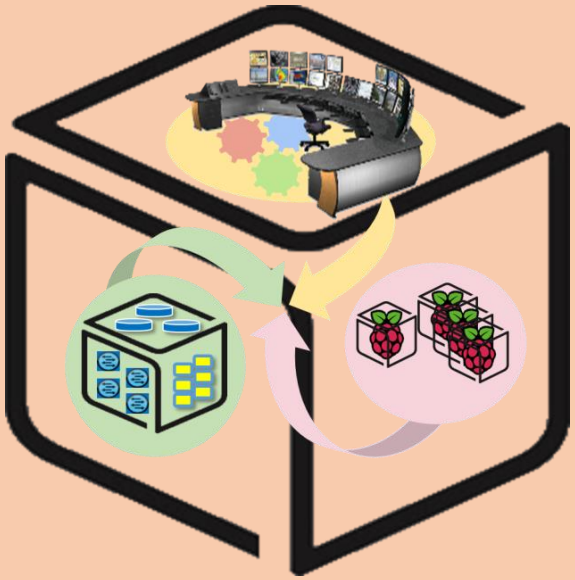
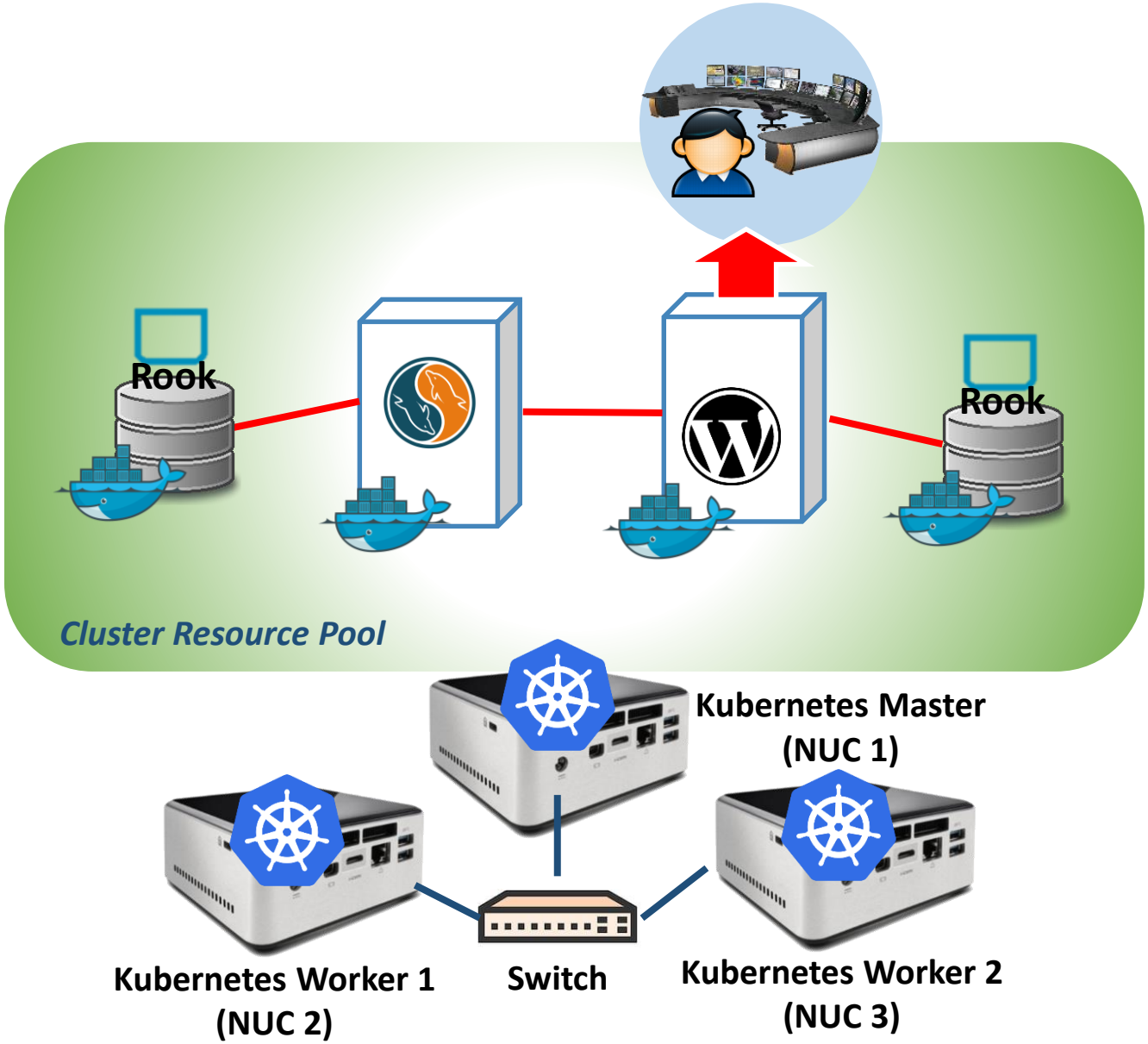
Computer Systems For AI-inspired Cloud Theory & Lab.

Lab #5: Cluster

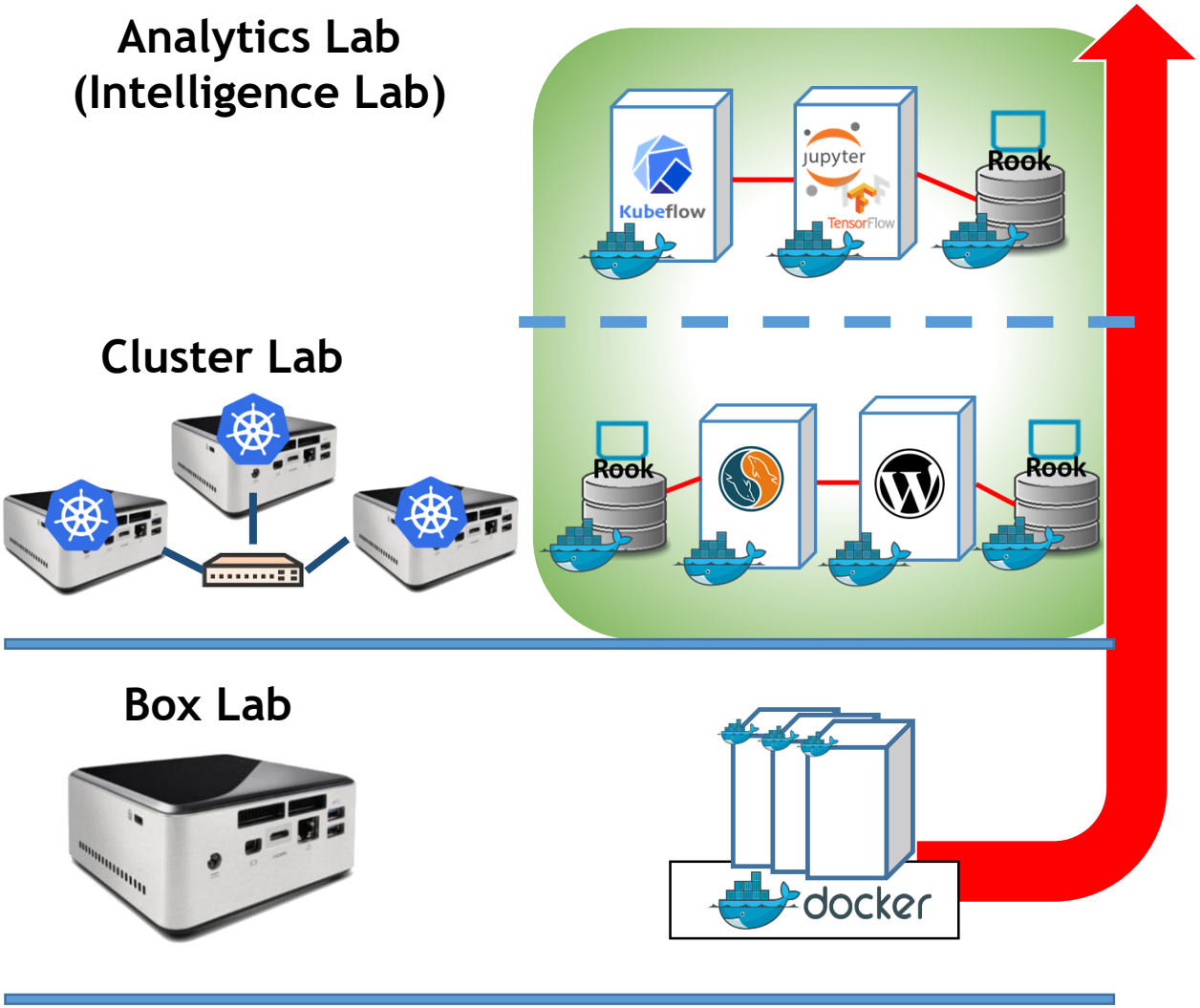


https://github.com/SmartX-Labs/SmartX-Mini-MOOC

Cluster Lab: Concept



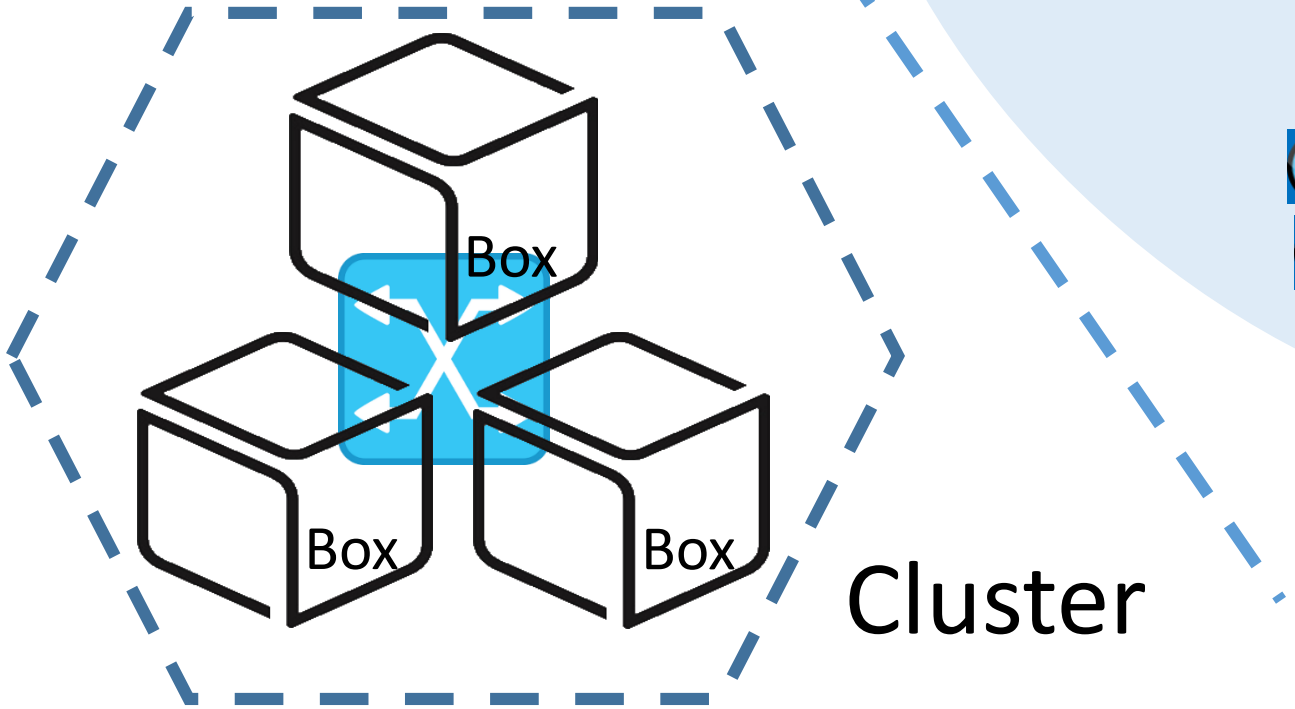
SmartX Labs #1/#5/#6: Relationship

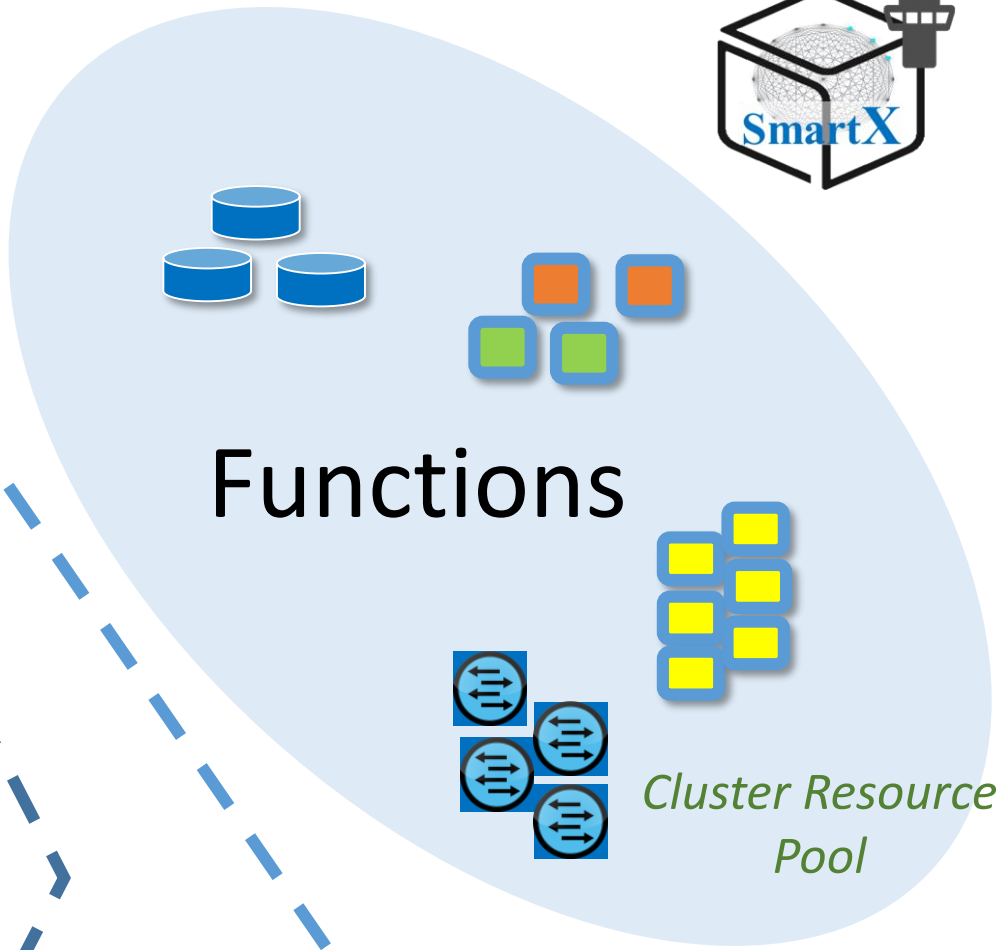


Theory

SmartX Cluster: Inter-connected SmartX Boxes

Computing Cluster is a form of computing in which a group of computers are linked together so that they can act like a single entity



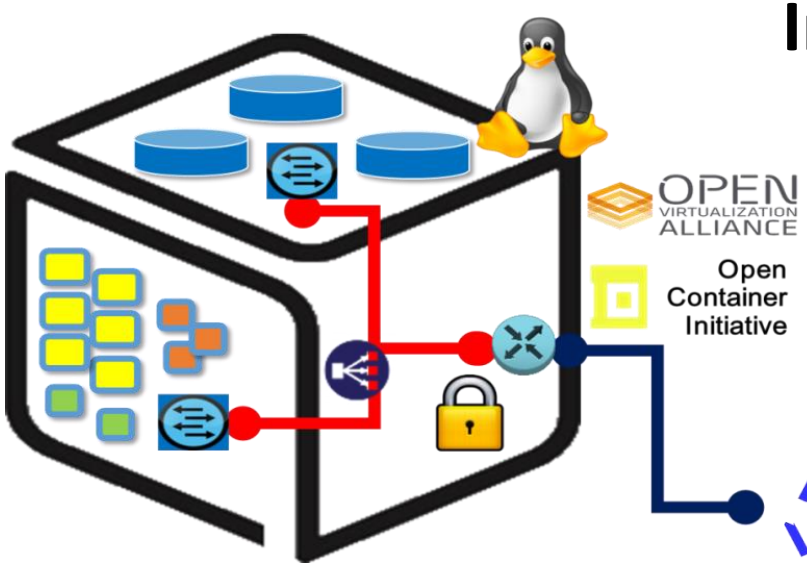


Inter-Connected Functions inside a Box & across Clusters

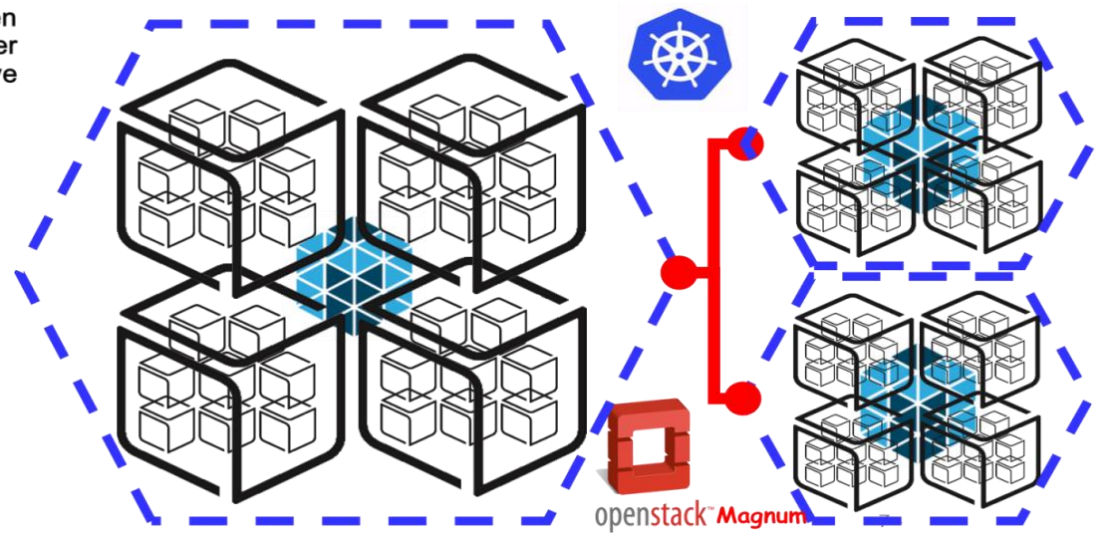
p+v+c Harmonization Challenge:

p(Baremetal) + v(VM) + c(Container)

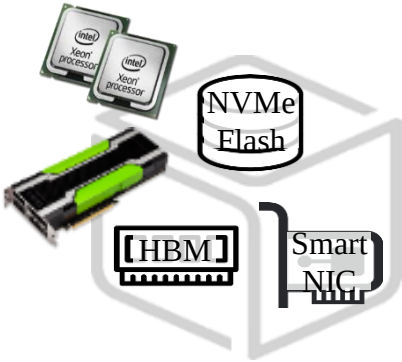
Inside a Box



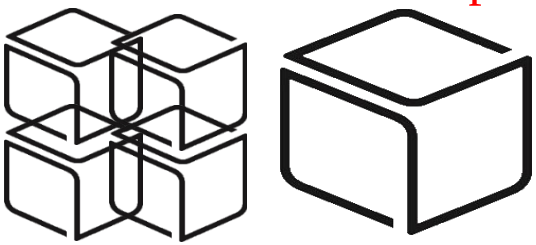
Across Clusters

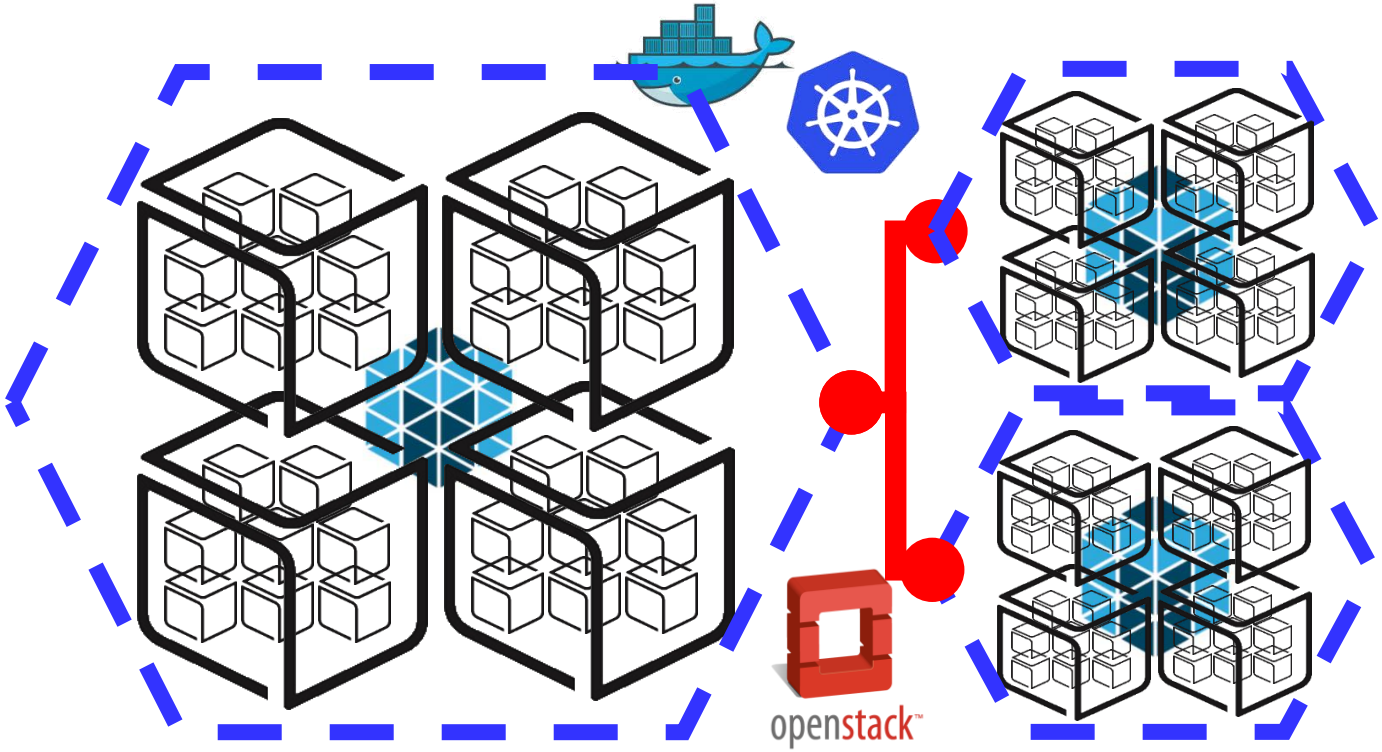


Computer System: Resource Scaling/Pooling with Clustering



Scale Out Scale Up





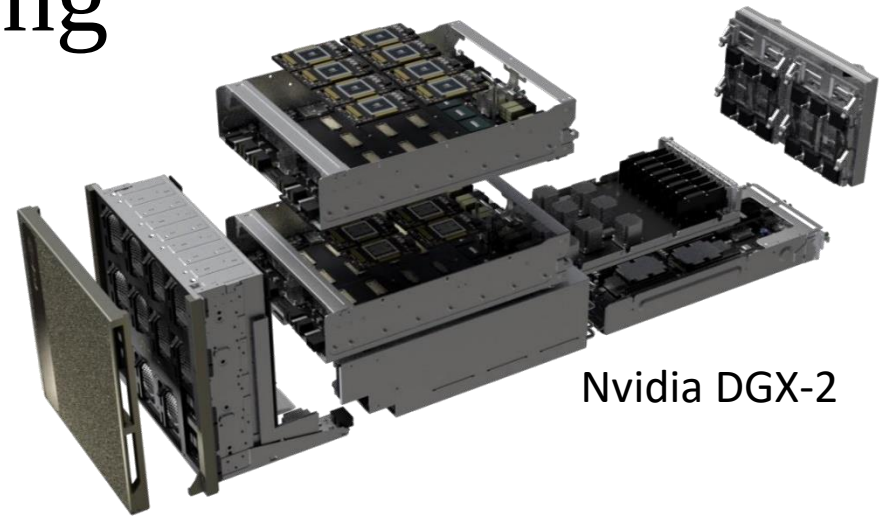
HPC +
HPDA (BigData)

➔ AI (ML/DL)

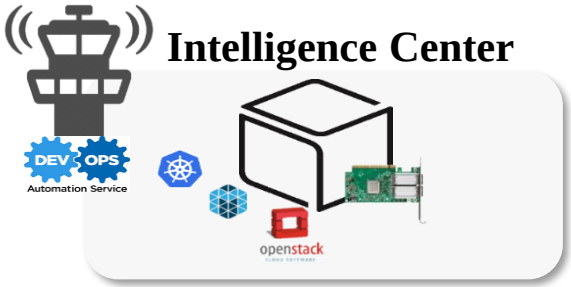
Cluster for AI Computing

HPC + HPDA (BigData)

➔ AI (ML/DL)



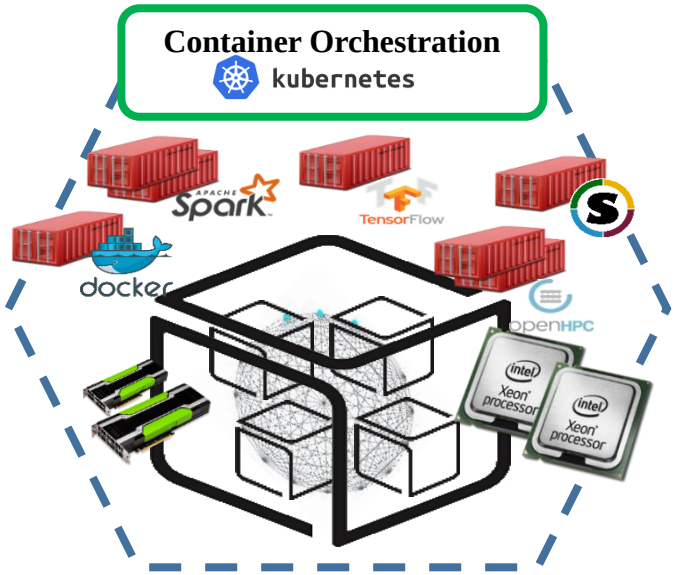
Nvidia DGX-2



Multi-node AI Computing Cluster with Optimized DL Tools

GPU Acceleration

GPU Cards



Cloud Storage Cluster

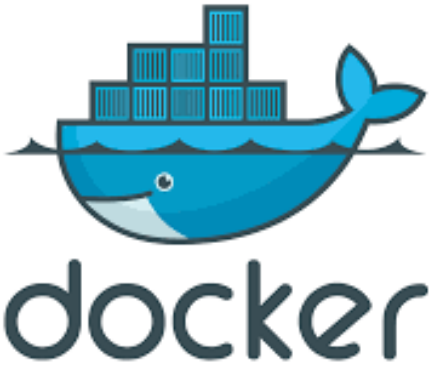
Parallel File System

NVMe SSDs

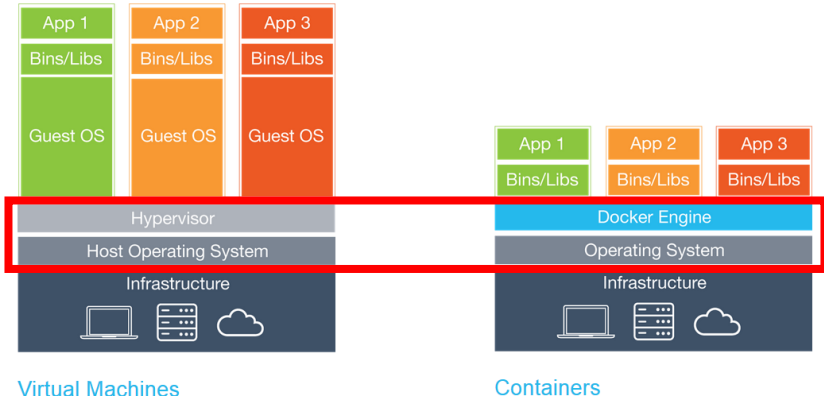
Cloud Storage

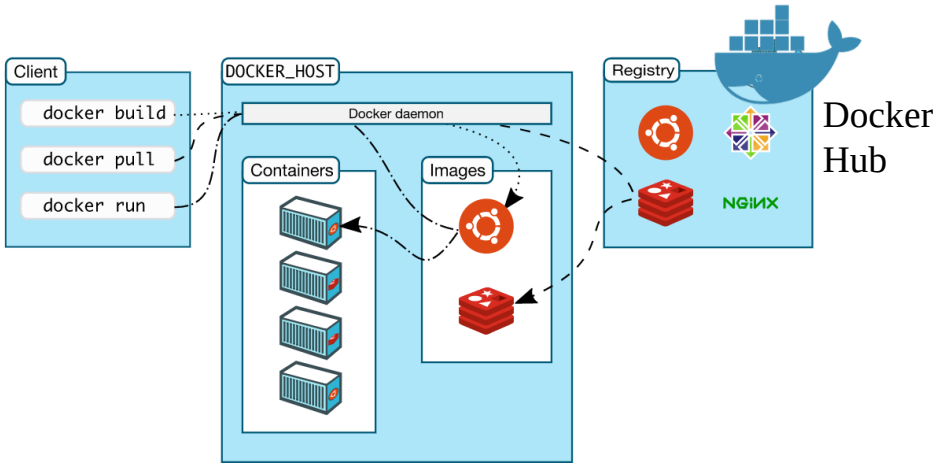
Smart NICs

Docker Containers



- **Docker** is an open platform for building, shipping and running distributed applications. It gives programmers, development teams and operations engineers the common toolbox they need to take advantage of the distributed and networked nature of modern applications.

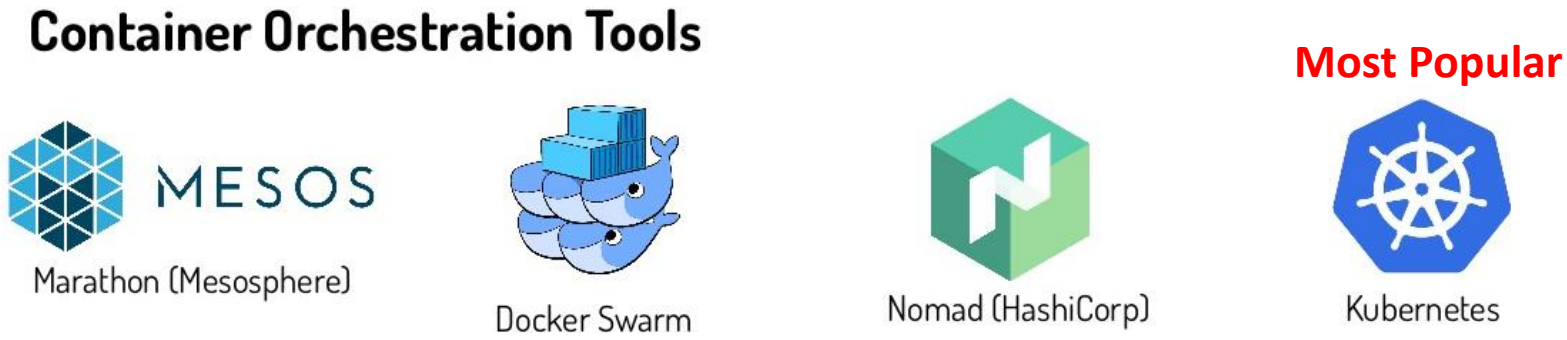




Since container uses host kernel,
OS of host should be Linux distribution.

https://www.docker.com/

Container Orchestration



Container orchestration refers to the process of organizing the work of individual components and application layers.

Container orchestration engines all allow users to control when containers start and stop, group them into clusters, and coordinate all of the processes that compose an application. Container orchestration tools allow users to guide container deployment and automate updates, health monitoring, and failover procedures.

https://www.hpe.com/us/en/what-is/container-orchestration.html

Container Orchestration: Kubernetes

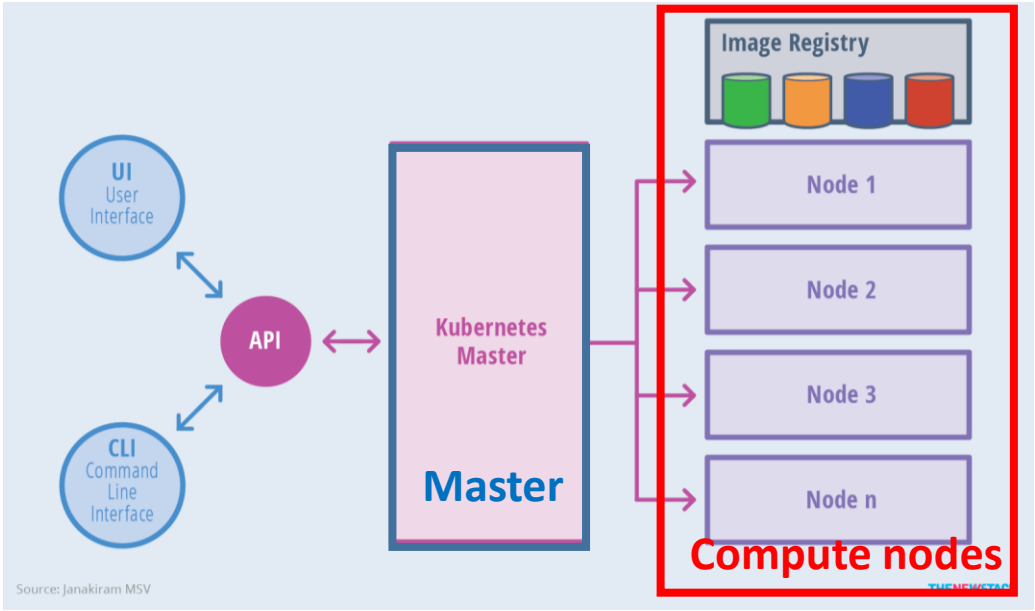
Kubernetes is an open-source system for automating deployment, scaling, and management of containerized applications.

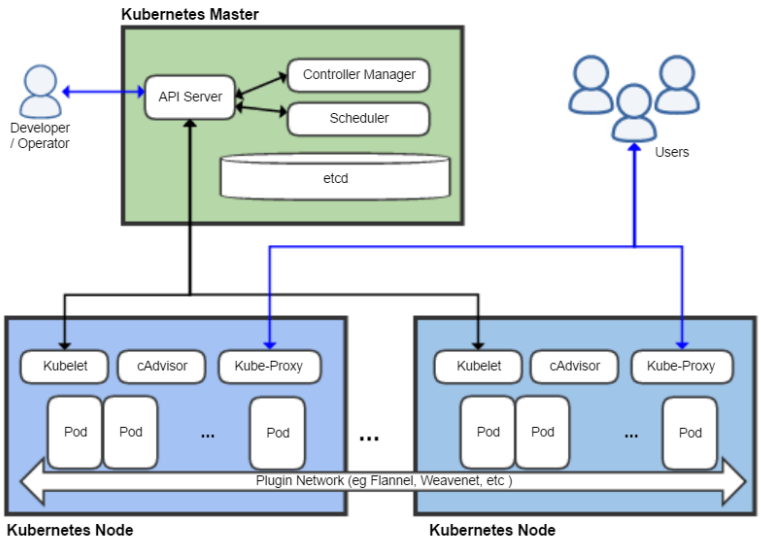
Kubernetes Features

- **Horizontal scaling**: Scale your application up and down with a simple command, with a UI, or automatically based on CPU usage.
- **Self-healing:** Restarts containers that fail, replaces and reschedules containers when nodes die, kills containers that don't respond to your user-defined health check, and doesn't advertise them to clients until they are ready to serve.
- **Service discovery and load balancing:** No need to modify your application to use an unfamiliar service discovery mechanism. Kubernetes gives containers their own IP addresses and a single DNS name for a set of containers, and can load-balance across them.
- **Storage Orchestration:** Automatically mount the storage system of your choice, whether from local storage, a public cloud provider

https://kubernetes.io/

Container Orchestration: Kubernetes





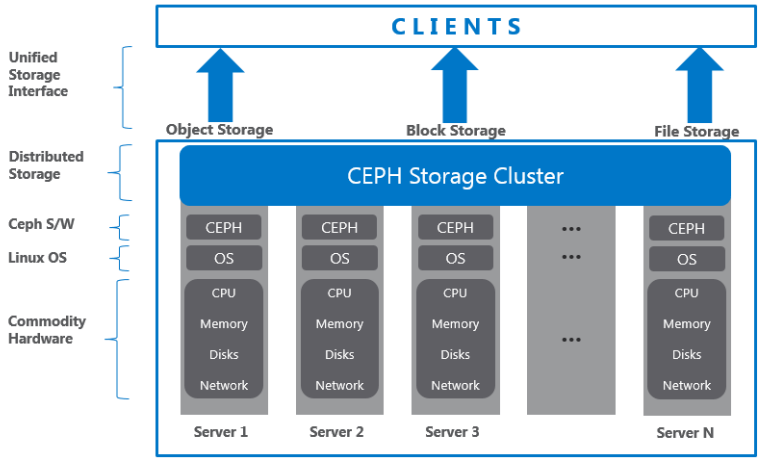
Kubernetes cluster consists of at least one master and multiple compute nodes.

The **master** is responsible for exposing the application program interface (**API**), scheduling the deployments and managing the overall cluster.

Pod is consists of one or more containers that are guaranteed to be co-located on the host machine and can share resources. Each pod is assigned a unique IP address within the cluster, which allows applications to use ports without the risk of conflict.

https://kubernetes.io/

Storage: Ceph and Rook (1/2)

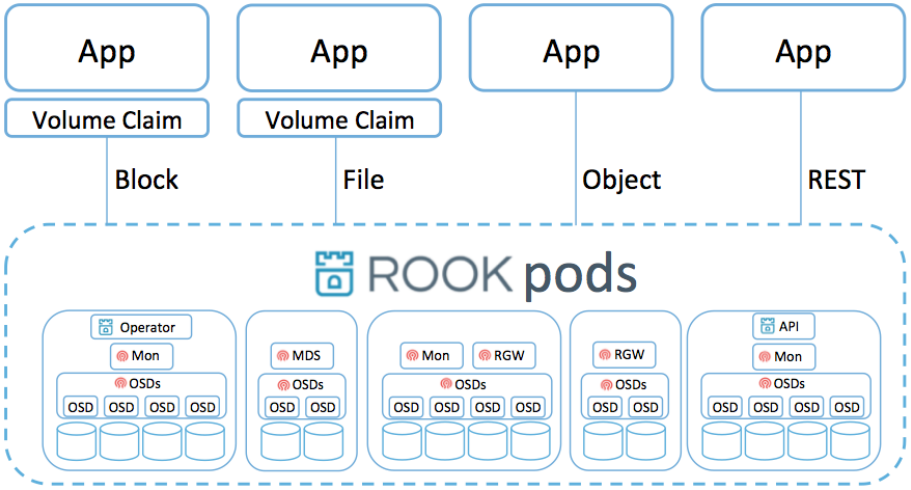


Ceph is a unified, distributed storage system designed for excellent performance, reliability and scalability.

Ceph provide Ceph Object Storage and/or Ceph Block Device services to Cloud Platforms, deploy a Ceph Filesystem or use Ceph for another purpose, all Ceph Storage Cluster deployments begin with setting up each Ceph Node, your network, and the Ceph Storage Cluster.

A Ceph Storage Cluster requires at least one Ceph Monitor, Ceph Manager, and Ceph OSD (Object Storage Daemon). The Ceph Metadata Server is also required when running Ceph Filesystem clients.

Storage: Ceph and Rook (2/2)

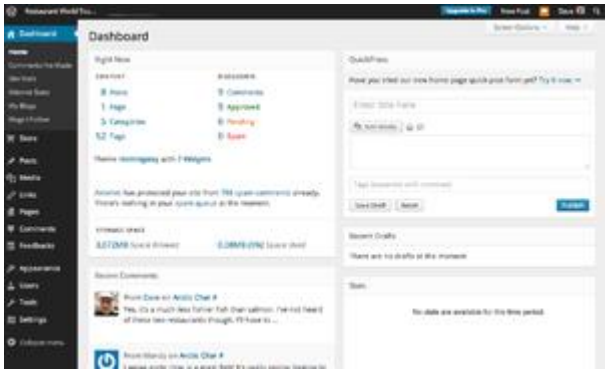


Rook is an open source cloud-native **Ceph storage orchestrator** for Kubernetes, providing the platform, framework, and support for a diverse set of storage solutions to natively integrate with cloud-native environments.

Rook turns storage software into self-managing, self-scaling, and self-healing storage services. It does this by automating deployment, bootstrapping, configuration, provisioning, scaling, upgrading, migration, disaster recovery, monitoring, and resource management. Rook uses the facilities provided by the underlying cloud-native container management, scheduling and orchestration platform to perform its duties.

https://github.com/rook/rook

WordPress: Sample Web Application



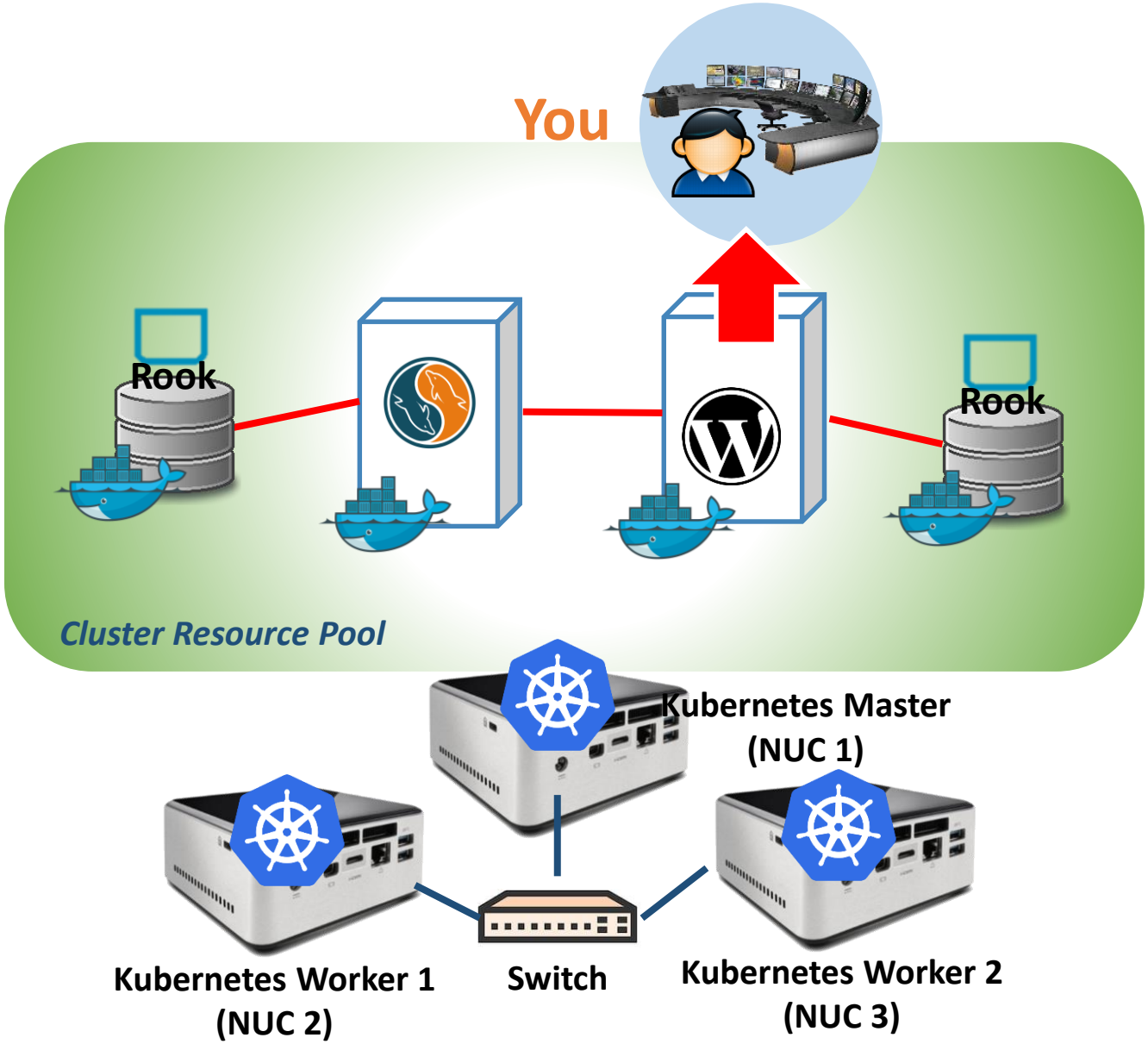
WordPress is the most popular online Open source publishing platform, currently powering more than 28% of the web. Used by more than 60 million websites,[5] including 30.6% of the top 10 million websites as of April 2018, WordPress is the most popular website management system in use.

You can start a blog or build a website in seconds without any technical knowledge!

https://wordpress.com/

Practice

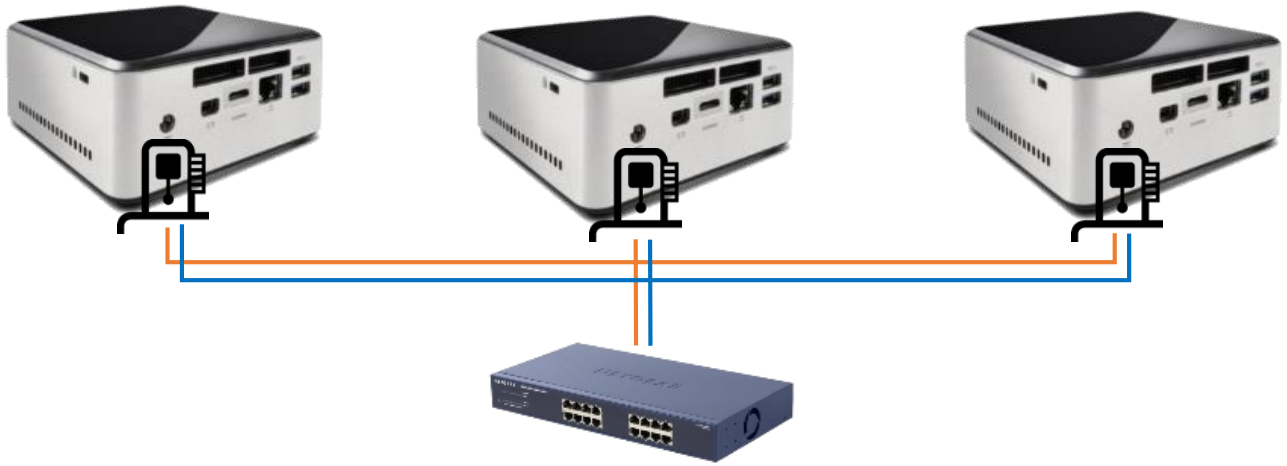
Cluster Lab: Concept



#0 - Lab Preparation (1/4)

Wired connection

NAME: NUC5i5MYHE (NUC PC)
CPU: i5-5300U @2.30GHz
CORE: 4
Memory: 16GB DDR3
HDD: 94GB



NAME: netgear prosafe 16 port gigabit switch(Switch)
Network Ports: 16 auto-sensing 10/100/1000 Mbps Ethernet ports

#0 - Lab Preparation (2/4)

- Boxes preparation: interconnect NUCs

For **NUC1**

In new terminal

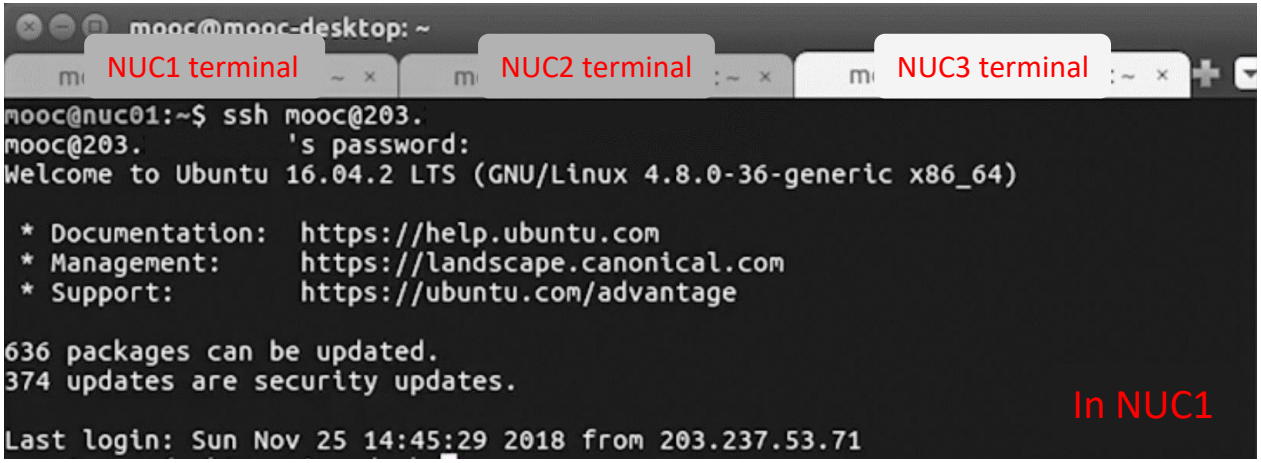
$ ssh <nuc2 name>@<nuc2 IP address>

> <nuc2 name>@<nuc2 IP address>'s password : <nuc2 pw>

In another new terminal

$ ssh <nuc3 name>@<nuc3 IP address>

> <nuc3 name>@<nuc3 IP address>'s password : <nuc3 pw>



#0 - Lab Preparation (3/4)

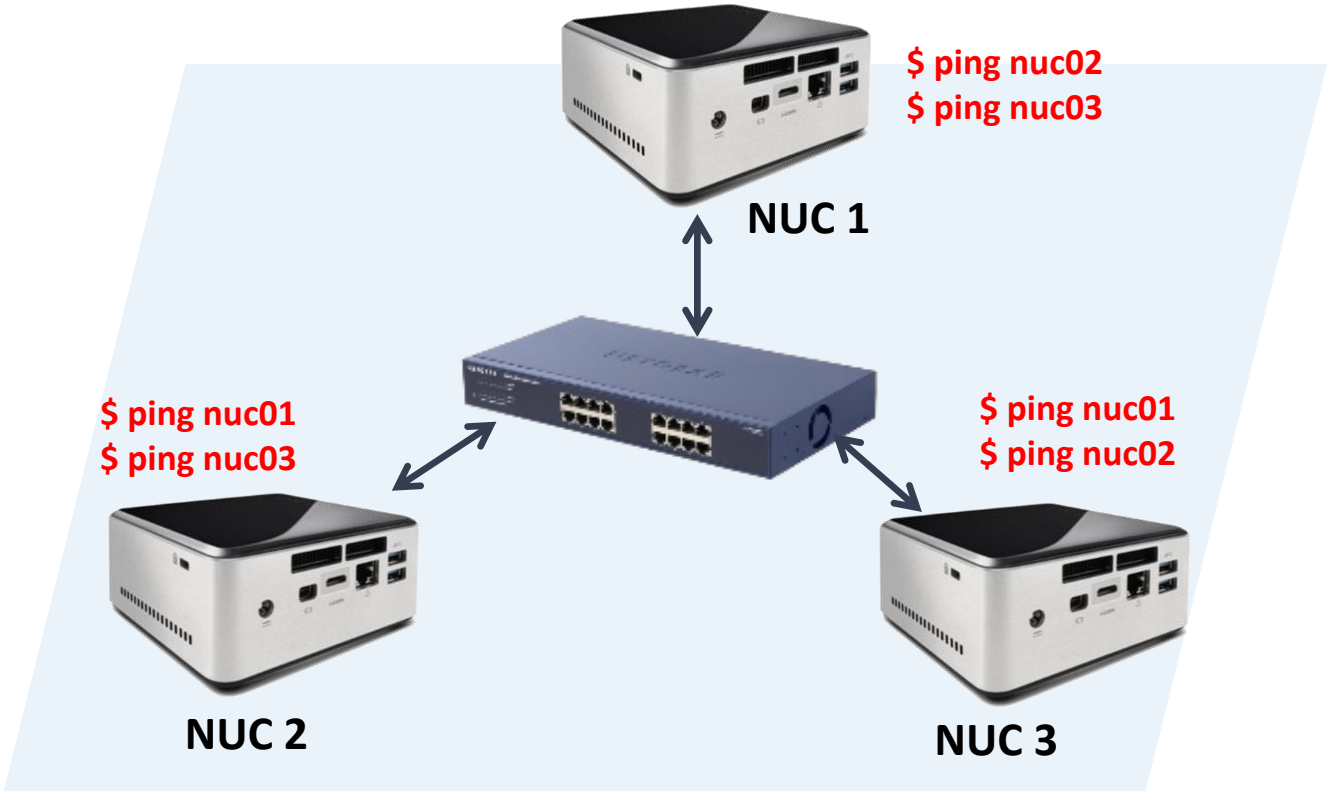
- Boxes preparation: configure hostname

1. From **NUC 1** :
   ```
   $ sudo hostname nuc01
   ```
2. From **NUC 2** :
   ```
   $ sudo hostname nuc02
   ```
3. From **NUC 3 :**
   ```
   $ sudo hostname nuc03
   ```

4. Edit /etc/hosts from **all NUCs** :
   ```
   $ sudo vi /etc/hosts
   ```
5. Append the following context into /etc/hosts :
   ```
   127.0.0.1       localhost
   IP Address of NUC 1   nuc01
   IP Address of NUC 2   nuc02
   IP Address of NUC 3   nuc03
   ```

#0 - Lab Preparation (4/4)

- Boxes preparation:
  Check full network connectivity between all boxes



#1-1 Preparations for Clustering: Prerequisites Installation

For **All NUCs**

A prerequisite for kubernetes: Docker Installation

```
$ sudo apt-get update
$ sudo apt-get install apt-transport-https ca-certificates curl software-properties-common
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
$ sudo apt-key fingerprint 0EBFCD88
$ sudo add-apt-repository \
  "deb [arch=amd64] https://download.docker.com/linux/ubuntu \
  $(lsb_release -cs) \
  stable"
$ sudo apt-get update
$ sudo apt-get install docker-ce
```

A prerequisite for ROOK : xfsprogs installation

```
$ sudo apt-get install xfsprogs
```

#1-2 Preparations for Clustering: Kubernetes Installation

Kubernetes Master
(NUC01)

Kubernetes Worker 1
(NUC02)

Kubernetes Worker 2
(NUC03)

We will deploy kubernetes on All NUCs and configure as above

#1-2 Preparations for Clustering: Kubernetes Installation

For **All NUCs**

```
$ sudo su
$ apt-get update && apt-get install -y apt-transport-https curl ipvsadm wget
$ curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | apt-key add -
$ cat <<EOF | tee /etc/apt/sources.list.d/kubernetes.list
> deb http://apt.kubernetes.io/ kubernetes-xenial main
> EOF
$ curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -

$ apt-get update
$ apt-get install -y --allow-downgrades kubelet=1.14.1-00 kubeadm=1.14.1-00
  kubectl=1.14.1-00 kubernetes-cni=0.7.5-00
$ exit
```

Disable root permissions

#1-3 Preparations for Clustering: Kubernetes Configuration

For **All NUCs**

If swap is not disabled, kubelet service will not start on the masters and nodes. First, you need to edit kubernetes configure file

```
$ sudo vi /etc/systemd/system/kubelet.service.d/10-kubeadm.conf
→ Add [ Environment="KUBELET_EXTRA_ARGS=--fail-swap-on=false" ]
$ sudo systemctl daemon-reload
$ sudo systemctl restart kubelet
```

```
[Service]
Environment="KUBELET_KUBECONFIG_ARGS=--bootstrap-kubeconfig=/etc/kubernetes/bootstrap-kubelet.conf --kubeconfig=/etc/kubernetes/kubelet.conf"
Environment="KUBELET_SYSTEM_PODS_ARGS=--pod-manifest-path=/etc/kubernetes/manifests --allow-privileged=true"
Environment="KUBELET_NETWORK_ARGS=--network-plugin=cni --cni-conf-dir=/etc/cni/net.d --cni-bin-dir=/opt/cni/bin"
Environment="KUBELET_DNS_ARGS=--cluster-dns=10.96.0.10 --cluster-domain=cluster.local"
Environment="KUBELET_AUTHZ_ARGS=--authorization-mode=Webhook --client-ca-file=/etc/kubernetes/pki/ca.crt"
Environment="KUBELET_CADVISOR_ARGS=--cadvisor-port=0"
Environment="KUBELET_CERTIFICATE_ARGS=--rotate-certificates=true --cert-dir=/var/lib/kubelet/pki"
Environment="KUBELET_EXTRA_ARGS=--fail-swap-on=false"
ExecStart=
ExecStart=/usr/bin/kubelet $KUBELET_KUBECONFIG_ARGS $KUBELET_SYSTEM_PODS_ARGS $KUBELET_NETWORK_ARGS $KUBELET_DNS_ARGS $KUBELET_AUTHZ_ARGS $KUBEL
~
```

Add the line like this

For **All NUCs**

Run the following command to disable swap immediately.

```
$ sudo swapoff -a
$ sudo sed -e '/\/swapfile/s/^/#/g' -i /etc/fstab
$ sudo sed -e '/\/swap\.img/s/^/#/g' -i /etc/fstab
```

#2-1 Clustering: Running Kubernetes Master

For **NUC1**

```
$ sudo systemctl daemon-reload
$ sudo systemctl restart kubelet
$ sudo kubeadm reset -f
$ sudo rm -rf /etc/cni/net.d
$ sudo ipvsadm --clear
$ sudo rm -rf /var/lib/rook
$ sudo kubeadm init  --ignore-preflight-errors=all
```

```
Then you can join any number of worker nodes by running the following on each as root:

kubeadm join                    --token z08hnt.so0bcgho51lpe1dk \
    --discovery-token-ca-cert-hash sha256:0060e9bf66f9b38609c3a6196fa0267570902385b77deb8130df3c901e1c3eca
```

You need to copy the command for joining kubernetes nodes (NUC2, NUC3)

Please copy it!

#2-1 Clustering: Running Kubernetes Master

For **NUC1**

Run these commands to make kubectl work for your non-root user.
When you run these commands, make sure that you are not in root

```
$ mkdir -p $HOME/.kube
$ sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
$ sudo chown $(id -u):$(id -g) $HOME/.kube/config
$ kubectl taint nodes --all node-role.kubernetes.io/master-
```

#2-2 Clustering: Joining k8s nodes to k8s master

You can join NUC2, NUC3 to k8s master(NUC1) by following commands

For **NUC2, NUC3**

```
$ sudo systemctl daemon-reload
$ sudo systemctl restart kubelet
$ sudo kubeadm reset -f
$ sudo rm -r /etc/cni/net.d
$ sudo ipvsadm --clear
$ sudo rm -rf /var/lib/rook
$ sudo kubeadm join NUC1 IP:6443 --token YOUR TOKEN --discovery-token-ca-cert-hash
YOUR HASH --ignore-preflight-errors=all → Paste the command you just copied
```

If ipvsadm is not installed,

```
$ sudo apt-get install ipvsadm
```

```
This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.
```

Check all nodes are joined

For **NUC1**

```
$ kubectl get node
```

```
netcs@nuc01:~$ kubectl get nodes
NAME      STATUS     ROLES     AGE       VERSION
nuc01     NotReady   master    1h        v1.11.2
nuc02     NotReady   <none>    6s        v1.11.2
nuc03     NotReady   <none>    4s        v1.11.2
```

#2-3 Clustering: k8s Network Plugin Installation

You MUST install a pod network add-on so that your pods can communicate with each other. We will use Weave in this lab.

For **NUC1**
Install Weave (pod network add-on)

```
$ kubectl apply -f "https://cloud.weave.works/k8s/net?k8s-version=$(kubectl version | base64 | tr -d '\n')"
```

For **NUC1**
Make sure Weave(network add-on) works

```
$ kubectl get nodes
$ kubectl get po -n kube-system -o wide
```

```
netcs@nuc01:~$ kubectl get nodes
NAME      STATUS    ROLES     AGE       VERSION
nuc01     Ready     master    1h        v1.11.2
nuc02     Ready     <none>    5m        v1.11.2
nuc03     Ready     <none>    5m        v1.11.2
```

```
netcs@nuc01:~$ kubectl get po -n kube-system
NAME                              READY     STATUS    RESTARTS   AGE
coredns-78fcdf6894-6zktx          1/1       Running   0          1h
coredns-78fcdf6894-glzvq          1/1       Running   0          1h
etcd-nuc01                        1/1       Running   0          1h
kube-apiserver-nuc01              1/1       Running   0          1h
kube-controller-manager-nuc01     1/1       Running   0          1h
kube-proxy-6d846                  1/1       Running   0          8m
kube-proxy-6n8qr                  1/1       Running   0          1h
kube-proxy-r82r7                  1/1       Running   0          8m
kube-scheduler-nuc01              1/1       Running   0          1h
weave-net-bw4dz                   2/2       Running   0          3m
weave-net-ns426                   2/2       Running   0          3m
weave-net-zbs26                   2/2       Running   0          3m
```

#2-3 Clustering: ROOK Installation

For **NUC1**

Set Privileges on cluster for ROOK

```
$ kubectl create clusterrolebinding permissive-binding \
  --clusterrole=cluster-admin \
  --user=admin \
  --user=kubelet \
  --group=system:serviceaccounts
```

For **NUC1**

Install ROOK Storage on your cluster

```
$ cd $HOME
$ git clone --single-branch --branch release-1.2 https://github.com/rook/rook.git
$ cd $HOME/rook/cluster/examples/kubernetes/ceph
$ kubectl create -f common.yaml
$ kubectl create -f operator.yaml
$ kubectl create -f cluster-test.yaml
```

#2-3 Clustering: ROOK Installation

For **NUC1**
Check rook-ceph-pod

```
$ watch kubectl get pod -n rook-ceph
```

```
Every 2.0s: kubectl get pod -n rook-ceph                                nuc1-NUC

AME                                                READY   STATUS    RESTARTS   AGE
csi-cephfsplugin-2c2bv                             3/3     Running   0          2m21s
csi-cephfsplugin-5q7z7                             3/3     Running   0          2m21s
csi-cephfsplugin-provisioner-775bb57749-n4h62      4/4     Running   0          2m21s
csi-cephfsplugin-provisioner-775bb57749-tnbkh      4/4     Running   0          2m21s
csi-rbdplugin-2qlnw                                3/3     Running   0          2m21s
csi-rbdplugin-m8sgk                                3/3     Running   0          2m21s
csi-rbdplugin-provisioner-65bdfb4644-5dt2p         5/5     Running   0          2m21s
csi-rbdplugin-provisioner-65bdfb4644-rp4rc         5/5     Running   0          2m21s
rook-ceph-mon-a-canary-776bd8bd9b-bszzv            1/1     Running   0          83s
rook-ceph-mon-b-canary-8f9d48567-bw2lb             1/1     Running   0          82s
rook-ceph-mon-c-canary-7d655d4f7d-8qw6s            1/1     Running   0          82s
rook-ceph-operator-8644d9bd85-2rdnh                1/1     Running   0          2m42s
rook-discover-8mnnk                                1/1     Running   0          2m24s
rook-discover-xd8r2                                1/1     Running   0          2m24s
```

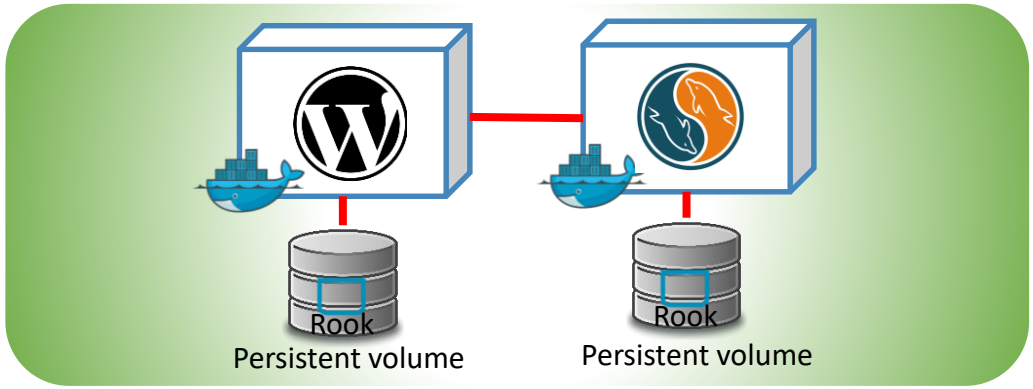
For **NUC1**
Install ToolBox and Execute ToolBox

```
$ cd $HOME/rook/cluster/examples/kubernetes/ceph
$ kubectl create -f toolbox.yaml
$ kubectl -n rook-ceph  rollout status deploy/rook-ceph-tools
$ kubectl -n rook-ceph exec -it $(kubectl -n rook-ceph get pod -l “app=rook-ceph-tools”
  -o jsonpath=‘{.items[0].metadata.name}’) bash
<in the toolbox>
$ watch ceph status
$ exit
```

For **NUC1** **Add StorageClass**

```
$ kubectl apply -f csi/rbd/storageclass-test.yaml
```

#2-4 Cluster Validation: Deploy Sample Web application



For **NUC1**

Deploy a Sample Web Application (WordPress) on your cluster.

```
$ kubectl create -f $HOME/rook/cluster/examples/kubernetes/mysql.yaml
$ kubectl create -f $HOME/rook/cluster/examples/kubernetes/wordpress.yaml
```

For **NUC1**

Check Wordpress containers are started

```
$ watch kubectl get pod
```

```
Every 2.0s: kubectl get po

AME                               READY   STATUS    RESTARTS   AGE
wordpress-595685cc49-ct9fl         1/1     Running   0          50s
wordpress-mysql-b78774f44-llgst    1/1     Running   0          56s
```

#2-4 Cluster Validation: Deploy Sample Web application

For **NUC1**

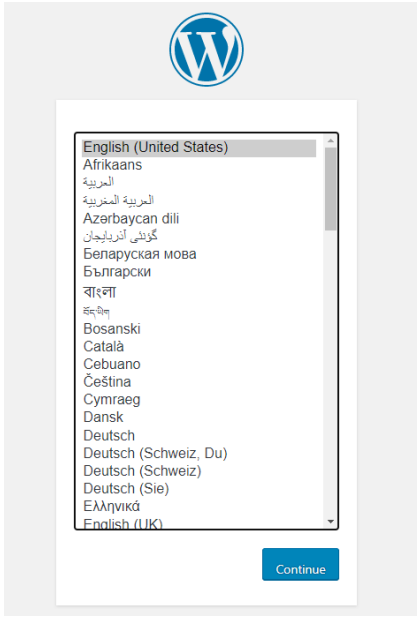
Check exposed port for Wordpress service.

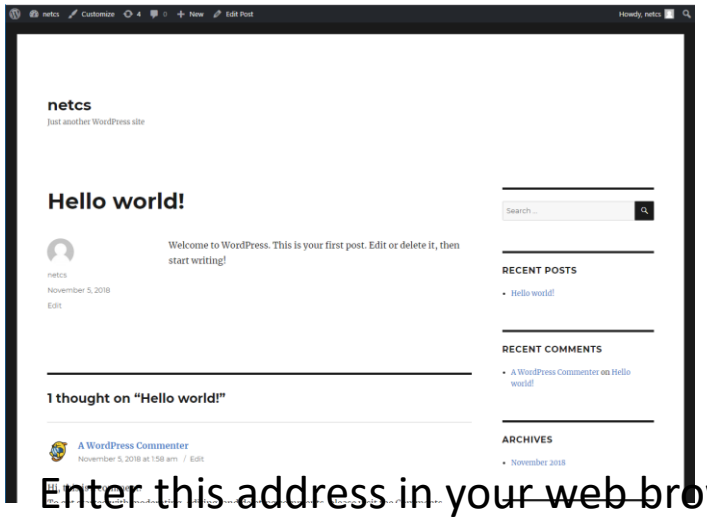
The address of your Wordpress Web is http://your NUC1 IP: exposed port

$ kubectl get svc

exposed port

```
nuc1@nuc1-NUC8i3BEK:~/rook/cluster/examples/kubernetes$ kubectl get svc
NAME              TYPE           CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
kubernetes        ClusterIP      10.96.0.1        <none>        443/TCP        160m
wordpress         LoadBalancer   10.107.116.108   <pending>     80:31671/TCP   20m
wordpress-mysql   ClusterIP      None             <none>        3306/TCP       20m
```





Enter this address in your web browser:
http://**your NUC1 IP**:**exposed port**

Review

Lab Summary

With Cluster Lab, you have experimented

1. How to physically connect multiple NUCs with a switch to create the targeted physical cluster?
2. How to install (deploy) and configure container-orchestrated software functions to enable the desired cluster role (e.g., HPC, HPDA, ML/DL, …)

3. Understanding how container orchestration is working: Can you distinguish Kubernetes master from Kubernetes nodes?

Thank You for Your Attention
Any Questions?

mini@smartx.kr