Introduction
Agenda

Start **building** a cloud!

Quick intro to **SUSE® Cloud architecture**

Learn about **HA in OpenStack and in SUSE® OpenStack Cloud**

Build an **HA cluster**

Build an **HA OpenStack cloud** on the cluster

**Break things!**
Workshop Environment
Workshop Environment

Relax ;-)

- We have plenty of time
- Whole build is also automated and idempotent
- You can take home the entire environment afterwards (available online)
- You can run on any machine with at least 16GB RAM
  - ... or 8GB at a push
  - (although that comes with limitations)
Workshop Environment

- We'll build a miniature cloud on a single machine
- VirtualBox hypervisor
- 4 VMs
  - Administration Server (Crowbar)
  - 2 Control Nodes in an HA cluster
  - 1 Compute Node
- Vagrant for rapid deployment
What is Vagrant?

"Creates and configures lightweight, reproducible, and portable development environments."

https://www.vagrantup.com/

- Not just for development
- Perfect for "kicking the tyres", demoing, testing etc.
- Cross-platform (Linux, MacOS X, Windows)
- Providers for libvirt, VirtualBox, VMware, Hyper-V, Docker, OpenStack, ...
Vagrant Inputs

• 1 or more Vagrant "box" – pre-built virtual appliances

• Vagrantfile: Ruby DSL file which defines:
  - which box(es) to use
  - virtual hardware required
  - virtual network topology
  - network ports to forward
  - hypervisor-specific settings
  - files to inject into appliance
  - commands to run in appliance

• files to inject
Using Vagrant: Crash Course

- `vagrant box add suse/cloud5-admin`
  - [https://vagrantcloud.com/suse/](https://vagrantcloud.com/suse/)
  - Also possible to add local boxes

- `vagrant up admin`

- `vagrant up controller1`

- `vagrant halt controller2`

- `vagrant destroy compute1`

- [https://docs.vagrantup.com/v2/getting-started/index.html](https://docs.vagrantup.com/v2/getting-started/index.html)
Workshop Vagrant Environment

- https://github.com/SUSE-Cloud/suse-cloud-vagrant
  - demos/HA/
  - vagrant/
    - Vagrantfile and configs/2-controllers-1-compute.yaml

- VirtualBox pre-installed
- 2 boxes pre-installed
  - suse/cloud5-admin and suse/sles11sp3
- 4 VMs
  - admin: SUSE OpenStack Cloud 5 Administration Server
  - controller1, controller2 (will form an HA cluster)
  - compute1
Exercise #1: Start the Build!

- Start up VirtualBox GUI
- `cd` to local copy of git repository
- `cd vagrant/`
- `vagrant up`
- All 4 VMs will be booted in sequence:
  - admin
  - controller1
  - controller2
  - compute1
SUSE® OpenStack Cloud Architecture Overview
SUSE® OpenStack Cloud Architecture

Admin Node
- SLES
- Chef Server
- Crowbar
- Software Mirror
- DHCP/TFTP

Control Node
- SLES
- Database
- Message Queue
- Openstack APIs
- Openstack Dashboard
- Openstack Scheduler
- Keystone
- Image Repo

Compute Node
- SLES
- Hypervisor
- OpenStack-Compute

Storage Node
- SLES
- CEPH/Rados
- OR-
- OpenStack Swift

Node Deployment & Management

Cloud Control
SUSE® OpenStack Cloud Control Node

- PostgreSQL database
- Image Service (Glance) for managing virtual images
- Identity (Keystone), providing authentication and authorization for all SUSE Cloud services
- Dashboard (Horizon), providing the Dashboard, which is a user Web interface for the SUSE Cloud services
- Nova API and scheduler
- Message broker (RabbitMQ)
SUSE® OpenStack Cloud Compute Nodes

• Pool of machines where instances run
• Equipped with RAM and CPU
• SUSE Cloud Compute (nova) service
  – Setting up, starting, stopping, migration of VMs
SUSE® OpenStack Cloud Network Node

- Networking Agents (Neutron)
- Providing access to the external world (North/South)
- Enabling intra-cloud communication (East/West)
- DVR technology decreases the bottleneck effect
Why an Install Framework?

SCARY AS HELL!
barclamp
High Availability and Cloud
Why High Availability?

“I can't have my systems go down. We lose $1,000,000 for every minute that we're down, and upper-management gets really 'excited' when that happens.”
High Availability for OpenStack

What might we want to protect?

• Admin server
  - core infrastructure: DNS, NTP, provisioning capabilities
• Controller node
  - OpenStack services
• Network node
  - OpenStack services
• Compute nodes
  - Hypervisor
  - VM instances (i.e. guests in the cloud)
Component Failure Impact

• Admin server
  - New cloud nodes require manual addition and configuration
  - Currently no ability to rediscover existing nodes on restart
  - No impact on currently operating cloud

• Control node
  - Cannot start or stop guest instances
  - No ability to rediscover existing nodes or guest VMs on restart
  - Some impact on currently deployed instances: loss of access to swift objects, etc.

• Network node
  - Some impact on currently deployed instances: loss of external networking
Pets vs. Cattle Metaphor

• Pets are given names like mittens.mycompany.com
• Each one is unique, lovingly hand-raised and cared for
• When they get ill, you spend money nursing them back to health

• Cattle are given names like vm0213.cloud.mycompany.com
• They are almost identical to other cattle
• When one gets ill, you shoot it and get another one
Component Failure Impact (continued)

• Compute node
  – Loss of VMs on that node
  – Recovery is by restart and re-provisioning of physical server
  – Can be mitigated through application design

• VM instances
  – Loss of workload
  – Recovery is by booting a replacement instance (cattle)
  – Can be mitigated through application design
Component Failure Assessment

- Control Node & Network Node
  - Highest priority
  - Recovery realistically requires complete cloud restart

- Compute Node & VM instances
  - Application level recovery is normal practice for existing clouds
  - Not existing “enterprise” expectation, but workaround exists for new workloads

- Admin Server
  - Least impact on deployed system
  - Operation can continue with no impact on end users
Status Quo of HA in OpenStack

• Community is now mostly converged on a standard architecture for an HA control plane involving Pacemaker and HAproxy

  – SUSE® was first vendor to release a supported implementation of this, via an update to SUSE Cloud 3 (May 2014)

• Towards a standard architecture for HA of compute nodes, using pacemaker_remote

  – Not for VM guests!

• HA for storage and network nodes is also still ongoing work

  – DVR, although not about HA, helps
Interlude
Exercise #2: Assign Aliases to Nodes

• Connect to admin node
  - `vagrant ssh admin` or
  - `ssh root@192.168.124.10` or
  - use VM console in VirtualBox

• Root password is `vagrant`

• Type `q` then `y` to accept the beta EULA

• Run `setup-node-aliases.sh`

• Point a browser at the Crowbar web UI
  - `http://localhost:3000`

• Check the 4 nodes are registered, named correctly, and in Ready state (green)
High Availability in SUSE OpenStack Cloud
HA in SUSE® OpenStack Cloud

• Administration Server
  - No longer a SPoF (Single Point of Failure)
  - Can have multiple DNS / NTP servers
  - Backup / restore script for cold or warm standby

• Control Plane & Network Node
  - Run services in a cluster
    - to ensure availability of data and service
  - Some OpenStack services are stateless
  - Some can run active/active, e.g. API endpoint services
  - The load balancer still needs protecting
  - Ordering constraints for services are required
  - Database and message queue need shared storage
HA of Control Plane

- Network
- Block Storage
- Compute
- Dashboard
- Image
- Object Storage
- Identity

Provides UI for
Provides UI for
Provides UI for
Provides UI for
Provides UI for
Provides UI for
Provides UI for
Provides Auth for
Provides Auth for
Provides Auth for
Provides Auth for
Provides Auth for
Provides Auth for

Provides network connectivity for
Provides volumes for
Stores images in
Stores disk files in

http://www.solinea.com
HA of Control Plane
HA of Control Plane

- Fully automated cluster setup through Pacemaker barclamp
- Simple, intuitive web UI
- Allows choice of cluster size and quantity
- Supports multiple strategies for STONITH and storage
- Uses SLE HAE components
  - Pacemaker, HAproxy, DRBD
- Architecture consistent with OpenStack community recommendations
HA of Control Plane

• Active/passive for PostgreSQL and RabbitMQ
  ‒ choice of replicated (DRBD) or shared storage
• Active/active for other services via HAproxy load balancer
  ‒ HAproxy itself is active/passive
  ‒ Innovative approach to Neutron L3 agent
    ‒ Not DVR (although not incompatible with it) nor VRRP
Setting Expectations

- *Not* fault tolerance
- Small outage of services is tolerated
- Automated recovery within small number of minutes
- “4 nines” availability (99.99% = ~53 mins/year)
- Maybe even five 9s achievable (4.32 mins/year) depending on context
- Some manual intervention may be necessary to repair a degraded (but still functioning) cluster
Simple cluster architecture

Control Node 1
- Orchestration
- Telemetry
- Dashboard
- Nova
- Glance
- Neutron
- Cinder
- Keystone
- RabbitMQ

Control Node 2
- Orchestration
- Telemetry
- Dashboard
- Nova
- Glance
- Neutron
- Cinder
- Keystone
- RabbitMQ

PostgreSQL

DRBD

Pacemaker Cluster
Recommended architecture

Services Cluster

- Node 1
- Node 2
- Node 3

- Orchestration
- Telemetry
- Dashboard
- Nova
- Glance
- Cinder
- Keystone

Network Cluster

- Node 1
- Node 2
- Node 3

- Neutron

Database Cluster

- Node 1
- Node 2

- RabbitMQ
- PostgreSQL
- DRBD or shared storage
Building a Pacemaker Cluster
… and a HA cloud!
Cluster Fencing

- Every HA cluster needs an out-of-band *fencing* mechanism. This is not optional!
- Simplistically, if cluster communications break down, consensus is lost, and multiple nodes may contend for the same data/service (“split brain” syndrome)
- Solution: **STONITH** (Shoot The Other Node In The Head)
- Popular fencing devices include IPMI, IBM RSA, HP iLO, Dell DRAC
- We'll use **SBD** (Storage-Based Death) which allows sending “poison pill” messages via shared block storage device.
Exercise #3: Build a Pacemaker Cluster

- In VirtualBox GUI, view settings for controller1 and controller2
  - Observe locations of extra disks (SBD and DRBD)
  - Which disk is shared?
- Point a browser at the Crowbar web UI
  - http://localhost:3000
- Follow instructions for deploying a Pacemaker cluster
  - SBD device is /dev/sdc
- On admin node:
  - Run tail -f /var/log/crowbar/chef_client/*
Exercise #4: Check Cluster Health

• Wait for chef-client to complete on both controller nodes
• Pacemaker proposal should finish applying and go green
• Connect to controller1 node
  - vagrant ssh controller or
  - connect to admin node and ssh controller1, or
  - use VM console in VirtualBox
• Root password (vagrant) and EULA as before
• Run crm_mon and check cluster has two nodes online
• Visit Hawk web UI on https://localhost:7630 and https://localhost:7631
  - Default credentials: hacluster / crowbar
Exercise #5: Build HA Cloud

• Follow remaining instructions for deploying the remaining barclamps
  - Locally available in demos/HA/manual-barclamps.md

• This will take quite a long time (at least 30 minutes)

• If you are feeling lazy, you can use a tool to this automatically:
  - crowbar batch --timeout 1800 \ build HA-cloud.yaml

• Watch Hawk web UI and crm_mon output as Crowbar/Chef automatically add resources to the cluster
Exercise #6: Simulate Failures

- Follow instructions for testing cluster failover
  - Locally available in demos/HA/cluster-failover.md
Exercise #7: Recover Degraded Cluster

- Follow instructions for recovering a degraded cluster
  - Locally available in demos/HA/cluster-recovery.md
If you made it this far, well done!
Unpublished Work of SUSE LLC. All Rights Reserved.

This work is an unpublished work and contains confidential, proprietary and trade secret information of SUSE LLC. Access to this work is restricted to SUSE employees who have a need to know to perform tasks within the scope of their assignments. No part of this work may be practiced, performed, copied, distributed, revised, modified, translated, abridged, condensed, expanded, collected, or adapted without the prior written consent of SUSE. Any use or exploitation of this work without authorization could subject the perpetrator to criminal and civil liability.

General Disclaimer

This document is not to be construed as a promise by any participating company to develop, deliver, or market a product. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. SUSE makes no representations or warranties with respect to the contents of this document, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. The development, release, and timing of features or functionality described for SUSE products remains at the sole discretion of SUSE. Further, SUSE reserves the right to revise this document and to make changes to its content, at any time, without obligation to notify any person or entity of such revisions or changes. All SUSE marks referenced in this presentation are trademarks or registered trademarks of Novell, Inc. in the United States and other countries. All third-party trademarks are the property of their respective owners.