Managing Mesos, Docker, and Chronos with Puppet

Roger Ignazio – Puppet Labs, Inc.
Niklas Quarfot Nielsen – Mesosphere, Inc.

ContainerCon 2015 – Seattle, WA
Who We Are

Roger Ignazio
QE Automation Engineer, Puppet Labs
@rogerignazio

Niklas Quarfot Nielsen
Distributed Systems Engineer, Mesosphere
@quarfot
Mesos In Action

mesosinaction.com

Code: cftwmesos
Agenda

- Deploying a Mesos cluster
- Building a Docker image
- Creating a Chronos job
- Demo
- Provisioning infrastructure
- Q & A
Audience Poll
About Mesos, Docker, Chronos

- **Mesos**
  - Represent many machines as a single entity
  - Advertise resources directly to applications

- **Docker**
  - Easily package and deploy apps and dependencies
  - Analogous to VMs, but minus the overhead

- **Chronos**
  - Distributed, highly available cron for Mesos
  - Run scheduled tasks in containers, incl. Docker
About Puppet

- Declare *desired state* for your infrastructure
- Wide range of OS support
- Idempotent
- Extensible – custom facts, types, providers
- Open source – Apache License, version 2
About Puppet

- Used by 25k+ companies worldwide
- 3,400+ modules available via Puppet Forge
- Puppet Enterprise
  - Flagship commercial product from Puppet Labs, Inc.
Intro to Puppet

[ -f /etc/debian_version ] && package_manager="apt-get"
[ -f /etc/redhat-release ] && package_manager="yum"

$package_manager install mesos

if [ $? == 0 ]; then
echo "zk://10.100.42.16:2181/mesos" > /etc/mesos/zk
service mesos-master restart
fi
Intro to Puppet

[ -f /etc/debian_version ] && package_manager="apt-get"
[ -f /etc/redhat-release ] && package_manager="yum"

$package_manager install mesos

if [ $? == 0 ]; then
echo "zk://10.100.42.16:2181/mesos" > /etc/mesos/zk
service mesos-master restart
fi

But what happens if we run this twice?
package { 'mesos':
    ensure => installed,
}

file { '/etc/mesos/zk':
    ensure => file,
    content => 'zk://10.100.42.16:2181/mesos',
    notify => Service['mesos-master'],
    require => Package['mesos'],
}

service { 'mesos-master':
    ensure => running,
}
“There’s no future in which there are fewer servers [or] fewer services, and there’s no future in which those services are less important.”

– Luke Kanies
Founder and CEO, Puppet Labs
Puppet’s Role

If Mesos is the abstraction layer for your applications, Puppet is the abstraction layer for infrastructure management
Puppet’s Role

If Mesos is the abstraction layer for your applications, Puppet is the abstraction layer for infrastructure management

But it’s also more...
Puppet’s Role

Custom types and providers can interact with external services (e.g. AWS, Chronos, ...)
Deploying Mesos, Docker, and Chronos with Puppet
Deployment Overview

- Install/configure Mesos, ZooKeeper, Docker
- Stage a Docker image on the Mesos slaves
- Install and configure Chronos
- Create a Chronos job
  - that runs in a Docker container
Deployment Overview

- Chronos
- ZK
- Mesos Master
  - Mesos Slave
    - Docker
  - Mesos Slave
    - Docker
  - Mesos Slave
    - Docker
Deployment Overview

- Intro to Puppet’s roles/profiles pattern
- Using the following Puppet modules
  - deric/mesos
  - deric/zookeeper
  - garethr/docker
  - puppetlabs/chronos

All of these modules are open source and available via the Puppet Forge: https://forge.puppetlabs.com
Deploying Mesos and ZooKeeper

- The deric/mesos module will
  1) Install/configure Mesosphere’s package repos
  2) Install the Mesos package
  3) Configure and start services

- The deric/zookeeper module will
  1) Install/configure Cloudera’s package repos
  2) Install the ZooKeeper package
  3) Configure and start services
Deploying Mesos (Master)

class role::mesos::master {
  include profile::base
  include profile::chronos
  include profile::mesos::master
  include profile::zookeeper
}

Deploying Mesos (Master)

class profile::mesos::master {
    include profile::mesos::common

    class { '::mesos::master':
        listen_address => $::ipaddress_eth0,
        work_dir => '/var/lib/mesos',
        options => {
            log_dir => '/var/log/mesos',
            quorum => '1',
        },
    }
}
Deploying ZooKeeper

class profile::zookeeper {
    include java

    class { '::zookeeper':
        client_ip => $::ipaddress_eth0,
        id => '1',
        repo => 'cloudera',
        require => Class['java'],
    }
}
Deploying Mesos (Slave)

```ruby
class role::mesos::slave {
  include profile::base
  include profile::docker
  include profile::mesos::slave
}
```
Deploying Mesos (Slave)

class profile::mesos::slave {
  include profile::mesos::common

  class { '::mesos::slave':
    listen_address => '::ipaddress_eth0',
    work_dir => '/var/lib/mesos',
    options => {
      log_dir => '/var/log/mesos',
    },
  }
}
Deploying Mesos (Common)

class profile::mesos::common {
    class {
        '::mesos':
        repo => 'mesosphere',
        zookeeper => 'zk://192.168.248.10:2181/mesos',
    }
}
Deploying Docker

● The garethr/docker module will
  1) Install and configure Docker
  2) Start services
  3) Stage Docker images for later use
Deploying Docker

# Call the garethr/docker module
include ::docker

# Reconfigure the Mesos slave
class { '::mesos::slave':
  ...
  options => {
    containerizers => 'docker,mesos',
    isolation => 'cgroups/cpu,cgroups/mem',
    executor_registration_timeout => '5mins',
  },
}
Deploying Chronos

- The puppetlabs/chronos module will
  1) Install and configure Chronos
  2) Start services
  3) Install custom type/provider chronos_job
Deploying Chronos

class profile::chronos {

    # Call the puppetlabs/chronos module
    include ::chronos

    ...

}
Building Docker Images with Puppet
Building Docker Images

Two approaches:

- **puppet agent** – pre-shared key to use existing Puppet infra when building the images
- **puppet apply** – directly apply manifests when building the images
Building Docker Images

Two approaches:

- **puppet agent** – pre-shared key to use existing Puppet infra when building the images
- **puppet apply** – directly apply manifests when building the images
Building Docker Images

FROM debian:wheezy

MAINTAINER Roger Ignazio <roger@puppetlabs.com>

WORKDIR /tmp

RUN curl -L -O -s https://apt.puppetlabs.com/puppetlabs-release-wheezy.deb
RUN dpkg -i puppetlabs-release-wheezy.deb
RUN apt-get update
RUN apt-get -q -y install puppet

COPY * ./

RUN puppet apply example.pp
Building Docker Images

```yaml
package { ['ruby', 'ruby-dev', 'build-essential']:
    ensure => installed,
}

package { 'httparty':
    ensure => installed,
    provider => gem,
}

file { '/usr/bin/query_mesos':
    ensure => file,
    mode   => '0755',
    source => '/tmp/query_mesos.rb',
}
```
Building Docker Images

Step 10: RUN puppet apply example.pp
    ---> Running in 12eda5e24ff8
Notice: Compiled catalog for 90c88c41cdaa.bad in environment production in 0.16 seconds
Notice: Package[build-essential]/ensure: ensure changed 'purged' to 'present'
Notice: File[/usr/bin/query_mesos]/ensure: defined content as '{md5}
e44268ac8e31f75f1aeee961d0ebe36b'
Notice: Package[ruby-dev]/ensure: ensure changed 'purged' to 'present'
Notice: Package[httparty]/ensure: created
Notice: Finished catalog run in 33.22 seconds
    ---> 1a8fefd724ee
Removing intermediate container 12eda5e24ff8
Successfully built 1a8fefd724ee
Staging Docker Images on Slaves

Using the garethr/docker Puppet module

```puppet
docker::image { 'rogerignazio/basic-puppet-example':
  image_tag => 'latest',
}
```

Equivalent to

```
$ docker pull rogerignazio/basic-puppet-example:latest
```
Creating a Chronos Job
Creating a Chronos Job

```json
chronos_job { 'query_mesos':
    command => 'query_mesos 192.168.248.10',
    job_schedule => 'R/2015-06-29T00:00:00.000Z/PT60s',
    container => {
        type => 'DOCKER',
        image => 'rogerignazio/basic-puppet-example',
    },
    cpus => 1.0,
    mem => 256,
    owner => 'roger@puppetlabs.com',
}
```
Demo

rji/containercon-2015-demo
Provisioning Infrastructure
Cloud Provisioning with AWS

- Declare AWS infrastructure as Puppet code
- Custom types and providers hit the AWS API
  - Ensures resources are in desired state
Cloud Provisioning with AWS

```yaml
ec2_instance {'mesos-slave-NN'}:
    ensure => present,
    region => 'us-west-2',
    image_id => 'ami-4dbf9e7d', # AWS RHEL 7.1 image
    instance_type => 'c4.xlarge', # 4 CPUs, 7.5 GB mem
    security_groups => ['mesos-aws-secgrp'],
```
Cloud Provisioning with AWS

- *Some* of the available resource types:
  - ec2_instance
  - ec2_securitygroup
  - ec2_vpc
  - elb_loadbalancer
  - route53_a_record

- A more complete example
Razor: Bare-metal Provisioning

- Auto-discover inventory
- Policy-based provisioning
- Pluggable “brokers”
- Razor is open source – Apache License, v2
Razor: Bare-metal Provisioning

- PXE Boot
- Razor Microkernel

- Ubuntu ISO
- RHEL ISO

Razor Server

Policy (Facts, IPMI, LLDP)

Puppet Master

role::mesos::master
role::mesos::slave
...

Razor Client (Admin)
Razor: Bare-metal Provisioning

For more information, check out

Q & A

Roger Ignazio – @rogerignazio
Niklas Quarfot Nielsen – @quarfot