Building Scaleable Cloud Infrastructure using the Red Hat OpenStack Platform

Will Foster – Sr. Systems Engineer, Red Hat
Dan Radez – Sr. Software Engineer, Red Hat
Kambiz Aghaiepour – Principal Software Engineer, Red Hat
What this talk is not.

- Deep-dive of OpenStack internal components
- How to build a standalone OpenStack environment
- Usage guide that is going to fit every scenario
- Tuning guide for OpenStack
What this talk is

- Deep-dive of our mixed usage OpenStack Cloud deployment
- Our design decisions to allow customers to build their own OpenStack Clouds
- One of many ways to tie together robust Open Source tools to deploy/manage OpenStack
- A reference architecture: deployment & tooling approach
Design Considerations
Build a Cloud design that...

- Uses standard x86 computing and network gear (you'd find in any enterprise DC)
- Managed with **1-2 people**
- Scales **horizontally**
- Flexible for various purposes
- Oriented towards customer self-service
- Will immediately make us rich and famous
- Will be immune to common infrastructure problems
Scaleout Lab Design

- 9 racks / 220 bare-metal nodes
  - *projected scale to 42 racks / 1024 nodes*

- mixed internal and external usage

- spine/leaf topology (40GbE/10GbE)

- Minimal VLAN/network complexity
Some of our use cases...

- Build your own OpenStack Cluster
- Bare-metal as a service
- Public CI / Build Infrastructure
Use Case: Build your own OpenStack Cluster

- Holistic component testing
- CI and Performance/Scale testing
- Break-fix replication on bare-metal
Use Case: Bare-metal as a service

- Temporal leasing of hardware assets
- Permanent, siloed dev/test
- Bootstrap/provisioning must be fully automated
Public CI / Build Infrastructure

- Triple-o and OpenStack upstream
- Upstream resource pool
Network Design Considerations

- Designed to scale horizontally with growth
- Employ minimal VLAN complexity
- OpenFlow 1.1 capable for future SDN enhancements
- Use standard enterprise equipment
Architecture Overview
Best Practices
Utility Services

- Consolidate services via Foreman
- Utilize config management
Consolidate Services Via Foreman

- Puppet, PXE, DHCP, DNS, via Foreman
- Ease administration cost
- Unify provisioning workflow
Utilize config management

- Puppetmaster functionality in Foreman
- Use distributed revision control system
... perhaps a talk for another time.
Systems Design

- Take Full Advantage of Kickstart
- Use Linux Software RAID
- Keep it Simple
Take Advantage of kickstart %post

- Do everything you can in %post
- Foreman makes this easy!
Use Linux Software RAID

- Flexibility in rapid provisioning > small increase in performance
- Modern CPU's can easily handle RAID overhead
Keep it Simple

- Use shared storage for anything important
- Nodes should be a commodity
- It should be faster to spin new than fix old
Path Forward
Evolution in Resource Delivery

- Bare-metal server
- Virtual Machine
- Cloud Resource
Provision me a Server ...

- Long lead time
- Dealing with hardware, networking challenges
- No ability for self-service on demand
Provision me a VM ...

- Faster lead time
- Dealing with some networking challenges
- No ability for self-service on demand without 3rd party software
OpenStack. You become the cloud provider!

- Users spin up/down VM instances on demand
- Users manage networking on demand (SDN, Neutron)
- Metering, usage tools, API driven.
- Open and modular component design
- Use existing hardware, networking gear, infrastructure.
Automation
Tooling and Deployment Considerations

- Scriptable CLI and APIs
- Utilize config management and automation everywhere
- Standardize on naming conventions, IP address allocation, etc.
Scriptable CLI and APIs

- Maintain a team-shared repository of tools and scripts
- Automate processes
Utilize config management and automation

- Templating for hardware management
- Config files, flat files in a distributed repository or SCM (git, etc.)
Standardize, Standardize, Standardize

- Match U-location with hostnames
- Maximize your DNS and IP addressing schemes to match different networks

Example: host01.example.com has last IP octet match across any VLAN memberships
Deployment
Deployment Tools

- Red Hat Linux OpenStack Platform 4.0 ("Havana")
  - On Red Hat Enterprise Linux 6.5
- Foreman 1.5 (Red Hat Satellite 6)
  - Node provisioning and deployment
  - Hostgroups – classify nodes by component
  - Nova Compute
  - Neutron Networker
  - Controller
  - OpenStack Storage (GlusterFS)
  - Puppetmaster
Foreman Hostgroups classify a node as a specific OpenStack component.

Edit host05-rack03.scale.openstack.engineering.redhat.com

Name: host05-rack03

Host Group:
- OpenStack
- OpenStack/controller

Environment:
- OpenStack/neutron
- OpenStack/nova-compute
- OpenStack/storage

Puppet CA:
- base_hostgroup
- base_hostgroup/test)

Puppet Master:
- base_hostgroup/test/Compute (Neutron)
- base_hostgroup/test/Controller (Neutron)
- base_hostgroup/test/LVM Block Storage
- base_hostgroup/test/Neutron Networker
- base_hostgroup/test/Swift Storage Node
- generic
- generic-Hadoop
- generic-Performance-Eng
- generic-Product-Mktg-Virt
- generic-RHEV
- generic-RHEV-VDSM-GA

Use this puppet server as a CA server
Use this puppet server as an initial Puppet Server or to execute puppet run!
Puppet classes transform the node into a new OpenStack role.

Edit host05-rack03.scale.openstack.engineering.redhat.com
Hostgroups used to associate different Puppet classes.

Included Classes
- gluster::server
- scalelab::collectd::services

Available Classes
- apache
- ceilometer
- cernmonger
- cinder
- concat
- firewall
- glance
- gluster
- gluster::client
- gluster::params
- haproxy
- heat
- horizon
- nova
- openstack
- pacemaker
- packstack
- qpid
- quickstack
- rsync
- scalelab
- scalestack
- ssh
- stdlib
- swift
- tempest
- vlan
Deployment Improved
StayPuft Foreman plugin provides easy, template-based provisioning.
StayPuft provides a consolidated service view prior to deployment

New OpenStack Deployment

Host Groups & Available Services
Distributed - Neutron Networking

- Controller (Neutron)
  - MySQL
  - Neutron (Controller)
  - Cinder
  - Heat
  - Ceilometer
  - qpid (non-HA)
  - Keystone (non-HA)
  - Glance (non-HA)

- Neutron Networker
  - Neutron - L3
  - DHCP
  - OVS

- LVM Block Storage
  - Cinder

- Swift Storage Node
  - Swift

- Compute (Neutron)
  - Neutron-compute
  - Neutron-ovs-agent
You can change OpenStack configuration options in one place.

New OpenStack Deployment

Cinder Service Configuration

- Cinder backend gluster: False
- Cinder backend iscsi: False
- Cinder db password: 10bf4c801867024d20b6de04f7df3649
- Cinder gluster servers: 192.168.0.4, 192.168.0.5, 192.168.0.6
- Cinder gluster volume: cinder
- Cinder user password: 64b9f77e30762d26d010f82b8e5458590a6
Decide what roles to assign to nodes in the deployment.

<table>
<thead>
<tr>
<th>Host Groups</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Compute (Neutron)</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Controller (Neutron)</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LVM Block Storage</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Neutron Networker</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Swift Storage Node</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Hosts</th>
<th>Assign</th>
<th>Name</th>
<th>MAC Address</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>b8ca3ae638b50</td>
<td>b8ca:3a:53:8b:50</td>
<td>Host::Managed</td>
</tr>
<tr>
<td></td>
<td></td>
<td>b8ca3ae60bd18</td>
<td>b8ca:3a:60:bd:18</td>
<td>Host::Discovered</td>
</tr>
<tr>
<td></td>
<td></td>
<td>b8ca3ae60bf20</td>
<td>b8ca:3a:60:bf:20</td>
<td>Host::Managed</td>
</tr>
<tr>
<td></td>
<td></td>
<td>b8ca3ae60bf20</td>
<td>b8ca:3a:60:bf:20</td>
<td>Host::Managed</td>
</tr>
</tbody>
</table>
Automated Scale Testing
Scale Testing Methodology

Old Way - “Build it and they might come”

- Wasteful of resources
- Not optimal for demand-driven workloads
Scale Testing Methodology

**New Way** - “Roomba” your Deployments

- Automated spin up/tear down based on published schedule
- Users and Developers can plan their time or modify schedule themselves
- Resources can be spun down after a period of inactivity
Scale Testing Methodology

• How it works
  – Node configuration driven by simple input file (YAML)
  – Input file is checked against current configuration
    • Changed? – Nodes are rebuilt automatically
    • Unchanged? – Things stay the same

- New calendar published appropriately
Deploying OpenStack with CLI Tools

- **Hammer CLI**
  - Command line Foreman tool
  - Easy to wrap inside automation scripts

- **IPMI Tools**
  - Use IPMI to control power state
  - Disk configuration (Hardware RAID)

We'll give you a quick demo!
70-node automated OpenStack deployment video
Lessons Learned

- Use the Foreman “Discovery” plugin
- Extend Scheduling to Users/Developers
Future Enhancements

● Utilize SDN and OpenFlow to Manage Network Devices
  – We shouldn’t be in the business of network administration
  – Automating creation/modification of VLANs and switch settings
  – Use vendor OpenStack drivers (e.g. Netapp’s Cinder driver)

● Extend Scheduling to Users/Developers
  – Schedule their own resources
  – Plan availability around their needs

● Automate Discovery & Inclusion of New Hardware
  – Use Foreman “Discovery” to grow deployments automatically
  – Add new resources to the pile seamlessly
Q & A

kambiz@redhat.com will@redhat.com dradez@redhat.com

Free Public OpenStack
http://trystack.org

Provisioning Demo Tools and Scripts used:
https://github.com/sadsfae/openstack-lab-tools

Next Up:
Deterministic Capacity Planning for OpenStack: Keith Basil & Sean Cohen (303 @ 1:20pm)
Introduction to OpenStack for Developers: Kambiz & Dan (212 @ 2:30pm)