



# NSO Kubernetes Lab

NSO Developers Connect- NYC

John Mullooly- Distinguished Architect- [jmullool@cisco.com](mailto:jmullool@cisco.com)

Fredrik Jansson- Principle Engineer

December 5th, 2019

*#GST #CISCOVT #CISCOSE*

Global  
Sales Training

# Agenda

- Introduction, Objectives  
Getting Started Tips and  
Lab Overview

# Agenda (cont.)

- LAB1 – Foundational
  - Docker
  - Kubernetes
  - Helm Charts
- Lab2
  - NSO as an Application deployed on K8s

# Housekeeping

- This is a 90%+ Hands-On Lab Class! Practice your keyboard Judo!
- Previous hands-on for K8s & Docker not required but helpful
- This lab is not a replacement for dedicated cloud native training
  - Many on-line classes, books and web sites available
- Use the Cisco AnyConnect VPN client to access your lab
  - You can use your own desktop tools! Use your own local ssh client & Browser
  - Makes cut and paste of commands much easier
- To learn, try and type everything from the the lab notes
- Use a simple editor as a scratch pad
  - Avoids “special character” insertion when cutting/pasting
  - Provides a record in case you need to re-do
- Have fun and goof around with the lab.
- Lab is up until Sunday and can be scheduled again (up to 5 days)- just email us

# What is a Container anyway?

- Containers offer a logical packaging mechanism in which applications can be abstracted from the environment in which they actually run
- Portable – write once, run anywhere
- “Immutable Infrastructure”
- A Container is a running instance of an image
- An image contains everything needed to run the application
- Leverages underlying operating system technologies:
  - Namespaces
  - Control Groups
  - Capabilities

# Container Orchestration (e.g. Kubernetes)

- Provision containers
- Manage container dependencies
- Enable discovery
- Handle container failure
- Scale containers
- Day 2 maintenance



# Kubernetes Objects

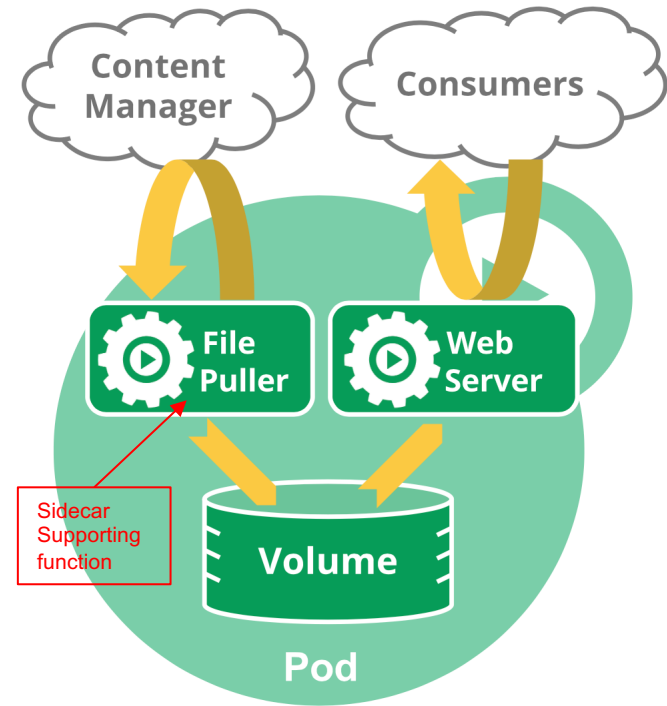
- K8s objects are “record of intent”
- Once created K8s ensures that the object is kept running
  - Desired state
- Object spec
  - Describes the desired state
- Object status
  - Actual state

```
application/simple_deployment.yaml

apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  selector:
    matchLabels:
      app: nginx
  minReadySeconds: 5
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:1.7.9
          ports:
            - containerPort: 80
```

# The pod

- Pods are the unit of deployment in Kubernetes
- Pods consist of one or more containers that share an IP address
- “IP per pod” model
- Containers within a pod can use localhost
- Must coordinate port usage within the pod

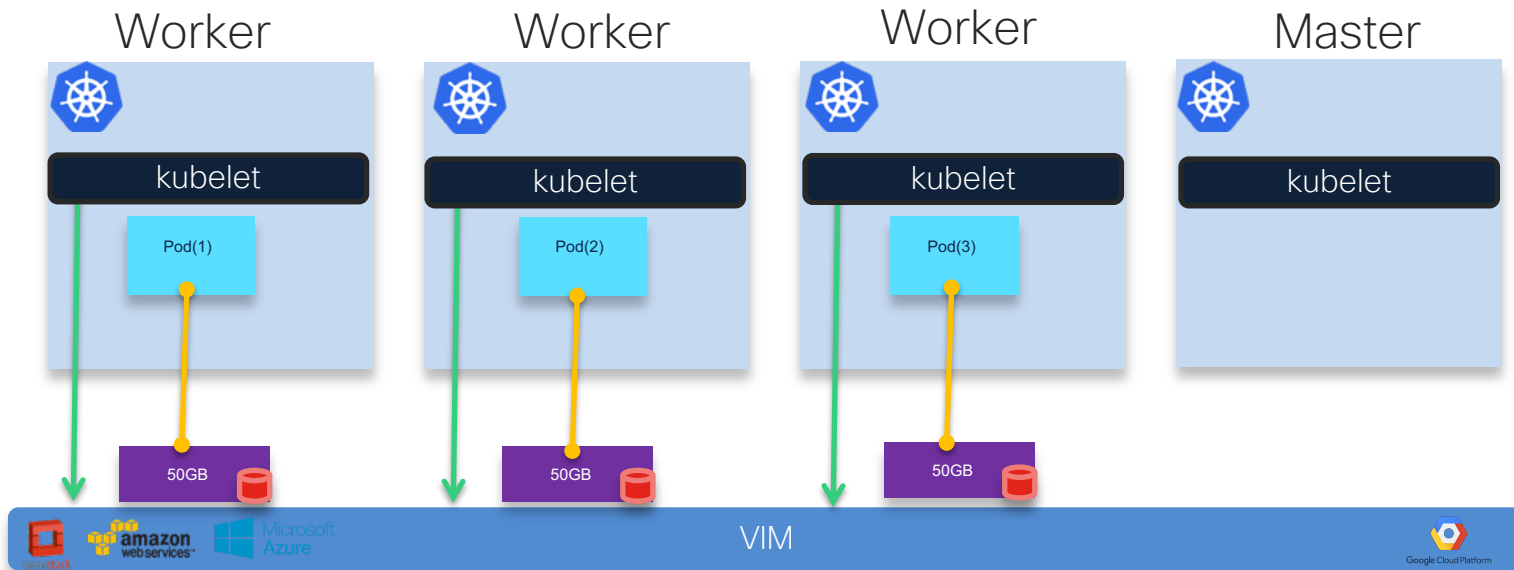




# StatefulSets

## Notes

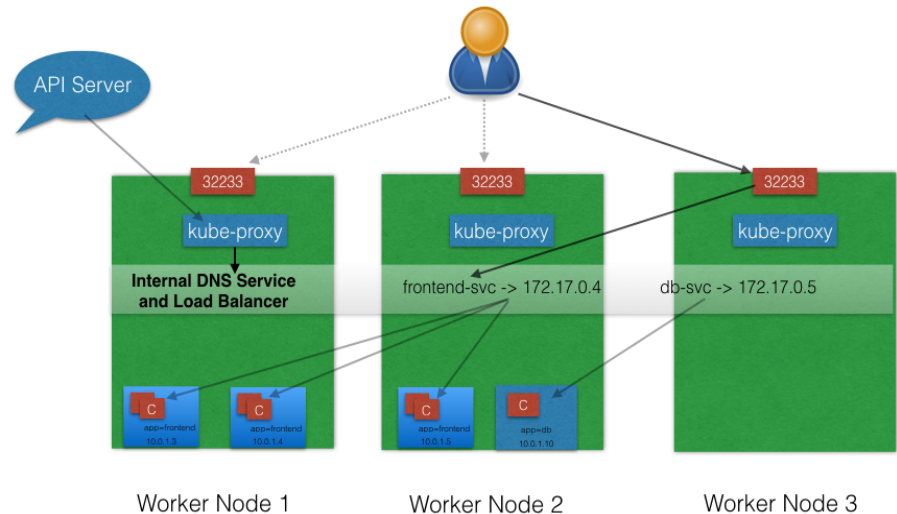
- Node Pinning
- IP consistency
- Will never be scheduled on another node
- Requires forced delete to move
- Requires a volume



# NodePort

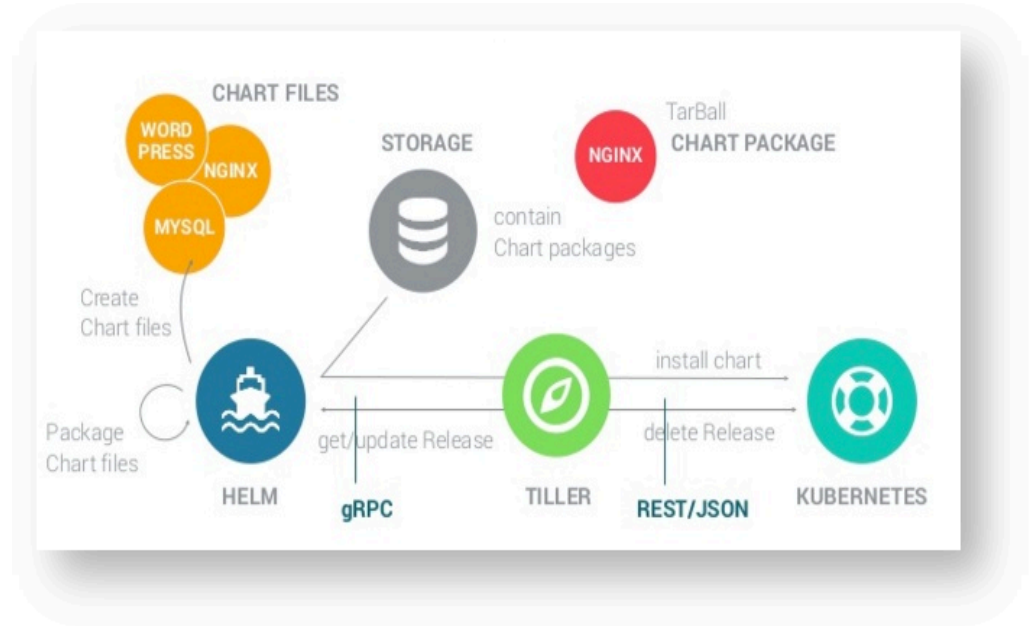
- K8s master allocates the same port on every node for your service
- Specified by: `--service-node-port-range` flag (default: `30000-32767`)
- You specify the port in the service resource

## Service - NodePort



# Helm

- Package manager for Kubernetes, analogous to yum or apt
- Organize Kubernetes objects in a packaged application
- In Helm, these packages are called *charts* (similar to debs or rpms)
- Supports versioning and rollback



# Why is running NSO in a container and with K8s interesting?

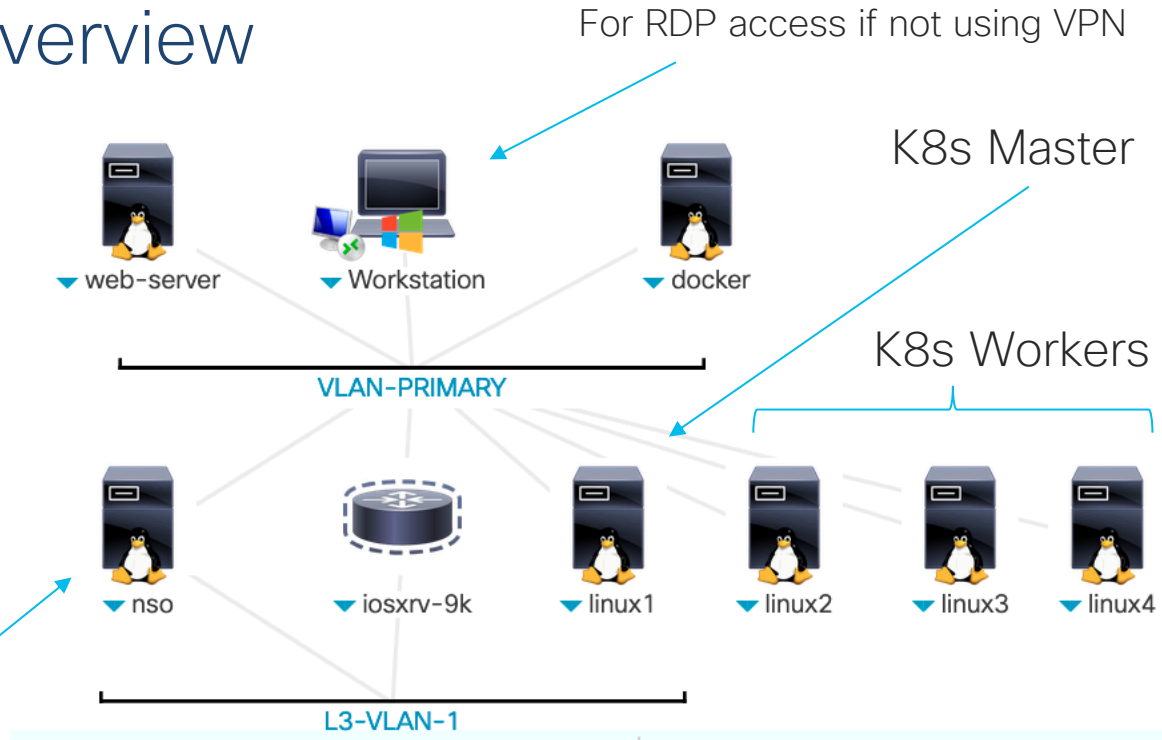
- Many customers are investing in container infrastructure
  - Harmonize how you distribute and run applications
  - Well known among micro service developers
- Take advantage of K8s for application re-startability
  - Benefit from k8s health check
  - Replicated Storage
  - You will do this in the lab
- NSO HA can/should still be used

# NSO in Container Caveats

- NSO is a database
  - stateful, hence needs persistent storage
  - mutable
- CDB is an in-memory DB
  - Memory consumption
  - k8s resource limits need be considered

NSO In a Container/k8s

# Lab Overview



This NSO for second part of lab (NSO driving K8s/HELM deployments)

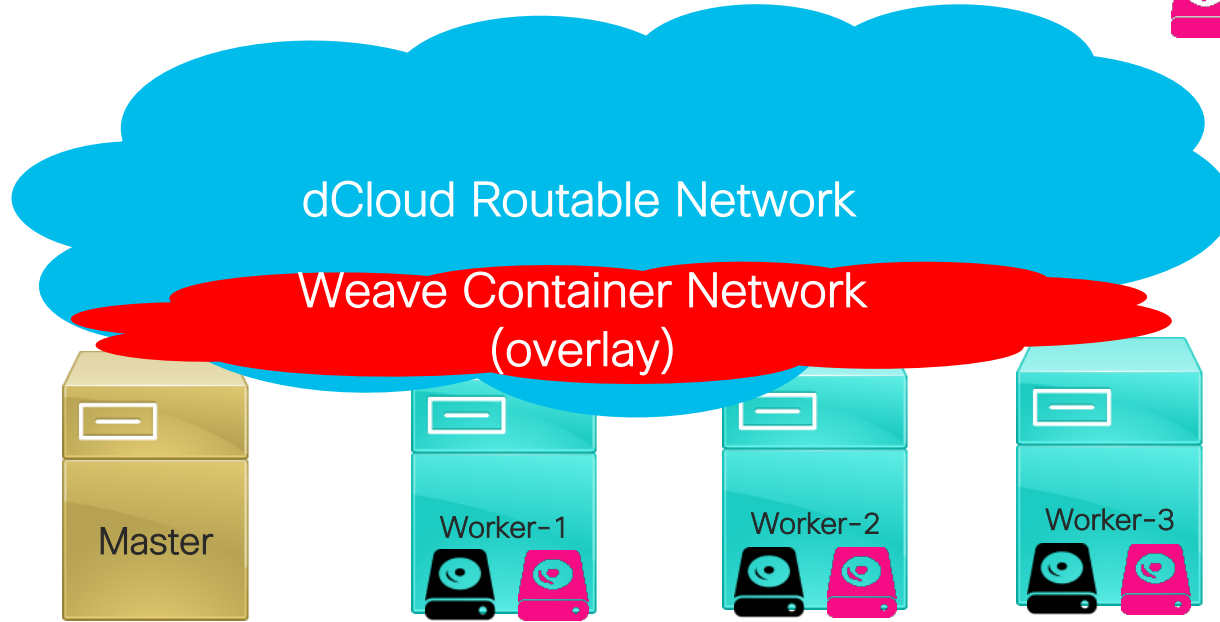
# Lab K8s Cluster Networking



Operating System



GlusterFS





# Lab flow

<b>Topology.....</b>	<b>3</b>
<b>Get Started .....</b>	<b>4</b>
<b>Explore the Environment .....</b>	<b>5</b>
<b>Using and moving around the Kubernetes environment: .....</b>	<b>5</b>
<b>GlusterFS.....</b>	<b>9</b>
<b>Helm .....</b>	<b>11</b>
<b>NSO in a Container Lab.....</b>	<b>16</b>
<b>Create NSO base container with a simple package .....</b>	<b>16</b>
<b>Run NSO base container with docker .....</b>	<b>18</b>
<b>Running the NSO base container as simple k8s stateful set.....</b>	<b>20</b>
<b>NSO Re-startability with K8s .....</b>	<b>23</b>
<b>Run NSO container with replicated storage .....</b>	<b>24</b>
<b>Run the NSO container with Helm .....</b>	<b>31</b>
<b>Using NSO to deploy your k8s Application Lab.....</b>	<b>33</b>
<b>Run the web server using Helm (again) .....</b>	<b>33</b>
<b>Local NSO .....</b>	<b>35</b>

# Cheat sheet

- `kubectl get nodes`
- `kubectl get nodes -owide`
- `kubectl get ns`
- `kubectl get pods --all-namespaces`
- `kubectl get pods -n <namespace>`
- `kubectl get pods --all-namespaces -owide`
- `kubectl get services`
- `kubectl apply -f <file.yaml>`
- `kubectl exec -n <namespace> <container/pod name> -it bash` (to access container shell)
- `helm upgrade --install <new-deployment-name> <path to helm directory>`
- `helm upgrade --install -f <values-override-file-path> <new-deployment-name> <path to helm directory>`
- `helm ls`

# Cisco Official Docker Image for NSO

- <https://gitlab.com/nso-developer/nso-docker/tree/master>

