Introduction to Kubernetes on UCS

Vallard Benincosa, Software Engineer
Lara Benito Concepcion, Systems Engineer
Michael Maurer, Systems Engineer
Chris Riviere, Consulting System Engineer
Cisco Spark

Questions?
Use Cisco Spark to communicate with the speaker after the session

How
1. Find this session in the Cisco Live Mobile App
2. Click “Join the Discussion”
3. Install Spark or go directly to the space
4. Enter messages/questions in the space

cs.co/ciscolivebot#BRKCLD-2091
An Introduction to Kubernetes on UCS

• UCS
• KUBAM
• Containers
• Kubernetes
• Conclusion
KUBAM
Kubernetes on UCS the easy way
Demo
UCS APIs offer the most complete solution for server automation
What is missing?

- MAC POOLS
- SERVICE PROFILES
- WWN POOLS
- BOOT POLICIES
- VNIC TEMPLATES
- UUID SUFFIX POOLS
- STORAGE PROFILES
- SAN CONNECTIVITY TEMPLATES
- FIRMWARE UPDATES
- VMEDIA POLICY
- LOCAL DISK POLICY
- BIOS POLICY
- SCREEN POLICIES
- POWER SYNC POLICIES
- LOCAL DISK POLICY
- MAINTENANCE POLICIES
- MEMORY POLICY
How to automate Operating System Installation?
Why would we want it?

Kubernetes

Automation

Operating System
Existing Solutions

- UCS Director
- Puppet Razor
- Cobbler
- Windows Deployment Services
- Rocks
- Crowbar
- Warewulf
- xCAT
- Bright Computing
- Symantic
- Blah blah blah blah blah

- Kubespray
- Kubeadm
- kubeUp
- MiniKube
- Kops
- BootKube
- Kube-aws
- Kismatic
- Kubicorn
- SimpleKube
PXE Boot!
PXE Boot!

Too Complicated!
PXE Boot!

Too Complicated!

Requires:
DHCP, TFTP, VLANs, synchronize network and server people.
vMedia Policy
Released with UCS 2.2(2c) in 2014

- Allows ISO image to associate with Service Profile
- Supports installation over NFS, CIFS, HTTP, HTTPS
Using vMedia Policies

1. Get ISO Image
2. Unpack ISO image
3. Edit / Change files for automation, multiple files, many lines
4. Create boot image, modify the ISO image
5. Create install image for every server
6. Start a Web Server
7. Assign Service Profile Template to Server
8. Create vMedia Policy
9. Assign vMedia Policy to Template
10. Troubleshoot

Complicated!

But we can automate!
Introducing

https://kubam.io
KUBAM Architecture

API    GUI    Ansible

KUBAM Docker Images

Communication over UCS management IP 1Gb

UCS Resources including Service Profiles created using UCSM Python SDK

CIMC vMedia mounted from the KUBAM server (OS & KS.img)

Communication over Blade 40Gb Network

OS Resources fetched from KUBAM server for Deployment

VM or Physical Server

Cisco live!
KUBAM components

2 containers

- kubam/kubam
  - Python Flask Image
  - Uses UCSM SDK to communicate with UCS
  - Tools to manipulate ISO images
  - Runs API service that calls UCS components

- Kubam/web
  - React Node.js Image
  - Calls API server
  - Based upon Hyperflex design
vMedia Policy

Introduced with 2.2(2c) in 2014

- Allows ISO image to automatically be mounted with the service profile on creation
- Allows variable name of images based on service profile name
- Supports NFS, CIFS, HTTP, HTTPS
Cisco and Google hybrid cloud solution

On Prem/Colo Data Center

Google Cloud

Istio: Multi-cloud Service Management

Focus of this deck

Networking | Security | Private Cloud Infrastructure | Consumption Management

consistent environment
<table>
<thead>
<tr>
<th>DNS name</th>
<th>Query Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenDNS</td>
<td>23.64 ms</td>
</tr>
<tr>
<td>Quad9</td>
<td>34.09 ms</td>
</tr>
<tr>
<td>Google</td>
<td>36.29 ms</td>
</tr>
<tr>
<td>Neustar</td>
<td>50.48 ms</td>
</tr>
<tr>
<td>Norton</td>
<td>51.18 ms</td>
</tr>
<tr>
<td>SafeDNS</td>
<td>66.11 ms</td>
</tr>
<tr>
<td>Comodo</td>
<td>69.99 ms</td>
</tr>
<tr>
<td>Verisign</td>
<td>75.63 ms</td>
</tr>
<tr>
<td>Yandex</td>
<td>126 ms</td>
</tr>
</tbody>
</table>
Big Data > Vast Data = Unique Perspective

- DNS Requests: 135B
- Daily Active Users: 90M
- Enterprises: 15K
- Countries: 160+
Umbrella for Data Centers

- **Best of Breed Recursive DNS**
  - 100% Uptime with Anycast routing
  - Fastest Recursive DNS Provider
  - Failovers/Redundancies
  - Visibility into DNS requests globally

- **Ease of Deployment (under 1 hour)**
  - Point DNS to OpenDNS
  - Register external IPs in dashboard
  - Configure policies for desired blocking/reporting

- 208.67.220.220
- 208.67.222.222
Umbrella for Data Centers

- **DNS Layer of Visibility**
  - Visibility into every request leaving the DC
  - Identify unwanted activity within DCs
    - Crypto-mining
    - Unauthorized callouts
    - Unexpected spike in traffic

- **DNS Layer of Security (optional)**
  - Addresses a gap in existing DC security posture
  - Port, Protocol, App agnostic
  - Indication of compromised infrastructure
  - Ransomware/Botnets
Containers
Act 1
Custom Applications into Production

Developer

App.py! So great!

Operations

Doesn’t work!

Oh, needs opencv3!

Still Doesn’t work!

Oh, needs opencv3 version 3.0.0, 3.1.1 doesn’t work!

Still Doesn’t work!!!
Act 2
Custom Applications into Production with Containers

Developer

App.py! So great!

Operations

App.py
opencv3
Python 3.4

App.py! So great!
Containers vs traditional infrastructure

- **Bare Metal**
  - Host OS
  - Server
  - Bins/Libs
  - App

- **Virtual Machine**
  - Host OS
  - Hypervisor
  - Server
  - Bins/Libs
  - Guest OS
  - App

- **Container**
  - Host OS
  - Docker Engine
  - Server
  - Bins/Libs
  - Guest OS
  - App
Docker overview
What does a Docker image look like?
Kubernetes
docker run -p 80:80 -d -name app app
for i in $(seq 3); do docker run -P -d -name app$i app; done
for j in \($\text{seq } -w 5\); do ssh ucs\$j for i in \($\text{seq } 3\); \ do docker run -P -d -name app\$j\$i app; done; done
Solutions Emerge

How to ensure each machine is used most efficiently?

Kubernetes

What application runs on which server?

How do you keep track of ports?

Mesos

How do we account for dynamic workloads?
Kubernetes Architecture
Kubernetes

Kubernetes is an open-source system for automating deployment, scaling, and management of containerized applications.

It groups containers that make up an application into logical units for easy management and discovery. Kubernetes builds upon 15 years of experience of running production workloads at Google, combined with best-of-breed ideas and practices from the community.
Kubernetes Architecture

Redundant Master Nodes
- Monitors cluster, pods, and services
- Stores state of cluster
- Only require 1 but should use 3-5

Kubernetes Worker Nodes
- Run Docker or other Container Runtime
- Receive Instructions from Master Nodes
- Report status back to Master nodes
- Proxy services to route traffic to containers
- Run infrastructure Pod services
- Place where all workload happens.
# Kubernetes Master Node

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>kube-controller-manager</td>
<td>Watches when nodes, pods, services go down</td>
</tr>
<tr>
<td>kube-scheduler</td>
<td>Selects Nodes for newly created Pods</td>
</tr>
<tr>
<td>kube-apiserver</td>
<td>Front-End API used by kubectl and web interface</td>
</tr>
<tr>
<td>etcd</td>
<td>Key-Value Store for all Kubernetes Cluster Data</td>
</tr>
</tbody>
</table>

- **master01**
- **master02**
- **master03**
Kubernetes Node

- **kube-controller-manager**: Watches when nodes, pods, services go down
- **kube-proxy**: Enables service abstraction with network rules
- **kubelet**: Listens for Instructions from API server and executes
- **docker**: Container Runtime environment

master01
master02
master03
Kubernetes Key Components
Kubernetes main Features

- Pods
- Deployments
- Services
- Ingress
Pods

- Group of one or more containers, shared storage, and options for how to run the containers
- Share IP address and port space
- Atomic unit of management

Source: http://kubernetes.io/docs/user-guide/pods/
Deployments

• Declare intent: How many replicas should be running of a given pod?
• Namespace
• Labels
• Ports that should be exposed

```yaml
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: my-nginx
spec:
  replicas: 2
  template:
    metadata:
      labels:
        run: my-nginx
      spec:
        containers:
          - name: my-nginx
            image: nginx
            ports:
              - containerPort: 80
```
Services

• Abstraction for the mortality of Pods
• Provide single stable name and address for a set of pods inside the cluster.

Source: http://kubernetes.io/docs/user-guide/services/
Ingress

• Abstraction for services
• An **Ingress** is a set of rules for directing inbound traffic to a service.
• An **Ingress Controller** is a service that listens for the creation of new services and does reverse proxy (nginx, traefik, f5 loadbalancer)

See: http://kubernetes.io/docs/user-guide/ingress/
Where we’re going

- Monitoring
- Multi UCS Domain Support
- UCS C-Series
- ACI
- Cisco Intersight
Conclusion

• **Containers**: simplify delivery of applications into production
• **Kubernetes**: simplifies containers
• **KUBAM**: simplifies Kubernetes on UCS
Questions?
Use Cisco Spark to communicate with the speaker after the session

How
1. Find this session in the Cisco Live Mobile App
2. Click “Join the Discussion”
3. Install Spark or go directly to the space
4. Enter messages/questions in the space

cs.co/ciscolivebot#BRKCLD-2091
• Please complete your Online Session Evaluations after each session

• Complete 4 Session Evaluations & the Overall Conference Evaluation (available from Thursday) to receive your Cisco Live T-shirt

• All surveys can be completed via the Cisco Live Mobile App or the Communication Stations

Don’t forget: Cisco Live sessions will be available for viewing on-demand after the event at www.ciscodive.com/global/on-demand-library/.
Continue Your Education

• Demos in the Cisco campus
• Walk-in Self-Paced Labs
• Tech Circle
• Meet the Engineer 1:1 meetings
• Related sessions
Thank you
You’re

CiscoLive!