The bridge to possible

Learning and Using Kubernetes with a Cluster of Raspberry Pis

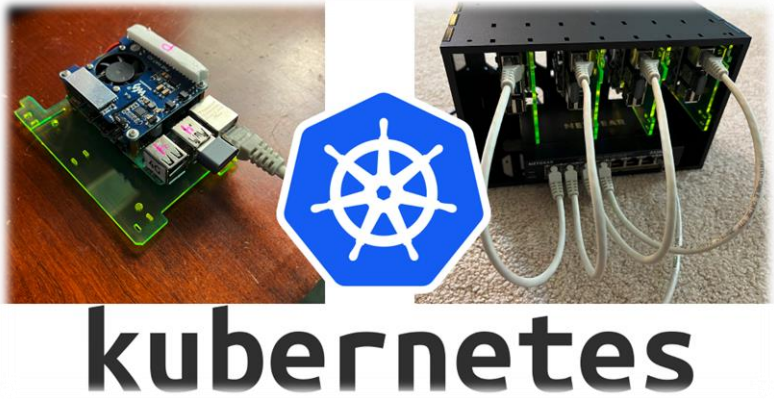
Part 2 of 2

Jason Davis, Distinguished Engineer
@SNMPguy

Agenda

- Introduction
- Some Basic Kubernetes commands
- Installing MetalLB
- Layer-2 network integration
- Layer-3 network integration
- Other Uses
- Conclusion

Recap



- Part 1, DEVNET-1279, we built a cluster of Raspberry Pis, installed microK8s and joined the nodes to a cluster
- In Part 2, DEVNET-2279 (THIS session), we will learn and use some microK8s administrative commands and configure MetalLB to interoperate with a routed network using BGP
- Join Part 2 directly if you have an existing microK8s cluster or Linux VMs already provisioned

NOTE: Cloud-based K8s clusters do not use MetalLB, as there are other mechanisms for network integration

Component View of Cluster


c-m
api
etcd
sched
kubelet
k-proxy
Pod
Node
kubelet
k-proxy
Pod
Node
kubelet
k-proxy
Pod
Node
kubelet
k-proxy
Pod
Node
Cluster

Get microK8s status

microk8s status

```
jason@pi-cluster-node1:~$ microk8s status
microk8s is running
high-availability: no
addons:
  enabled:
    dns                  # (core) CoreDNS
    helm                 # (core) Helm - the package manager for Kubernetes
    helm3                # (core) Helm 3 - the package manager for Kubernetes
    hostpath-storage     # (core) Storage class; allocates storage from host directory
    metallb              # (core) Loadbalancer for your Kubernetes cluster
    storage              # (core) Alias to hostpath-storage add-on, deprecated
  disabled:
    cert-manager         # (core) Cloud native certificate management
    community            # (core) The community addons repository
    dashboard            # (core) The Kubernetes dashboard
    ha-cluster           # (core) Configure high availability on the current node
    host-access          # (core) Allow Pods connecting to Host services smoothly
    ingress              # (core) Ingress controller for external access
    kube-ovn             # (core) An advanced network fabric for Kubernetes
    mayastor             # (core) OpenEBS MayaStor
    metrics-server       # (core) K8s Metrics Server for API access to service metrics
   . . .
```

Run from controller node

Get K8s nodes

```
microk8s kubectl get nodes -o wide
```

```
jason@pi-cluster-node1:~$ microk8s kubectl get nodes -o wide
NAME                STATUS   ROLES    AGE   VERSION   INTERNAL-IP     EXTERNAL-IP   OS-IMAGE
KERNEL-VERSION      CONTAINER-RUNTIME
pi-cluster-node1    Ready    <none>   23d   v1.26.0   192.168.1.161   <none>        Ubuntu 22.04.1 LTS
5.15.0-1021-raspi   containerd://1.6.8
pi-cluster-node2    Ready    <none>   22d   v1.26.0   192.168.1.162   <none>        Ubuntu 22.04.1 LTS
5.15.0-1021-raspi   containerd://1.6.8
pi-cluster-node3    Ready    <none>   21d   v1.26.0   192.168.1.163   <none>        Ubuntu 22.04.1 LTS
5.15.0-1021-raspi   containerd://1.6.8
pi-cluster-node4    Ready    <none>   21d   v1.26.0   192.168.1.164   <none>        Ubuntu 22.04.1 LTS
5.15.0-1021-raspi   containerd://1.6.8
```

Determine node function by label

```
microk8s kubectl get nodes -l 'node.kubernetes.io/microk8s-controlplane'
microk8s kubectl get nodes -l '!node.kubernetes.io/microk8s-controlplane'
```

```
jason@pi-cluster-node1:~$ microk8s kubectl get nodes -l 'node.kubernetes.io/microk8s-
controlplane'
NAME                STATUS   ROLES    AGE    VERSION
pi-cluster-node1    Ready    <none>   139d   v1.26.0
jason@pi-cluster-node1:~$ microk8s kubectl get nodes -l '!node.kubernetes.io/microk8s-
controlplane'
NAME                STATUS   ROLES    AGE   VERSION
pi-cluster-node2    Ready    <none>   30m   v1.26.0
pi-cluster-node3    Ready    <none>   24m   v1.26.0
pi-cluster-node4    Ready    <none>   23m   v1.26.0
```

Labeling for custom querying

```
microk8s kubectl label node <node> node-role.kubernetes.io/<label_info>
```

```
jason@pi-cluster-node1:~$ microk8s kubectl label node pi-cluster-node1 node-
role.kubernetes.io/controller=controller
node/pi-cluster-node1 labeled
jason@pi-cluster-node1:~$ microk8s kubectl label node pi-cluster-node2 node-
role.kubernetes.io/worker=worker
node/pi-cluster-node2 labeled
jason@pi-cluster-node1:~$ microk8s kubectl label node pi-cluster-node3 node-
role.kubernetes.io/worker=worker
node/pi-cluster-node3 labeled
jason@pi-cluster-node1:~$ microk8s kubectl label node pi-cluster-node4 node-
role.kubernetes.io/worker=worker
node/pi-cluster-node4 labeled
```

Get K8s services

```
microk8s kubectl get services
```

```
jason@pi-cluster-node1:~$ microk8s kubectl get services
NAME         TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.152.183.1   <none>        443/TCP   139d
```

Create a deployment

Container hostname: microbot-7dc9ffb6cf-mpfz7

```
jason@pi-cluster-node1:~$ microk8s kubectl create deployment microbot \
--image=cdkbot/microbot-arm64:latest
deployment.apps/microbot created
jason@pi-cluster-node1:~$ microk8s kubectl scale deployment microbot --replicas=4
deployment.apps/microbot scaled

jason@pi-cluster-node1:~$ microk8s kubectl get pods -A
NAMESPACE     NAME                                       READY   STATUS    RESTARTS
default       microbot-7dc9ffb6cf-5khxp                  1/1     Running
default       microbot-7dc9ffb6cf-8qlsb                  1/1     Running
default       microbot-7dc9ffb6cf-stgqd                  1/1     Running
default       microbot-7dc9ffb6cf-vnxcl                  1/1     Running
kube-system   coredns-6f5f9b5d74-ggj94                   1/1     Running
kube-system   hostpath-provisioner-69cd9ff5b8-mvffv      1/1     Running
```

microbot is a convenient docker image to demonstrate K8s functions

It is VERY small; based on Alpine Linux with NGINX web-server showing a static image and the K8s node identifier

https://hub.docker.com/r/cdkbot/microbot-arm64

Expose the service

- Initially we'll define a service as a NodePort
- Each node will expose the same port number individually
- Directed queries to the node's IP with the identified port will get that pod instance's output

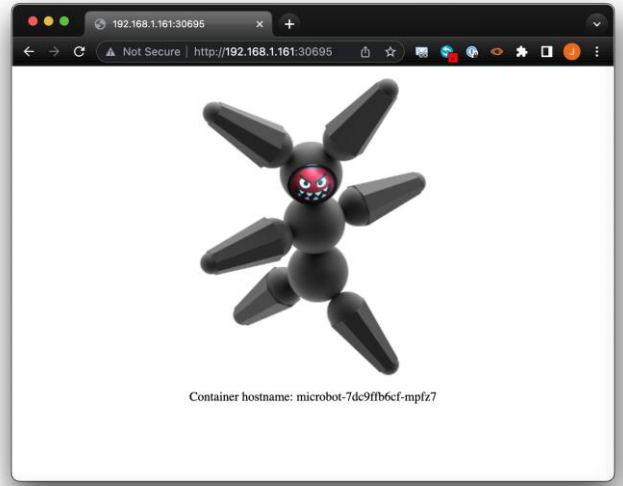
```
jason@pi-cluster-node1:~$ microk8s kubectl expose deployment microbot --type=NodePort --
port=80 --name=microbot-service
service/microbot-service exposed
```

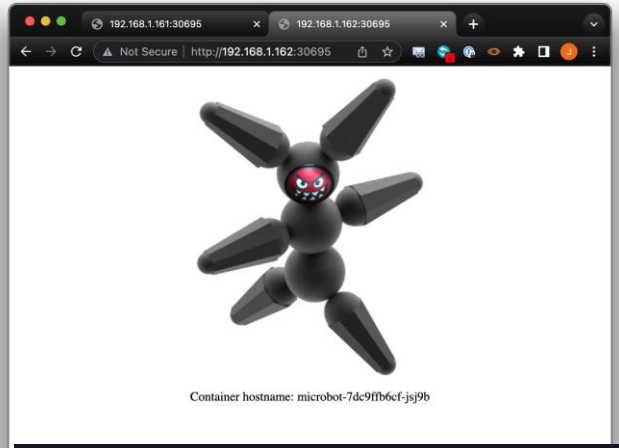
View the deployment

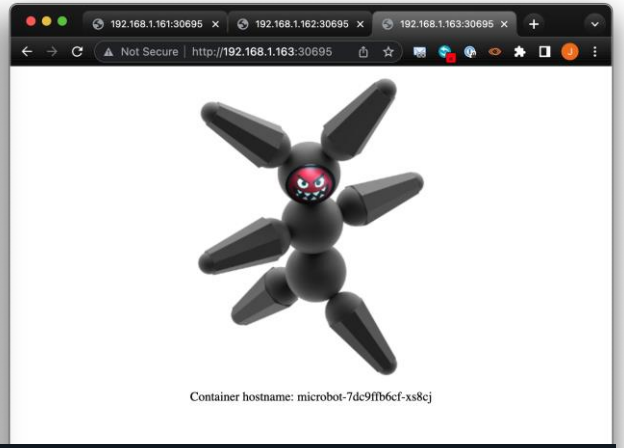
```
jason@pi-cluster-node1:~$ microk8s kubectl get all -n default
NAME                                READY   STATUS    RESTARTS   AGE
pod/microbot-7dc9ffb6cf-jsj9b       1/1     Running   0          4m15s
pod/microbot-7dc9ffb6cf-mpfz7       1/1     Running   0          4m15s
pod/microbot-7dc9ffb6cf-xs8cj       1/1     Running   0          4m15s
pod/microbot-7dc9ffb6cf-zcxt4       1/1     Running   0          4m15s

NAME                        TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)        
service/kubernetes          ClusterIP   10.152.183.1    <none>        443/TCP        22d
service/microbot-service    NodePort    10.152.183.98   <none>        80:30695/TCP   5s

NAME                        READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/microbot    4/4     4            4           13m

NAME                                   DESIRED   CURRENT   READY   AGE
replicaset.apps/microbot-7dc9ffb6cf    4         4         4       13m
```

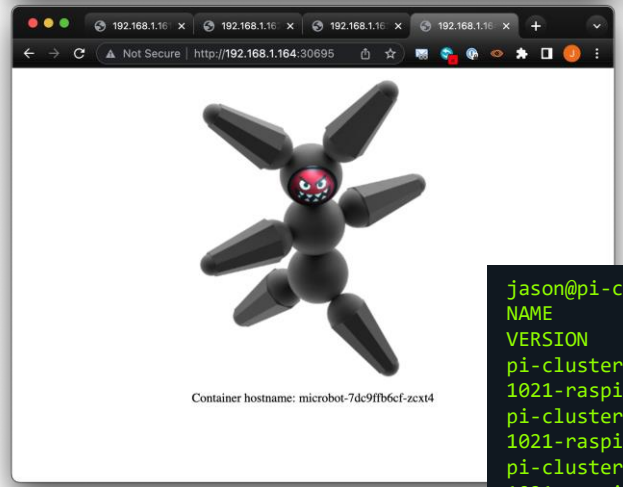
Great!

Now we can access via NodePort externally



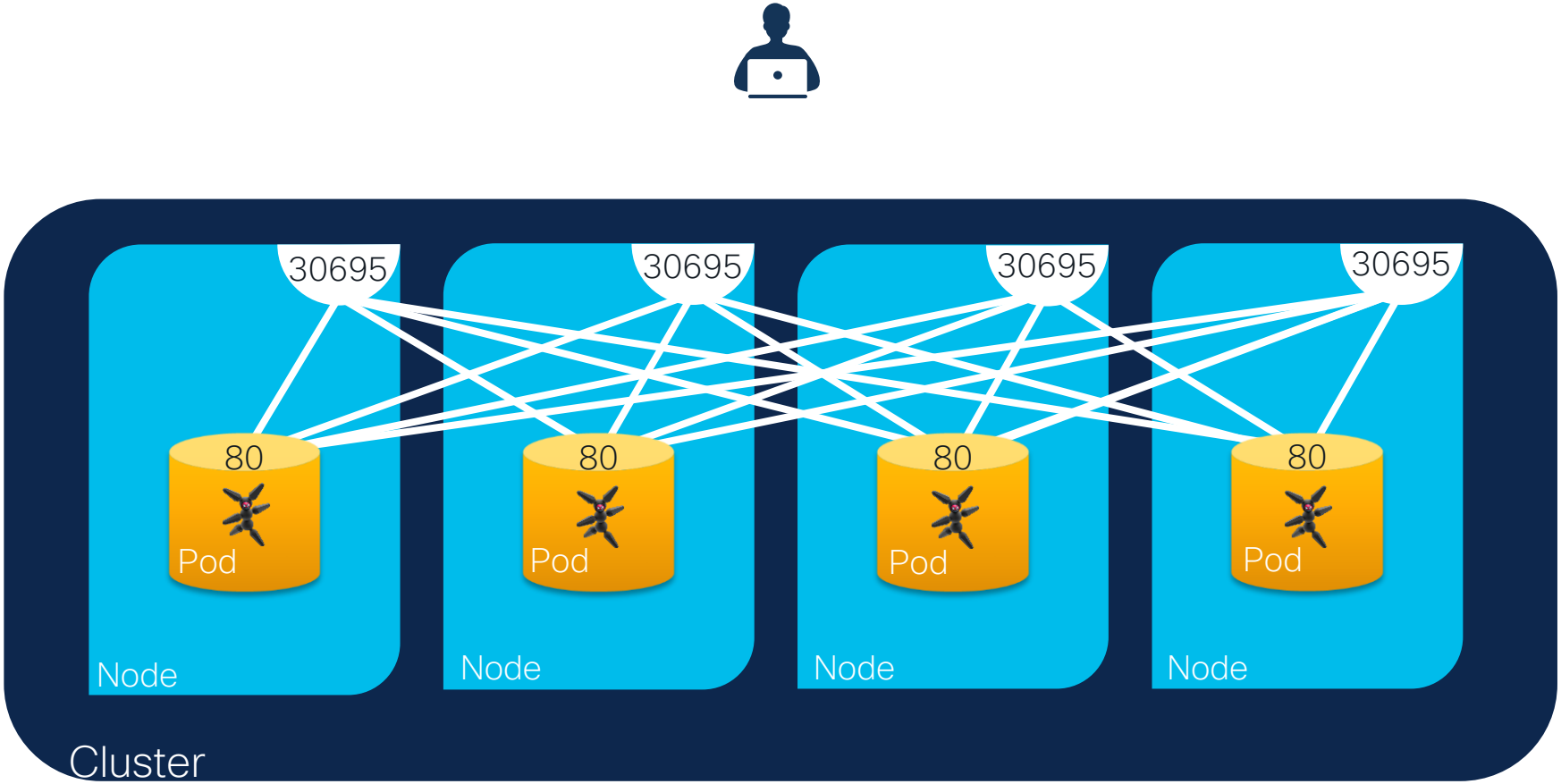







```
jason@pi-cluster-node1:~$ microk8s kubectl get pods,svc -o wide
NAME                                  READY   STATUS    RESTARTS   AGE   IP           NODE
pod/microbot-7dc9ffb6cf-jsj9b         1/1     Running   0          32m   10.1.27.5    pi-cluster-node2
pod/microbot-7dc9ffb6cf-mpfz7         1/1     Running   0          32m   10.1.1.7     pi-cluster-node1
pod/microbot-7dc9ffb6cf-xs8cj         1/1     Running   0          32m   10.1.81.3    pi-cluster-node3
pod/microbot-7dc9ffb6cf-zcxt4         1/1     Running   0          32m   10.1.100.6   pi-cluster-node4

NAME                      TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE   SELECTOR
service/kubernetes        ClusterIP   10.152.183.1    <none>        443/TCP        22d   <none>
service/microbot-service  NodePort    10.152.183.98   <none>        80:30695/TCP   28m
app=microbot
```

```
jason@pi-cluster-node1:~$ microk8s kubectl get nodes -o wide
NAME              STATUS   ROLES    AGE   VERSION   INTERNAL-IP     EXTERNAL-IP   OS-IMAGE             KERNEL-
VERSION      CONTAINER-RUNTIME
pi-cluster-node1   Ready    <none>   23d   v1.26.0   192.168.1.161   <none>        Ubuntu 22.04.1 LTS   5.15.0-
1021-raspi   containerd://1.6.8
pi-cluster-node2   Ready    <none>   22d   v1.26.0   192.168.1.162   <none>        Ubuntu 22.04.1 LTS   5.15.0-
1021-raspi   containerd://1.6.8
pi-cluster-node3   Ready    <none>   21d   v1.26.0   192.168.1.163   <none>        Ubuntu 22.04.1 LTS   5.15.0-
1021-raspi   containerd://1.6.8
pi-cluster-node4   Ready    <none>   21d   v1.26.0   192.168.1.164   <none>        Ubuntu 22.04.1 LTS   5.15.0-
1021-raspi   containerd://1.6.8
```



MetalLB

Connect Externally with LoadBalancer

- Get MetalLB as a K8s manifest
- https://metallb.universe.tf/installation/#installation-by-manifest

```
microk8s kubectl apply -f
https://raw.githubusercontent.com/metallb/metallb/v0.13.7/config/manifests/metallb-
native.yaml
```

Apply MetalLB manifest

```
jason@pi-cluster-node1:~$ microk8s kubectl apply -f
https://raw.githubusercontent.com/metallb/metallb/v0.13.7/config/manifests/metallb-native.yaml
namespace/metallb-system created
customresourcedefinition.apiextensions.k8s.io/addresspools.metallb.io created
customresourcedefinition.apiextensions.k8s.io/bfdprofiles.metallb.io created
customresourcedefinition.apiextensions.k8s.io/bgpadvertisements.metallb.io created
customresourcedefinition.apiextensions.k8s.io/bgppeers.metallb.io created
customresourcedefinition.apiextensions.k8s.io/communities.metallb.io created
customresourcedefinition.apiextensions.k8s.io/ipaddresspools.metallb.io created
customresourcedefinition.apiextensions.k8s.io/l2advertisements.metallb.io created
serviceaccount/controller created
serviceaccount/speaker created
role.rbac.authorization.k8s.io/controller created
role.rbac.authorization.k8s.io/pod-lister created
clusterrole.rbac.authorization.k8s.io/metallb-system:controller created
clusterrole.rbac.authorization.k8s.io/metallb-system:speaker created
rolebinding.rbac.authorization.k8s.io/controller created
rolebinding.rbac.authorization.k8s.io/pod-lister created
clusterrolebinding.rbac.authorization.k8s.io/metallb-system:controller created
clusterrolebinding.rbac.authorization.k8s.io/metallb-system:speaker created
secret/webhook-server-cert created
service/webhook-service created
deployment.apps/controller created
daemonset.apps/speaker created
validatingwebhookconfiguration.admissionregistration.k8s.io/metallb-webhook-configuration created
jason@pi-cluster-node1:~$
```

Review the metallb-system namespace created

```
jason@pi-cluster-node1:~$ microk8s kubectl get all -n metallb-system
NAME                                  READY   STATUS    RESTARTS   AGE
pod/controller-577b5bdfcc-7fflq       1/1     Running   0          31s
pod/speaker-bsr9f                     1/1     Running   0          31s
pod/speaker-k4txn                     1/1     Running   0          31s
pod/speaker-mk999                     1/1     Running   0          31s
pod/speaker-nmgdw                     1/1     Running   0          31s

NAME                      TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)   AGE
service/webhook-service   ClusterIP   10.152.183.227   <none>        443/TCP   31s

NAME                      DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR            AGE
daemonset.apps/speaker    4         4         4       4            4           kubernetes.io/os=linux   31s

NAME                         READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/controller   1/1     1            1           31s

NAME                                    DESIRED   CURRENT   READY   AGE
replicaset.apps/controller-577b5bdfcc   1         1         1       31s
jason@pi-cluster-node1:~$
```

Define and deploy a Layer-2 configmap

https://metallb.universe.tf/configuration/#layer-2-configuration

```yaml
---
apiVersion: metallb.io/v1beta1
kind: IPAddressPool
metadata:
  name: default
  namespace: metallb-system
spec:
  addresses:
    - 192.168.1.170-192.168.1.179
---
apiVersion: metallb.io/v1beta1
kind: L2Advertisement
metadata:
  name: default
  namespace: metallb-system
```

Use an available address range in the same subnet as the INTERNAL-IPs of your nodes

Check with: microk8s kubectl get nodes -o wide

save as 'cr-metallb-l2arp.yaml'

Apply the YAML definition

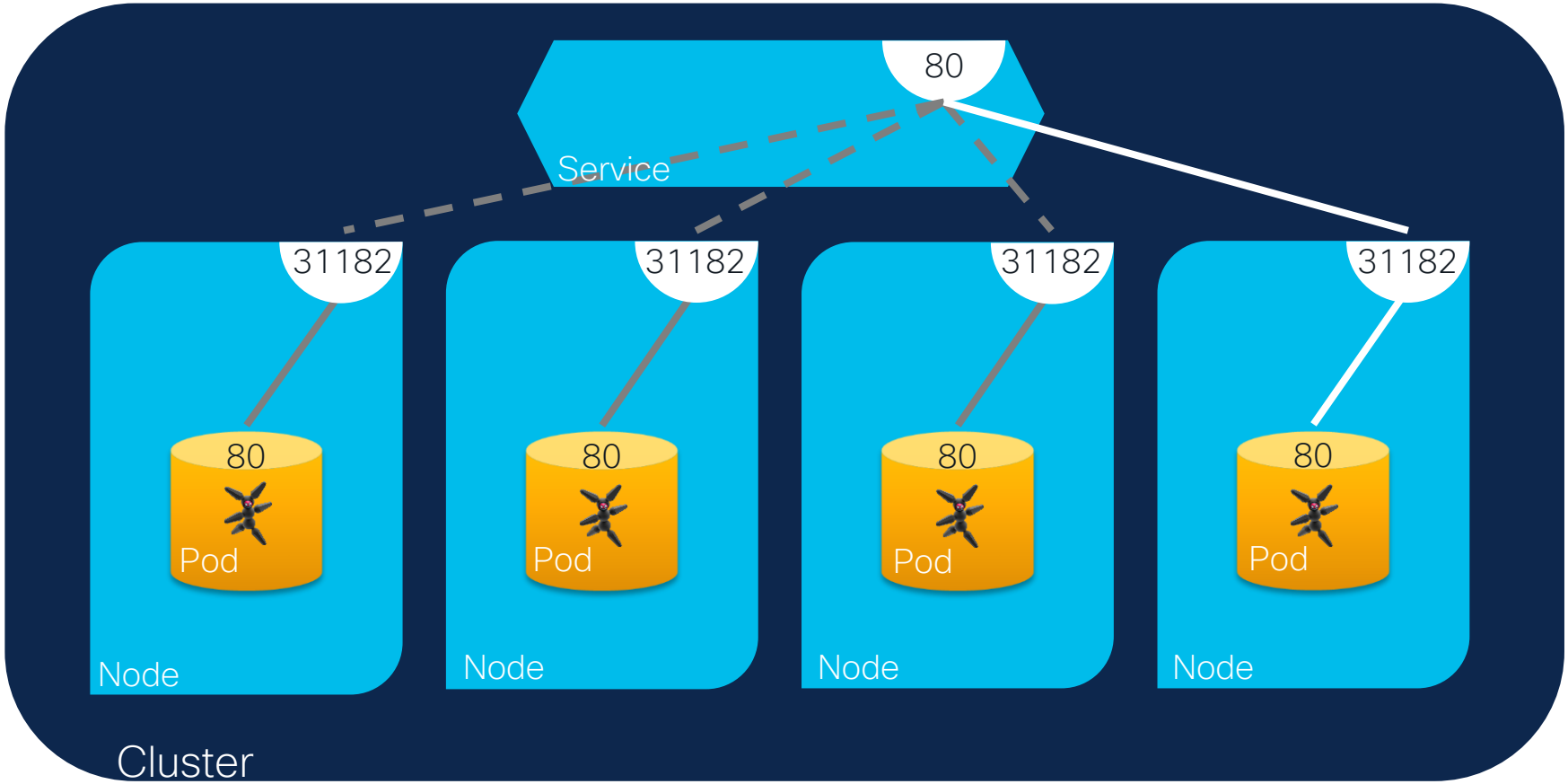
```
jason@pi-cluster-node1:~$ microk8s kubectl apply -f cr-metallb-l2arp.yaml
ipaddresspool.metallb.io/default created
l2advertisement.metallb.io/default created
```

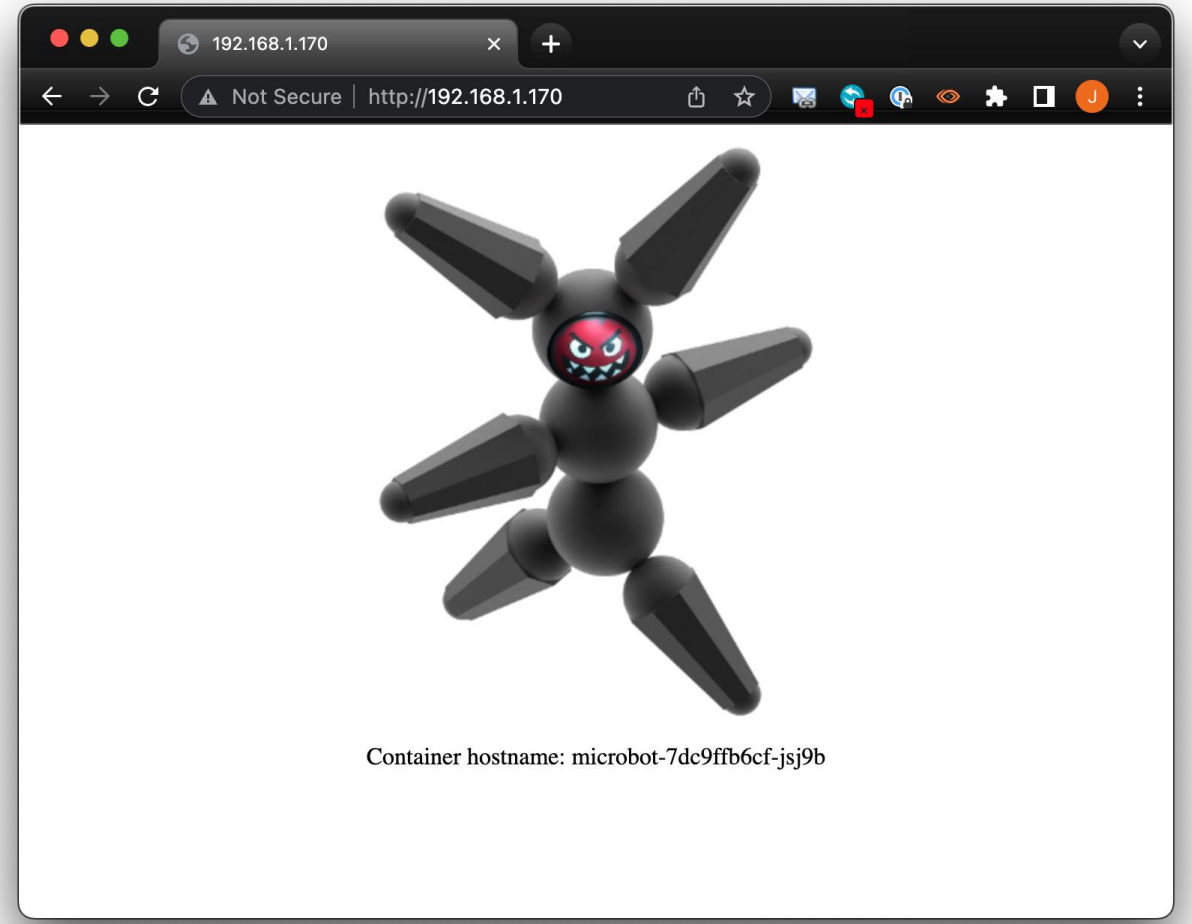
Apply service and get status

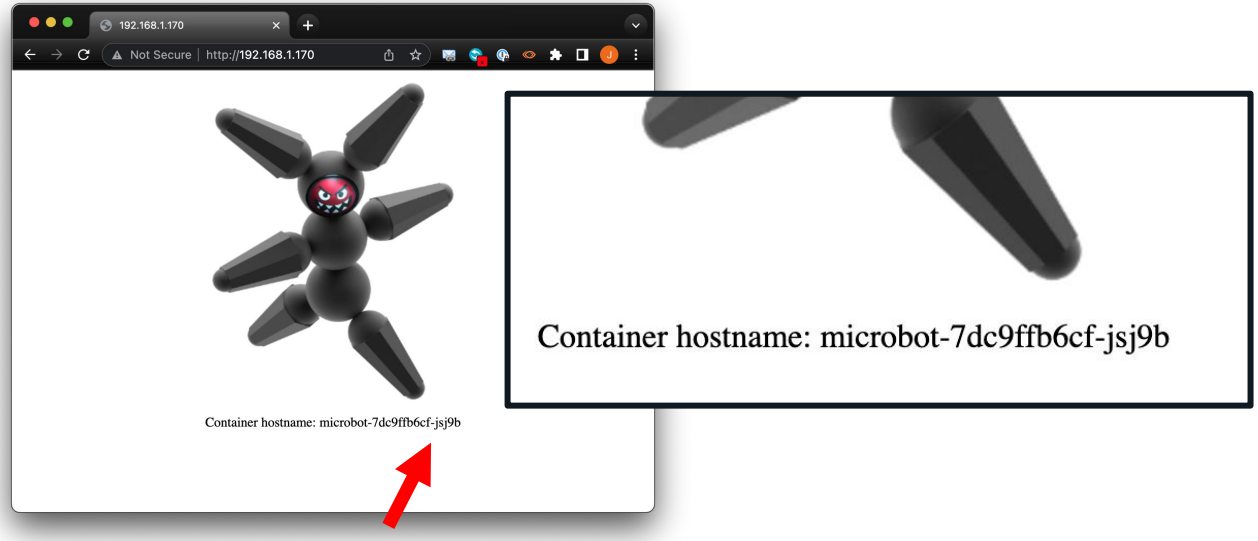
```
jason@pi-cluster-node1:~$ microk8s kubectl get services -A
NAMESPACE     NAME               TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)                  AGE
default       kubernetes         ClusterIP   10.152.183.1    <none>        443/TCP                  22d
default       microbot-service   NodePort    10.152.183.98   <none>        80:30695/TCP             92m
kube-system   kube-dns           ClusterIP   10.152.183.10   <none>        53/UDP,53/TCP,9153/TCP   21d
jason@pi-cluster-node1:~$ microk8s kubectl delete service microbot-service
service "microbot-service" deleted
jason@pi-cluster-node1:~$ microk8s kubectl expose deployment microbot --type=LoadBalancer --port=80 --
name=microbot-service
service/microbot-service exposed
jason@pi-cluster-node1:~$ microk8s kubectl get svc -A
NAMESPACE        NAME               TYPE           CLUSTER-IP       EXTERNAL-IP     PORT(S)                  AGE
default          kubernetes         ClusterIP      10.152.183.1     <none>          443/TCP                  140d
metallb-system   webhook-service    ClusterIP      10.152.183.198   <none>          443/TCP                  18h
kube-system      kube-dns           ClusterIP      10.152.183.10    <none>          53/UDP,53/TCP,9153/TCP   18h
default          microbot-service   LoadBalancer   10.152.183.22    192.168.1.170   80:31182/TCP             72s
jason@pi-cluster-node1:~$
```

Apply service and get status

```
jason@pi-cluster-node1:~$ microk8s kubectl describe service/microbot-service
Name:                     microbot-service
Namespace:                default
Labels:                   app=microbot
Annotations:              <none>
Selector:                 app=microbot
Type:                     LoadBalancer
IP Family Policy:         SingleStack
IP Families:              IPv4
IP:                       10.152.183.41
IPs:                      10.152.183.41
LoadBalancer Ingress:     192.168.1.170
Port:                     <unset>  80/TCP
TargetPort:               80/TCP
NodePort:                 <unset>  31182/TCP
Endpoints:                10.1.1.7:80,10.1.100.6:80,10.1.27.5:80 + 1 more...
Session Affinity:         None
External Traffic Policy:  Cluster
Events:
  Type    Reason        Age   From                Message
  ----    ------        ----  ----                -------
  Normal  IPAllocated   3m3s  metallb-controller  Assigned IP ["192.168.1.170"]
  Normal  nodeAssigned  3m3s  metallb-speaker     announcing from node "pi-cluster-node4" with protocol "layer2"
```



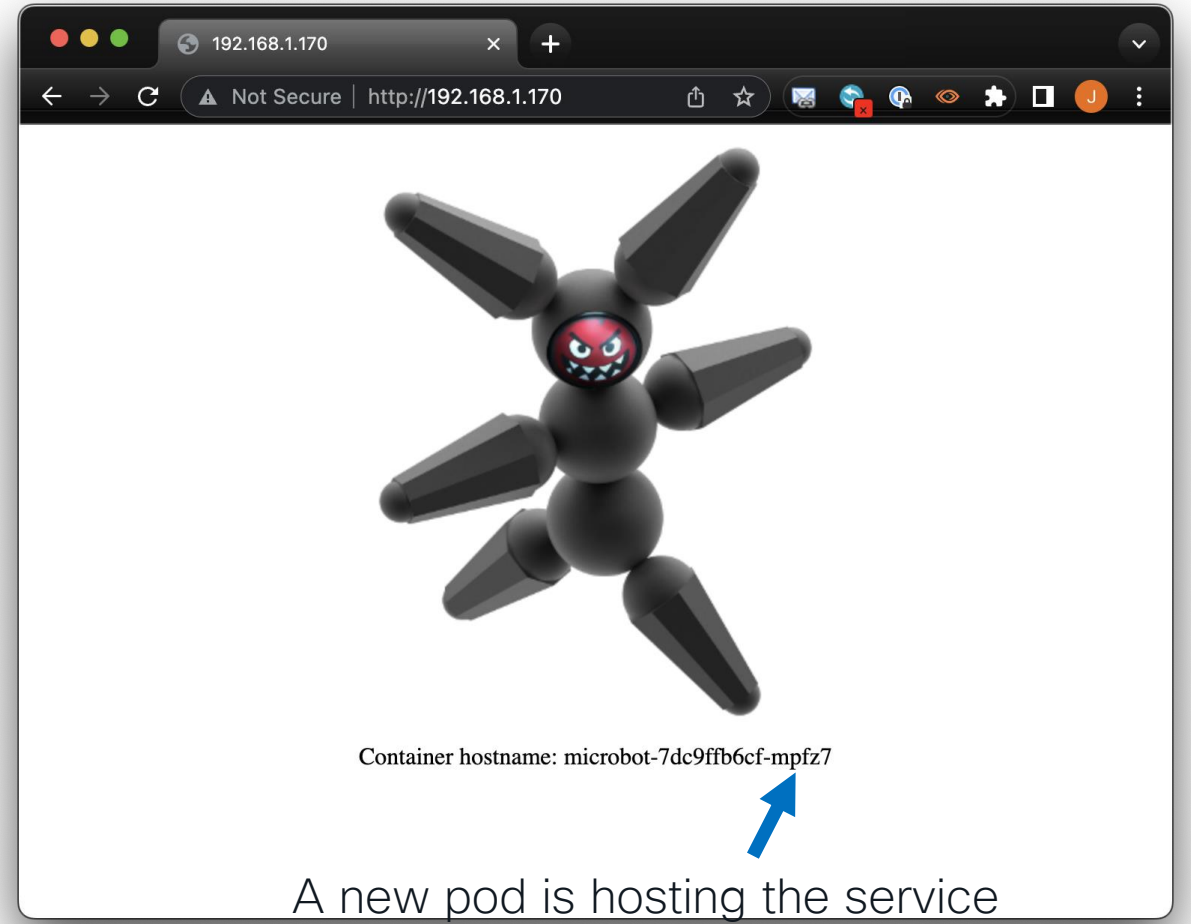




Now delete the pod hosting this microbot instance and see another takeover

```
jason@pi-cluster-node1:~$ microk8s kubectl get pods -o wide
NAME                        READY   STATUS    RESTARTS   AGE   IP          NODE
microbot-7dc9ffb6cf-mpfz7   1/1     Running   0          25h   10.1.1.7    pi-cluster-node1
microbot-7dc9ffb6cf-ffpf2   1/1     Running   0          22h   10.1.27.8   pi-cluster-node2
microbot-7dc9ffb6cf-xs8cj   1/1     Running   0          25h   10.1.81.3   pi-cluster-node3
microbot-7dc9ffb6cf-jsj9b   1/1     Running   0          25h   10.1.100.6  pi-cluster-node4
```

```
jason@pi-cluster-node1:~$ microk8s kubectl delete pod/microbot-7dc9ffb6cf-jsj9b
pod "microbot-7dc9ffb6cf-jsj9b" deleted
jason@pi-cluster-node1:~$
jason@pi-cluster-node1:~$ microk8s kubectl get pods -o wide
NAME                        READY   STATUS              RESTARTS   AGE   IP           NODE
microbot-7dc9ffb6cf-mpfz7   1/1     Running             0          25h   10.1.1.7     pi-cluster-node1
microbot-7dc9ffb6cf-ffpf2   1/1     Running             0          22h   10.1.27.8    pi-cluster-node2
microbot-7dc9ffb6cf-xs8cj   1/1     Running             0          25h   10.1.81.3    pi-cluster-node3
microbot-7dc9ffb6cf-zcxt4   1/1     ContainerCreating   0          5s    10.1.100.6   pi-cluster-node4
jason@pi-cluster-node1:~$
jason@pi-cluster-node1:~$ microk8s kubectl get pods -o wide
NAME                        READY   STATUS    RESTARTS   AGE   IP           NODE
microbot-7dc9ffb6cf-mpfz7   1/1     Running   0          25h   10.1.1.7     pi-cluster-node1
microbot-7dc9ffb6cf-ffpf2   1/1     Running   0          22h   10.1.27.8    pi-cluster-node2
microbot-7dc9ffb6cf-xs8cj   1/1     Running   0          25h   10.1.81.3    pi-cluster-node3
microbot-7dc9ffb6cf-zcxt4   1/1     Running   0          15s   10.1.100.6   pi-cluster-node4
```



A new pod is hosting the service

Limitations on MetalLB Layer-2

- Depends on ARP/ND
- Single-node Bottleneck
- Slowish fail-over performance

Now for Magic…

Integration with BGP Routing

image by Vectorportal.com

Reconfigure microk8s

- Remove earlier L2/ARP config

```
jason@pi-cluster-node1:~$ microk8s kubectl delete -f ./cr-metallb-l2arp.yaml
ipaddresspool.metallb.io "default" deleted
l2advertisement.metallb.io "default" deleted
```

Reconfigure microk8s

- Create new L3/BGP config/CR

```yaml
---
apiVersion: metallb.io/v1beta2
kind: BGPPeer
metadata:
  name: bgppeering
  namespace: metallb-system
spec:
  myASN: 64512
  peerASN: 64512
  peerAddress: 192.168.1.180
---
apiVersion: metallb.io/v1beta1
kind: IPAddressPool
metadata:
  name: address-pool
  namespace: metallb-system
spec:
  addresses:
    - 192.168.1.170-192.168.1.179
---
apiVersion: metallb.io/v1beta1
kind: BGPAdvertisement
metadata:
  name: bgpadvertisement
  namespace: metallb-system
```

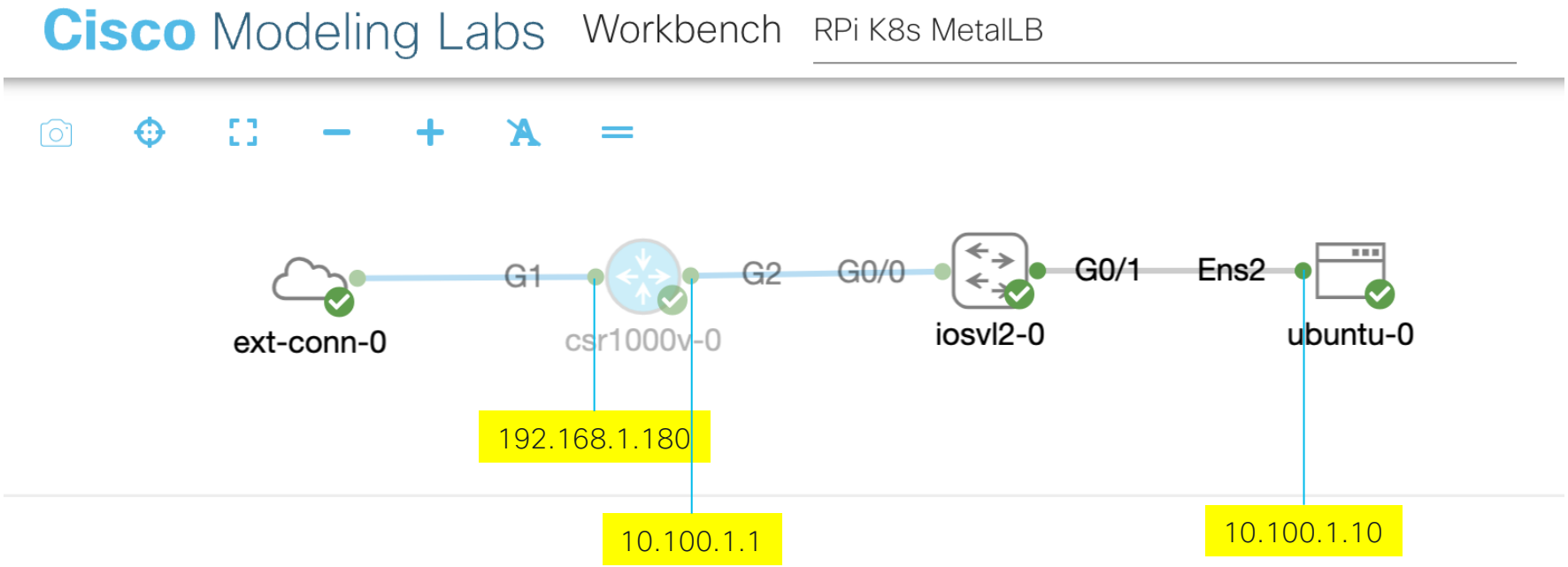
save as ‘cr-metallb-l3bgp.yaml’

Reconfigure microk8s

- Apply new L3/BGP config/CR

```
jason@pi-cluster-node1:~$ microk8s kubectl apply -f ./cr-metallb-l3bgp.yaml
bgppeer.metallb.io/bgppeering created
ipaddresspool.metallb.io/address-pool created
bgpadvertisement.metallb.io/bgpadvertisement created
jason@pi-cluster-node1:~$
```

Use Cisco Modeling Labs (CML) to build up BGP Connectivity



Configure CSR1000v for BGP routing

```
router bgp 64512
 bgp log-neighbor-changes
 redistribute connected
 neighbor 192.168.1.161 remote-as 64512
 neighbor 192.168.1.161 description rpi-cluster-node-1
 neighbor 192.168.1.162 remote-as 64512
 neighbor 192.168.1.162 description rpi-cluster-node-2
 neighbor 192.168.1.163 remote-as 64512
 neighbor 192.168.1.163 description rpi-cluster-node-3
 neighbor 192.168.1.164 remote-as 64512
 neighbor 192.168.1.164 description rpi-cluster-node-4
!
```

Cisco Modeling Labs Workbench RPi K8s MetalLB

DASHBOARD TOOLS ADMIN

ADD NODES

ext-conn-0 — G1 — csr1000v-0 — G2 — G0/0 — iosvl2-0 — G0/1 — Ens2 — ubuntu-0

NODE INFO SIMULATE CONNECTIVITY CONSOLE VNC EDIT CONFIG INTERFACES

SERIAL LINE 0 DOWNLOAD HISTORY SETTINGS

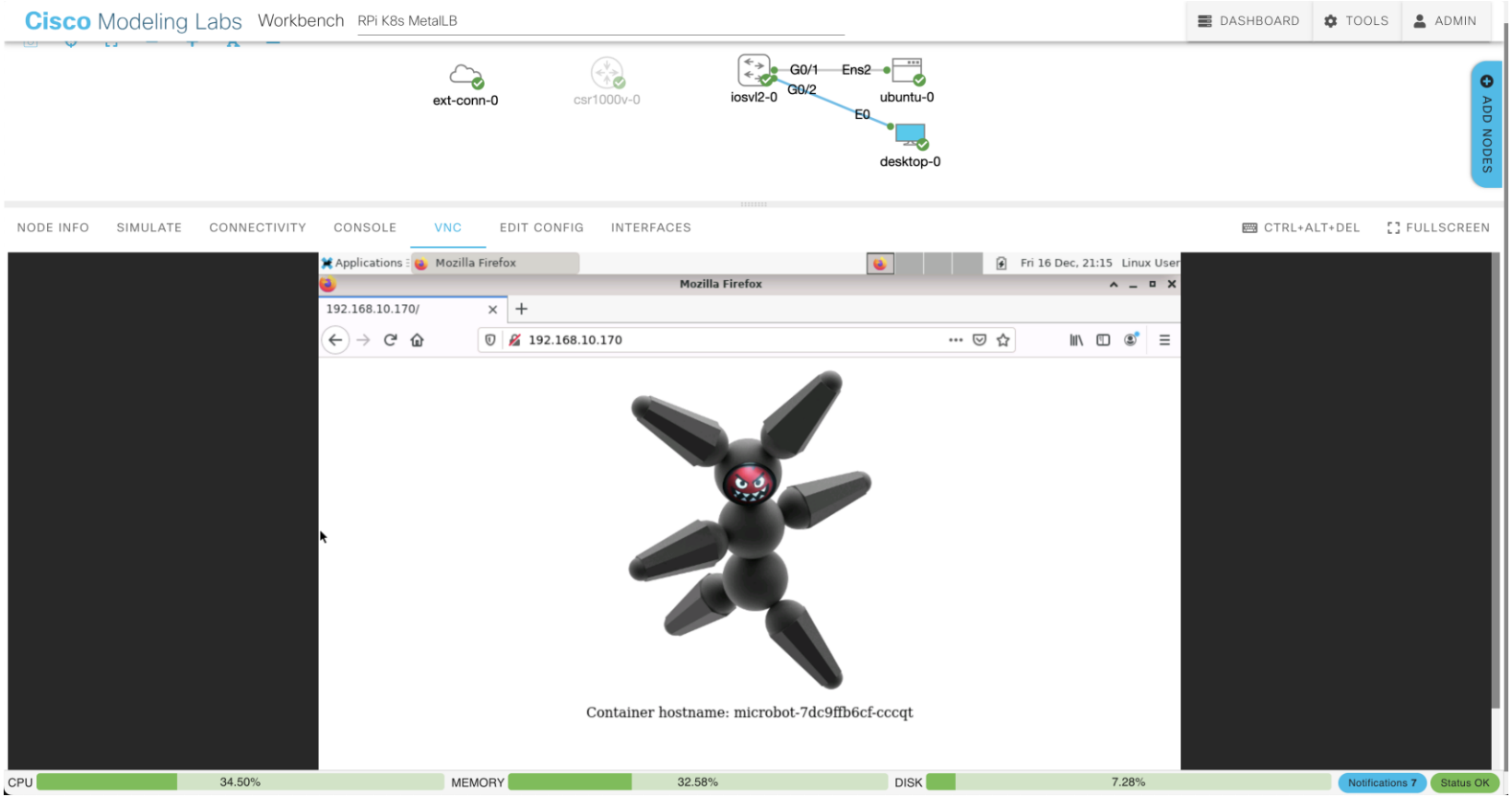
```
*Dec  5 21:32:55.871: %SYS-6-TTY_EXPIRE_TIMER: (exec timer expired, tty 0 (0.0.0.0)), user
inserthostname_here>show ip route
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, m - OMP
       n - NAT, Ni - NAT inside, No - NAT outside, Nd - NAT DIA
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       H - NHRP, G - NHRP registered, g - NHRP registration summary
       o - ODR, P - periodic downloaded static route, l - LISP
       a - application route
       + - replicated route, % - next hop override, p - overrides from PfR
       & - replicated local route overrides by connected

Gateway of last resort is 192.168.1.1 to network 0.0.0.0

S*    0.0.0.0/0 [254/0] via 192.168.1.1
      10.0.0.0/8 is variably subnetted, 2 subnets, 2 masks
C        10.100.1.0/24 is directly connected, GigabitEthernet2
L        10.100.1.1/32 is directly connected, GigabitEthernet2
      192.168.1.0/24 is variably subnetted, 3 subnets, 2 masks
C        192.168.1.0/24 is directly connected, GigabitEthernet1
B        192.168.1.170/32 [20/0] via 192.168.1.162, 00:57:24
L        192.168.1.180/32 is directly connected, GigabitEthernet1
inserthostname_here>
```

CPU 26.60% MEMORY 28.97% DISK 6.83% Notifications 32 Status OK

Use Ubuntu node to do Console curl

```
cisco@inserthostnamehere:~$ curl http://192.168.1.170
<!DOCTYPE html>
<html>
  <style type="text/css">
    .centered
      {
      text-align:center;
      margin-top:0px;
      margin-bottom:0px;
      padding:0px;
      }
  </style>
  <body>
    <p class="centered"><img src="microbot.png" alt="microbot"/></p>
    <p class="centered">Container hostname: microbot-66c45859f5-4gxwm</p>
  </body>
</html>
cisco@inserthostnamehere:~$
```

. . . Or Desktop node with VNC



Other Uses

Another idea for the cluster – Network testing

- Network Multitool project
- https://github.com/wbitt/Network-MultiTool
- Includes useful tools like curl, mtr (my traceroute), tcptraceroute, arping, iperf3, tshark, rsync, nc (netcat), socat and ab (ApacheBench)
- All in a 220 Mbyte container

```
jason@pi-cluster-node1:~$ microk8s kubectl create deployment multitool --
image=praqma/network-multitool:alpine-extra
jason@pi-cluster-node1:~$
jason@pi-cluster-node1:~$ microk8s kubectl exec -it pod/multitool-<pod_instance>
-- /bin/bash
```

Network-Multitool / mtr

```
bash-5.1# which mtr
/usr/sbin/mtr
bash-5.1# mtr -c 100 developer.cisco.com
bash-5.1#
```

```
                                                My traceroute  [v0.94]
multitool-6d5cffb4d7-ndq85 (10.1.100.8) -> developer.cisco.com                    2023-01-09T16:55:42+0000
Keys:  Help   Display mode   Restart statistics   Order of fields   quit
                                                                   Packets               Pings
 Host                                                             Loss%   Snt   Last   Avg  Best  Wrst StDev
 1. 10.1.100.1                                                     0.0%   100    0.5   0.4   0.3   0.6   0.1
 2. gateway.[redacted]                                             0.0%   100    0.6   0.6   0.4   0.8   0.1
 3. 192.168.1.254                                                  0.0%   100    1.0   1.1   0.7   1.5   0.2
 4. 99-178-168-1.lightspeed.rlghnc.sbcglobal.net                   0.0%   100    4.1   4.4   1.9   7.5   1.4
 5. 99.173.76.21 reply)                                            5.0%   100    2.7   4.5   2.1   7.2   1.4
 6. (waiting for reply)
 7. (waiting for reply)
 8. (waiting for reply)
 9. 32.130.24.165                                                  0.0%   100   19.5  16.6  11.4  22.0   2.6
10. cr1.cl2oh.ip.att.net                                           0.0%   100   21.1  16.9  10.9  23.1   2.6
11. 12.123.10.37 reply)                                            0.0%   100   12.5  12.7   9.8  16.2   1.4
12. 12.117.216.210                                                 0.0%   100   11.5  14.8  11.0  50.0   5.0
13. (waiting for reply)
14. (waiting for reply)
15. (waiting for reply)
16. (waiting for reply)
17. (waiting for reply)
18. (waiting for reply)
19. server-18-67-65-49.iad89.r.cloudfront.net                      0.0%   100   15.0  13.5  11.0  16.7   1.4
```

Network-Multitool / mtr

```
bash-5.1# mtr -z -j -c 10 developer.cisco.com
{
    "report": {
        "mtr": {
            "src": "multitool-6d5cffb4d7-ndq85",
            "dst": "developer.cisco.com",
            "tos": 0,
            "tests": 10,
            "psize": "64",
            "bitpattern": "0x00"
        },
        "hubs": [
            {
                "count": 1,
                "host": "10.1.100.1",
                "ASN": "AS???",
                "Loss%": 0.0,
                "Snt": 10,
                "Last": 0.381,
                "Avg": 0.426,
                "Best": 0.348,
                "Wrst": 0.521,
                "StDev": 0.047
            },
            {
                "count": 2,
                "host": "gateway-[redacted]
                "ASN": "AS???",
                "Loss%": 0.0,
                "Snt": 10,
                "Last": 0.566,
                "Avg": 0.535,
                "Best": 0.407,
                "Wrst": 0.61,
```

Secure Shell VT220 Xmodem ZOC2301.LOG 01:22:53 10/33

Network-Multitool / ab (ApacheBench)

```
bash-5.1# ab -n 1000 -c 20 https://developer.cisco.com/
This is ApacheBench, Version 2.3 <$Revision: 1879490 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking developer.cisco.com (be patient)
Completed 100 requests
Completed 200 requests
Completed 300 requests
Completed 400 requests
Completed 500 requests
Completed 600 requests
Completed 700 requests
Completed 800 requests
Completed 900 requests
Completed 1000 requests
Finished 1000 requests


Server Software:        CloudFront
Server Hostname:        developer.cisco.com
Server Port:            443
SSL/TLS Protocol:       TLSv1.2,ECDHE-RSA-AES128-GCM-SHA256,2048,128
Server Temp Key:        X25519 253 bits
TLS Server Name:        developer.cisco.com

Document Path:          /
Document Length:        6681 bytes
```

Network-Multitool / ab (ApacheBench)

```
Concurrency Level:      20
Time taken for tests:   5.313 seconds
Complete requests:      1000
Failed requests:        0
Total transferred:      7828012 bytes
HTML transferred:       6681000 bytes
Requests per second:    188.22 [#/sec] (mean)
Time per request:       106.260 [ms] (mean)
Time per request:       5.313 [ms] (mean, across all concurrent requests)
Transfer rate:          1438.84 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:       51   76   5.9     76      97
Processing:    12   25   7.2     25     175
Waiting:       12   22   6.5     21     163
Total:         78  101   8.8    101     226

Percentage of the requests served within a certain time (ms)
  50%    101
  66%    104
  75%    105
  80%    107
  90%    110
  95%    113
  98%    118
  99%    121
 100%    226 (longest request)
```

Other ideas

- Pi-Hole to block DNS and web advertisements in your browser
- Development of microservices using Flask containers with simple Python scripts
- Learning and practical experience development with Load balancing and how to swap out/in updated docker containers of microservices

In Conclusion - What did we learn?

- How to provision Raspberry Pis for Kubernetes use
- Basic Kubernetes (and microK8s) commands
- Integrating a Kubernetes cluster in an L2/ARP network with MetalLB
- Integrating a Kubernetes cluster in an L3/BGP network with MetalLB
- Hosting basic container services as replicated deployments
- A few fun practical tool use cases with K8s images

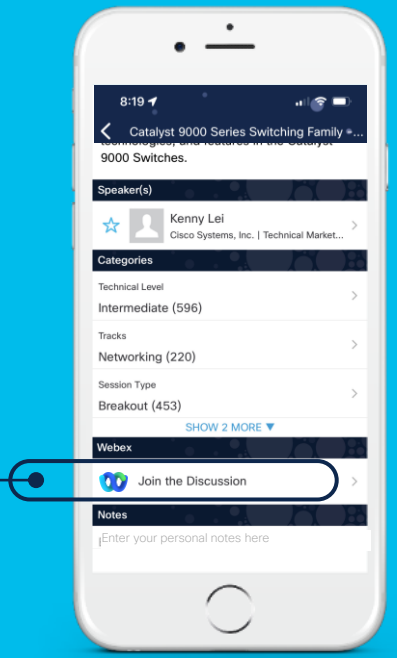
Cisco Webex App

Questions?

Use Cisco Webex App to chat
with the speaker after the session

How

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



Webex spaces will be moderated until February 24, 2023.

Complete your Session Survey

- Please complete your session survey after each session. Your feedback is important.
- All surveys can be taken in the Cisco Events Mobile App or by logging in to the Session Catalog and clicking the " Attendee Dashboard" at https://www.ciscolive.com/emea/learn/sessions/session-catalog.html

Continue Your Education

Visit the Cisco Showcase for related demos.

Book your one-on-one Meet the Engineer meeting.

Attend any of the related sessions at the DevNet, Capture the Flag, and Walk-in Labs zones.

Visit the On-Demand Library for more sessions at ciscolive.com/on-demand.

The bridge to possible

Thank you

cisco Live!

CISCO Live!

ALL IN