

VMware Advanced Customer Engagements (ACE) Team

# Tanzu Kubernetes Grid Integrated (TKGI) Workload Isolation using NSX-T Micro Segmentation

May 2020

# Table of Contents

- Introduction 3
- Use-case 5
  - Traffic flows.....5
  - Assumptions.....5
- K8s Resources 5
  - Pod Definition .....5
  - Deployments And Services.....5
- K8s Resource Deployments 6
  - K8 Deployments .....6
  - Current Traffic Flow .....12
- Configure Micro Segmentation with NSX-T 14
- Test Traffic Flow 28

## Introduction

Containers provide a great deal of benefits to your development pipeline and leverage resource isolation but they're not designed with strong security boundaries or workload isolation. Tanzu Kubernetes Grid Integrated (TKGI) or Enterprise PKS (PKS) integrates with VMware NSX-T Data Center (NSX-T) and built on top of VMware vSphere to provide an agile software defined infrastructure to build cloud-native applications. NSX-T primarily focuses on networking and security for these applications and used to create stronger workload isolation.

In TKGI, workload isolation can be performed by physically isolating workload at the vSphere level or by logically isolating workloads within the TKGI deployed clusters (at cluster or namespace isolation). The following design patterns can be considered for workload isolation.

### Design patterns for Workload Isolation

#### Traditional DMZ design

- Full stack DMZ
- Hybrid stack DMZ
- Shared stack DMZ

Use Network Profiles in Enterprise PKS using the same NSX-T

Use Micro segmentation (NSX-T)

Use Kubernetes Network Profiles

Physical Isolation and Expensive

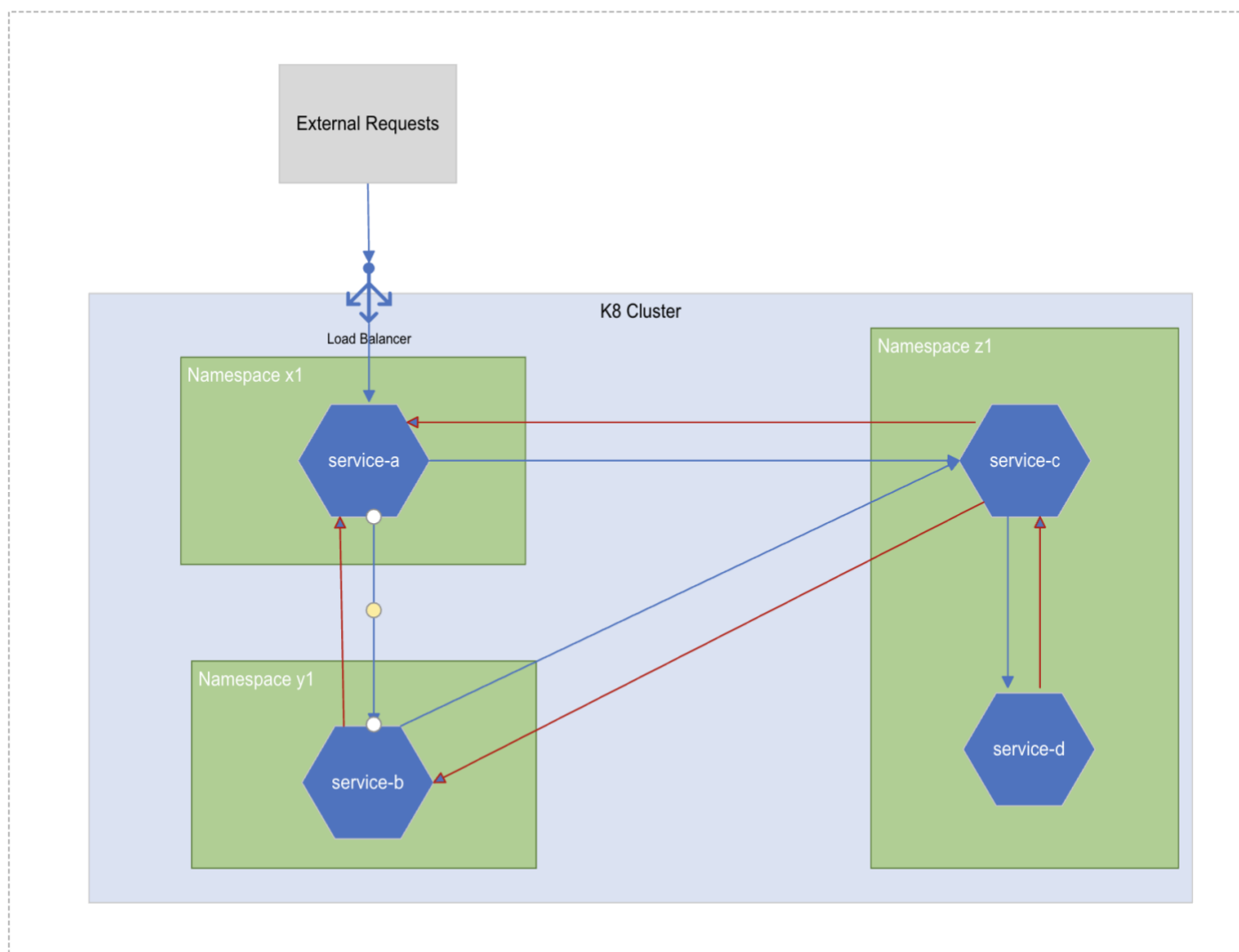
Logical Isolation and Less Expensive

In this document, we focus on how you can leverage the micro-segmentation feature within NSX-T to provide workload isolation. NSX-T comes with a distributed firewall that can provide complete control of both North-South Traffic but also East-West Traffic and can isolate workloads, even if they are next to each other. For example, traditional firewalls only isolate network traffic between network VLANs or segments but not within a network segment. But with NSX-T distributed firewall, you can create rules to isolate workload on the same segment and with Kubernetes tags, you can isolate even POD to POD communication.

In this document, we take a simple application that has several components or services. These services are required to communicate with each other in a very defined manner. For example, service-a need to communicate with service-c and service-b but not with any other service. Similarly, service-c needs to communicate with service-d but not with service-a or service-b.

In such a scenario, we look at how you can isolate the workload using NSX-T and this is all done dynamically as pods are created and destroyed.

Now, let us assume that all ingress traffic from outside the cluster comes into service-a. service-a can be running in namespace x1 and is a typically ingress controller like nginx or contour. service-a routes to service-b or service-c and service-c might need to talk to database service like that defined in service-d. This diagram shows which workload should be allowed and what is not between the services.



## Use-case

The k8s cluster has three namespaces (namespace x1, y1 and z1). All ingress into the cluster is restricted to namespace x1 and pod service-a.

Namespace y1 runs a pod service-b and namespace z1 runs pods service-c and service-d.

## Traffic flows

All ingress traffic into the cluster can only be serviced by service-a on namespace x1

Service-b allows traffic from service-a

Service-b denies traffic from service-c

Service-c allows traffic from service-a and service-b

Service-d allows traffic from only service-c

Service-c denies traffic from service-d

## Assumptions

K8s cluster exists and there is integration with NSX-T

## K8s Resources

### Pod Definition

We will be using a pod definition with 2 containers

- nginx (to service web requests)
- busybox with curl (container to test service to service traffic)

### Deployments And Services

service-a, service-b, service-c and service-d are identical and run the above pod definition on their respective namespaces, they run two replicas and are exposed as services svc-service-a, svc-service-b, svc-service-c, svc-service-d

## K8s Resource Deployments

This section goes through the steps to set up the K8 environment with the deployments in their respective namespaces. At the end of the setup a test will be performed to establish that the traffic flows between the different services deployed

### K8 Deployments

#### 1. Login to k8s cluster

```
pks login -a <pks-api> -u <pksuser> -p <pks-password> -k
```

Eg.

```
pks login -a pks.corp.local -u pksadmin -p VMware1! -k
```

#### 2. Get Kubeconfig

```
pks get-kubeconfig <cluster-name> -a <pks-api> -u <pksuser> -p <pks-password> -k
```

Eg.

```
pks get-kubeconfig ci-cluster -a pks.corp.local -u pksadmin -p VMware1! -k
```

#### 3. Change context

```
kubectl config use-context ci-cluster
```

#### 4. Create Namespace x1, y1, z1

```
kubectl create ns x1
```

```
kubectl create ns y1
```

```
kubectl create ns z1
```

```
ubuntu@cli-vm:~/dmz$ kubectl create ns x1
namespace/x1 created
ubuntu@cli-vm:~/dmz$
ubuntu@cli-vm:~/dmz$ kubectl create ns y1
namespace/y1 created
ubuntu@cli-vm:~/dmz$
ubuntu@cli-vm:~/dmz$ kubectl create ns z1
namespace/z1 created
```

## 5. Check if namespace is created

```
kubectl get ns
```

```
ubuntu@cli-vm:~/dmz$ kubectl get ns
NAME                STATUS    AGE
default             Active    12d
jenkinso            Active    11d
kube-node-lease     Active    12d
kube-public         Active    12d
kube-system         Active    12d
pks-system          Active    11d
x1                 Active    13s
y1                 Active    13s
z1                 Active    13s
```

## 6. Create Deployments & Services

Download the yaml file or copy contents to a local file Eg. micro.yaml . This file declares the K8 resources required . Create the necessary resources in their respective namespaces

<https://github.com/riazvm/nsxtk8smicrosegmentation/blob/master/yaml/micro.yaml>

```
kubectl apply -f micro.yaml
```

```
ubuntu@cli-vm:~/dmz$ vi micro.yaml
ubuntu@cli-vm:~/dmz$ kubectl apply -f micro.yaml
deployment.apps/service-a created
service/svc-service-a created
deployment.apps/service-b created
service/svc-service-b created
deployment.apps/service-c created
service/svc-service-c created
deployment.apps/service-d created
service/svc-service-d created
ubuntu@cli-vm:~/dmz$
```



## 7. Check services and pods created in each namespace

kubectl get all -n x1

```
ubuntu@cli-vm:~/dmz$ kubectl get all -n x1
```

NAME	READY	STATUS	RESTARTS	AGE
pod/service-a-84965f57cc-nhbrx	2/2	Running	0	105s
pod/service-a-84965f57cc-q72fh	2/2	Running	0	105s

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
service/svc-service-a	ClusterIP	10.100.200.143	<none>	80/TCP	105s

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
deployment.apps/service-a	2/2	2	2	105s

NAME	DESIRED	CURRENT	READY	AGE
replicaset.apps/service-a-84965f57cc	2	2	2	105s

kubectl get all -n y1

```
ubuntu@cli-vm:~/dmz$ kubectl get all -n y1
```

NAME	READY	STATUS	RESTARTS	AGE
pod/service-b-86d8c888c7-6r9kp	2/2	Running	0	2m39s
pod/service-b-86d8c888c7-jpwf9	2/2	Running	0	2m39s

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
service/svc-service-b	ClusterIP	10.100.200.12	<none>	80/TCP	2m39s

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
deployment.apps/service-b	2/2	2	2	2m39s

NAME	DESIRED	CURRENT	READY	AGE
replicaset.apps/service-b-86d8c888c7	2	2	2	2m39s

kubectl get all -n z1

```
ubuntu@cli-vm:~/dmz$ kubectl get all -n z1
```

NAME	READY	STATUS	RESTARTS	AGE
pod/service-c-5c5fc5c857-d9sqm	2/2	Running	0	2m52s
pod/service-c-5c5fc5c857-jxqbh	2/2	Running	0	2m52s
pod/service-d-69f59f4cb9-f94rs	2/2	Running	0	2m52s
pod/service-d-69f59f4cb9-pkplt	2/2	Running	0	2m52s

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
service/svc-service-c	ClusterIP	10.100.200.117	<none>	80/TCP	2m52s
service/svc-service-d	ClusterIP	10.100.200.211	<none>	80/TCP	2m52s

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
deployment.apps/service-c	2/2	2	2	2m52s
deployment.apps/service-d	2/2	2	2	2m52s

NAME	DESIRED	CURRENT	READY	AGE
replicaset.apps/service-c-5c5fc5c857	2	2	2	2m52s
replicaset.apps/service-d-69f59f4cb9	2	2	2	2m52s

## 8. Expose service-a as a load-balancer service

kubectl expose deployment service-a \

--name=service-a-lb --port=80 --target-port=8080 --type=LoadBalancer --namespace=x1

```
ubuntu@cli-vm:~/dmz$ kubectl expose deployment service-a \
> --name=service-a-lb --port=80 --target-port=8080 --type=LoadBalancer --namespace=x1
service/service-a-lb exposed
```

## 9. Check the external URL/IP address assigned to the service (make note of the first IP address under External-IP).

```
kubectl get svc -n x1
```

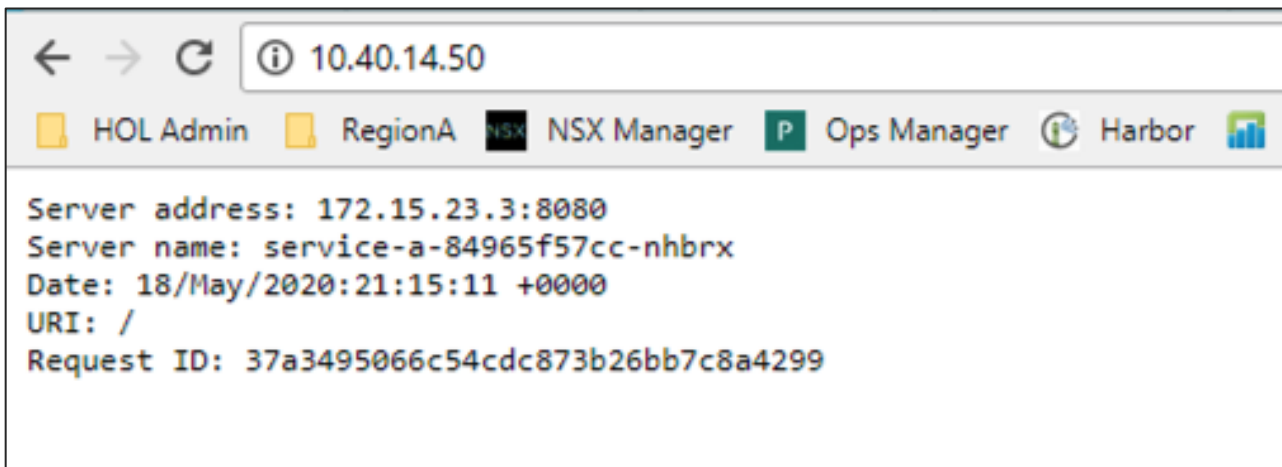
```
ubuntu@cli-vm:~/dmz$ kubectl get svc -n x1
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
service-a-lb	LoadBalancer	10.100.200.49	10.40.14.50	80:32650/TCP	12s
svc-service-a	ClusterIP	10.100.200.143	<none>	80/TCP	15m

## Current Traffic Flow

### Traffic flow from external network to k8 cluster.

Open a browser and browse to the external ip from the previous step



service-a can be reached from the external network through the load balancer which is created in NSX-T

service-b, service-c and service-d are not exposed and hence are not reachable from the external network.

### Traffic flow between pods

We will be using the busy-box container within service-a pod and use curl to check for responses between service.

Source Service – service-a (namespace x1)

Target Service – service-b (namespace y1)

1. Get pods running on namespace x1

```
kubectl get po -n x1
```

```
ubuntu@cli-vm:~/dmz$ kubectl get po -n x1
```

NAME	READY	STATUS	RESTARTS	AGE
service-a-84965f57cc-nhbrx	2/2	Running	0	31m
service-a-84965f57cc-q72fh	2/2	Running	0	31m

```
ubuntu@cli-vm:~/dmz$
```

2. Get service name for service-b running on namespace y1

```
kubectl get svc -n y1
```

```
ubuntu@cli-vm:~/dmz$ kubectl get svc -n y1
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
svc-service-b	ClusterIP	10.100.200.12	<none>	80/TCP	38m

3. Exec into the busybox container running on a service-a pod

```
kubectl exec -n <namespace> <podname> -it -c busybox -- /bin/sh
```

Eg

```
kubectl exec -n x1 service-a-84965f57cc-nhbrx -it -c busybox -- /bin/sh
```

```
ubuntu@cli-vm:~/dmz$ kubectl exec -n x1 service-a-84965f57cc-nhbrx -it -c busybox -- /bin/sh
/home #
/home #
```

4. Use curl to reach service-b in namespace y1

```
curl http://svc-service-b.y1
```

NOTE: the service name is prepended by svc in this case, this is defined in the yaml we used to create the K8 resources. The service name is also appended by the namespace.

5. This results in a successful response

```
ubuntu@cli-vm:~/dmz$ kubectl exec -n x1 service-a-84965f57cc-nhbrx -it -c busybox -- /bin/sh
/home # curl http://svc-service-b.y1
Server address: 172.15.13.3:8080
Server name: service-b-86d8c888c7-jpwf9
Date: 18/May/2020:21:38:00 +0000
URI: /
Request ID: cfe8d330cac2894b204a1b57082e10cc
```

6. Use the curl command to check the service response of service-d in namespace z1

curl <http://svc-service-d.z1>

This should be successful as well

```
ubuntu@cli-vm:~/dmz$ kubectl exec -n x1 service-a-84965f57cc-nhbrx -it -c busybox -- /bin/sh
/home # curl http://svc-service-b.y1
Server address: 172.15.13.3:8080
Server name: service-b-86d8c888c7-jpwf9
Date: 18/May/2020:21:38:00 +0000
URI: /
Request ID: cfe8d330cac2894b204a1b57082e10cc
/home # curl http://svc-service-d.z1
Server address: 172.15.9.4:8080
Server name: service-d-69f59f4cb9-f94rs
Date: 18/May/2020:21:40:51 +0000
URI: /
Request ID: 9acb24c07338cadfa53b59c99f042c47
```

At this point all services should be able to communicate to each other. The only traffic allowed from outside the cluster is to service-a via the loadbalancer.

## Configure Micro Segmentation with NSX-T

This section goes through the configuration of NSX-T DFW rules that are required to allow, restrict or drop pod to pod traffic.

### 1. Check labels on the pods

```
kubectl get pod --show-labels -n x1
```

Check the labels on the pods in namespace y1 and z1 as well. All tags are defined as app={service name}. e.g.: app=service-a

```
ubuntu@cli-vm:~/dmz$ kubectl get pod --show-labels -n x1
NAME                                READY   STATUS    RESTARTS   AGE   LABELS
service-a-84965f57cc-nhbrx          2/2     Running   0           80m   app=service-a,pod-template-hash=84965f57cc
service-a-84965f57cc-q72fh          2/2     Running   0           80m   app=service-a,pod-template-hash=84965f57cc
ubuntu@cli-vm:~/dmz$ kubectl get pod --show-labels -n yi
No resources found.
ubuntu@cli-vm:~/dmz$ kubectl get pod --show-labels -n y1
NAME                                READY   STATUS    RESTARTS   AGE   LABELS
service-b-86d8c888c7-6r9kp          2/2     Running   0           81m   app=service-b,pod-template-hash=86d8c888c7
service-b-86d8c888c7-jpwf9          2/2     Running   0           81m   app=service-b,pod-template-hash=86d8c888c7
ubuntu@cli-vm:~/dmz$ kubectl get pod --show-labels -n z1
NAME                                READY   STATUS    RESTARTS   AGE   LABELS
service-c-5c5fc5c857-d9sqm          2/2     Running   0           81m   app=service-c,pod-template-hash=5c5fc5c857
service-c-5c5fc5c857-jxqbh          2/2     Running   0           81m   app=service-c,pod-template-hash=5c5fc5c857
service-d-69f59f4cb9-f94rs          2/2     Running   0           81m   app=service-d,pod-template-hash=69f59f4cb9
service-d-69f59f4cb9-pkplt          2/2     Running   0           81m   app=service-d,pod-template-hash=69f59f4cb9
```

## 2. Check tags for pods in NSXT

Login to NSXT and Navigate to Advanced Networking & Security → Switching → Ports

The pods service-a , service-b, service-c and service-d would have a logical port assigned to them



# NSX-T & K8s Micro Segmentation

vm NSX-T

Search

Refresh

Help

Admin

HomeNetworkingSecurityInventoryPlan & TroubleshootSystemAdvanced Networking & Security

SwitchesPortsSwitching Profiles

NetworkingSwitchingRoutersNATDHCPIPAMLoad BalancingSecurityPartner ServicesToolsInventory

+ ADD

EDIT

DELETE

ACTIONS

Search

Logical Port	ID	Admin Status	Operational Status	Switching Profile	Attachment	Logical Switch
<input type="checkbox"/> pks-592ff063-c3b0-46b4-81ce-0bcfaafd62-jenkins-jenkins-operator-6d5477b784-kf6mv	83b4...ef30	Up	Up	nsx-default-...	VIF:c30a...6...	pks-592ff06...
<input type="checkbox"/> pks-592ff063-c3b0-46b4-81ce-0bcfaafd6246-default-busybox-7b87695f88-jgc4f	c32c...6e0d	Up	Up	nsx-default-...	VIF:4868...0...	pks-592ff06...
<input type="checkbox"/> pks-592ff063-c3b0-46b4-81ce-0bcfaafd6246-jenkins-jenkins-example	5a4e...d747	Up	Up	nsx-default-...	VIF:e615...ac...	pks-592ff06...
<input type="checkbox"/> pks-592ff063-c3b0-46b4-81ce-0bcfaafd6246-kube-system-coredns-6f9bcd8956-6znfz	71c7...15df	Up	Up	nsx-default-...	VIF:bcd0...2...	pks-592ff06...
<input type="checkbox"/> pks-592ff063-c3b0-46b4-81ce-0bcfaafd6246-kube-system-coredns-6f9bcd8956-cgl4n	5ea4...5954	Up	Up	nsx-default-...	VIF:291a...15...	pks-592ff06...
<input type="checkbox"/> pks-592ff063-c3b0-46b4-81ce-0bcfaafd6246-kube-system-coredns-6f9bcd8956-kmcp	5124...5bc8	Up	Up	nsx-default-...	VIF:c249...c...	pks-592ff06...
<input type="checkbox"/> pks-592ff063-c3b0-46b4-81ce-0bcfaafd6246-pks-system-fluent-bit-hxh69	168a...87fe	Up	Up	nsx-default-...	VIF:1cb3...7...	pks-592ff06...
<input type="checkbox"/> pks-592ff063-c3b0-46b4-81ce-0bcfaafd6246-pks-system-fluent-bit-whqkw	9d62...4418	Up	Up	nsx-default-...	VIF:fde4...b...	pks-592ff06...
<input type="checkbox"/> pks-592ff063-c3b0-46b4-81ce-0bcfaafd6246-pks-system-fluent-bit-xrbzx	e1f9...edd0	Up	Up	nsx-default-...	VIF:ad5a...fc...	pks-592ff06...
<input type="checkbox"/> pks-592ff063-c3b0-46b4-81ce-0bcfaafd6246-pks-system-validator-847cb99cc-pzhrl	ce14...4bed	Up	Up	nsx-default-...	VIF:d653...8...	pks-592ff06...
<input type="checkbox"/> pks-592ff063-c3b0-46b4-81ce-0bcfaafd6246-switch-to-t1-port	ce3f...1342	Up	Up	nsx-default-...	LR:6a16...80...	pks-592ff06...
<input type="checkbox"/> pks-592ff063-c3b0-46b4-81ce-0bcfaafd6246-x1-service-a-84965f57cc-nhbrx	df5d...57ed	Up	Up	nsx-default-...	VIF:792d...2...	pks-592ff06...
<input checked="" type="checkbox"/> pks-592ff063-c3b0-46b4-81ce-0bcfaafd6246-x1-service-a-84965f57cc-q72fh	8a84...69ce	Up	Up	nsx-default-...	VIF:3f3e...1b...	pks-592ff06...
<input type="checkbox"/> pks-592ff063-c3b0-46b4-81ce-0bcfaafd6246-y1-service-b-86d8c888c7-6r9kp	508f...e0ce	Up	Up	nsx-default-...	VIF:6ed0...c...	pks-592ff06...
<input type="checkbox"/> pks-592ff063-c3b0-46b4-81ce-0bcfaafd6246-y1-service-b-86d8c888c7-jpwf9	4ae3...3c64	Up	Up	nsx-default-...	VIF:6184...6...	pks-592ff06...
<input type="checkbox"/> pks-592ff063-c3b0-46b4-81ce-0bcfaafd6246-z1-service-c-5c5fc5c857-d9sqm	cabf...625f	Up	Up	nsx-default-...	VIF:8f16...f412	pks-592ff06...
<input type="checkbox"/> pks-592ff063-c3b0-46b4-81ce-0bcfaafd6246-z1-service-c-5c5fc5c857-jxqbh	ba4b...f539	Up	Up	nsx-default-...	VIF:53b9...1...	pks-592ff06...
<input type="checkbox"/> pks-592ff063-c3b0-46b4-81ce-0bcfaafd6246-z1-service-d-69f59f4cb9-f94rs	dald...ada3	Up	Up	nsx-default-...	VIF:5918...9...	pks-592ff06...
<input type="checkbox"/> pks-592ff063-c3b0-46b4-81ce-0bcfaafd6246-z1-service-d-69f59f4cb9-pkplf	7894...93cb	Up	Up	nsx-default-...	VIF:d2c0...2...	pks-592ff06...

COLUMNS

REFRESH

Last Updated: 2 minutes ago

BACKNEXT

1 - 50 of 87 Logical Ports



Switches **Ports** Switching Profiles
[+ ADD](#)
[EDIT](#)
[DELETE](#)
[ACTIONS](#)

<input type="checkbox"/>	Logical Port <span>↑</span>
<input type="checkbox"/>	pkcs-592ff063-c3b0-46b4-81ce-0bcfaafd6246-jenkins-jenkins-example
<input type="checkbox"/>	pkcs-592ff063-c3b0-46b4-81ce-0bcfaafd6246-default-busybox-7b87695f88-jgc4f
<input type="checkbox"/>	pkcs-592ff063-c3b0-46b4-81ce-0bcfaafd6246-jenkins-jenkins-example
<input type="checkbox"/>	pkcs-592ff063-c3b0-46b4-81ce-0bcfaafd6246-kube-system-coredns-6f9bcd8956-6znfz
<input type="checkbox"/>	pkcs-592ff063-c3b0-46b4-81ce-0bcfaafd6246-kube-system-coredns-6f9bcd8956-cgl4n
<input type="checkbox"/>	pkcs-592ff063-c3b0-46b4-81ce-0bcfaafd6246-kube-system-coredns-6f9bcd8956-kmcpc
<input type="checkbox"/>	pkcs-592ff063-c3b0-46b4-81ce-0bcfaafd6246-pks-system-fluent-bit-hxh69
<input type="checkbox"/>	pkcs-592ff063-c3b0-46b4-81ce-0bcfaafd6246-pks-system-fluent-bit-whqxw
<input type="checkbox"/>	pkcs-592ff063-c3b0-46b4-81ce-0bcfaafd6246-pks-system-fluent-bit-xrbzx
<input type="checkbox"/>	pkcs-592ff063-c3b0-46b4-81ce-0bcfaafd6246-pks-system-validator-847cb99cc-pzhrl
<input type="checkbox"/>	pkcs-592ff063-c3b0-46b4-81ce-0bcfaafd6246-switch-to-t1-port
<input type="checkbox"/>	pkcs-592ff063-c3b0-46b4-81ce-0bcfaafd6246-x1-service-a-84965f57cc-nhbrx
<input type="checkbox"/>	pkcs-592ff063-c3b0-46b4-81ce-0bcfaafd6246-x1-service-a-84965f57cc-q72fh
<input type="checkbox"/>	pkcs-592ff063-c3b0-46b4-81ce-0bcfaafd6246-y1-service-b-86d8c888c7-6r9kp
<input checked="" type="checkbox"/>	pkcs-592ff063-c3b0-46b4-81ce-0bcfaafd6246-y1-service-b-86d8c888c7-jpww9
<input type="checkbox"/>	pkcs-592ff063-c3b0-46b4-81ce-0bcfaafd6246-z1-service-c-5c5fc5c857-d9sqm
<input type="checkbox"/>	pkcs-592ff063-c3b0-46b4-81ce-0bcfaafd6246-z1-service-c-5c5fc5c857-jxqbh
<input type="checkbox"/>	pkcs-592ff063-c3b0-46b4-81ce-0bcfaafd6246-z1-service-d-69f59f4cb9-f94rs
<input type="checkbox"/>	pkcs-592ff063-c3b0-46b4-81ce-0bcfaafd6246-z1-service-d-69f59f4cb9-pkpllt

Click on one of the Logical ports associated with the services we are working with and notice that the tags on NSXT is the labels defined for the pods in K8s.

Switches **Ports** Switching Profiles

+ ✎ 🗑 ⚙

**Logical Port** ↑

- ☐ pks-592ff063-c3b0-46b4-81ce-0bcfaafd62...
- ☐ pks-592ff063-c3b0-46b4-81ce-0bcfaafd62...
- ☐ pks-592ff063-c3b0-46b4-81ce-0bcfaafd62...
- ☐ pks-592ff063-c3b0-46b4-81ce-0bcfaafd62...
- ☐ pks-592ff063-c3b0-46b4-81ce-0bcfaafd62...
- ☐ pks-592ff063-c3b0-46b4-81ce-0bcfaafd62...
- ☐ pks-592ff063-c3b0-46b4-81ce-0bcfaafd62...
- ☐ pks-592ff063-c3b0-46b4-81ce-0bcfaafd62...
- ☐ pks-592ff063-c3b0-46b4-81ce-0bcfaafd62...
- ☐ pks-592ff063-c3b0-46b4-81ce-0bcfaafd62...
- ☐ pks-592ff063-c3b0-46b4-81ce-0bcfaafd62...
- ☐ pks-592ff063-c3b0-46b4-81ce-0bcfaafd62...
- ☒ pks-592ff063-c3b0-46b4-81ce-0bcfaafd62... >
- ☐ pks-592ff063-c3b0-46b4-81ce-0bcfaafd62...
- ☐ pks-592ff063-c3b0-46b4-81ce-0bcfaafd62...
- ☐ pks-592ff063-c3b0-46b4-81ce-0bcfaafd62...
- ☐ pks-592ff063-c3b0-46b4-81ce-0bcfaafd62...
- ☐ pks-592ff063-c3b0-46b4-81ce-0bcfaafd62...
- ☐ pks-592ff063-c3b0-46b4-81ce-0bcfaafd62...
- ☐ pks-592ff063-c3b0-46b4-81ce-0bcfaafd62...

< > 1-50 / 87

**pks-592ff063-c3b0-46b4-81ce-0bcfaafd6246-x1-service-a-84965f57cc-nhb**

Overview Monitor Manage ▾ Related ▾

Name pks-592ff063-c3b0-46b4-81ce-0bcfaafd6246-x1-service-a-84965f57cc-nhb

ID df5da5ee-8b43-4b2a-9990-5f749b9c57ed

Location

Description

Admin Status ● Up

Attachment VIF:792d07b7-dff1-4892-8a1e-85ee07f825ac

Logical Switch pks-592ff063-c3b0-46b4-81ce-0bcfaafd6246-x1-0

Created May 18, 2020 1:55:53 PM by pks-592ff063-c3b0-46b4-81ce-0bcfaafd6246

Last Updated May 18, 2020 1:55:53 PM by pks-592ff063-c3b0-46b4-81ce-0bcfaafd6246

▾ Address Bindings | REFRESH

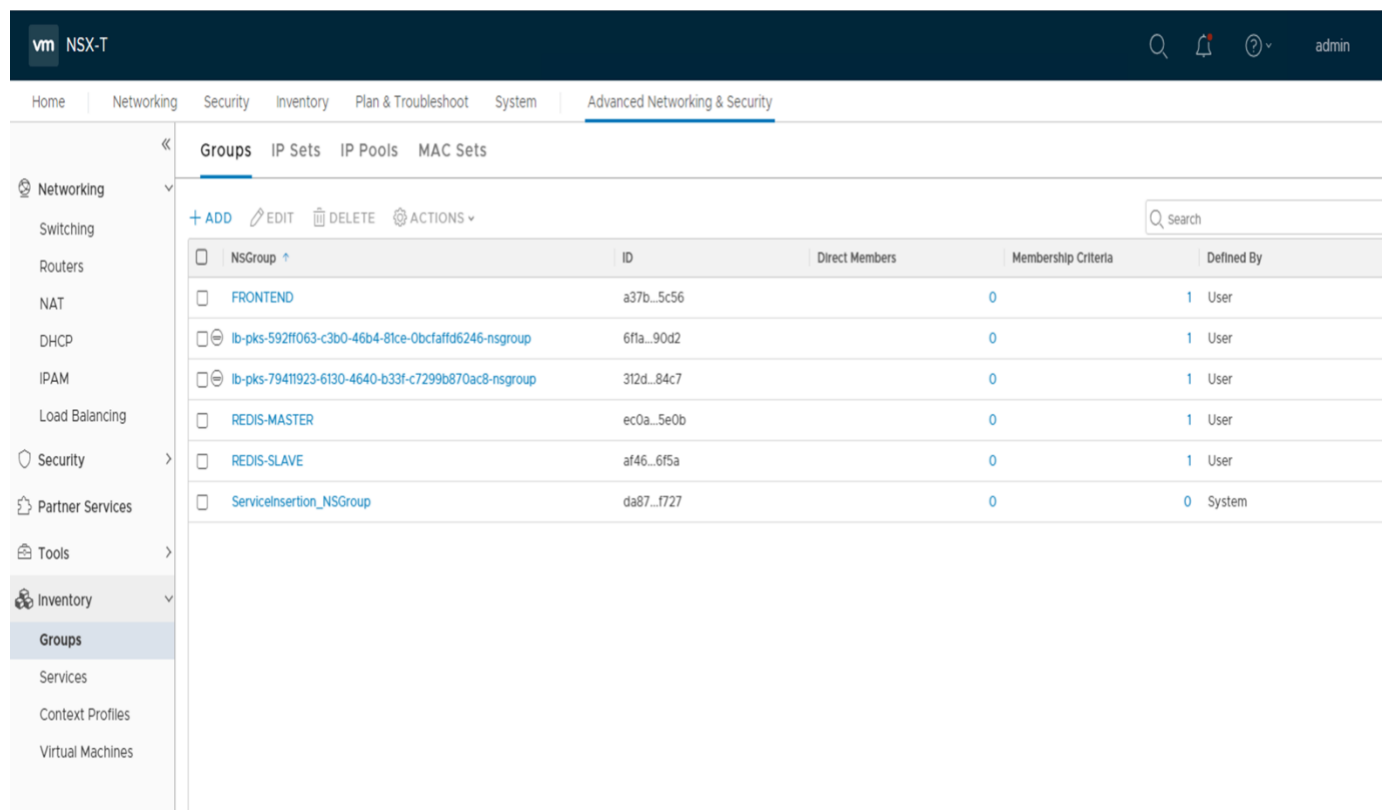
- > Auto Discovered Bindings
- > Manual Bindings
- > Ignore Bindings
- > Realized Bindings

▾ Tags | MANAGE

Tag	Scope
1.2.0	ncp/version
pks-592ff063-c3b0-46b4-81ce-0bcfaafd6246	ncp/cluster
x1	ncp/project
f2cb34bf-c7e1-4e3c-9c2e-8b9505bb16f3	ncp/project_uid
service-a-84965f57cc-nhb	ncp/pod
67dee814-8246-4858-9f06-9c607bbc30b5	ncp/pod_uid
service-a	app

### 3. Create NSX-T NSGroup (NSX-T Security Group)

Login to NSXT and Navigate to Advanced Networking & Security → Inventory → Groups



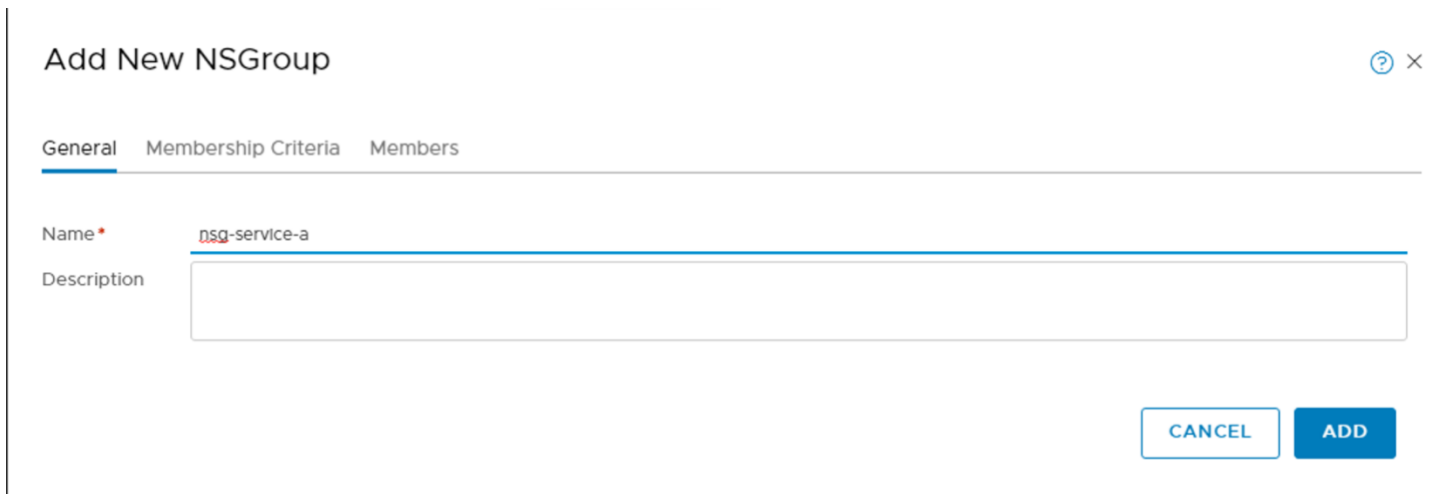
The screenshot shows the NSX-T web interface. The top navigation bar includes 'vm NSX-T' and user 'admin'. The main navigation tabs are Home, Networking, Security, Inventory, Plan & Troubleshoot, System, and Advanced Networking & Security. The left sidebar shows a tree view with categories like Networking, Security, Partner Services, Tools, and Inventory. Under Inventory, 'Groups' is selected. The main content area shows the 'Groups' page with tabs for Groups, IP Sets, IP Pools, and MAC Sets. A table lists existing NSGroups:

NSGroup	ID	Direct Members	Membership Criteria	Defined By
FRONTEND	a37b...5c56	0	1	User
lb-pks-592ff063-c3b0-46b4-8fce-0bcffaf6246-nsgroup	6f1a...90d2	0	1	User
lb-pks-79411923-6130-4640-b33f-c7299b870ac8-nsgroup	312d...84c7	0	1	User
REDIS-MASTER	ec0a...5e0b	0	1	User
REDIS-SLAVE	af46...6f5a	0	1	User
ServiceInsertion_NSGroup	da87...f727	0	0	System

Create a new NSGroup by clicking on ADD

Create an NSGroup for service-a

NSGroup name – nsg-service-a



**Add New NSGroup** ⓘ ×

General Membership Criteria Members

Name \* nsg-service-a

Description

CANCEL ADD

Click on Membership Criteria

And add the following criteria

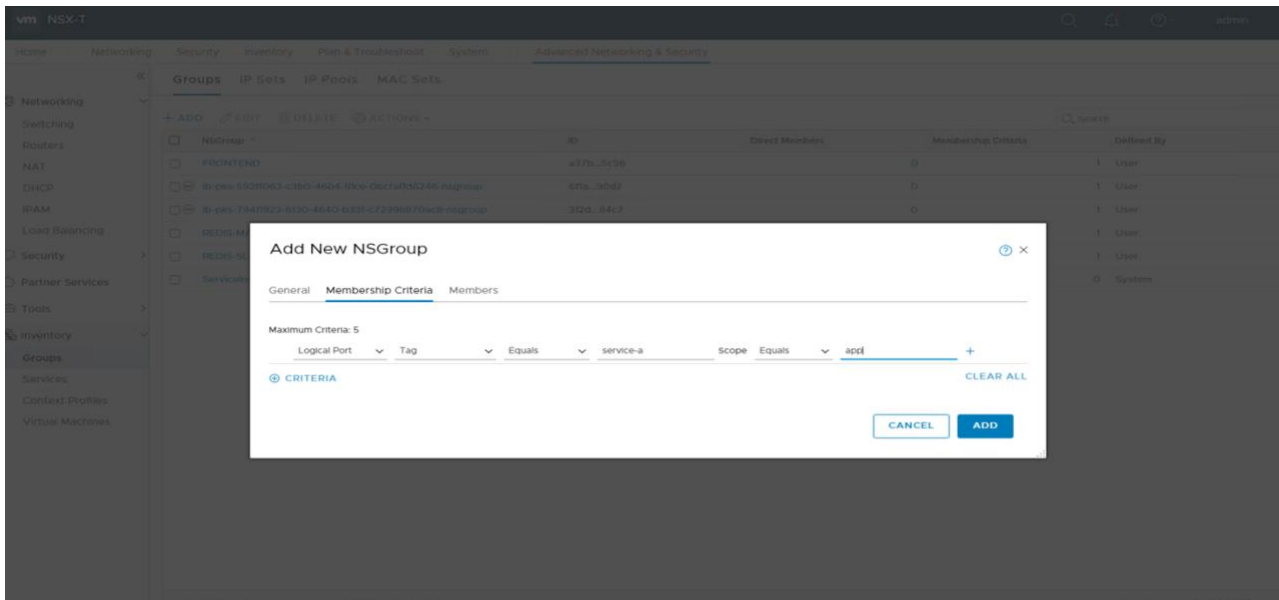
Logical Port > Tag > Equals > service-a > Scope > equals > app

Note these are the values of the tag retrieved in step 2.

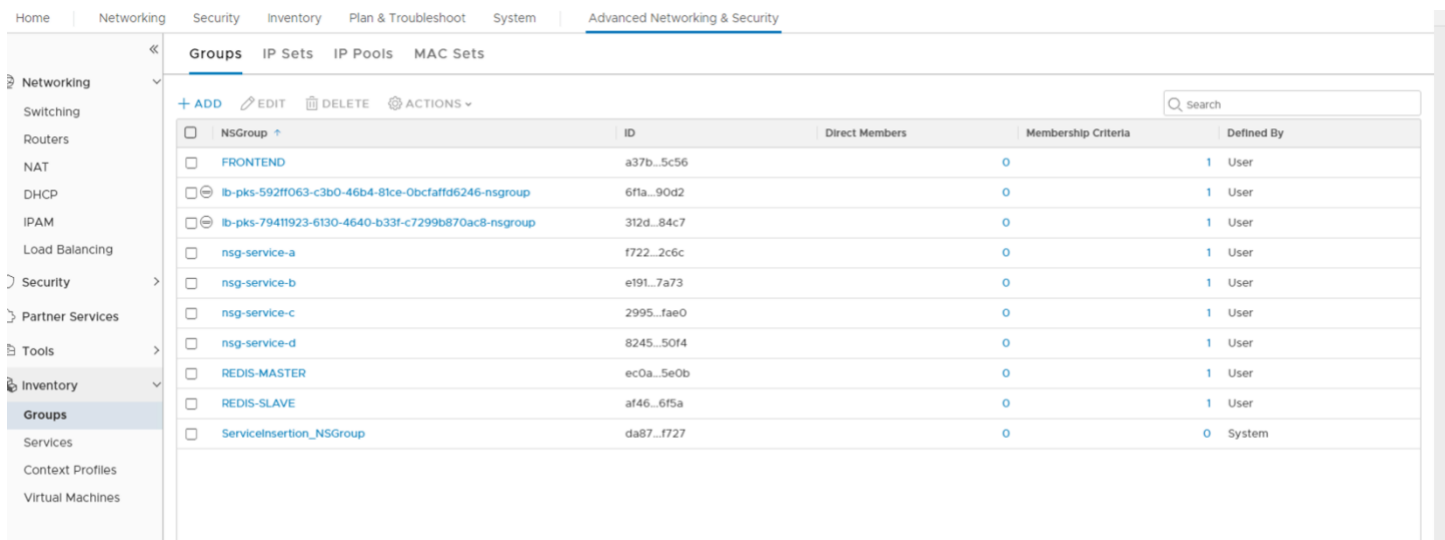
For example, if you defined a Kubernetes tag of “app=service-a” then “service-a” should be used for “Tag” value for the Logical Port and “app” should be used for the “scope”.

If you want to further refine the membership criteria to include only pods in a namespace, you can use the namespace name as the tag and scope should be defined as “ncp/project”. So for this service, the tag will be “x1” and scope “ncp/project”.

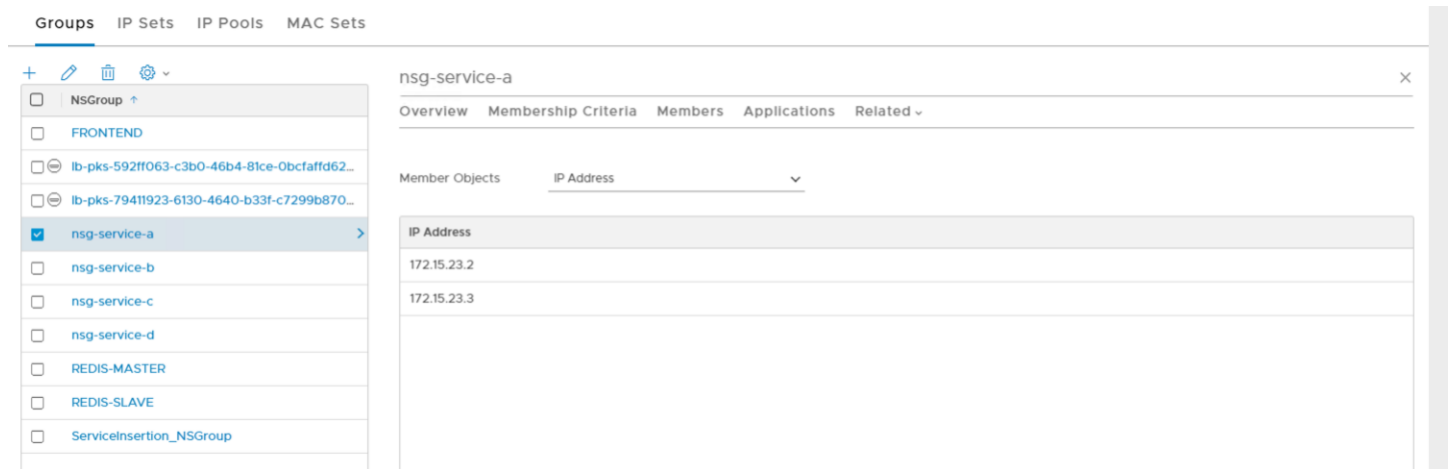
You can use multiple tags to define your membership criteria.



Repeat the same to create NSGroups for service-b, service-c and service-d



Click on the newly created groups and select Members, check IP addresses



Run the following kubectl command from the terminal to verify that the ip-address match the pods ip addresses

`kubectl get po -n x1 -o wide`

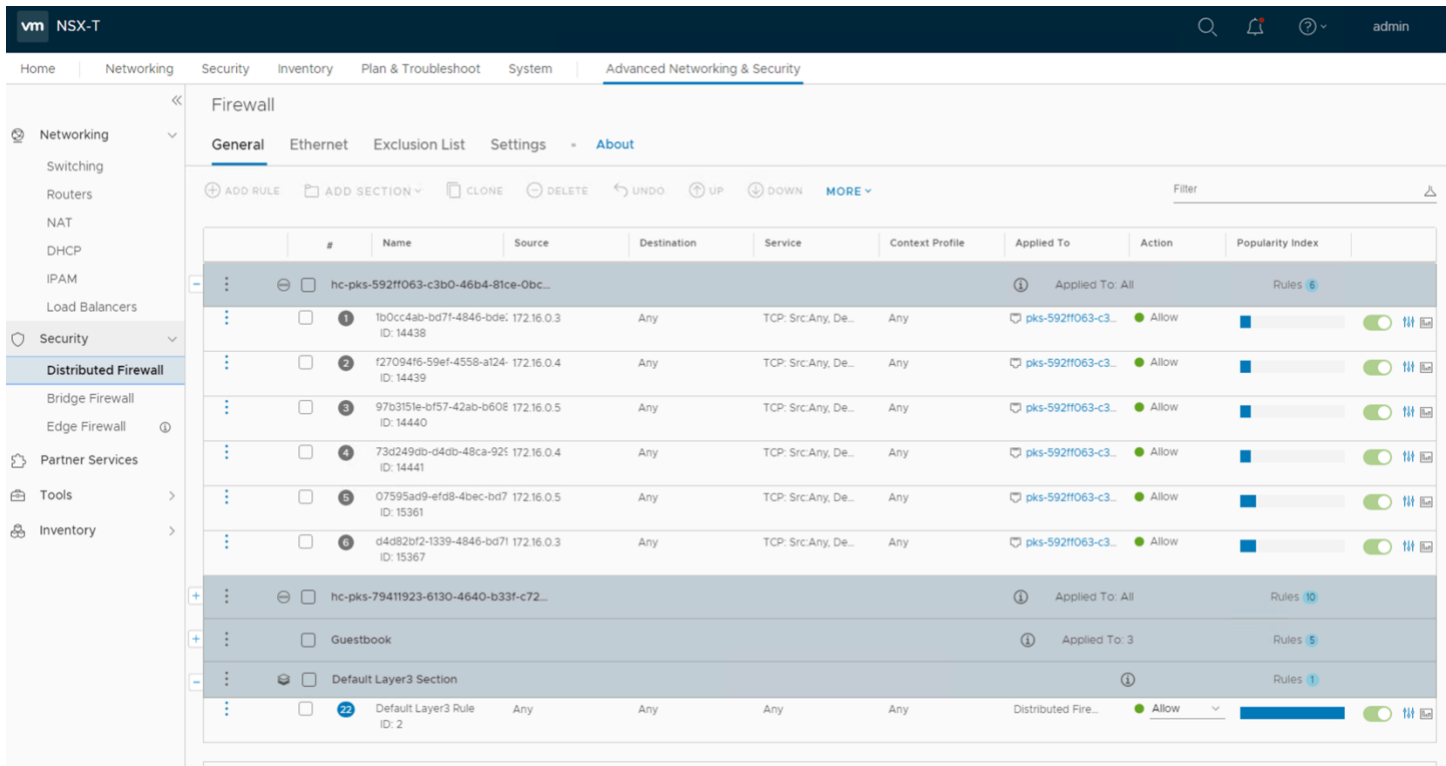
```
ubuntu@cli-vm:~/dmz$ kubectl get po -n x1 -o wide
```

NAME	READY	STATUS	RESTARTS	AGE	IP	NODE	NOMINATED NODE	READINESS GATES
service-a-84965f57cc-nhbrx	2/2	Running	0	127m	172.15.23.3	a60cfb28-f6d8-4022-8dcb-c202aa9335c3	<none>	<none>
service-a-84965f57cc-q72fh	2/2	Running	0	127m	172.15.23.2	f9be16c4-7396-41c6-9c97-7b5280f165f0	<none>	<none>

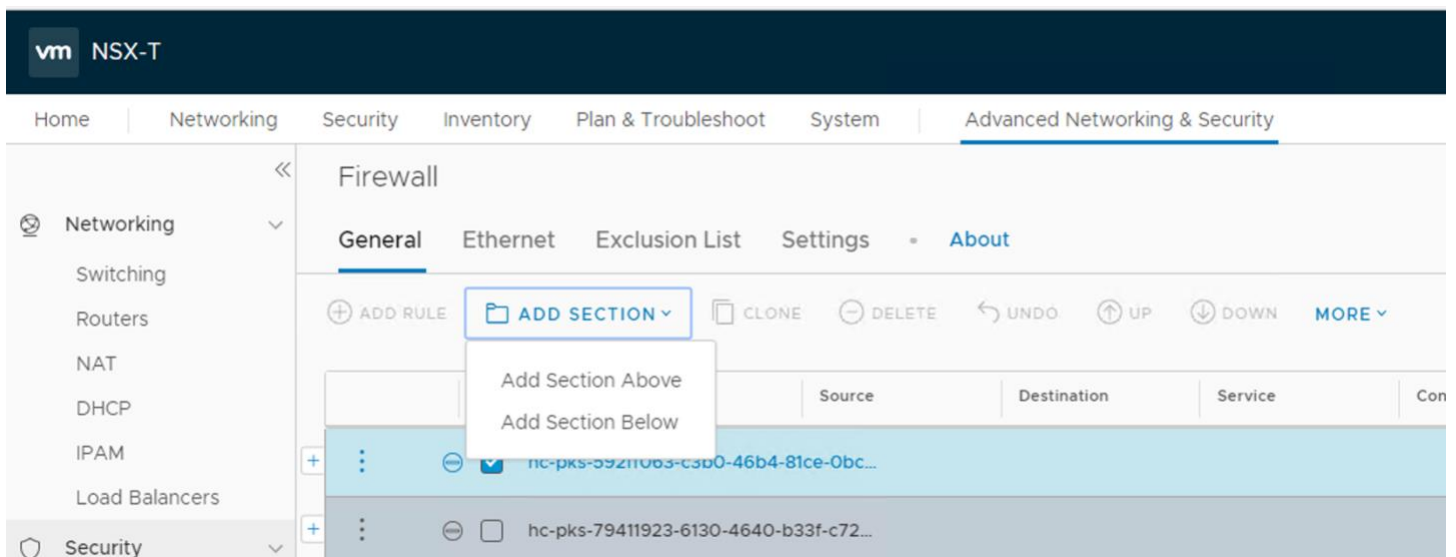
Repeat the same for the other NSGroups created

## 5. Create DFW rules

Login to NSXT and navigate to Advanced Networking & Security → Security → Distributed Firewall



Select the first section listed and navigate to ADD SECTION → Add Section Above



Section Name – PtoPDFW-MicroSegmentation

Object Type – NSGroup

Select – nsg-service-b, nsg-service-c and nsg-service-d

## New Section



Add a new section to organize firewall rules. For example, you might want to have rules for sales and engineering departments in separate sections.

Section Name PtoPDFW-MicroSegmentation

Section Properties ☐ Enable TCP Strict ⓘ ☐ Enable Stateless Firewall ⓘ

Applied To

ⓘ Section level 'Applied To' entities mentioned here, will take precedence over rule level 'Applied To' entities of the same section.

Object Type: NSGroup ▼

Available Objects

<input type="checkbox"/>	Name
<input type="checkbox"/>	nsg-service-a
<input type="checkbox"/>	nsg-service-b
<input type="checkbox"/>	nsg-service-c
<input type="checkbox"/>	nsg-service-d
1 - 9 of 9 Objects	

Selected Objects

<input type="checkbox"/>	Name	Object Type
<input type="checkbox"/>	nsg-service-b	NSGroup
<input type="checkbox"/>	nsg-service-c	NSGroup
<input type="checkbox"/>	nsg-service-d	NSGroup
Max limit: 128		3 Objects

[CREATE NEW NSGROUP](#)

OK

CANCEL



Select the newly created PtoPDFW-MicroSegmentation segment and click on Add rule

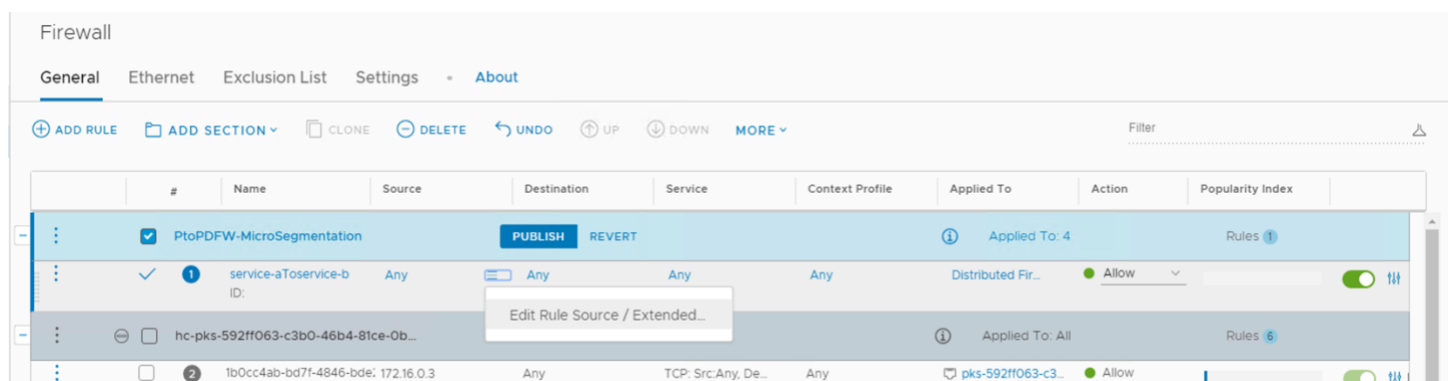
Rule 5: Denies all Traffic to nsg-service-a, nsg-service-b , nsg-service-c and nsg-service-d

ID : DenyAll  
 Source : Any  
 Destination: Any  
 Service: Any  
 ContextProfile: Any  
 Action : Reject

Rule 4: Allow service-a to service-b traffic

ID : service-aToservice-b  
 Source : nsg-service-a  
 Destination: nsg-service-b  
 Service: Any  
 ContextProfile: Any  
 Action : Allow

To select the Source click on the icon next to the source filed and click on Edit Rule Source



Select Object Type as NSGroup

### Specify Source / Extended Source | service-aToservice-b

Specify source for the rule. You can provide container objects or IP addresses from which the communication originated.

Negate Source: Off ☐

Container Objects (1) | IP Addresses (0) | Identity NSGroup ⓘ

Select one or more objects for the source field to the firewall rule

Object Type: NSGroup ▼

Available Objects

<input type="checkbox"/>	Name
<input type="checkbox"/>	🏠 nsg-service-a
<input type="checkbox"/>	🏠 nsg-service-b
<input type="checkbox"/>	🏠 nsg-service-c
<input type="checkbox"/>	🏠 nsg-service-d

1 - 9 of 9 Objects

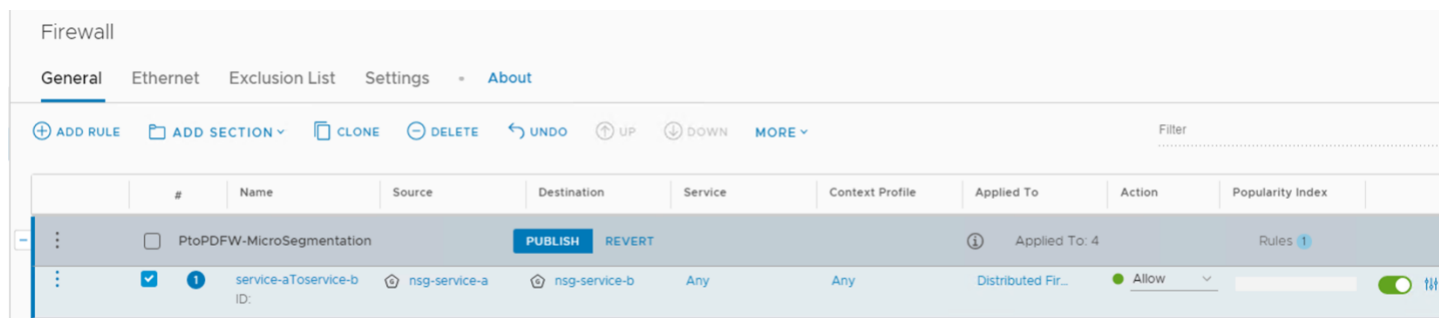
Selected Objects

<input type="checkbox"/>	Name	Object Type
<input type="checkbox"/>	🏠 nsg-service-a	NSGroup

Max limit: 128 1 Objects

[CREATE NEW NSGROUP](#)

Similarly select nsg-service-b for destination.



Rule 3: Allow service-a to service-c traffic

ID : service-aToservice-c  
 Source : nsg-service-a  
 Destination: nsg-service-c  
 Service: Any  
 ContextProfile: Any  
 Action : Allow

Rule 2: Allow service-b to service-c traffic

ID : service-bToservice-c  
 Source : nsg-service-b  
 Destination: nsg-service-c  
 Service: Any  
 ContextProfile: Any  
 Action : Allow

Rule 1: Allow service-c to service-d traffic

ID : service-cToservice-d  
 Source : nsg-service-c  
 Destination: nsg-service-d  
 Service: Any  
 ContextProfile: Any

Action : Allow

Firewall

General Ethernet Exclusion List Settings About

+

ADD RULE

ADD SECTION ▾

CLONE

DELETE

UNDO

UP

DOWN

MORE ▾

Filter

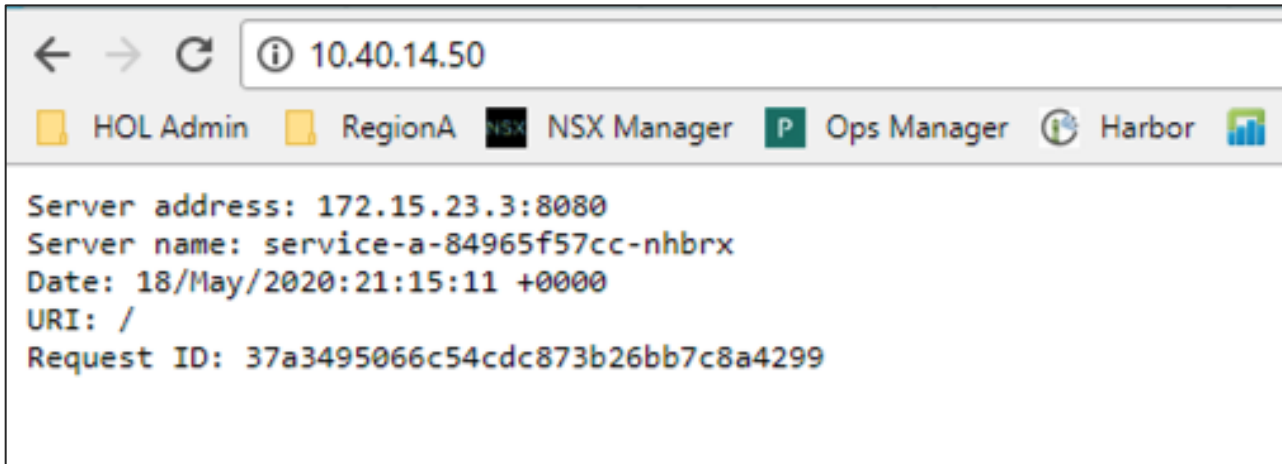
	#	Name	Source	Destination	Service	Context Profile	Applied To	Action	Popularity Index	
	<input checked="" type="checkbox"/>	PtoPDFW-MicroSegmentation					<div><div>1</div>Applied To: 3</div>			Rules <div>5</div>
	<input checked="" type="checkbox"/>	<div>1</div> service-cToService-d ID: 15401	<div></div> nsg-service-c	<div></div> nsg-service-d	Any	Any	Distributed Fir...	<div><div></div>Allow ▾</div>		<div><div></div><div></div></div> <div><div></div></div>
	<input checked="" type="checkbox"/>	<div>2</div> service-bToService-c ID: 15401	<div></div> nsg-service-b	<div></div> nsg-service-c	Any	Any	Distributed Fir...	<div><div></div>Allow ▾</div>		<div><div></div><div></div></div> <div><div></div></div>
	<input checked="" type="checkbox"/>	<div>3</div> service-aToService-c ID: 15401	<div></div> nsg-service-a	<div></div> nsg-service-c	Any	Any	Distributed Fir...	<div><div></div>Allow ▾</div>		<div><div></div><div></div></div> <div><div></div></div>
	<input checked="" type="checkbox"/>	<div>4</div> service-aToService-b ID: 15400	<div></div> nsg-service-a	<div></div> nsg-service-b	Any	Any	Distributed Fir...	<div><div></div>Allow ▾</div>		<div><div></div><div></div></div> <div><div></div></div>
	<input checked="" type="checkbox"/>	<div>5</div> DenyAll ID: 15399	Any	Any	Any	Any	Distributed Fir...	<div><div></div>Reject ▾</div>		<div><div></div><div></div></div> <div><div></div></div>

Select PtoPDFW-Microsegmentation and click on Publish

## Test Traffic Flow

**Traffic flow from external network to k8 cluster.**

Open a browser and browse to the external ip from the previous step



service-a can be reached from the external network through the load balancer which is created in NSX-T

service-b, service-c and service-d are not exposed and hence are not reachable from the external network.

### Traffic flow between pods

We will be using the busy-box container within service-a pod and use curl to check for responses between service.

Source Pod	Destination Pod	Result
service-a	service-b	Success
service-a	service-c	Success
service-a	service-d	Fail
service-b	service-c	Success
service-b	service-d	Fail

service-c	service-d	Success
service-c	service-b	Fail
service-d	service-c	Fail
service-d	service-b	Fail
service-c	service-a	Fail
service-b	service-a	Fail

1. Get pods running on namespace x1

```
kubectl get po -n x1
```

```
ubuntu@cli-vm:~/dmz$ kubectl get po -n x1
NAME                                READY   STATUS    RESTARTS   AGE
service-a-84965f57cc-nhbrx         2/2     Running   0           31m
service-a-84965f57cc-q72fh         2/2     Running   0           31m
ubuntu@cli-vm:~/dmz$
```

2. Get service name for service-b running on namespace y1

```
kubectl get svc -n y1
```

```
ubuntu@cli-vm:~/dmz$ kubectl get svc -n y1
NAME            TYPE        CLUSTER-IP    EXTERNAL-IP  PORT(S)    AGE
svc-service-b   ClusterIP   10.100.200.12 <none>       80/TCP     38m
```

3. Exec into the busybox container running on a service-a pod

```
kubectl exec -n <namespace> <podname> -it -c busybox -- /bin/sh
```

Eg

```
kubectl exec -n x1 service-a-84965f57cc-nhbrx -it -c busybox -- /bin/sh
```

```
ubuntu@cli-vm:~/dmz$ kubectl exec -n x1 service-a-84965f57cc-nhbrx -it -c busybox -- /bin/sh
/home #
/home #
```

Use curl to reach service-b in namespace y1

```
curl http://svc-service-b.y1
```

NOTE: the service name is prepended by svc in this case, this is defined in the yaml we used to create the K8 resources. The service name is also appended by the namespace.

4. This results in a successful response

```
ubuntu@cli-vm:~/dmz$ kubectl exec -n x1 service-a-84965f57cc-nhbrx -it -c busybox -- /bin/sh
/home # curl http://svc-service-b.y1
Server address: 172.15.13.3:8080
Server name: service-b-86d8c888c7-jpwwf9
Date: 18/May/2020:21:38:00 +0000
URI: /
Request ID: cfe8d330cac2894b204a1b57082e10cc
```

5. Use the curl command to check the service response of service-c in namespace z1

```
curl http://svc-service-c.z1
```

This should be successful as well

6. Use the curl command to check the service response of service-d in namespace z1

```
curl http://svc-service-d.z1
```

This would fail

```
/home # curl http://svc-service-d.z1  
curl: (7) Failed connect to svc-service-d.z1:80; Connection refused
```

Exec into each pod and check connectivity between pods.





VMware, Inc. 3401 Hillview Avenue Palo Alto CA 94304 USA Tel 877-486-9273 Fax 650-427-5001 [www.vmware.com](http://www.vmware.com).

Copyright © 2019 VMware, Inc. All rights reserved. This product is protected by U.S. and international copyright and intellectual property laws. VMware products are covered by one or more patents listed at [vmware.com/go/patents](http://vmware.com/go/patents). VMware is a registered trademark or trademark of VMware, Inc. and its subsidiaries in the United States and other jurisdictions. All other marks and names mentioned herein may be trademarks of their respective companies. Item No: vmw-wp-tech-temp-word-102-proof 5/19