Dependable Cloud Computing with OpenStack

Johannes Eschrig, **Sven Knebel**, Nicco Kunzmann

HPI Cloud Symposium "Operating the Cloud"

Hasso-Plattner-Institut Potsdam, 03.11.2015

(http://www.alphasandesh.com/blog/wp-content/uploads/2012/12/smtp-server.jpg)

Dependable Cloud Computing with OpenStack

- **Mission:** Create an OpenStack in the Box
- **Goal:** Create an OpenStack system for evaluation the dependability of fault tolerance mechanisms in OpenStack
- **2 Main Tasks:**
  1. Reproducible environment with OpenStack system
  2. Experiments for testing the dependability of OpenStack

Requirements

- Simple setup
  - Small setups complicated enough for thorough testing
  - Easy to simulate failures
  - Fast turnaround
  - Works on limited hardware
  - Easy to use for future (student) projects
- Multiple nodes/small cluster
- Potential for customization/extension to compare different setups
- Reproducibility of experiments

DevStack

HPI Hasso Plattner Institut

- **Development environment for OpenStack**
  - Multi-Node possible, but made for single node
- Custom configuration scripts
  - Documentation for multi-node and high-availability setups scarce
  - Hard to extend for new use cases
- Can DevStack dependability experiments be generalized to OpenStack?

Commercial Distributions

- Have comfortable installers for cluster-deployments
- Integrate with system management tools
  - (Ubuntu Juju, RedHat Spacewalk, …)
- Setup "production-grade" configurations
  - Require many nodes → high hardware demands
  - High-availability features built in
  - A lot of pre-configuration not intended for user change

→ **Results can't be generalized, hard to modify**

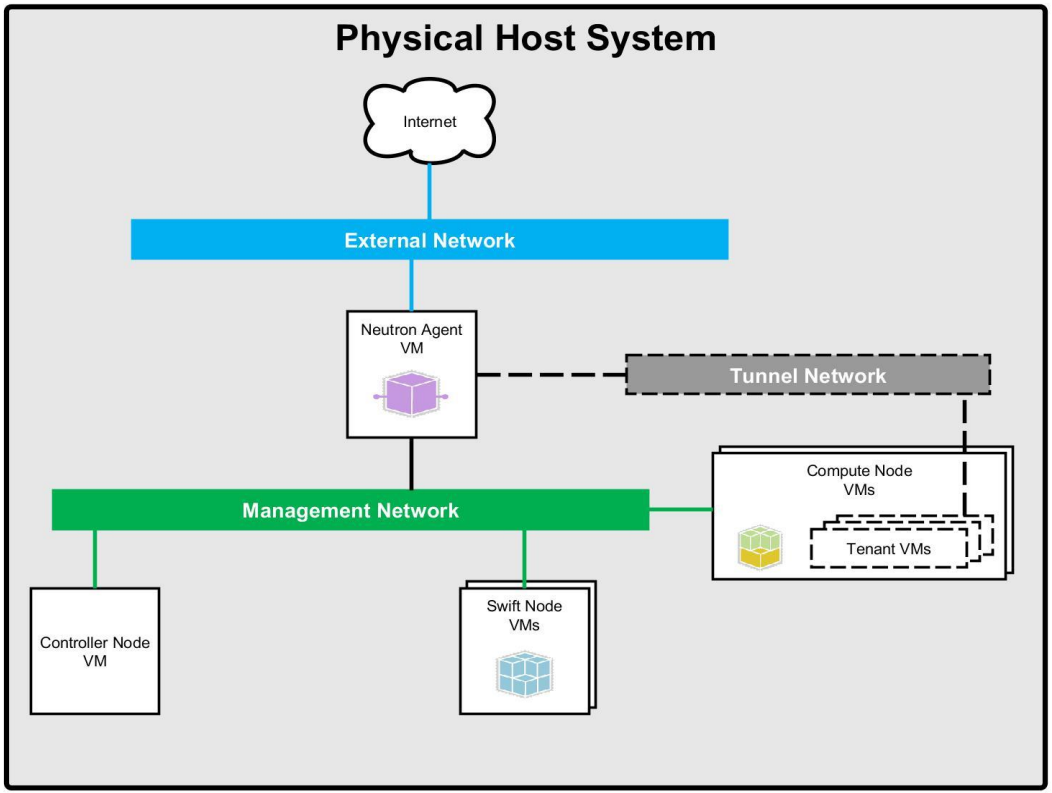
HP Helion Community Edition

- Has all-in-one demo mode
  - Nodes are created as Virtual Machines
- Great idea, but
  - Install takes a long time
  - Doesn‘t survive reboot of host machine
  - Install uses Triple-O
    - Elegant, but hard to understand
    - Not documented enough for us

Our Own Setup

- All-in-one “cluster” with virtualized nodes
- Very simple
  - Minimum interesting number of nodes and services
  - Leaves room for modifications/extensions
- Install process based closely on official OpenStack install docs
- Integration with experimentation framework

Reproducible Test Environment
(Our OpenStack Architecture)



OpenStack Installation

- Virtual machines using KVM
  - Widely supported
  - Nested virtualization

- Virtual Machines and networks are managed using libvirt
  - Virtual networks implemented as Linux bridges

- Installation using
  - Bash scripts to create Virtual Machines
  - cloud-init to inject configuration in Virtual Machines
  - Ansible to set up OpenStack on Virtual Machines

OpenStack Installation
Ansible

+

HPI Hasso Plattner Institut

- Ansible hosts file

```
[controller]
          root@192.168.100.11
[network]
          root@192.168.100.21
[compute]
          root@192.168.100.31
          root@192.168.100.32
[objectstorage]
          root@192.168.100.41
          root@192.168.100.42
```

OpenStack Installation

+

HPI Hasso Plattner Institut

- 02_install_and_configure_the_controller_node_components.yml

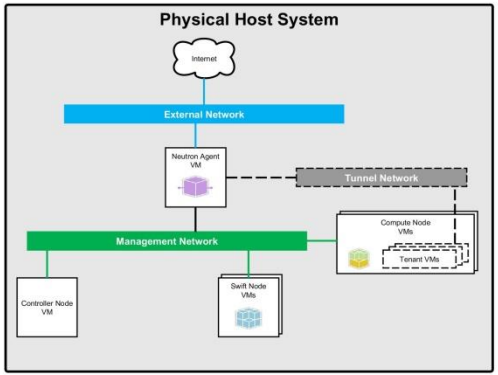
```
---
# http://docs.openstack.org/kilo/install-guide/install/apt/content/swift-install-controller-node.html
- name:  "Add Object Storage > Install and configure the controller node > install and configure the controller node components"
  hosts: controller
  roles:
    - config
  sudo: True
  gather_facts: True
  tasks:

    - name: "1. Install the packages"
      apt: pkg={{item}} state=installed
      with_items:
      - swift
      - swift-proxy
      - python-swiftclient
      - python-keystoneclient
      - python-keystonemiddleware
      - memcached

    - name: "2. Create the /etc/swift directory"
      file: path=/etc/swift state=directory mode=0755

    - name: "4. Edit the /etc/swift/proxy-server.conf file and complete the following actions"
      template: src=proxy-server.conf.j2 dest=/etc/swift/proxy-server.conf owner={{ ansible_user_id }}
```

OpenStack Installation
Ansible

```
Installation took 15m 28s.
To use the Horizon Web interface visit http://controller/horizon/
Logins are
    user: "demo" password: "demo"
    user: "admin" password: "admin"
  as specified in the configuration.
To use SSH execute
  ssh ubuntu@compute1
  ssh ubuntu@compute2
  ssh ubuntu@controller
  ssh ubuntu@network
  ssh ubuntu@object1
  ssh ubuntu@object2
```

OpenStack Installation
Results

- Full installation in 10-20 minutes
- Snapshot and restore all virtual nodes
- Package versions can be frozen to allow installing the same set up
- “Light” setup runs on normal (8 GB RAM, quadcore) workstation
- Full setup (2 compute and 2 Swift nodes) on 64 GB RAM, 16 Core workstation



Experiments on OpenStack

- Experiments vs Tests?
  - Tests have a clear success/failure condition
- Experiment contains
  - Where and how do we trigger a fault?
  - How can we observe resulting failures

Running Experiments on OpenStack

- Scripted using bash and Ansible
- Data about cluster available from installation configuration

- Stages
  - Setup
  - Observe
  - Break
  - Observe
  - (Heal)
  - (Observe)

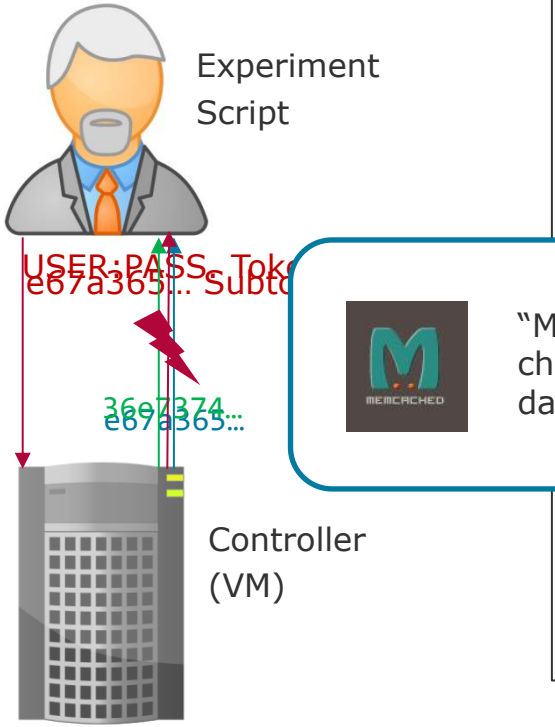
Experiment: Keystone Tokens

- Keystone: Authentication Service

- Authenticates Users, manages their permissions
- Authenticates actions using/between services

- Tokens:
  - created by users
  - carry a subset of their permissions
  - are passed with API requests
  - can be used by services to authenticate actions against other APIs

Experiment: Keystone Tokens




```
* SETUP
  - on controller, get token from Keystone
* Running checks
  - attempting to create new token using first token
    OK! Token created.
       36e737434cc446a7a3d9ab95e9746cb9
```

"Memcached is an *in-memory* key-value store for small chunks of arbitrary data (strings, objects) from results of database calls, API calls, or page rendering."

http://memcached.org/

Dependable Cloud Computing with OpenStack

1. Simple all-in-one virtualized cluster setup of OpenStack
2. Framework for reproducible experiments

- Our Code is on Github!

http://git.io/vlu6Y