OpenStack on Kubernetes: Make OpenStack and Kubernetes Fail-Safe

Seungkyu Ahn (ahnsk@sk.com)
Jaesuk Ahn (jay.ahn@sk.com)
Open System Lab
Network IT Convergence R&D Center
SK Telecom

Wil Reichert (wil@solinea.com)
Solinea
What will happen

- Introduction
- Kubernetes/OpenStack
- Demo Starts
- CI
- Demo Ends
Introducing Our Company

SKT
• No. 1 Mobile Service Provider in Korea with 50% market share
• We has been at the forefront of developing and commercializing new wireless technologies (recently, 4G LTE 5band CA with max 700Mbps)
• We are exploring more than network; Especially around AI and Media.
• We actively participate open source project; OCP, TIP, ONOS, Ceph, OpenStack, etc.

Solinea
• professional services partner that accelerates enterprise cloud adoption
• Technology agnostic, always working in the best interest of our clients
• Our clients are primarily Global Fortune 1000 organizations in multiple industry verticals
This is Totally Community Effort

SK Telecom

Solinea

Wil: CI/CD & K8S

Sungkyu Ahn: OpenStack & K8S

Jaesuk Ahn: OpenStack & K8S

Robert Choi: OpenStack & Automation

Jawon Choo: OpenStack & Kolla

Dan Kim: OpenStack & K8S

KOLLA

Large Contributing OpenStack Operator WG

Containers!

Cloud Native!
Current (previous) Way

SPEC

DEV

TEST

UPSTREAM

Community Code

OpenStack Package

Configuration Management

Deployment Automation

Requirements

Flexible Configuration

Deployment Architecture
- Network
- Storage
- Appliance Integration
- Configuration Tuning

Hardware/Appliance Purchase

Deployment

QA

Triage

Monitoring

Trouble Sh.

Analysis

Patch

Upgrade

Tuning

Capacity Mgmt.

Scale-out

Development

OpenStack

Deployment

Production

Operation
Previous Product Pain-Points

• Even Update (patch) is challenging

• Upgrade – gosh, what I can say.

• Deployment issue – snow-flake env. vs. cattle
  • Not single huge scale OpenStack, but many small/medium OpenStacks

• Lack of flexible configuration management capability in “standardized manner”

• Very difficult to integrate with our own stuffs (Ceph, SDN Controller, Datacenter Operation Platform, etc)
Continuous Loop

SPEC
DEV
TEST
Community Code
OpenStack Package
Configuration Management
Deployment Automation
Deployment Architecture
- Network
- Storage
Appliance Integration
Configuration Tuning
Requirements
Flexible Configuration
Deployment
Hardware/Appliance Purchase
Deployment Automation
QA
Triage
Monitoring
Trouble Sh.
Analysis
Upgrade
Capacity Mgmt.
Scale-out
OpenStack
Production
Development
Deployment
Operation
Triage
Monitoring
Trouble Sh.
Analysis
Upgrade
Capacity Mgmt.
Scale-out
Why OpenStack on Kubernetes?

Better way to deliver OpenStack and manage its Lifecycle

• Reducing Overhead: Dependency Management

• “Easy and Fast” Multiple Deployment in “Standardized” way

• Upgrade/Update/Rollback

• Easy Scaling and Healing
Key Technologies

- Kubernetes (Control Plane Orchestration)
- Helm (Application Lifecycle Management Automation)
- CI/CD Pipeline leveraging Jenkins
- OpenStack-helm (Managing OpenStack on Kubernetes)
- Kolla (Containerizing OpenStack)
- ONOS/SONA (OpenStack Network Management)
- Ceph (Storage)
Our Plan

• Production-Ready by the end of 2017

• First Production within 2017 (IT Infrastructure)

• Expanding to more deployment (Media, NFV) in 2018 & Putting more APPs on this “Streamline”
Overall Architecture
Today’s Demo System

- Jenkins Master
- Jenkins Slave
  - Kubernetes Master
  - Kubernetes Node
  - Tiller
- Helm Repo
- git
- Helm CLI
- Kubectl
What is the HA target?

- **Kubernetes Master**
  - Etcd
  - API server (load balance)
  - Scheduler (leader election)
  - Controller manager (leader election)

- **OpenStack Controller (Keystone, Glance, Nova, Cinder, Neutron)**
  - API server (load balance)
  - Scheduler (Nova, Cinder)
  - MariaDB
  - RabbitMQ
  - Neutron network node (SONA)
Kubernetes 3-Masters

kube-master01 kube-master02 kube-master03

node01 node00

ceph
Kubelet

KUBELET_OPTS="--kubeconfig=/etc/kubernetes/kubelet.conf
--require-kubeconfig=true
--hostname-override=kube-master01
--logtostderr=false
--log-dir=/var/log/kubernetes
--pod-manifest-path=/etc/kubernetes/manifests
--allow-privileged=true
--v=0
--register-schedulable=false
--cluster-dns=10.96.0.10
--cluster-domain=cluster.local"
etcd yaml

--name kube-master01
--listen-client-urls http://192.168.30.13:2379,
    http://127.0.0.1:4001
--data-dir /var/etcd/data
--initial-cluster-token 5d3903915c2cda30174970d84075f0a
--initial-cluster kube-master01=http://192.168.30.13:2380,
    kube-master02=http://192.168.30.14:2380,
    kube-master03=http://192.168.30.15:2380
--initial-cluster-state new
kube-apiserver yaml

```
- /usr/local/bin/kube-apiserver
  --etcd-servers=http://127.0.0.1:2379
  --storage-backend=etcd3
  --insecure-bind-address=127.0.0.1
  --insecure-port=8080
  --secure-port=6443
  --admission-control=NamespaceLifecycle,LimitRanger,ServiceAccount,
                         PersistentVolumeLabel,DefaultStorageClass,ResourceQuota
  --service-cluster-ip-range=10.96.0.0/16
  --tls-crt-file=/etc/kubernetes/pki/apiserver.crt
  --tls-private-key-file=/etc/kubernetes/pki/apiserver-key.pem
  --token-auth-file=/etc/kubernetes/pki/kube-token
  --service-account-key-file=/etc/kubernetes/pki/apiserver-key.pem
  --allow-privileged
  --anonymous-auth=false
```
kube-controller-manager yaml

- kube-controller-manager
  --master=127.0.0.1:8080
  --cluster-cidr=172.16.0.0/16
  --cluster-name=kubernetes
  --allocate-node-cidrs=true
  --service-account-private-key-file=/etc/kubernetes/pki/apiserver-key.pem
  --root-ca-file=/etc/kubernetes/pki/ca.crt
  --cluster-signing-cert-file=/etc/kubernetes/pki/ca.crt
  --cluster-signing-key-file=/etc/kubernetes/pki/ca-key.pem
  --v=0
  --leader-elect=true
kube-scheduler yaml

- /usr/local/bin/kube-scheduler
  --master=127.0.0.1:8080
  --v=0
  --leader-elect=true
kube-proxy yaml

```yaml
securityContext:
  privileged: true
command:
  - /bin/sh
  - -c
  - /usr/local/bin/kube-proxy
    --kubeconfig=/run/kubeconfig
    --cluster-cidr=10.96.0.0/16
    --v=0
```
OpenStack Controller & Compute

- MariaDB
- RabbitMQ
- Keystone
- Glance
- Cinder
- NOVA
- Neutron

kubernetes worker-nodes (label: controller)

- VM
- NOVA
- Neutron

kubernetes worker-nodes (label: compute)
OpenStack Controller & Compute

- **OpenStack Process 1** (nova-api)
- **OpenStack Process 2** (nova-api)
- **OpenStack Process 3** (nova-api)

**kubernetes worker-nodes** (label: controller)

**kubernetes worker-nodes** (label: compute)

- **VM**
- **NOVA**
- **Neutron**
Database clustering (3 node)

- jobs/mariadb-seed
- po/mariadb-0
- po/mariadb-1
- po/mariadb-2

- joiner (3th mariadb)
  --wsrep_cluster_address=gcomm://172.16.56.7,172.16.75.5,172.16.8.15
Neutron network (1 nic)
OpenStack-Helm Neutron chart

```
... network:
  interface:
    external: veth0
    default: br-data
ml2:
  agent:
    tunnel_types: vxlan
  type_drivers:
    - flat
    - vxlan
  ovs:
    auto_bridge_add: null
    bridge_mappings: null
neutron:
  default:
    l3_ha_network_type: vxlan
    debug: 'True'
...
```
Simplified Overlay Network Architecture

OpenStack
- Nova
- Neutron

Sets Switching/Routing flow rules

Compute-01
- nova-compute
- Hypervisor

Compute-02
- nova-compute
- Hypervisor

SONA(ONOS)
- OpenStackNetworking
  - Proxies ARP, DHCP
- vRouter
  - Sets NAT flow rules

Gateway Node Group
- br-router (OVS)
- Quagga

Hypervisor

Controls external connectivity

BGP / OSPF Multipath

East-West Traffic

North-South Traffic

OpenStack Networking

Proxies ARP, DHCP

Provisions virtual machine

Sets Switching/Routing flow rules

Sets NAT flow rules

VXLAN tunnel

BGP / OSPF Multipath
Live Demo 1
CI Presentation - Wil
Moving parts - some numbers

- Git repositories: 34 local, 10 upstream
- Deployment configurations: 4
- Supported Environments: 4
- Charts per deployment: 12
- Unique docker images in each deployment: 23
- Pods & jobs in a single deployment: 34
- Pods & jobs in a 3 node HA deployment: 85
Kinds of Workflows

- Build Kolla Container
- Build Helm Chart
- Test
- Deploy / upgrade
Kinds of Jobs

- Commit / PR builds - validate internal changes
- Live system deploys - validate component upgrades
- Nightly builds - validate merge CI / upstream
- Nightly redeploy - validate clean slate deploys
- Upstream builds - identify incoming breaking changes
When you’ve got twice as many virtualization technologies, things fall over twice as fast.
Managing the Clutter

• No upstream forks
• Isolated CI build & test environments
• Incremental upgrades
• Constant rebuilds
Testing starts from the ground up
Deployed Kubernetes

- Kubernetes e2e
- Custom
- Heapster*
Testing starts from the ground up
Kolla Container Builds

• Bats

• Clair
Testing starts from the ground up
Helm charts

- Helm test
- Partial tempest runs
- Built in (e.g. horizon)
Testing starts from the ground up
Deployed Openstack

- Full tempest runs
- Rally
Deployment is only the Beginning

- Manual validations
- Launch custom application on OpenStack
- Additional security tests on running containers
- Performance baselines and thresholds
Failure testing

- Risk - Many pods are run — privileged & with host networking
- Layered solutions present a lot more interesting failure cases
- HA rules change
- Openstack HA behaviors mapped onto Kubernetes HA behavior
Upgrades Become Boring

- All changes can be simulated multiple times prior to live rollout
- Minor component patches become trivial
- OpenStack upgrades become predictable
Challenges (pitfalls)

- Operational Burden
- Limitations
- OpenStack as Cloud Native Apps? Feasible? Needed? Gaps and Improvement?
- Kubernetes Stability (fast moving projects)
- Security
THE END