Integrating the Cloud with Puppet
About me:

Dan Bode
Some Dude at PuppetLabs

@bodepd

bodepd <on> freenode
Who is this talk for?

Cloud Users

Puppet beginners
It will cover

why integrate?

explanation of Puppet’s architecture as it applies to integration

using Puppet to model VM instances
Why Integrate?
Cloud

Provisions virtual machines

deployVirtualMachine

Self Service API

VM1
Puppet

VMs -> Applications

deployApacheServer

Self Service API

VM1

Make me an apache server

Here are your instructions

Puppet Master

Tuesday, February 26, 13
Together

PaaS

deployAppStack

Self Service API

DB1  Apache1  Apache2  LB
Puppet
2 run modes

puppet apply

client/server
Puppet Client/Server

- Classifier
- Master
- VM1
- Modules
- Facts
- Catalog
Facter

- Classifier
- Modules
- Master
- Facts
- Catalog
- VM1
Facter

$ facter

architecture => x86_64
domain => local
fqdn => DansLapTop.local
id => danbode
ec2_instance_id => abc123abc123abc123
operatingsystem => ‘Ubunbtu’
osfamily => ‘Debian’
......
Facter

Available as top scope variables from manifests

ie: $::fact_name

Creating custom facts is easy.
Modules

Sharable Puppet content
Module Forge

http://forge.puppetlabs.com/puppetlabs/apache
Classes/defines compose resources
Resources

Describe the configuration state of individual system elements.
user { 'dan': # a user named dan
  ...

user { 'dan': 
  ensure => present,
  # a user named dan
  # should exist
  ...
user { 'dan':                # a user named dan
  ensure => present,       # should exist
  shell => '/bin/bash',    # with this shell
}
Puppet DSL and resources
Puppet DSL

Composes collections of resources.
class webserver {
  package { 'apache2': ... }
  file { '/etc/apache2/apache2.conf':
    ...
    require => Package['apache2'],
  }
  service { 'apache2':
    ...
    subscribe => File['/etc/apache2/apache2.conf']
  }
}
configure a node

include webserver
Classification (maps roles as classes)
Site manifest

(/etc/puppet/manifests/site.pp)

Map a host’s certname to content from a module

node /^my_node/ {
    include apache
}

Tuesday, February 26, 13
The master can call out to arbitrary executables to figure out how a node should be classified.
Puppet Client/Server

Classifer → Master

VM1

Catalog

Modules

Facts
Integration is all about Classification
Using metadata/userdata

deployApacheServer (with metadata=’puppet_class=apache’)
Using metadata/userdata

deployApacheServer (with metadata='puppet_class=apache')

Self Service API

VM1

I was provisioned with metadata puppet_class=apache

Puppet Master

Tuesday, February 26, 13
Using metadata/userdata

deployApacheServer (with metadata='puppet_class=apache')

Self Service API

VM1

Puppet Master

Oh cool! You must be an apache server

I was provisioned with metadata puppet_class=apache
Determine role based on facts

deployVirtualMachine (with metadata)
Determine role based on facts

- deployVirtualMachine (with metadata)
- populate facter metadata service
Determine role based on facts

- deployVirtualMachine (with metadata)
- populate facter metadata service
- use fact for classification

node default {
  include $::meta_data_role
}

Tuesday, February 26, 13
Pros

- simple

- classification information set during provisioning process
Cons

- hosts become authoritative over their role
- a single rooted host can pretend to be anyone else
- metadata/userdata is not always read/write
Using instance annotation data

deployApacheServer (with group='apache')

Self Service API

VM1

Puppet Master

here is my id

Let me consult the cloud system

You were provisioned as an apache server

You were provisioned as an apache server
Using instance annotation data

deployApacheServer (with group='apache')

Self Service API

VM1
Using instance annotation data

deployApacheServer (with group='apache')

Self Service API

here is my id

VM1

Puppet Master
Using instance annotation data

deployApacheServer (with group='apache')

Self Service API

VM1

here is my id

Puppet Master

Let me lookup your role based on your id
Using instance annotation data

deployApacheServer (with group=‘apache’)

Self Service API

here is my id

VM1

Puppet Master

Let me lookup your role based on your id

You were provisioned as an apache server

Tuesday, February 26, 13
Pros

- provisioning credentials are used to determine role
- annotation field likely updatable
Cons

- puppetmaster must have API credentials
- may require a custom ENC
Decouple role assignment from provisioning

After provisioning is completed, ssh into a machine, set a custom fact (using facts.d), and trigger a puppet run.

pros - you can easily execute a script to install and bootstrap puppet

cons - extra step
facts.d

facts.d comes with stdlib
(http://forge.puppetlabs.com/puppetlabs/stdlib)

it converts any ‘key=value’ pairs listed in /etc/facts.d/*/*.txt into facts
VM provisioning with Puppet (experimental! use cases appreciated)
class my_app_stack {
  cloudstack_instance { 'foo4':
    ensure => present,
    group => 'role=db',
  }
  cloudstack_instance { 'foo3':
    ensure => present,
    group => 'role=apache',
  }
}
Use resource defaults for common settings

Cloudstack_instance {
    image => 'CentOS 5.6 key+pass',
    flavor => 'Small Instance',
    zone => 'ACS-FMT-001',
    network => 'puppetlabs-network',
    keypair => 'dans_keypair4',
}

cloudstack_instance { 'foo4':
    ensure => $::ensure,
    group => 'role=db',
}

cloudstack_instance { 'foo3':
    ensure => $::ensure,
    group => 'role=apache',
}

Tuesday, February 26, 13
More issues of trust