vFORUM

# CONTAINERS AND KUBERNETES WITH NSX


Anthony Burke
Solutions Architect
Network and Security Business Unit,
Customer Success | Ambassador, Office of the CTO
@pandom_


vmware

POSSIBLE BEGINS WITH YOU

# AGENDA

- Containers
- Kubernetes
- Kubernetes and NSX-T
- Planespotter

vmware

vFORUM

# Containers

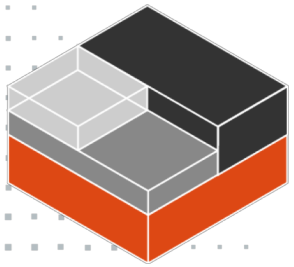
## What is a container?

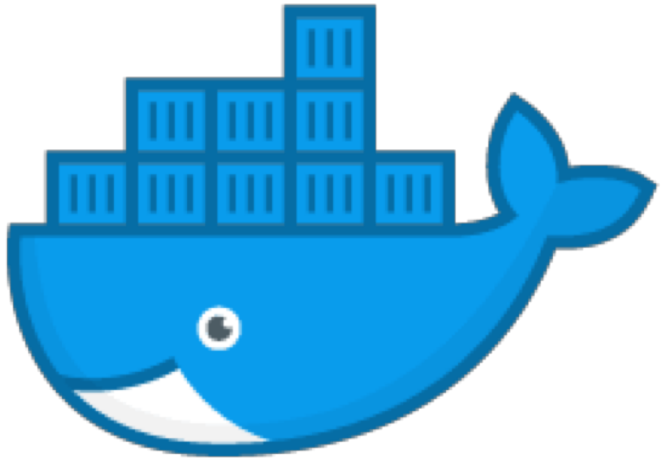
vmware

vFORUM

# We're running containers

## A container runtime





docker

rkt

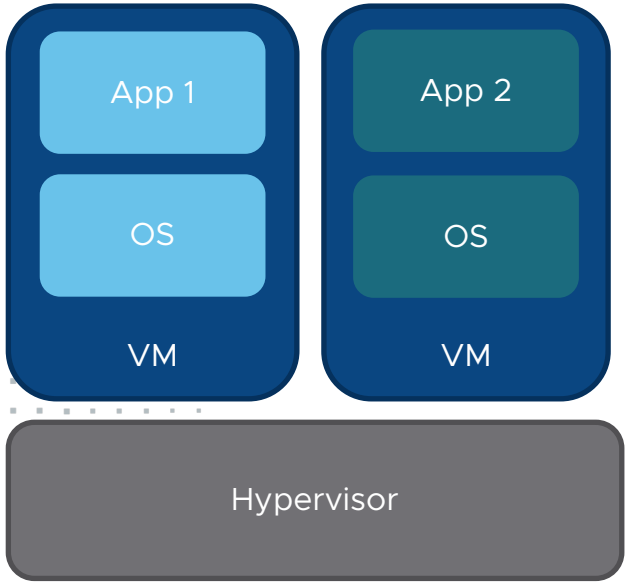
# How containers are “sold” – made human friendly

Virtual Machine can run one or many apps

Libraries and executables are shared

Consume the resources allocated to VM

Each VM has a full OS / App / Binaries

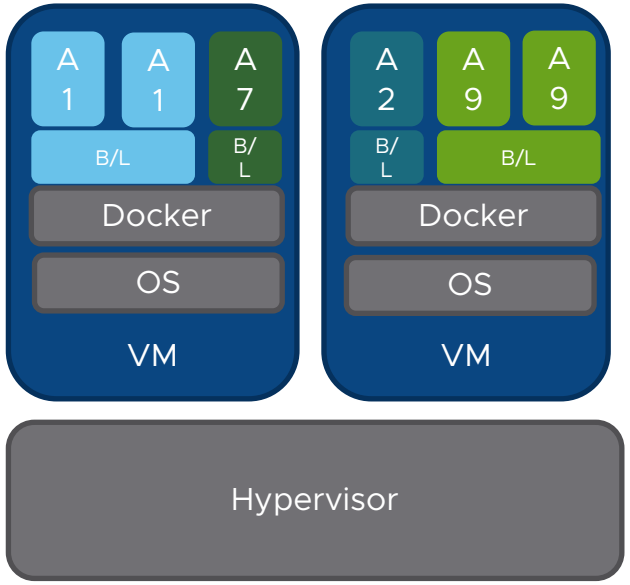



A container runs a process along with its dependencies

Share host OS kernel

Process isolation from other apps on same host

Lightweight – Just the app (NGINX, Redis, MySQL)

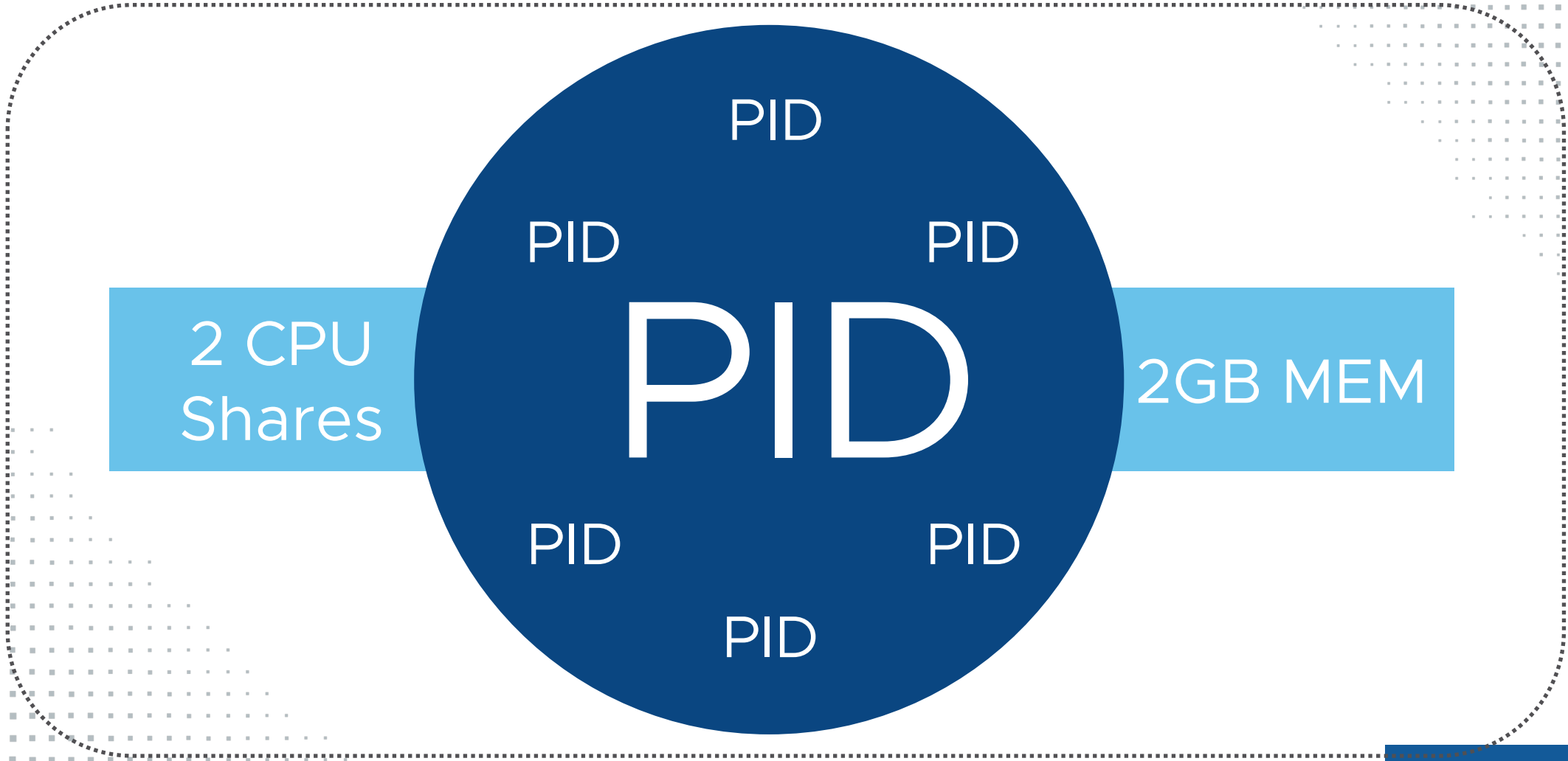



vmware

vFORUM


That
is a lie

# How linux sees ‘containers’



# Containers are a lie

## What containers really are

```
vmware@k8s-node01a:~$ top
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
 7714 root      20   0  604352 108412  37740 S   7.3  2.7 601:48.51 kubelet
 1560 root      20   0 1188268  55248   8548 S   2.0  1.4 170:38.38 dockerd
24349 nobody    20   0  791068 285364  11412 S   1.7  7.1 101:51.55 prometheus
 3951 root      20   0   45224  27868  17296 S   1.0  0.7  38:03.55 coredns
 2690 root      20   0   43832  25944  16288 S   0.7  0.6  38:09.25 kube-proxy
24007 root      20   0   29132   7444   2892 S   5.9  0.2   9:40.25 alertmanager
```

```
vmware@k8s-node01a:~$ pstree -p 24349
prometheus(24349)─┬─{prometheus}(24378)
                  ├─{prometheus}(24379)
                  ├─{prometheus}(24380)
                  ├─{prometheus}(24384)
                  ├─{prometheus}(24386)
                  ├─{prometheus}(24388)
                  ├─{prometheus}(24389)
```

```
vmware@k8s-node01a:~$ pstree -p 3951
coredns(3951)─┬─{coredns}(3984)
              ├─{coredns}(3985)
              ├─{coredns}(3986)
              ├─{coredns}(3987)
              ├─{coredns}(3992)
              ├─{coredns}(4009)
```

# Control Groups

## Allocating resources to a Namespace

```
vmware@k8s-node01a:~$ ls /sys/fs/cgroup/memory/kubepods/besteffort/pod0bbd0491-cbb9-11e8-8efc-005056a3e23d/
18bcb52de19c8426afe024a5f364da352f65602a4d9ee5047c6a13ab59b1ae1c  memory.kmem.usage_in_bytes
b203deb02da6774ba9a3f65a439a5b5135c1dc50436554350ce8608bb2276022  memory.limit_in_bytes
cgroup.clone_children                                             memory.max_usage_in_bytes
cgroup.event_control                                              memory.move_charge_at_immigrate
cgroup.procs                                                      memory.numa_stat
memory.failcnt                                                    memory.oom_control
memory.force_empty                                                memory.pressure_level
memory.kmem.failcnt                                               memory.soft_limit_in_bytes
memory.kmem.limit_in_bytes                                        memory.stat
memory.kmem.max_usage_in_bytes                                    memory.swappiness
memory.kmem.slabinfo                                              memory.usage_in_bytes
memory.kmem.tcp.failcnt                                           memory.use_hierarchy
memory.kmem.tcp.limit_in_bytes                                    notify_on_release
memory.kmem.tcp.max_usage_in_bytes                                tasks
memory.kmem.tcp.usage_in_bytes
vmware@k8s-node01a:~$ cat /sys/fs/cgroup/memory/kubepods/besteffort/pod0bbd0491-cbb9-11e8-8efc-005056a3e23d/memory.kmem.limit_in_bytes
640000000
vmware@k8s-node01a:~$ ls /sys/fs/cgroup/cpuacct/kubepods/besteffort/pod0bbd0491-cbb9-11e8-8efc-005056a3e23d/
18bcb52de19c8426afe024a5f364da352f65602a4d9ee5047c6a13ab59b1ae1c  cpuacct.stat          cpu.cfs_quota_us   tasks
b203deb02da6774ba9a3f65a439a5b5135c1dc50436554350ce8608bb2276022  cpuacct.usage         cpu.shares
cgroup.clone_children                                             cpuacct.usage_percpu  cpu.stat
cgroup.procs                                                      cpu.cfs_period_us     notify_on_release

vmware@k8s-node01a:~$ cat /sys/fs/cgroup/cpuacct/kubepods/besteffort/pod0bbd0491-cbb9-11e8-8efc-005056a3e23d/cpu.shares
2
```

# Demo: Docker up

1. Pull basic hello world
2. Validate image
3. Run image
4. Browse Website
5. Attach to the namespace

1. Live demo on macOS
2. Backup video

# Kubernetes

## Schedule at scale

vmware

vFORUM

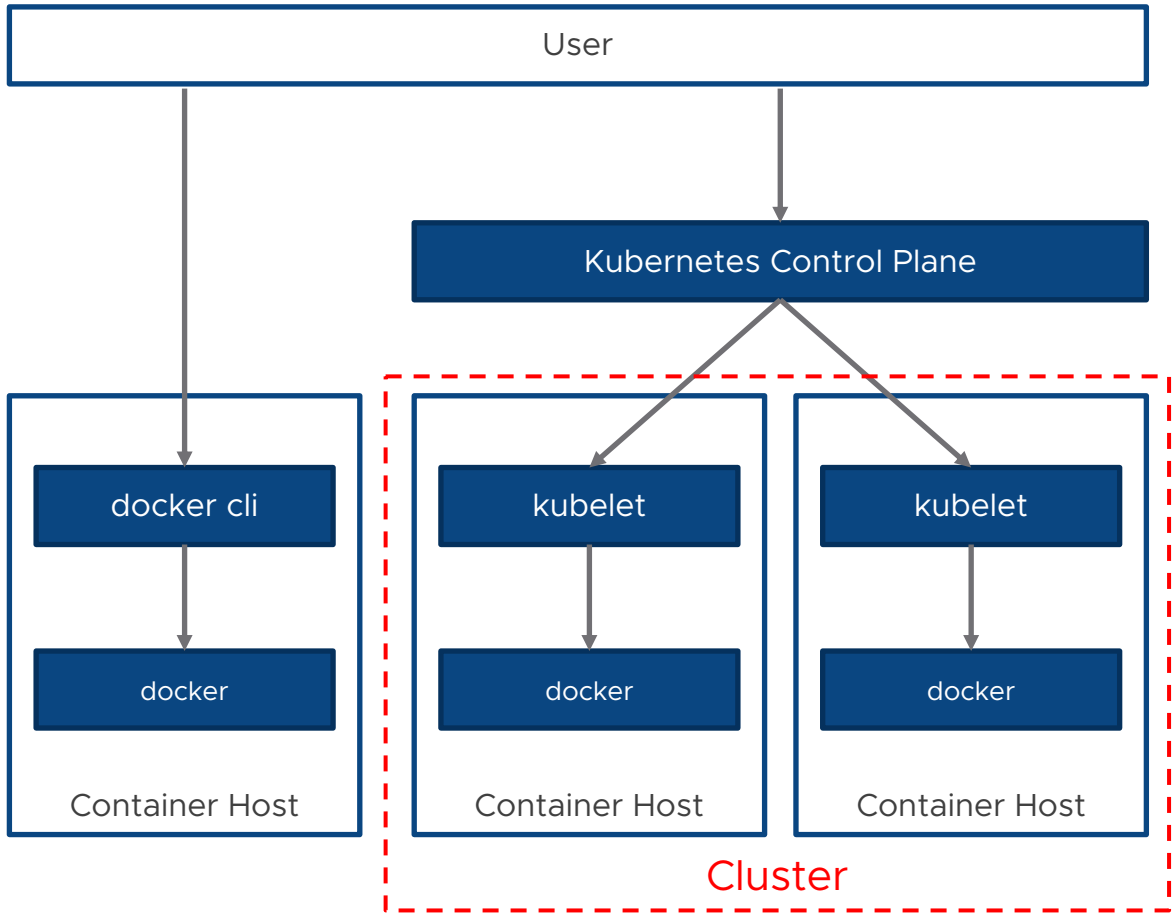
# Why is Kubernetes needed?

Orchestration across tens, hundreds, and thousands of containers

Developers don't care about storage/network/infrastructure nuances – they just want to deploy their application

Kubernetes centralizes orchestration providing

- Container Scheduling
  - AZs/Distribution/Behavior
- Container Management
  -Monitoring/Recovery/Updates/Logging
- Service Endpoints
  - Discovery/HA/Load Balancing/Auto Scale
- External Services
  - Network Services/Storage and Volume management

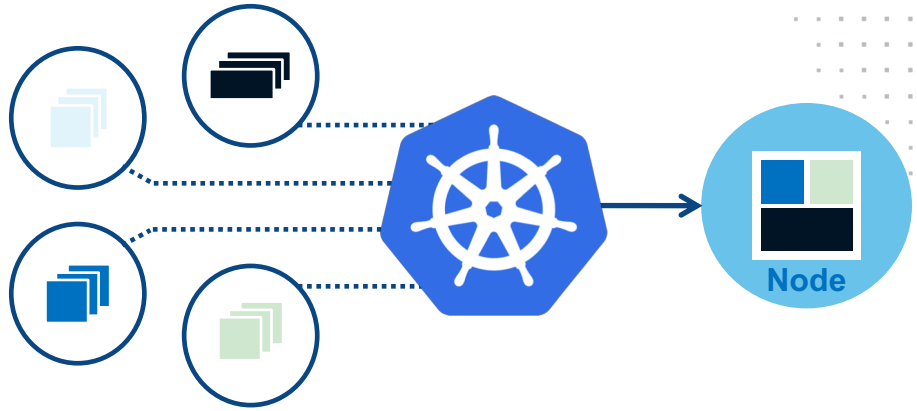


# What is Kubernetes used for?

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

**Major Features**

- Deploy applications quickly and predictably
- Scale applications on the fly
- Roll out new features seamlessly
- Optimize use of hardware
- Self-healing(Restart, replace and reschedule containers)



# Kubernetes

## Basic Objects

Namespaces

ReplicaSets/ReplicationController

Pod

Service

Ingress

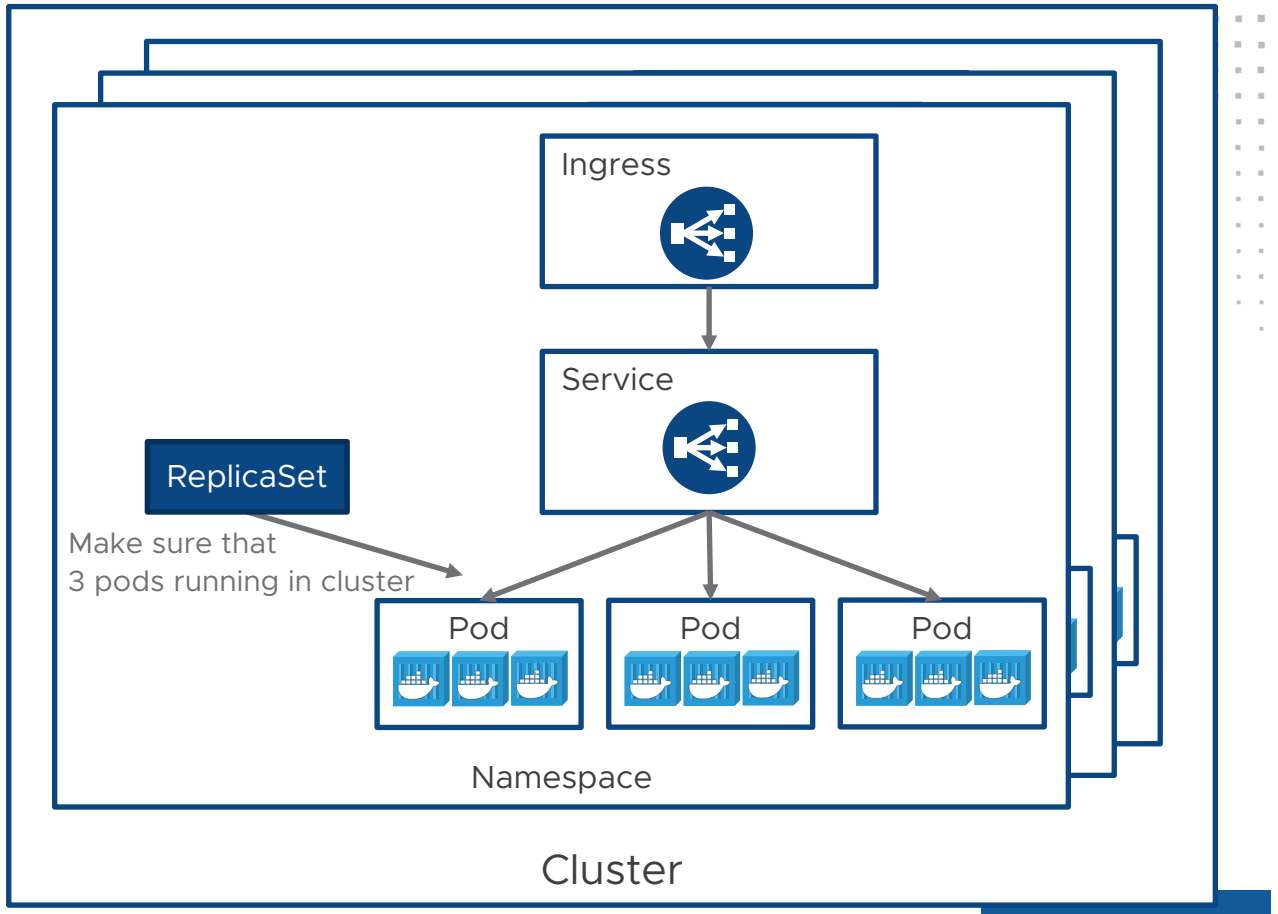
NetworkPolicy

Label

Kubernetes Cluster specific

*Node*

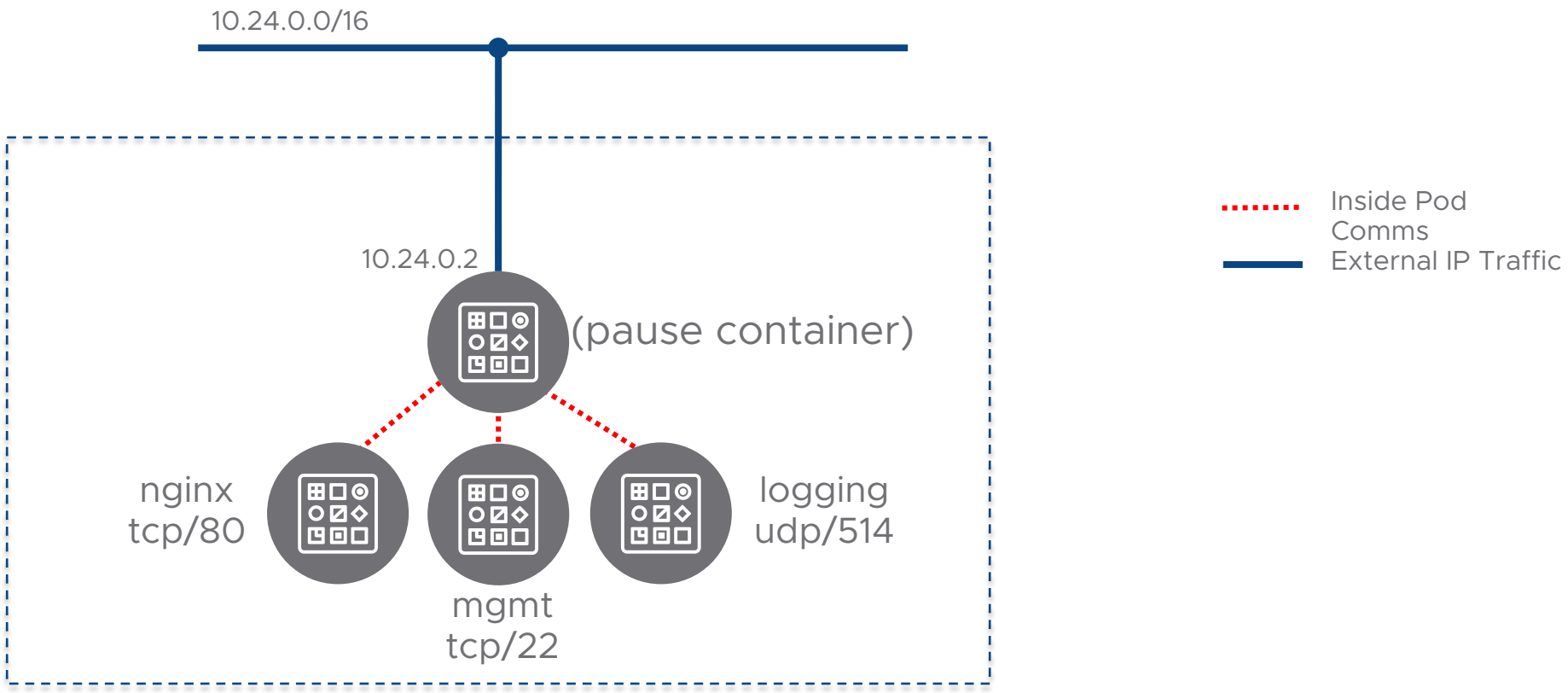
*Cluster*




vmware

vFORUM

# Kubernetes Pod




- A Pod is a group of one or more containers that shares an IP address and a data volume
- A function can be made up on many pods delivering a service such as nginx delivering a web function
- A namespace can have many pods providing different functions such as nginx, redis, mysql for apps.

# Kubernetes Labels

Labels are used as metadata for a resource

Labels provide a way of matching for use in memberships, inclusion, exclusion, or identification

In the case of multiple requirements, all must be satisfied so the comma separator acts as a logical *AND* (&&) operator.

An empty label selector (that is, one with zero requirements) selects every object in the collection.

A null label selector (which is only possible for optional selector fields) selects no objects.

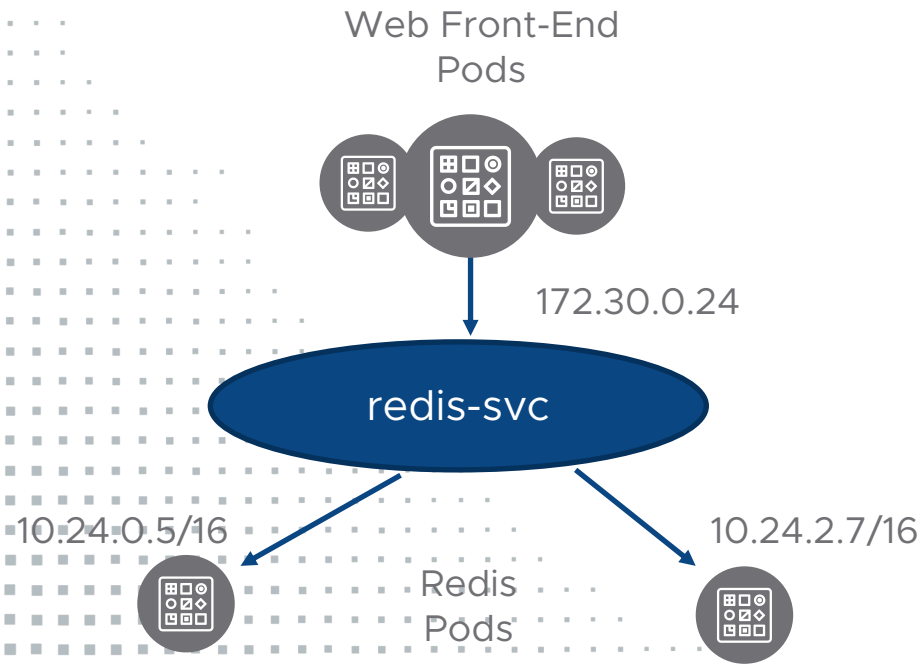
```
vmware@k8s-node01a:~$ cat example-svc.yaml
apiVersion: v1
kind: Service
metadata:
labels:
    app: nginx
    role: web
  name: nginx-demo-svc
spec:
  ports:
  - name: tcp
    port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: nginx
    role: web
  type: ClusterIP
```

```
vmware@k8s-node01a:~$ cat example-networkpolicy.yaml
# Relatively Newer object supports matchLabels and matchExpressions
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: nginx-app-policy
spec:
  podSelector:
    matchLabels:
      app: nginx
      role: app
  policyTypes:
  - Ingress
  ingress:
  - from:
    - podSelector:
        matchLabels:
          app: nginx
        matchExpressions:
          - {key: role, operator: In, values: [web, db, client]
```

https://kubernetes.io/docs/concepts/overview/working-with-objects/labels/

# Kubernetes Service

```
vmware@k8s-node01a:~$ kubectl describe svc redis-svc
Name:                   redist-svc
Namespace:              default
Labels:                 name=redis-svc
Selector:               name=redis-svc
Type:                   ClusterIP
IP:                     172.30.0.24
Port:                   <unnamed> 6379/TCP
Endpoints:   10.24.0.5:6379, 10.24.2.7:6379
```

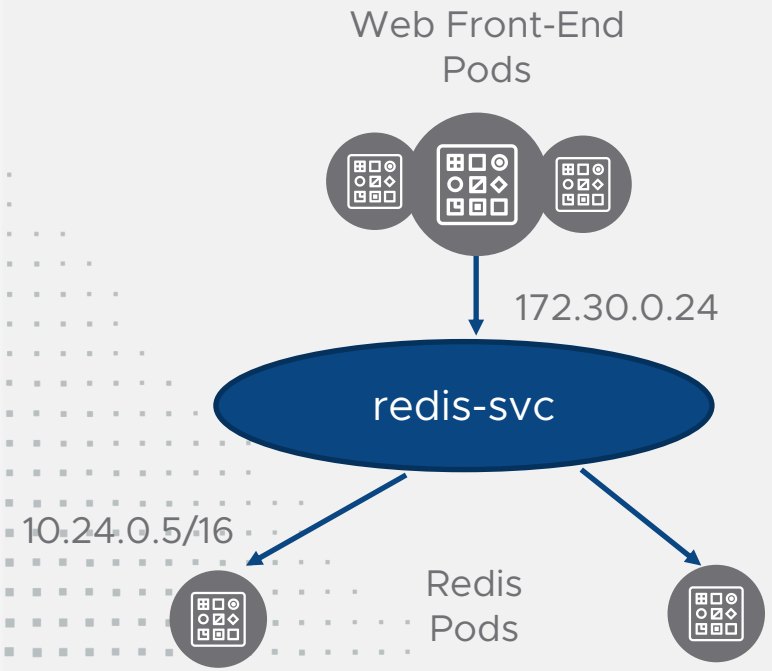



## Kubernetes Service

- **Gist:** A Kubernetes Service is an abstraction which defines a logical set of Pods
- **East/West Load-Balancing**: In terms of networking a service usually contains a cluster IP, which is used as a Virtual IP reachable internally on all Nodes
- **IPTables**: In the default upstream implementation IPTables is used to implement distributed east/west load-balancing
- **DNS**: A service is also represented with a DNS names, e.g. '*redis-svc.cluster.local*' in the Kubernetes dynamic DNS service (CoreDNS) or through environment variable injection
- **External Access**: A K8s Service can also be made externally reachable through **all** Nodes IP interface using 'NodePort' exposing the Service through a specific UDP/TCP Port
- **Type**: In addition to ClusterIP and NodePort, some cloud providers like GCE & OpenStack support using the type 'LoadBalancer' to configure an external LoadBalancer to point to the Endpoints (Pods)

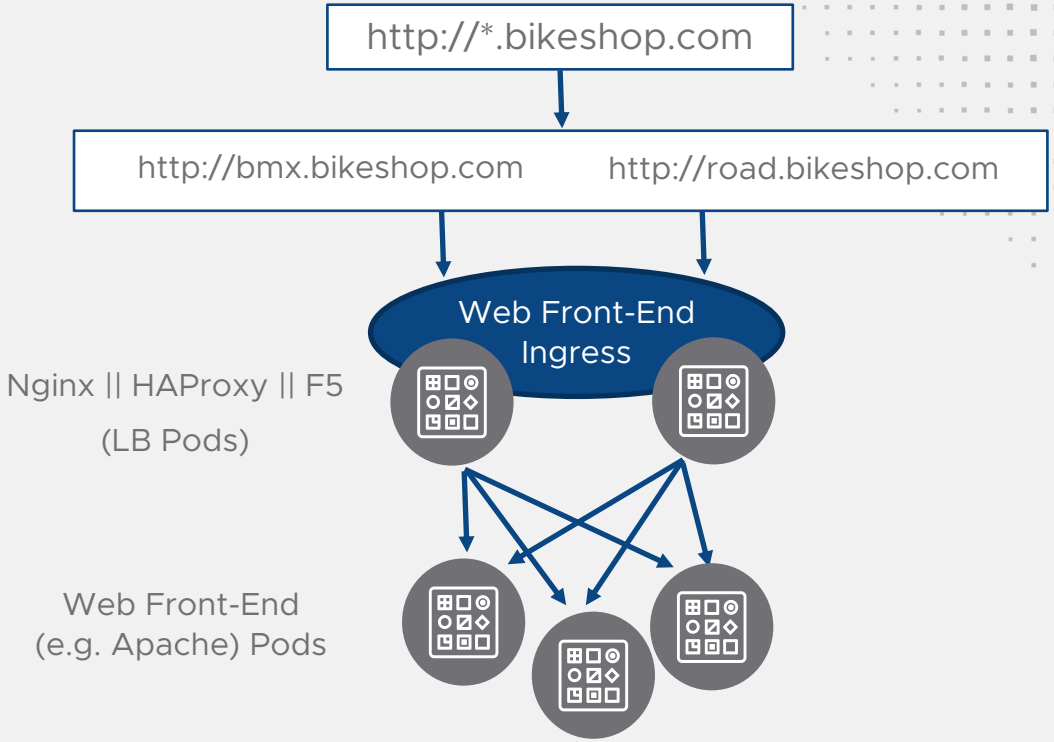
# Kubernetes Load Balancing

## East-West Load Balancing




East-West Load Balancing is provided through Kubernetes Service using ClusterIP & IPTables

## North-South Load Balancing




Can be achieved through Kubernetes Ingress or External third Party Load Balancer using NodePort

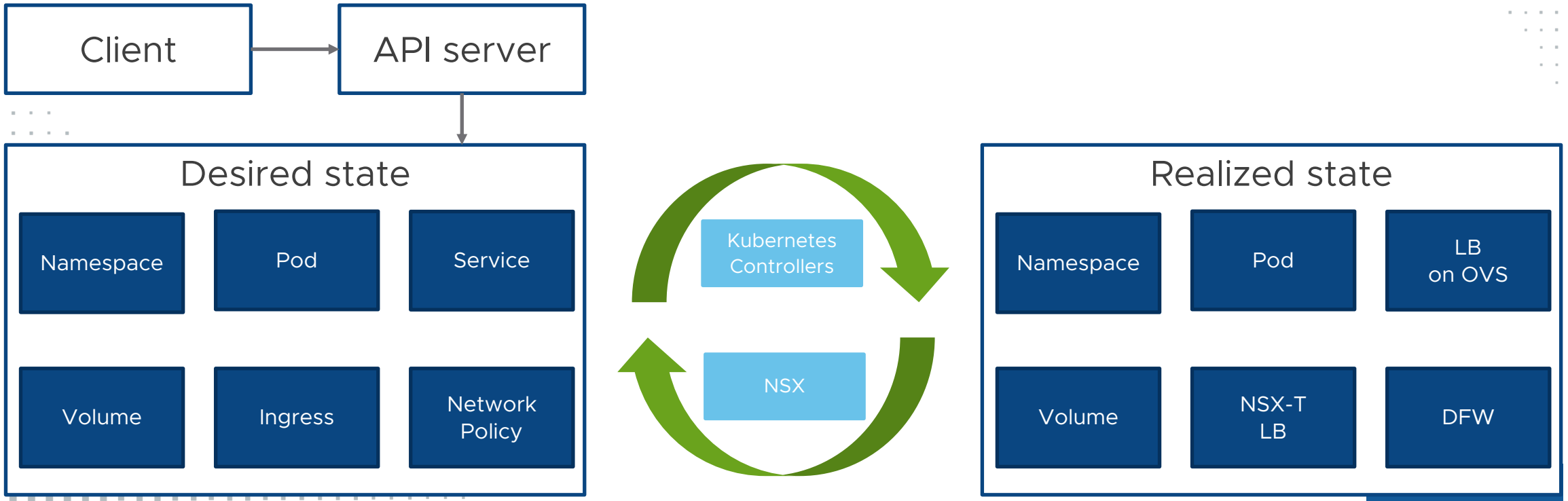
# Kubernetes Objects (Resource Types)

| Component | Description |
| --- | --- |
| Pods | A grouping of one or more containers as an atomic unit |
| Namespaces | A way to organize items in a cluster (+ RBAC) |
| Labels, Annotations & Selectors | Tags for component grouping and methods to access them |
| Service Discovery | An object associated to a label selector to provide a LB and Service DNS |
| ReplicaSets | A cluster wide Pod manager providing Pod scaling |
| DaemonSets | A Pod manager to ensure a Pod is scheduled across a Cluster Node set |
| StatefulSets | Replicated Pods where each Pod gets an indexed hostname |
| Jobs | A Pod which runs until the process returns a successful termination |
| Deployments | Manages the release rollout of new versions of Pods |

# How does Kubernetes do its thing?

## Concept

Kubernetes control plane work to make the actual state based of desired state which is made by user/client. There are some running controllers which has loop which gets rid of loop between desired state and realized state.



# Demo: Kubernetes

1. Schedule a single nsx-demo RC
2. Scale pods

1. Live demo via Singapore
2. Backup video

# Kubernetes and NSX-T

## NSX-T as a Network Provider

vmware

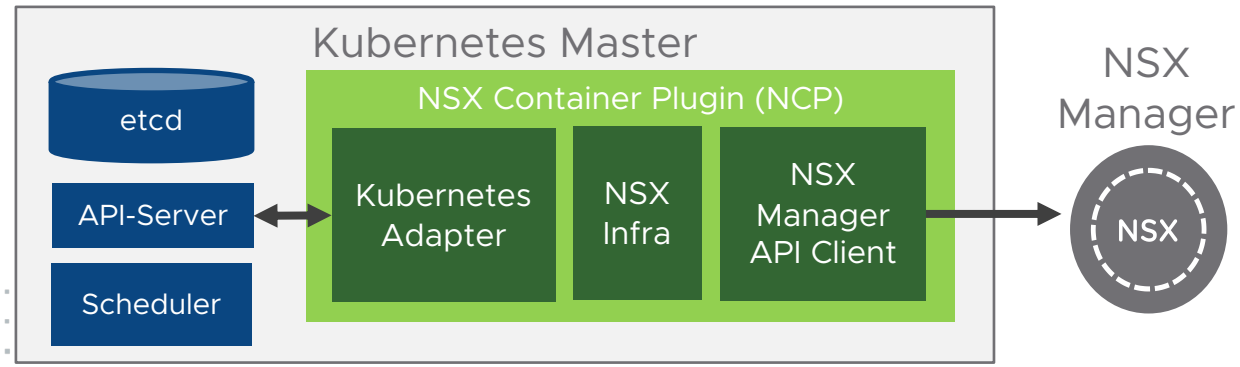
vFORUM

# NSX-T Container Plugin

## Facilitating the NSX-T - K8s Integration

NSX Container Plugin (NCP) is the core component that provides integration between K8s and the NSX Manager.

- Runs as a container inside a K8s pod
- Monitors for changes of relevant objects (namespaces, pods) on K8s API server
- Creates the relevant NSX Objects (logical switches, routers) on creation of K8s constructs
- NCP is built in a modular way, so that individual adapters can be added for different CaaS and PaaS systems

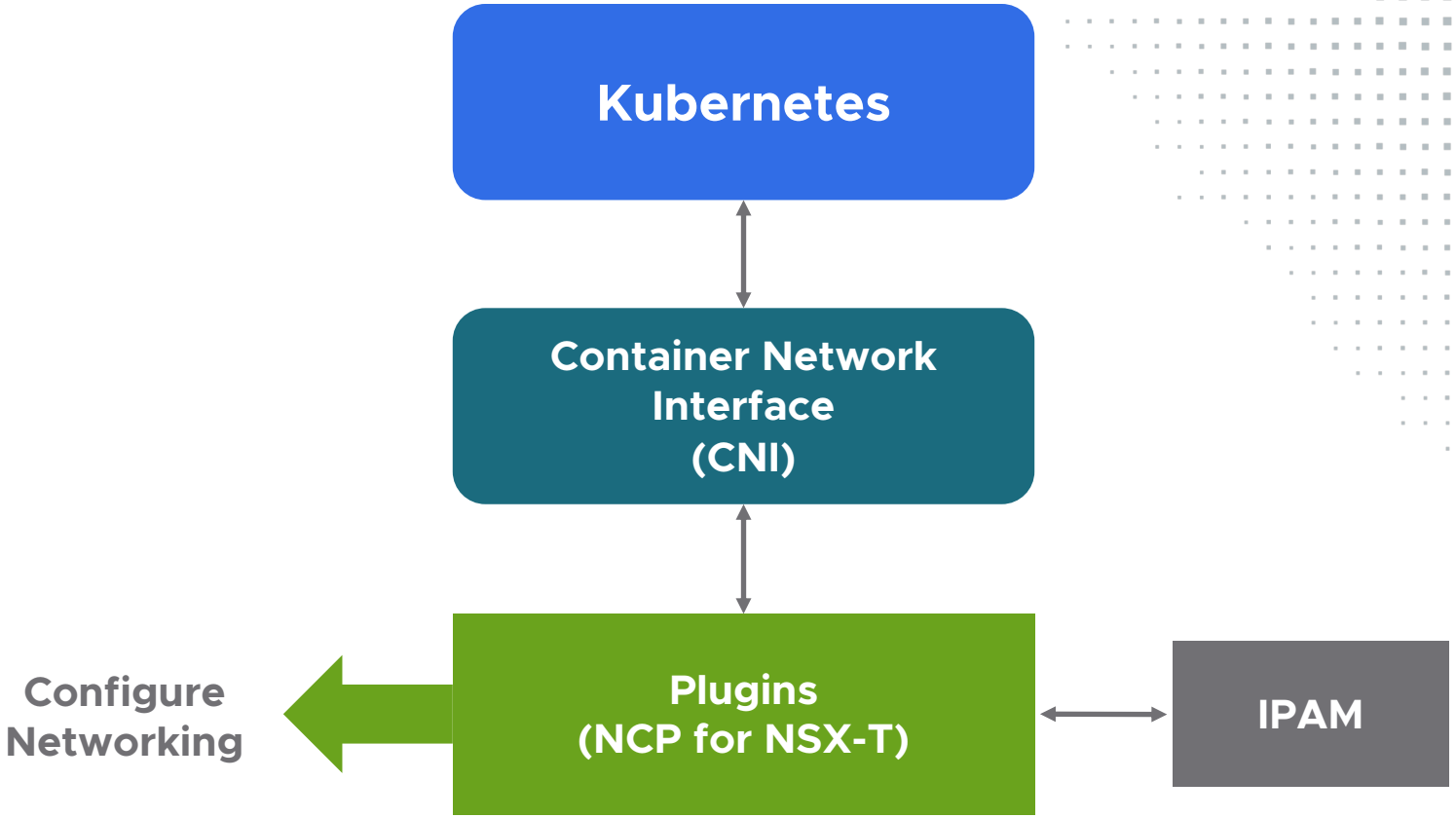


# Container Networking Interface

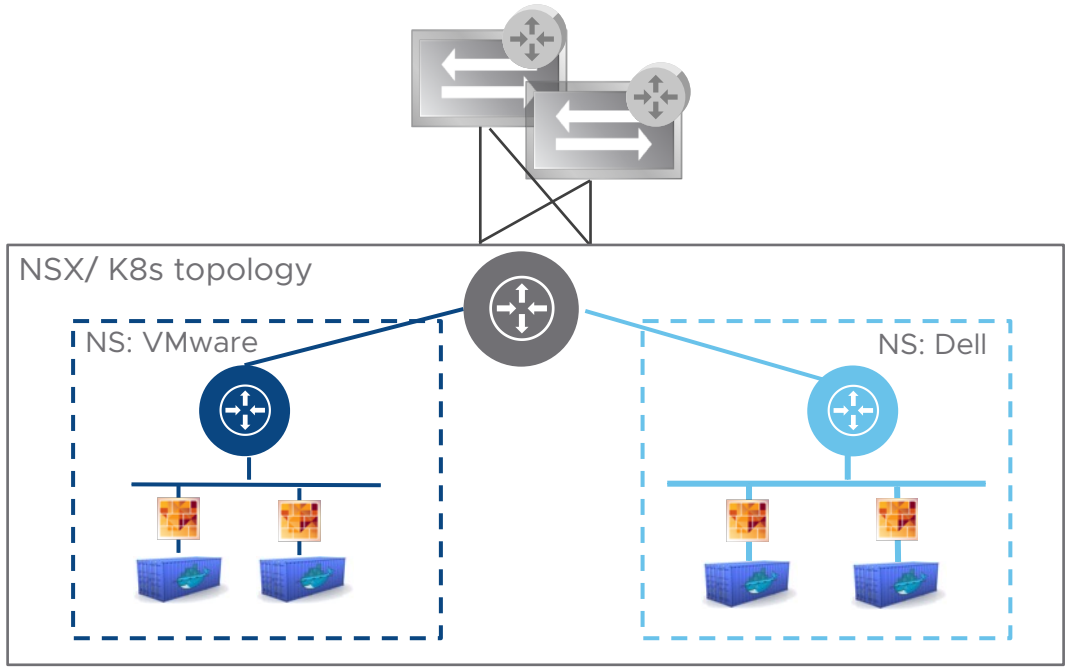
Connecting the Dots …

## Network Container Plugin (NCP)

- Common interface between container runtime and implementation
- Plugins responsible for the network plumbing – IPAM/switching/firewall/load balancing – for container workloads
- Kubernetes commands stay the same; NSX does networking functions
- NSX Container Plugin (NCP) results in the NSX-T to Kubernetes integration



# Kubernetes NSX Topology

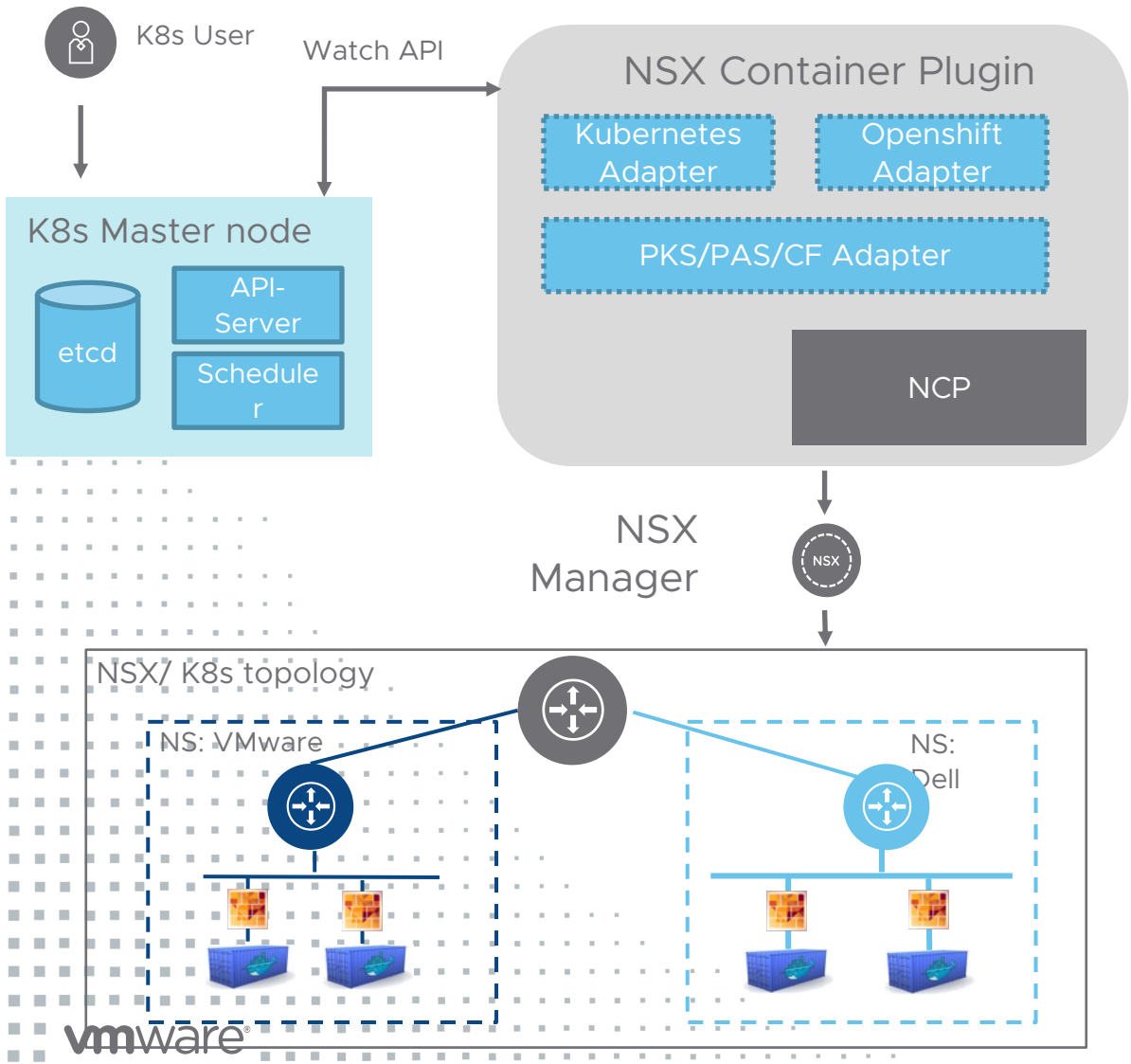



## Kubernetes NSX Topology

- **Namespaces:** Dynamically builds a separate network topology per K8s namespace, every K8s namespace gets one or mode logical switches and one Tier-1 router
- **Nodes**: Are not doing IP routing, every Pod has its own logical port on a NSX logical switch. Every Node can have Pods from different Namespaces and with this from different IP Subnets / Topologies
- **Firewall:** Every Pod has DFW rules applied on its Interface
- **Routing**: High performant East/West and North/South routing using NSX's routing infrastructure, including dynamic routing to physical network
- **Visibility and troubleshooting:** Every Pod has a logical port on the logical switch with:
  - Counters
  - SPAN / Remote mirroring
  - IPFIX export
  - Traceflow / Port-Connection tool
  - Spoofguard
- **IPAM:** NSX is used to provide IP Address Management by supplying Subnets from IP Block to Namespaces, and Individual IPs and MAC to Pods

# Kubernetes and NSX components

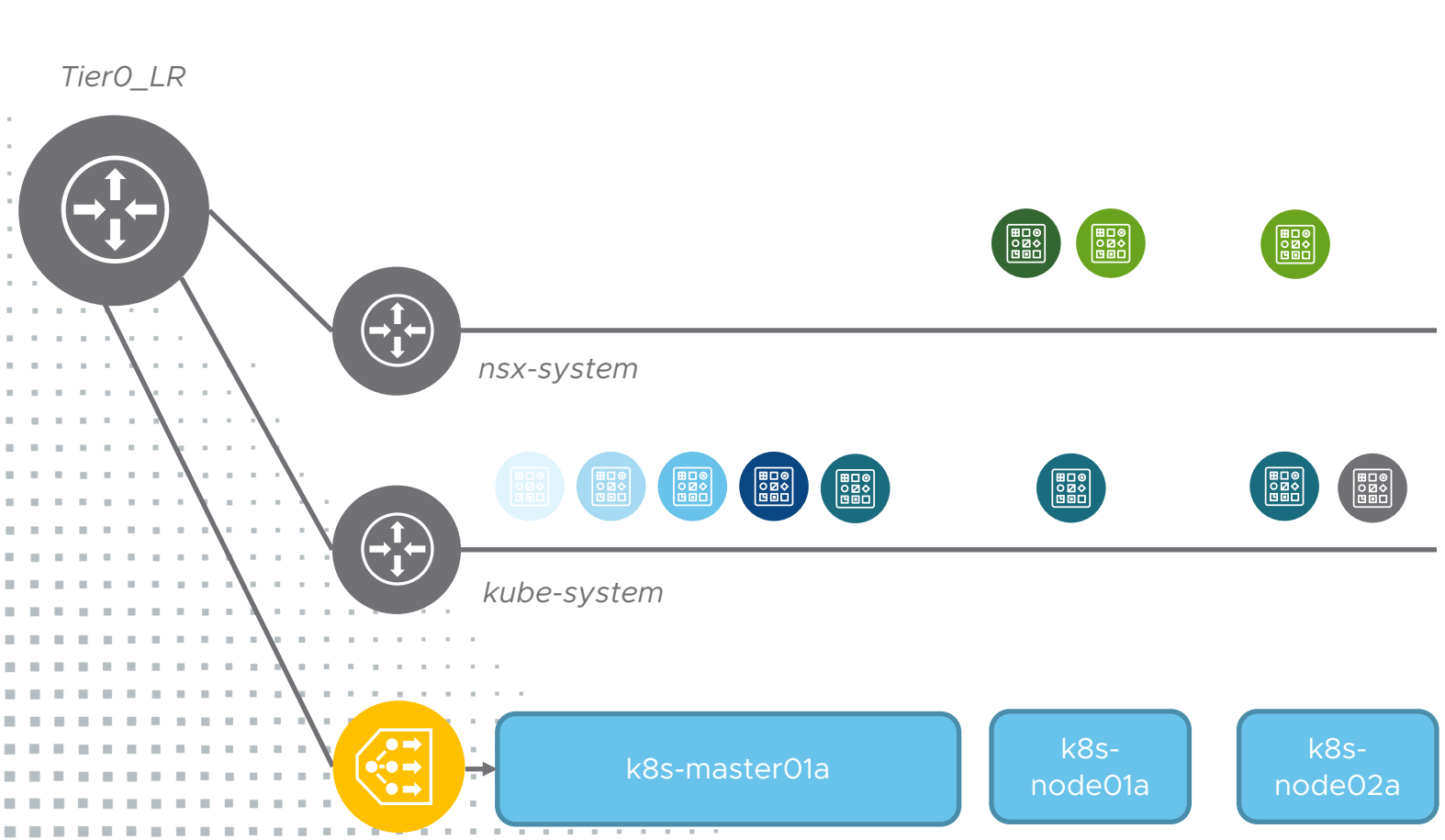
## Network Container Plugin (NCP)




### Network Container Plugin (NCP)

- **NSX Container Plugin:** NCP is a software component provided by VMware in form of a container image, e.g. to be run as a K8s Pod
- **Adapter layer:** NCP is build in a modular way, so that individual adapters can be added for different CaaS and PaaS systems
- **NSX Infra layer**: Implements the logic that creates topologies, attaches logical ports, etc. based on triggers from the Adapter layer
- **NSX API Client**: Implements a standardized interface to the NSX API

# Ground floor topology for NSX-T and Kubernetes integration





vmware

vFORUM

# Planespotter

Putting it all together

vmware

vFORUM

# What is Planespotter?

Planespotter is a free, open-source application, built with a “Cloud-native” mindset

- Built for demonstration of key Kubernetes and NSX resources
- Pods, Replication controllers, StatefulSets, NetworkPolicy, Ingress, Services

Free!

- You can get planespotter
- You can use planespotter
- You can use planespotter on Minikube, GKE, VKE, AKS, Vanilla K8s.

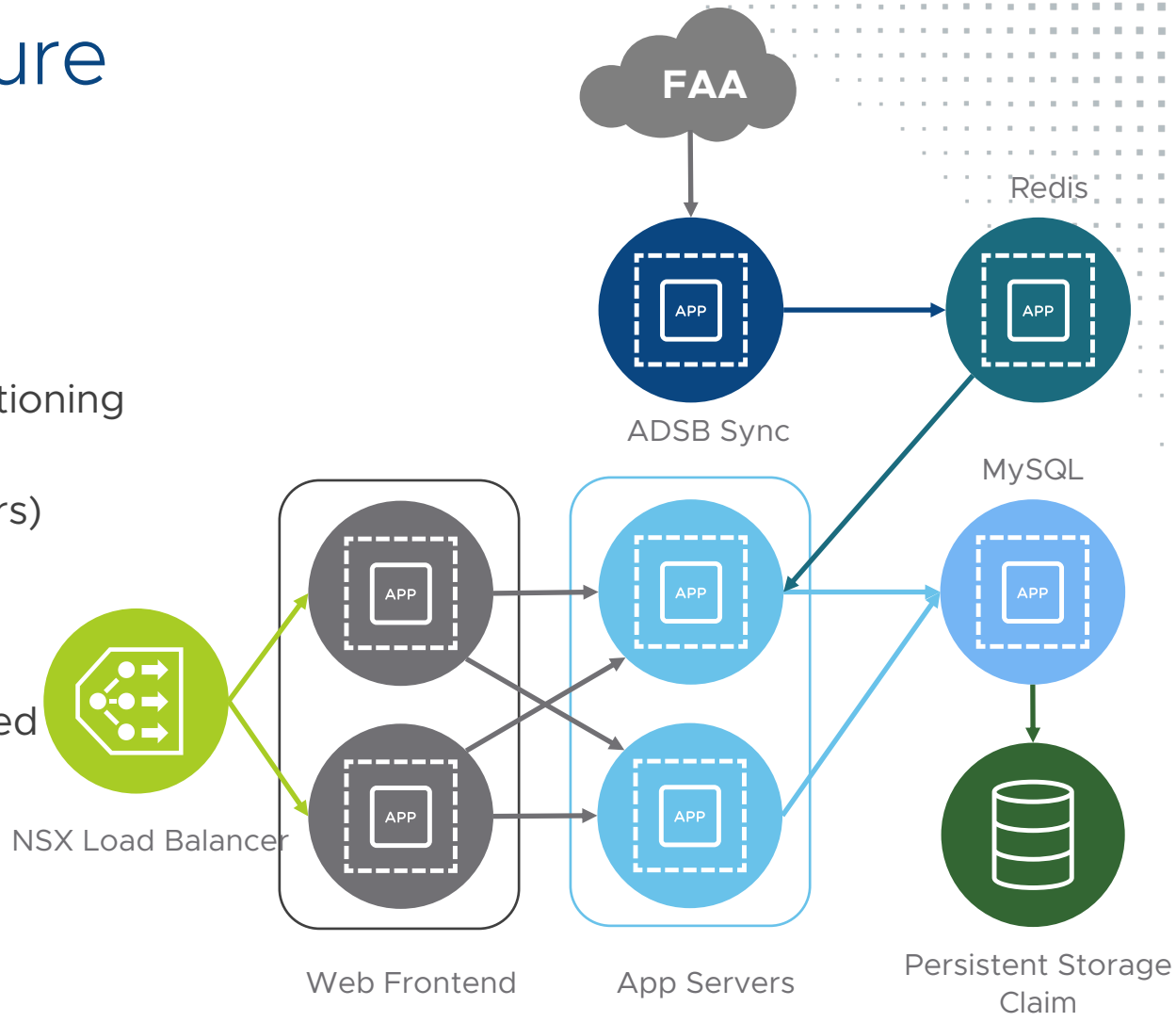
# Planespotter architecture

Defined by 5 YAML files

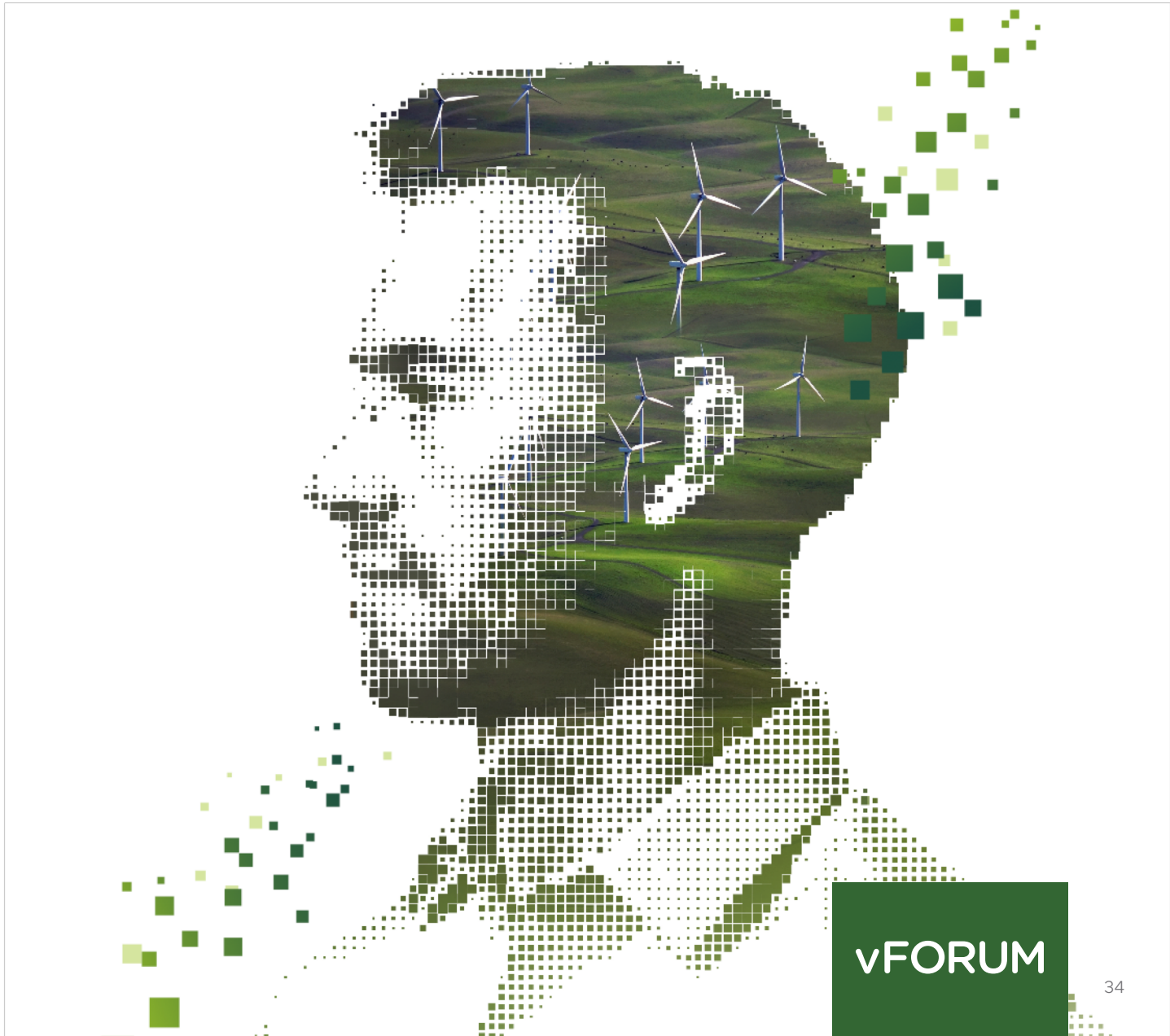
- Mysql – storing static plane information
- Redis - in-memory DB for real time positioning
- Live Position ingestion
- API and Correlation servers (App Servers)
- Web Frontend

Pre-reqs

- Kubernetes/NSX-T integration configured
- Storage class defined or a default set
- Any ops tools configured



# DEMO TIME



vmware

vFORUM

# Putting Planespotter together

## Demo time

The order of operations to create the planespotter app is as follows:

- Create Namespace
- Create a Storage Class
- Create Persistent Volume Claim
- Create a Stateful set
- Create an application deployment
- Create a frontend deployment
- Create a redist deployment
- Apply network policies

Use Kubernetes to validate
Use NSX-T ops tools to validate

Coming up next

# Migration, Connectivity and Security for the Hybrid cloud with NSX Hybrid Connect

1:30PM – 2:30PM

# Thank you

vmware

vFORUM

# POSSIBLE BEGINS WITH YOU

vmware

vFORUM