Automating CloudStack with Puppet

Puppet Camp Silicon Valley
David Nalley
david@gnsa.us
@ke4qqq
#whoami

- Recovering sysadmin
- Committer on Apache CloudStack
- Fedora Project Contributor
- Fan of “The Phoenix Project”
The plan

- Overview of Apache CloudStack
- Using puppet to manage your CloudStack-based VMs
- Using puppet to manage your VM deployment.
What is CloudStack?

- Open source IaaS platform
- ASLv2 licensed
- History tl;dr
  - began development in 2008
  - production deployments by 2009
  - open sourced in 2010
  - moved to ASF in 2012
Design goals

- Integrate with untold number of yet to be identified hardware.
- Provide an API platform on which to run cloud operations.
- Orchestrate hardware resources that may be protected by a firewall.
- Horizontally scalable management layer.
- Enable the best data paths to accomplish cloud operations.
- A beautiful and functional UI
Architectural Overview

- Division of physical resources
- Storage
- Borg drones VMs
- Networking
- Management and orchestration
Physical hosts

- Hypervisors
  - KVM
  - Xenserver
  - XCP
  - VMware
- Baremetal (with IPMI)
Clusters

- Collections of hosts
- Typically 1-15 hosts in a cluster
- Homogenity
  - Network
  - Hypervisor
  - CPU type
Clusters

- Hosts share storage
- Fault domain for individual VM availability
- Lowest level for allocation decisions
Pods

- Collection of clusters
- Typically a rack or row of racks
- Can contain multiple types of hypervisors
- Largely just an arbitrary division
Zones

- Typically a datacenter
- Single networking model within a zone
- Visible to the end user
Storage

- CloudStack doesn't really provide storage, but does consume and orchestrate it.
Local Storage

- Typically faster than SAN/NAS
- Failure of a host means loss of a VM
- Can be far more scalable than trying to scale a large traditional storage platform
Primary (shared) storage

- Shared at the cluster level
- Where running disk images live
- All hosts in the cluster can write to the resource
- Most commonly NFS and iSCSI, but essentially anything the hypervisor can mount
- 'New' storage types like Ceph RBD
Secondary Storage

- Primary storage is focused on running VMs, Secondary storage is focused on immutable items.
  - Snapshots
  - Disk images
  - ISOs
- Zone wide storage resource
- Can employ object storage
# Borg drone VMs

<table>
<thead>
<tr>
<th>Labor</th>
<th>Transport</th>
<th>Maintenance</th>
<th>Tactical</th>
<th>Assimilation</th>
<th>Medical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Two regular arms</td>
<td>Two regular arms</td>
<td>One mech arm</td>
<td>Heavy disruptor arm</td>
<td>Two long mech arms</td>
<td>Silver mech arm</td>
</tr>
<tr>
<td>Med gray</td>
<td>Med gray</td>
<td>DK gray upper body</td>
<td>Dark and bulky</td>
<td>Dk gray upper body</td>
<td>Lt fleshy plates</td>
</tr>
<tr>
<td>Open abdomen</td>
<td>Light shoulders</td>
<td>Lt gray thighs</td>
<td>Shoulder armature</td>
<td>Lt gray arms and thighs</td>
<td>Slim</td>
</tr>
<tr>
<td>Slim</td>
<td>Backpack</td>
<td>Backpack</td>
<td>Heavy armor plates</td>
<td>Bulky upper body</td>
<td>Blue lights</td>
</tr>
<tr>
<td>No lights</td>
<td>No lights</td>
<td>Red lights</td>
<td>Green lights</td>
<td>Yellow lights</td>
<td></td>
</tr>
</tbody>
</table>
Console Proxy VM

- AJAX-based VNC console access
- Allows CloudStack to deal with auth\{n,z\} for console access.
- Abstracts away hypervisor access
- Not a replacement for ssh/RDP but no more painful than DRAC/iLO
- Stateless; horizontally scalable
Secondary Storage VM

- Secondary Storage is the resource, the SSVM that handles the following operations:
  - Copying snapshots from primary to secondary storage
  - Copying disk images from secondary to primary storage
  - Making all items stored in secondary storage downloadable and a place to transfer items into secondary storage
  - Aging the snapshots according to policy
Networking Model: VLANs

- Traditional L2 isolation
- CloudStack given a block of VLANs and allocates them on demand
- Each account gets allocated at least one VLAN.
- Inherent limitations of VLANs
Networking model: L3 Isolation

- L3 isolation; aka Security Groups
- Pushes ACLs down to each hypervisor host
- Far more scalable, decentralized (more Borg)
- Filter at the bridge device
Security Groups
Network Model: SDN

- OVS (GRE overlay tunnels)
- Nicira NVP
- Others rapidly appearing:
  - BigSwitch
  - Midokura
Virtual networking hardware

- DHCP
- VLAN allocation
- Firewall
- NAT/Port forwarding
- Routing
- VPN
- Load Balancing
Virtual networking hardware

- Cisco Nexus 1000v
- NetScaler VPX
- F5 Big IP virtual edition
Physical networking hardware

- Juniper SRX
- F5 BigIP LB
- NetScaler
Management Server

- Management server is stateless, horizontally scalable platform for orchestrating all of the resources.
- Provides isolation in what is assumed to be a multi-tenant environment
API

- EC2/S3 translation layer
- CloudStack native API:
  http://incubator.apache.org/cloudstack/docs/api
More info

- http://incubator.apache.org/cloudstack
- #cloudstack on irc.freenode.net
- cloudstack-users-subscribe@incubator.apache.org
Using puppet to manage VMs

- Being able to deploy 500 VMs in 10 minutes means you need some method to classify and apply configuration management.
- Most of the work for this awesomeness was done by Jason Hancock (@jsnby)
A couple of upfront goals

- Minimize the number of templates
- Have all instances receive config via Puppet
- Zero manual intervention
Make one API call to launch a VM, and get out of the way and watch the automation do wonderful things.
A word about auto-signing

- You can use auto-signing.
- Automatically signs any cert from a given domain
- Potential security issues if folks can connect to your puppetmaster
- You can pre-seed templates with a signed key - but there are gotchas
Run puppet ASAP

- Turn off splay - you want to minimize the time that the box remains unconfigured.
- Make sure puppet is configured to start on boot (enable the service, not cron)
Classifying nodes - options

- {site,node}.pp
- hostname-based regex
- PE/Dashboard
- ENC
- facts
- $other_things
facts
What to base a fact on...

http://incubator.apache.org/cloudstack/docs/api/apidocs-4.0.0/user/deployVirtualMachine.html

| userdata          | an optional binary data that can be sent to the virtual machine upon a successful deployment. This binary data must be base64 encoded before adding it to the request. Currently only HTTP GET is supported. Using HTTP GET (via querystring), you can send up to 2KB of data after base64 encoding. |
Sample userdata

role=webserver
location=datacenter1
environment=production
Custom fact for userdata

- http://s.apache.org/acs_userdata
Implementing ::role in puppet

Everyone is a default node.
No need to add nodes to site.pp or use an ENC.
import 'base'

node default {
  include base
}

Sample, skeletonized, site.pp
class base.pp {
  # Includes that apply to all machines

  case $::role {
    'somerole': {
      include somemodule
    }
    'otherrole': {
      include someothermodule
    }
  }
}
This is only the beginning...

- Setting environment, purging terminated instances, and more.
- Check out Jason's blog: http://geek.jasonhancock.com
Making puppet deploy your infrastructure in CloudStack

- Most of the real work that follows was done by Dan Bode.
deployVirtualMachine API

- CloudStack provides an API for provisioning machines.
Puppet...

- converts freshly provisioned VMs into functional machines ready to do work.
When combined....

you can start from nothing, deploy the machines and wind up with a completely automated deployment system.
CloudStack resources in puppet

https://github.com/bodepd/cloudstack_resources

Still a bit raw....potentially unstable, use with caution, file bug reports and patches.
Defining application stacks

class my_app_stack {
  cloudstack_instance {'web1':
    ensure => present,
    group  => 'role=web',
  }
  cloudstack_instance {'db1':
    ensure => present,
    group  => 'role=db',
  }
}
Setting defaults

Cloudstack_instance {
  image => 'Fedora18_x86_64',
  flavor => 'm1.medium',
  zone => 'SanJose',
  network => 'default own',
  keypair => 'my_secret_keypair',
}

cloudstack_instance { 'web1':
  ensure => $::ensure,
  group => 'role=web',}
Now machines and their configuration are deployable all from puppet.

(This also exists for GCE and OpenNebula.)