm.define "slave" do |slave|
        slave.vm.network "private_network", ip: "192.168.10.101"
        slave.vm.hostname = "slave"
    end
end
```

PERCONA LIVE

Test VMs

- Start all VMs ("master" and "slave")
  - vagrant up
- Stop VM "slave"
  - vagrant halt slave
- Check status of VM "slave"
  - vagrant status slave
- Remove VM "slave"
  - vagrant destroy slave

Git Setup

- $REPO/tutorial/$TREE/git_setup.md
- Install
  - Git
- Configure
  - .gitconfig

.gitconfig

- git config --global user.name “…”
- git config --global user.email “…”
- cat ~/.gitconfig

```
[user]
   name = Robert Barabas
   email = robert.barabas@example.com
```

Ansible Setup

- $REPO/tutorial/$TREE/ansible_setup.md
- Install
  - Ansible
- Configure
  - ansible.cfg (to be configured later)

Ansible Insights - About

- Automation tool
- Written in python
- Agentless (plain SSH or python)
- Idempotent
- Easy to learn
- Relatively new (2012)
- Supports *NIX primarily (Windows: >1.7)

Ansible Insights - History

- 1993 - CF Engine v1
- 2005 - Puppet, Capistrano
- 2007 - Vlad the Deployer
- 2009 - Chef
- 2010 - Vagrant
- 2011 - Salt, Fabric
- 2012 - Ansible

Ansible Insights - Terminology

- Management Workstation
  - *NIX machine
  - Some extra requirements
- Managed Node
  - Where the magic happens!

Ansible Insights - Terminology

- Inventory
  - definition of host groups
  - common settings for hosts
  - can be extended and/or dynamically generated

Ansible Insights - Terminology

- Playbook
  - top level “plan”
  - tasks that run against a group of hosts

Ansible Insights - Terminology

- Tasks
  - the actual steps that execute
  - execute sequentially
  - idempotent
  - can use “facts” to make smart decisions
  - leverage modules to get the job done

Ansible Insights - Terminology

- Modules
  - basic building blocks of Ansible
  - execute actions
  - programmable

Ansible Insights - Terminology

- Roles
  - means to code reuse
  - abstract set of tasks

Ansible Insights - Operation Modes

- Operation Modes
  - Push
    - Run play on Management Workstation
  - Pull
    - remote git repo
    - cron job executes play(s) locally

Ansible Insights - Requirements

- SSH
  - OpenSSH or Paramiko
  - Access, permissions
    - Deploy user vs. operating user

Ansible Insights - Requirements

- Git
  - Remote repository for Pull Mode
  - Local repo on Management Workstation

Ansible Insights - Requirements

- Python
    - Already installed most of the time (LSB)
    - Management Workstation (>2.6)
    - Managed Hosts (>2.4)

Ansible Insights - Requirements

- Additional Python modules
  - python-simplejson (python 2.4)
  - libselinux-python (for SELinux management)

Ansible Insights - Simple inventory

- cat local

```
[localhost]
127.0.0.1 ansible_connection=local
```

Ansible Insights - Basic commands

- ansible -i local -m setup localhost
  - shows “facts” for the machine

Ansible Insights - Basic commands

- ansible -i local -m ping localhost
  - validates connection

Ansible Insights - Basic commands

- ansible -i local -a uptime localhost
  - hidden / implicit command module (-m command)
  - runs “uptime” command on machine

Ansible Insights - Simple play

- cat uptime.yml

```
---
- name: Show uptime
  hosts: localhost
  tasks:
    - name: run uptime
      shell: uptime
      register: uptime
    - name: show uptime
      debug: var=uptime
```

Ansible Insights - Basic commands

- ansible-play -i local uptime.yml
  - runs tasks to register and show uptime

Ansible Insights - Configuration

- Per system
  - /etc/ansible.cfg
- Per user
  - ~/ansible.cfg
- Per “project” (exec dir)
  - ${PROJECT_HOME}/ansible.cfg

Ansible Insights - Simple configuration

- cat ansible.cfg

```
[defaults]
hostfile = local
```

Ansible Insights - Using configuration

- Now rerun previous commands *without* “-i local”
  - ansible -m ping localhost
  - ansible -m setup localhost
  - ansible -a uptime localhost
  - ansible-play uptime.yml

Initialize Ansible Repository

- Prereqs
  - cd demo/
  - ls -la Vagrantfile
- Initialize Git repo in *your* project directory
  - git init .

Create inventory file

- cat hosts

```
[master]
192.168.10.100

[slave]
192.168.10.101

[all:children]
master
slave
```

Ensure VMs are running

- vagrant up
- vagrant status

```
Current machine states:

master                    running (virtualbox)
slave                     running (virtualbox)

This environment represents multiple VMs. The VMs are all listed
above with their current state. For more information about a specific
VM, run `vagrant status NAME`.
```

Test connectivity (long)

- ansible -u root -i hosts -m ping \ --private-key=~/. vagrant.d/insecure_private_key all

```
192.168.10.100 | success >> {
    "changed": false,
    "ping": "pong"
}

192.168.10.101 | success >> {
    "changed": false,
    "ping": "pong"
}
```

Setup Ansible config

- cat ansible.cfg

```
[defaults]
remote_user = root
private_key_file = ~/.vagrant.d/insecure_private_key
hostfile = hosts
```

Test connectivity (short)

- ansible -m ping all

```
192.168.10.100 | success >> {
    "changed": false,
    "ping": "pong"
}

192.168.10.101 | success >> {
    "changed": false,
    "ping": "pong"
}
```

Create playbook for OS setup

- plays/setup_os.yml
  - modules used: yum, service, selinux
  - iteration
  - no interaction between tasks
  - leverages the idea of idempotency

Create playbook for MySQL setup

- plays/setup_mysql.yml
  - some interaction between tasks
  - uses includes, facts, asserts, handlers, filters!
  - new modules: template, shell
  - references additional files (templates, includes)

Promoting reusability with roles

- New play
  - plays/setup_server.yml
- New roles
  - roles/os/
  - roles/mysql/

Create playbook for database load

- New play
  - plays/setup_sakila.yml
- Features
  - new modules
    - get_url
    - unarchive

Create playbook for replication setup

- New play
  - plays/clone_mysql.yml
- Features
  - async
  - new modules
    - file

Putting the pieces together

- New play
  - site.yml
- Features
  - includes
    - plays/setup_server.yml
    - plays/setup_sakila.yml
    - plays/clone_mysql.yml

Questions?

- ???