Kubernetes Fundamentals

Tommy Falgout
Kendall Roden

Introduction

Tommy Falgout
2 years at Microsoft
20+ years as Unix/Linux developer
Cloud Solution Architect

@lastcoolname https://lastcoolnameleft.com
thfalgou@microsoft.com

Kendall Roden

3 years at Microsoft
Premier Developer Consultant

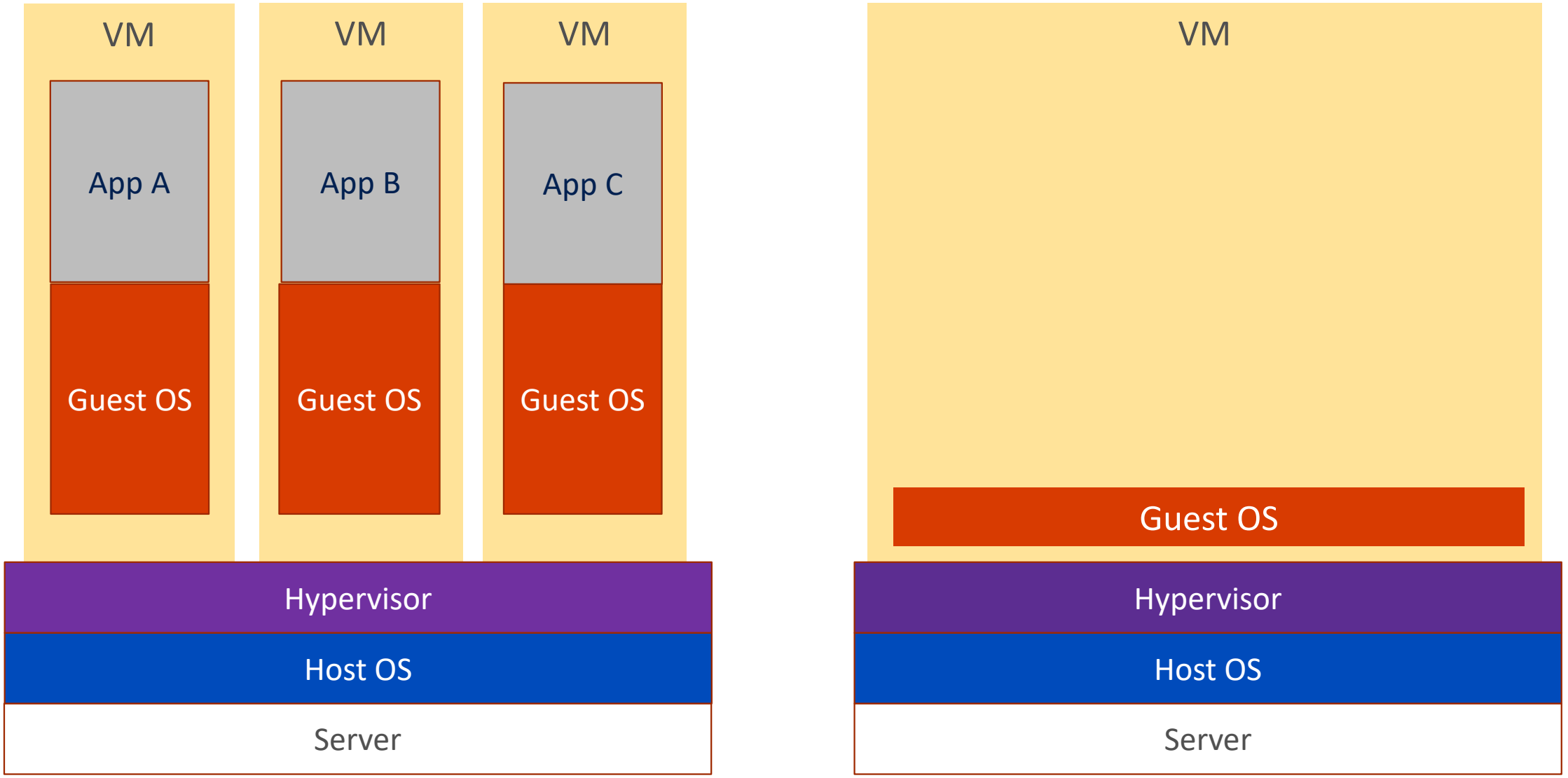
@KendallRoden
keroden@microsoft.com

Session learning objectives

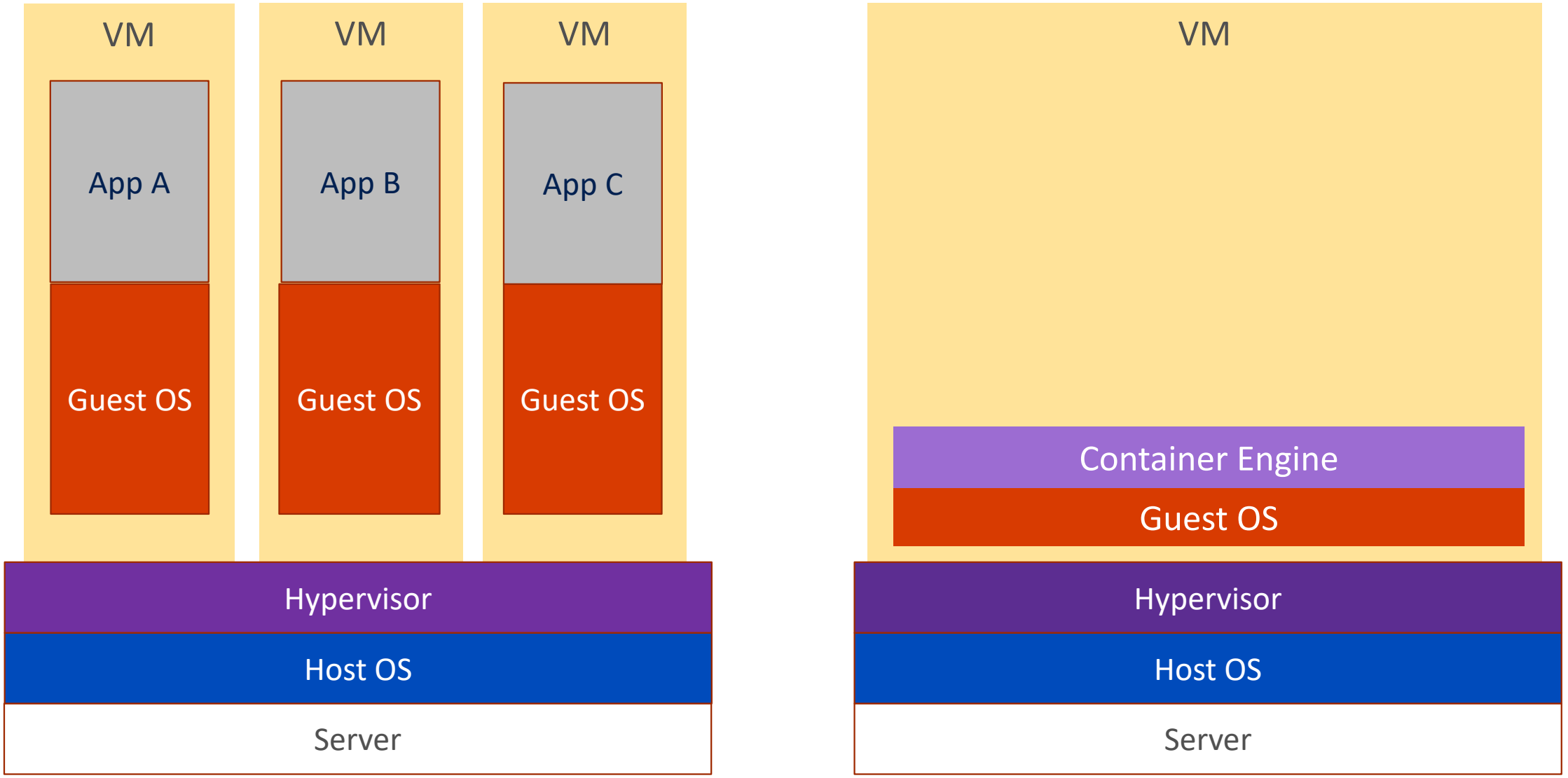
At the end of this session, you should be better able to…

- Explain how Kubernetes works and how it can improve the scalability and maintainability of production deployments
- Understand how managed Kubernetes offerings such as AKS can simplify the implementation of Kubernetes

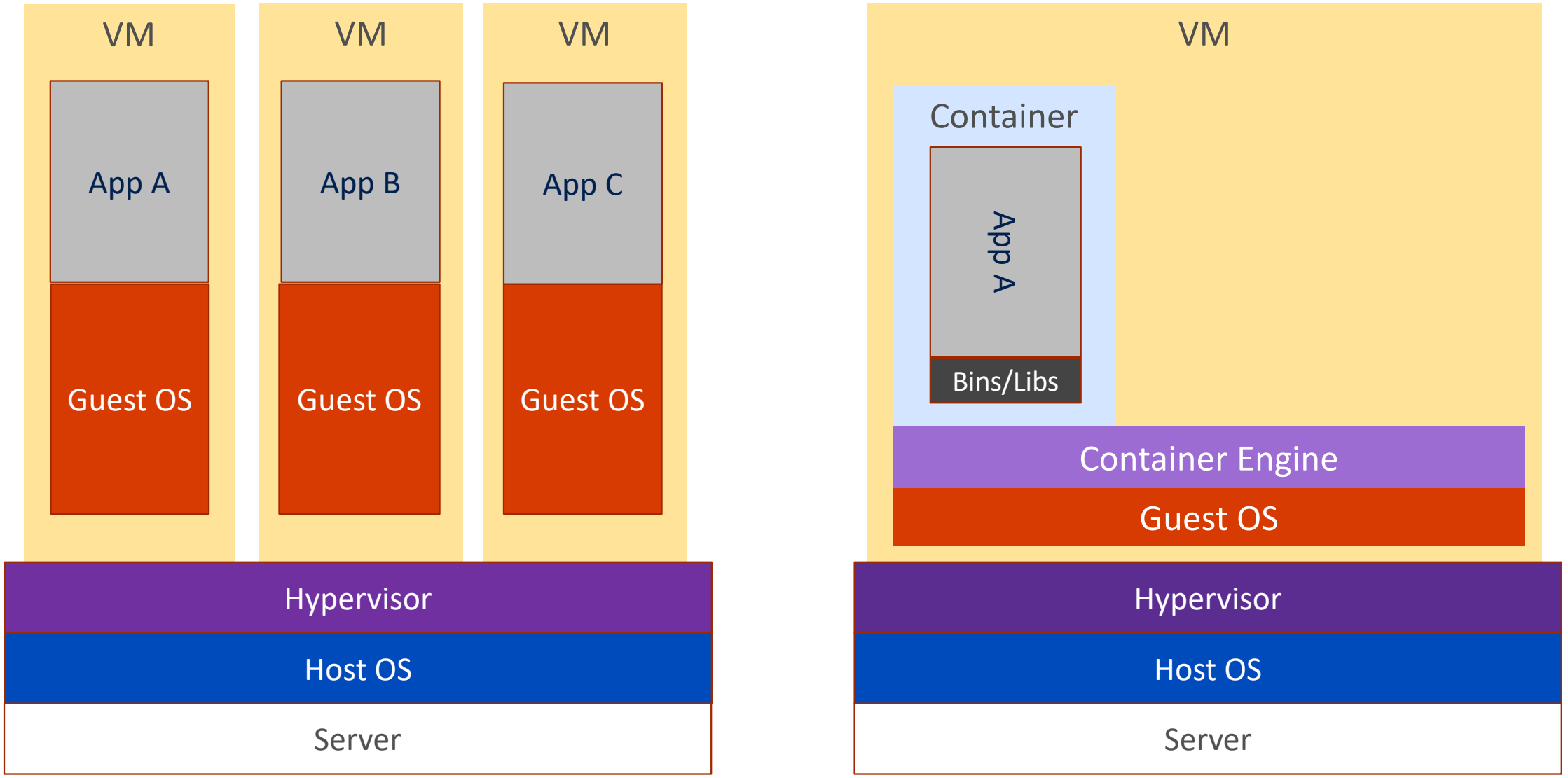
What is a Container?



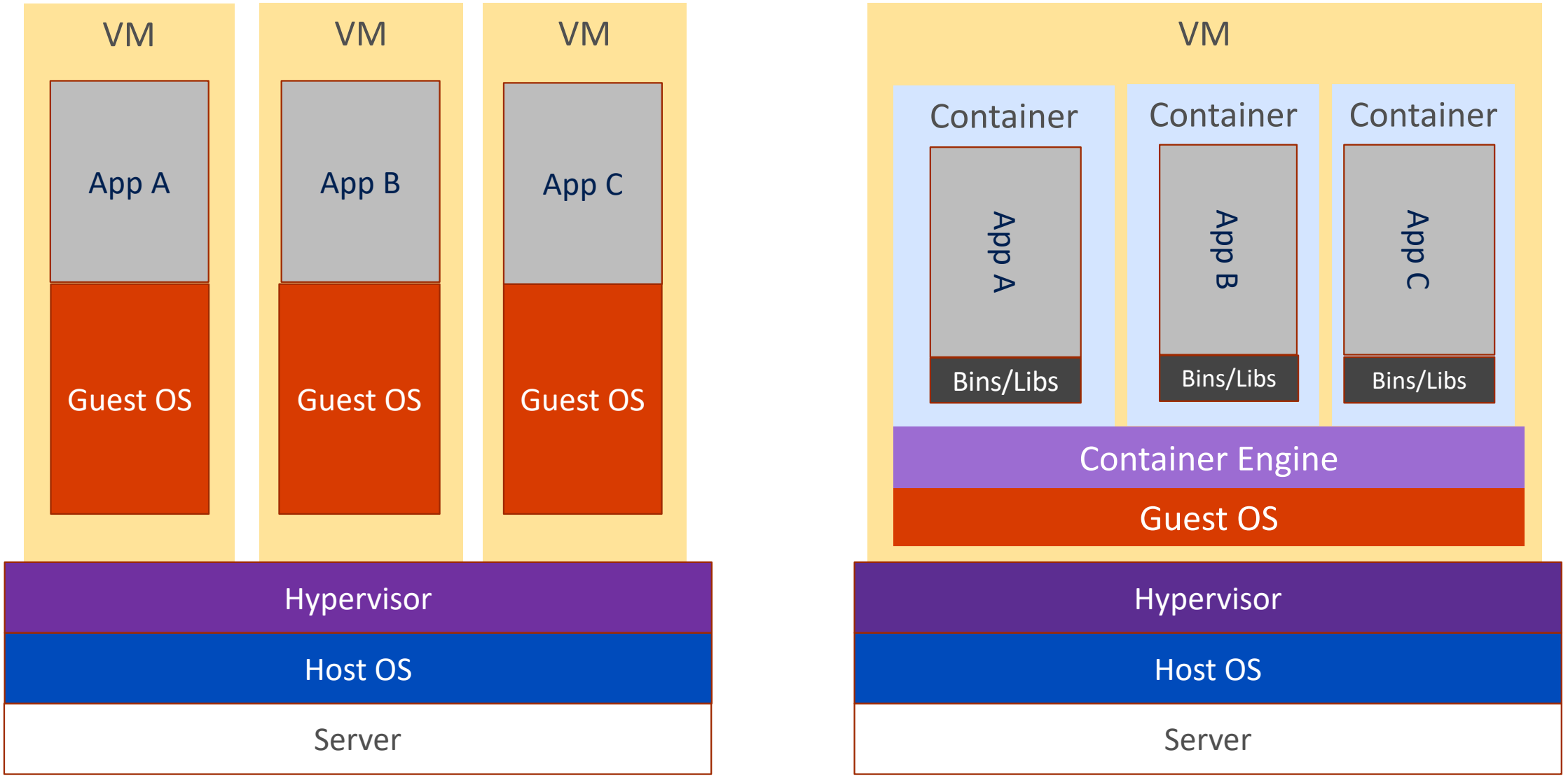
What is a Container?



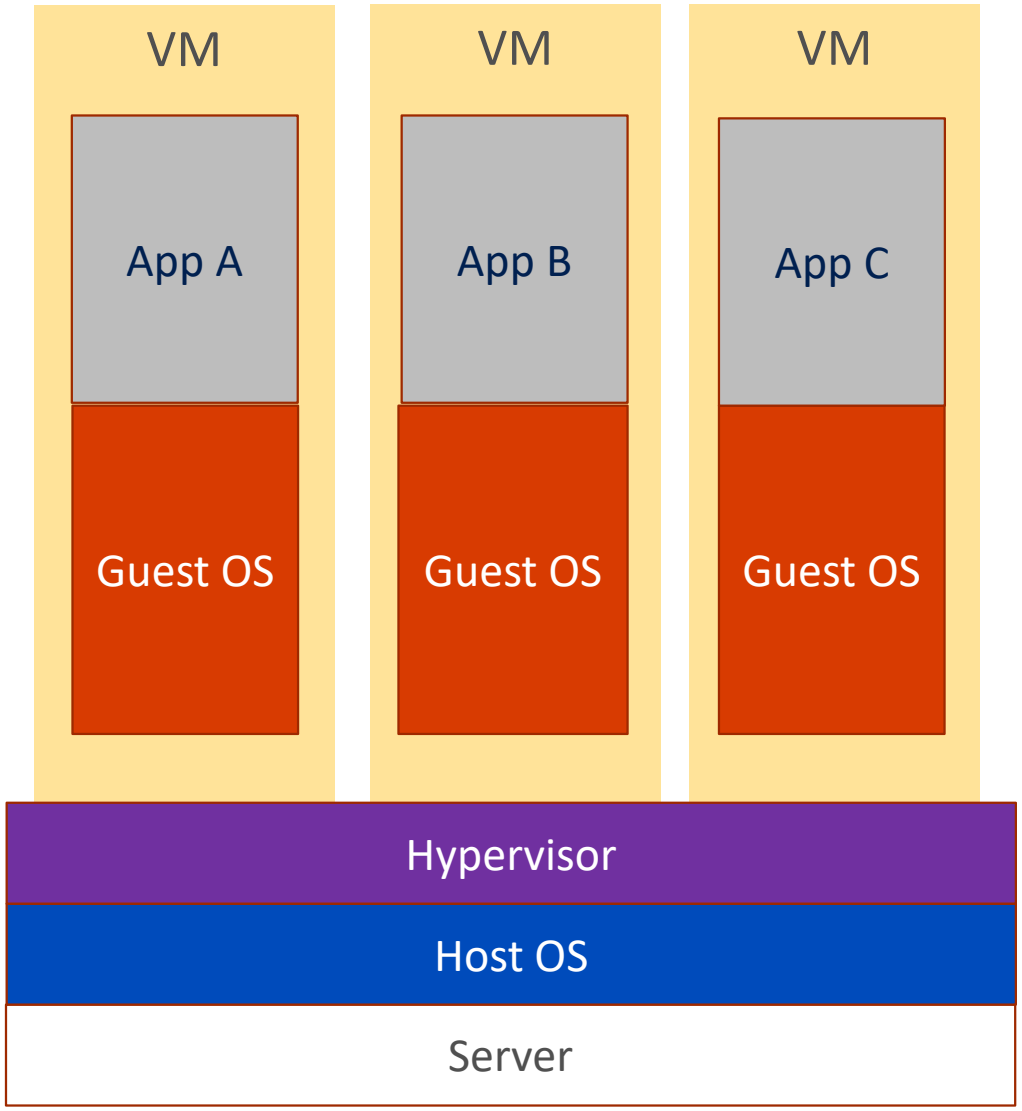
What is a Container?

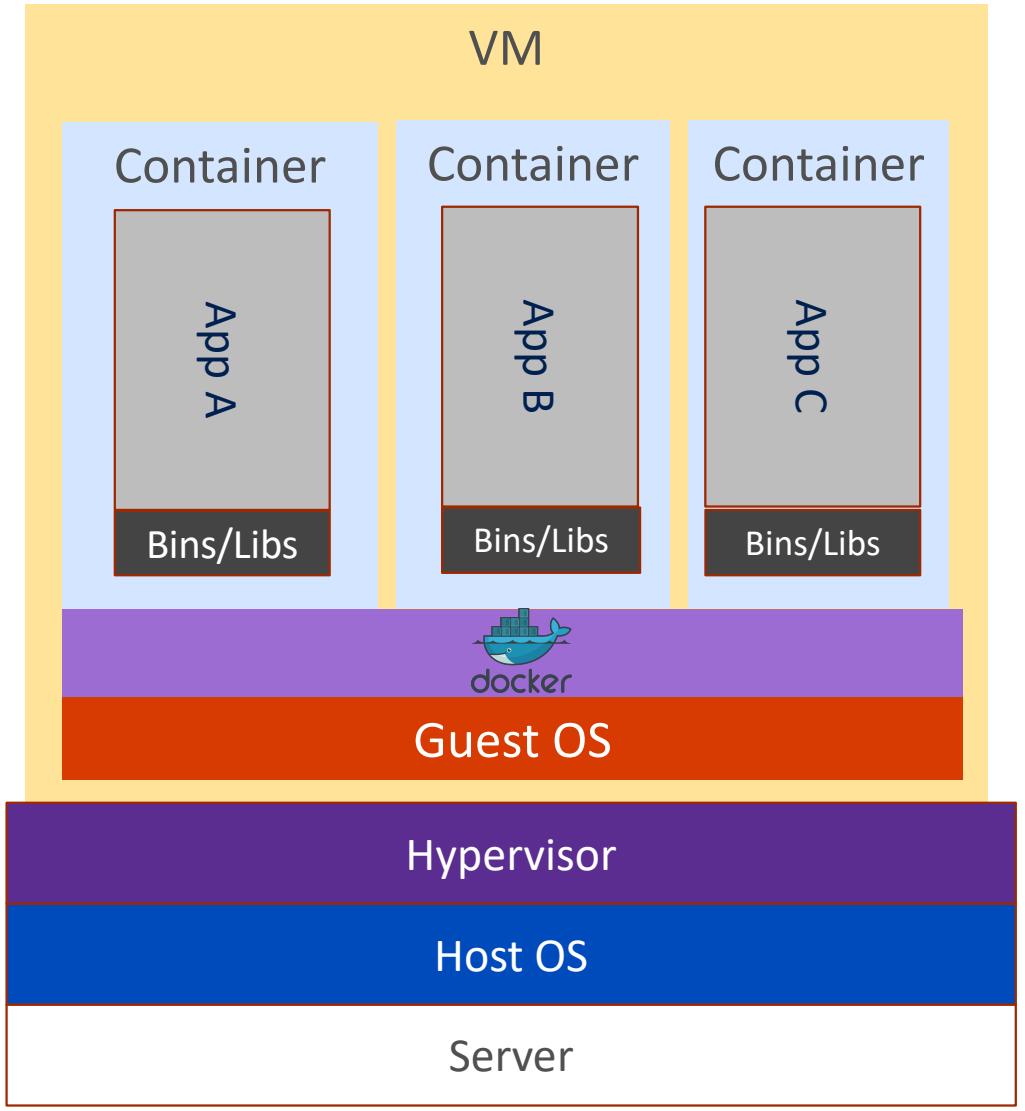


What is a Container?



What is a Container?





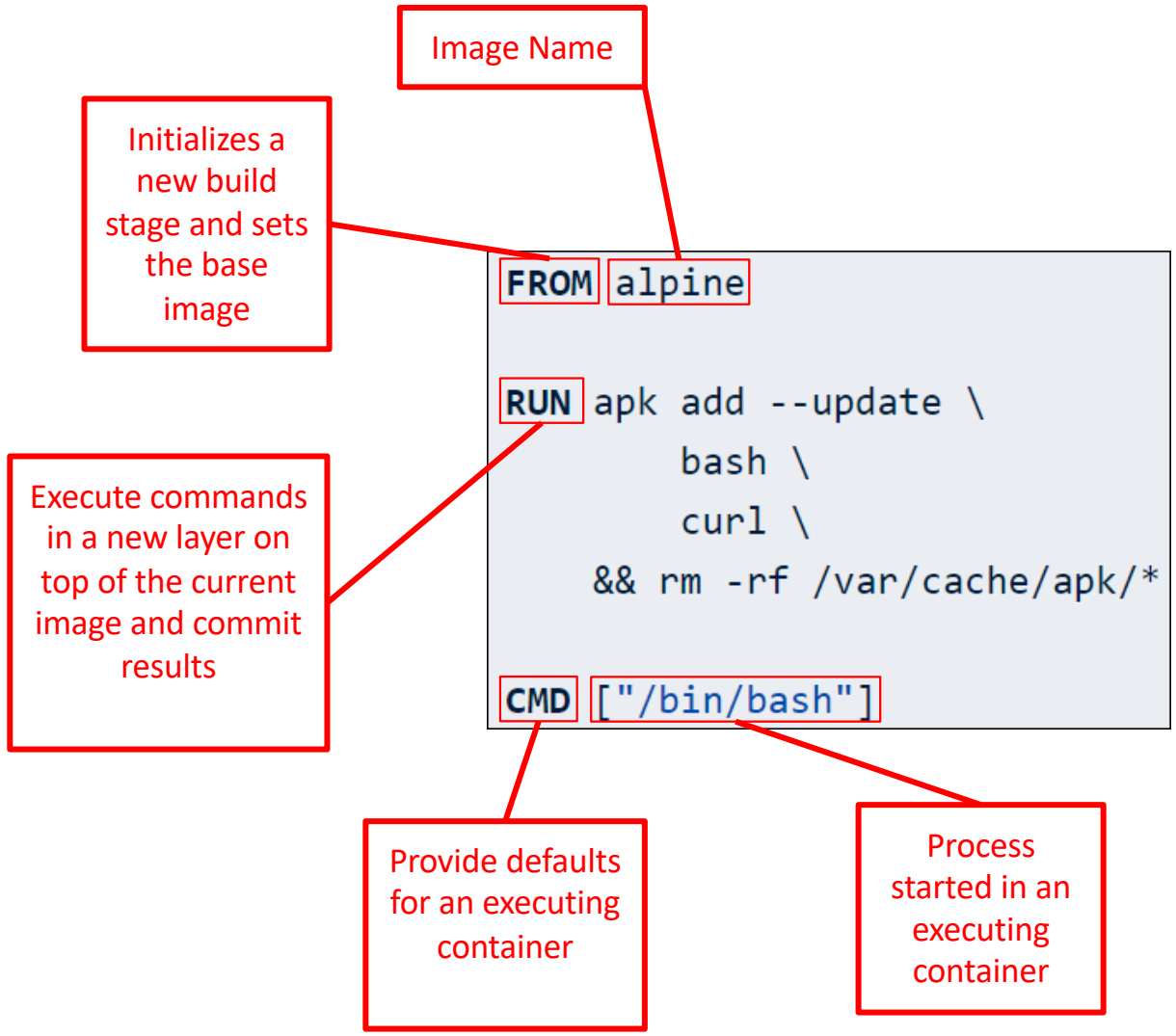
How do we create containers?

We start with the **Dockerfile**

Captures the steps to build a container image

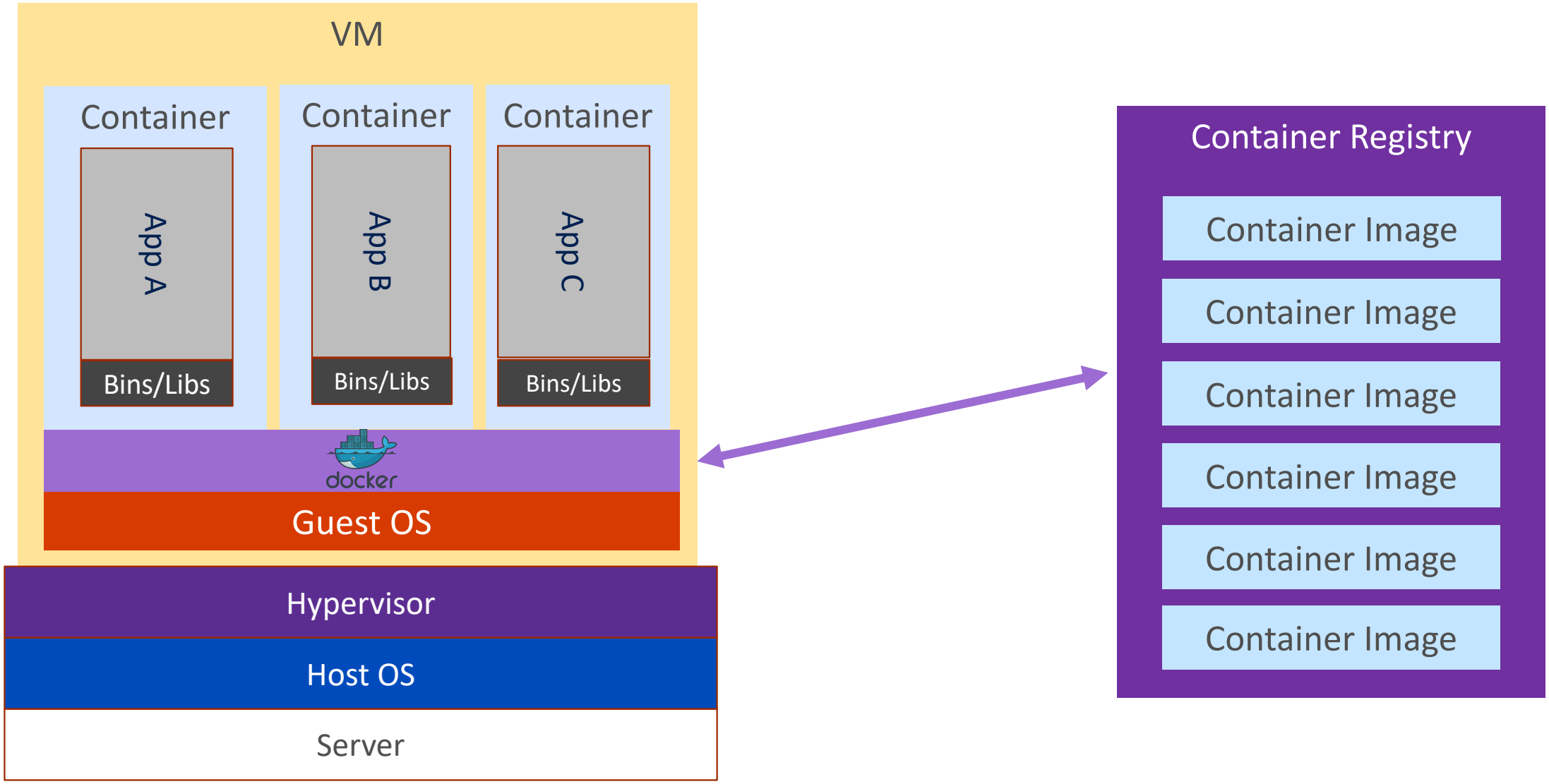
Version-able asset in your DevOps flows

Configuration-as-Code

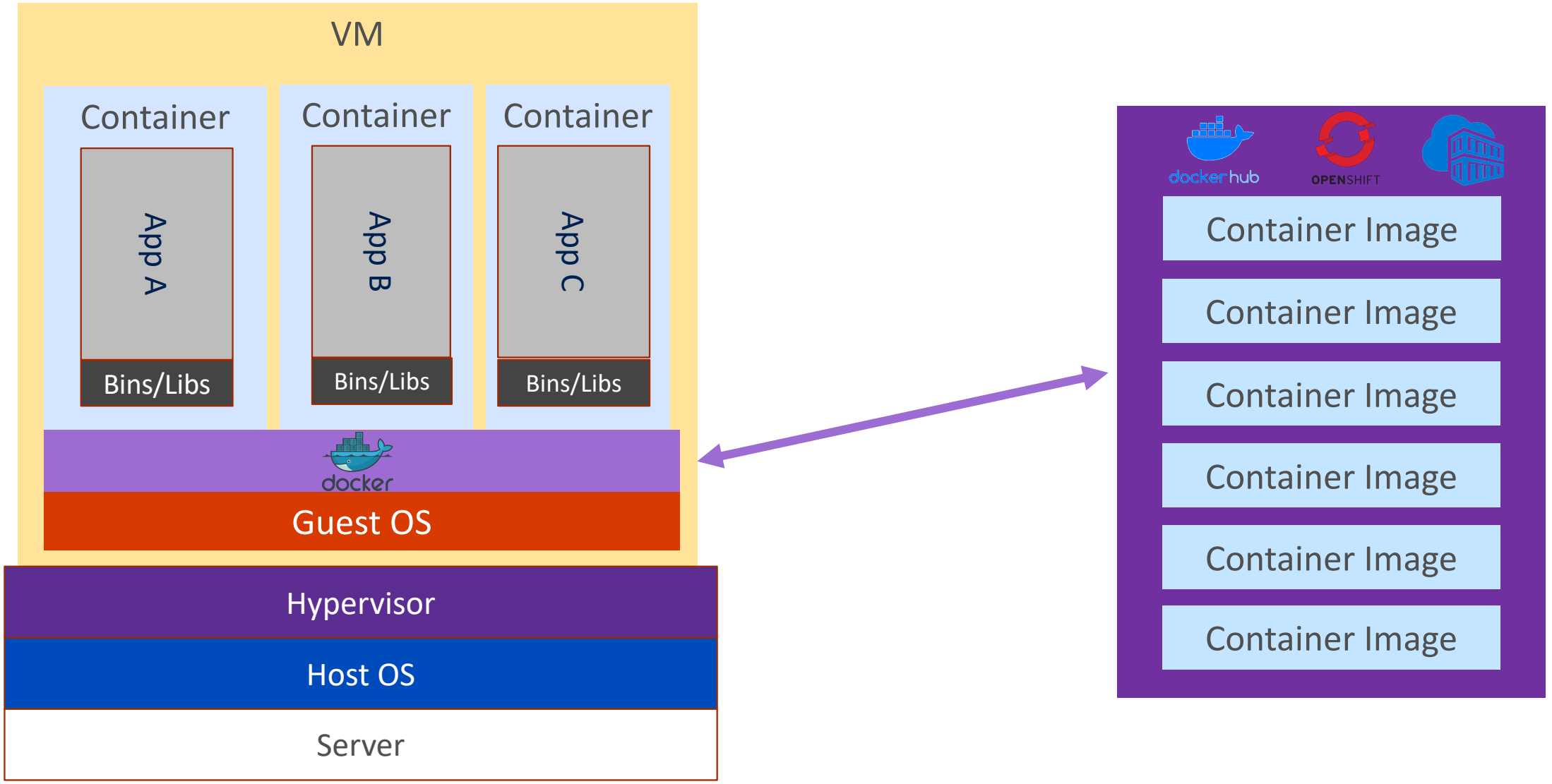


Now we need a container registry…

What is a Container Registry?



What is a Container Registry?

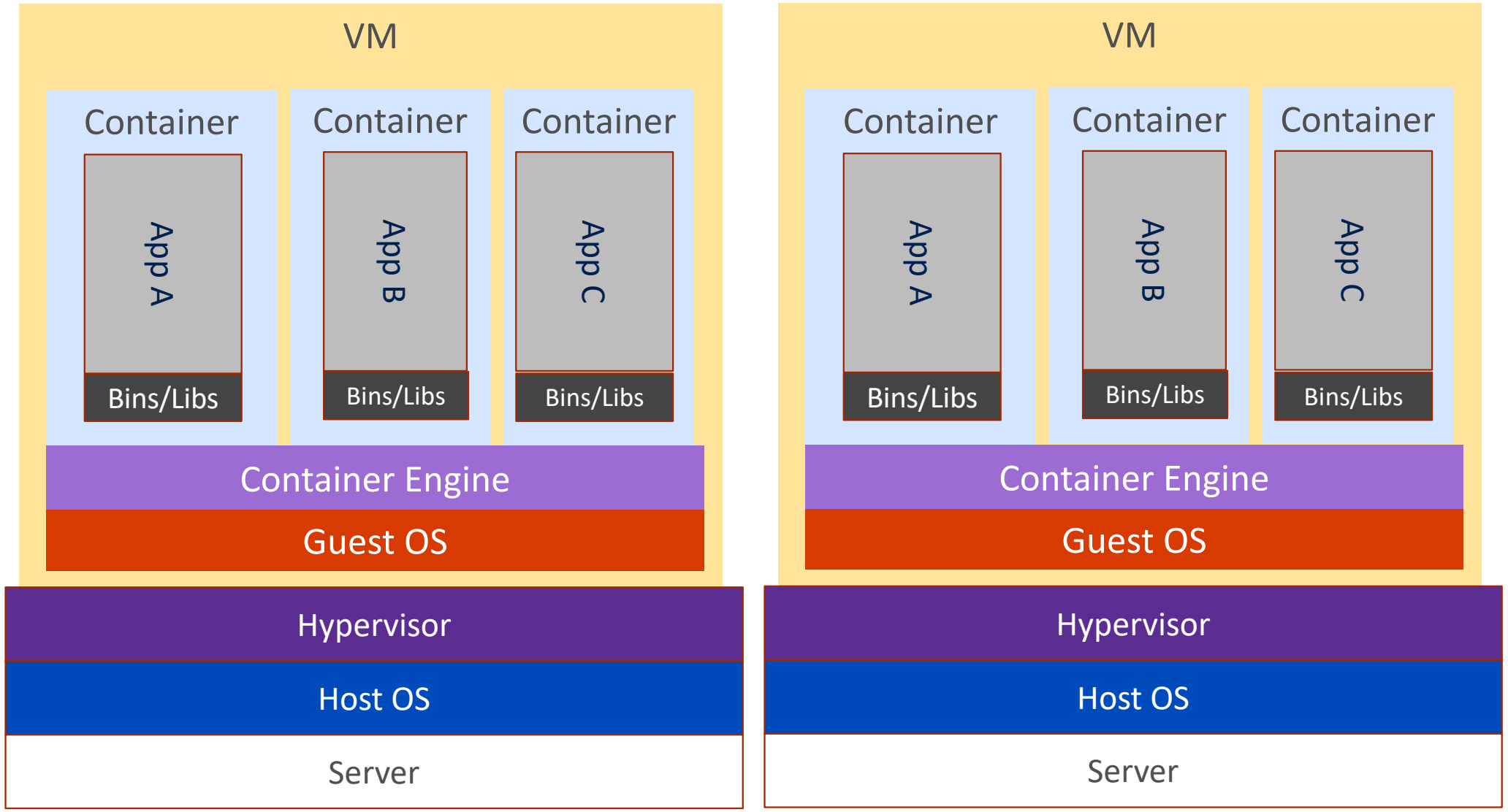


Ok, we’ve got containers.

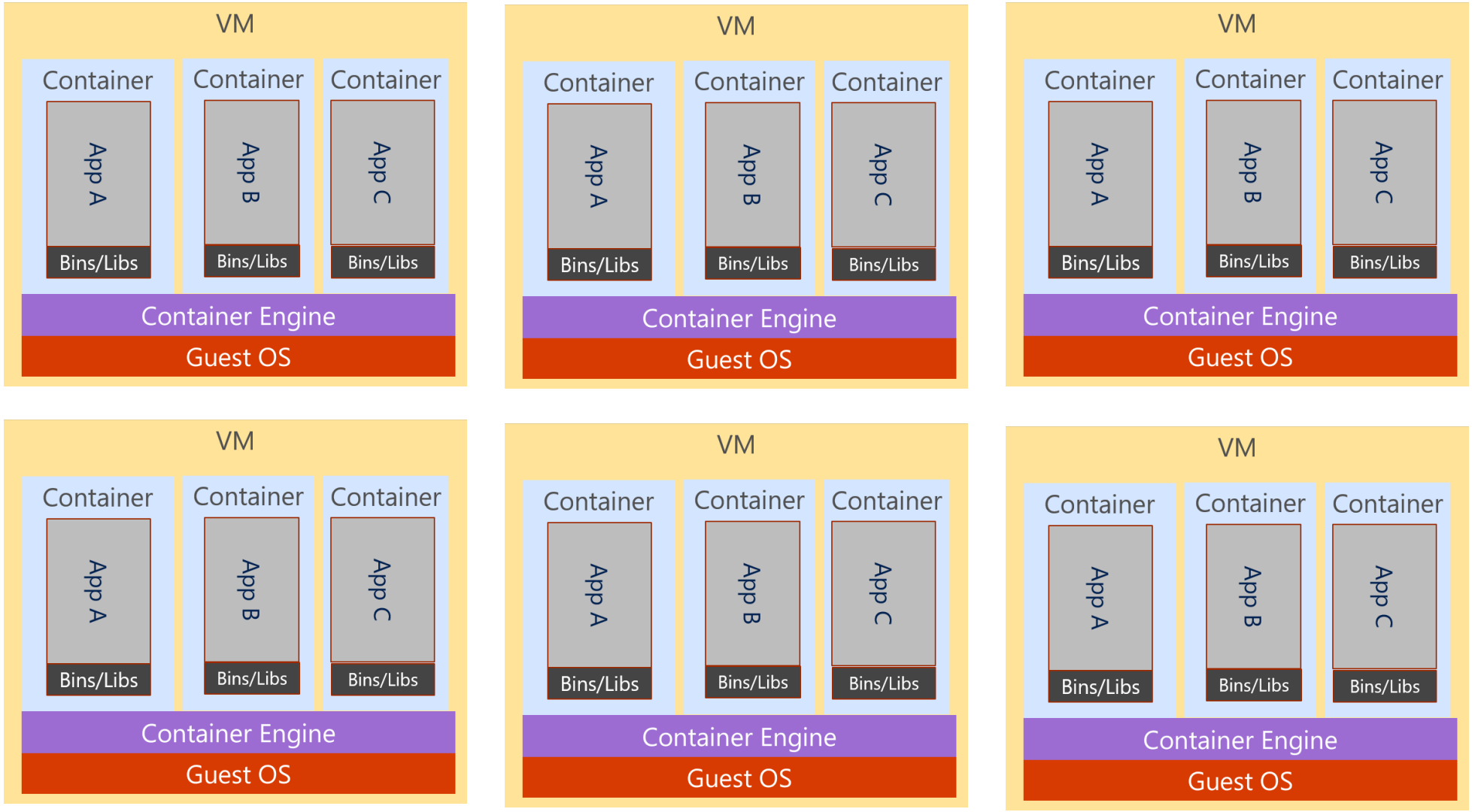
And a registry.

Now what?

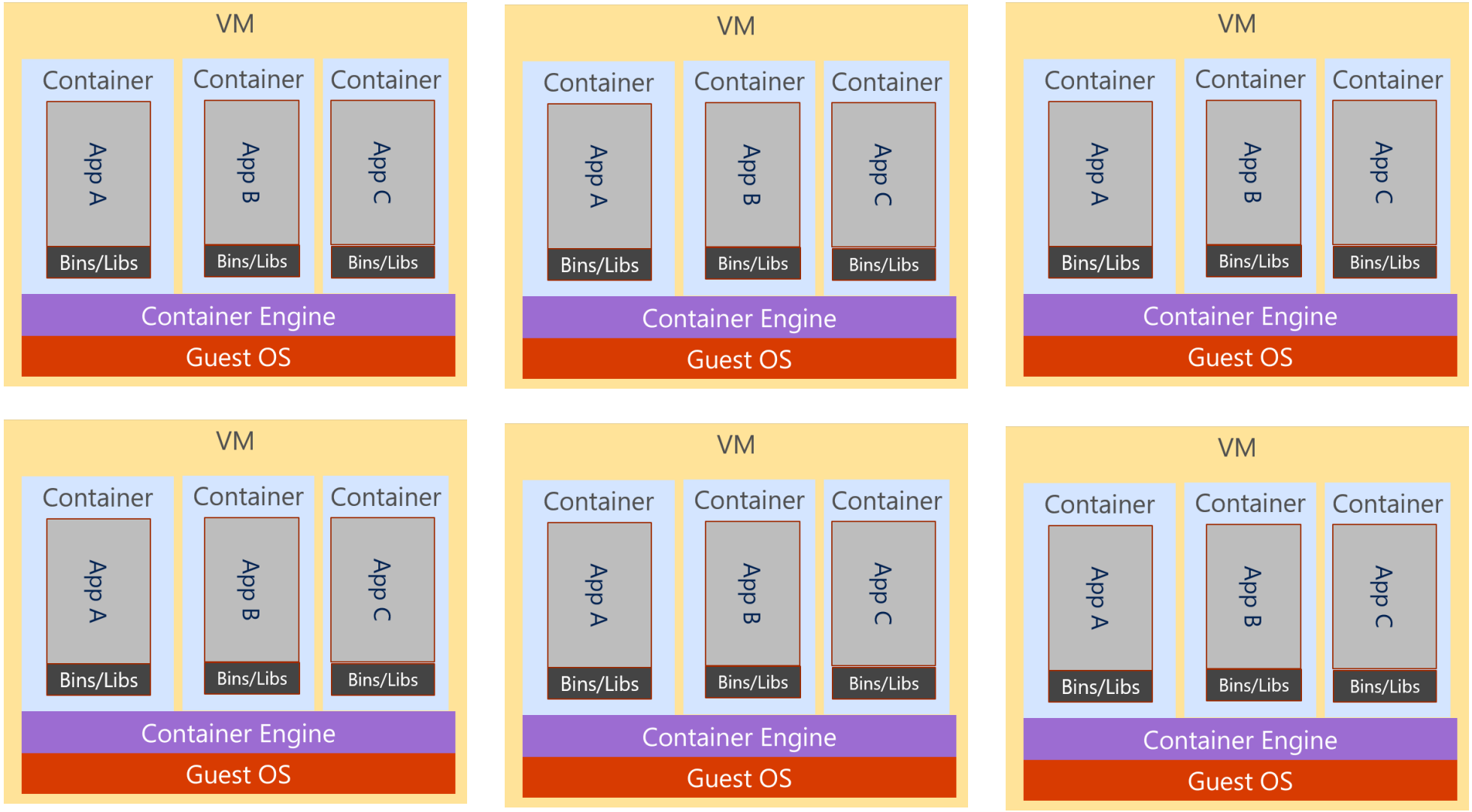
We are going to want more VMs with containers!



And more… and more!

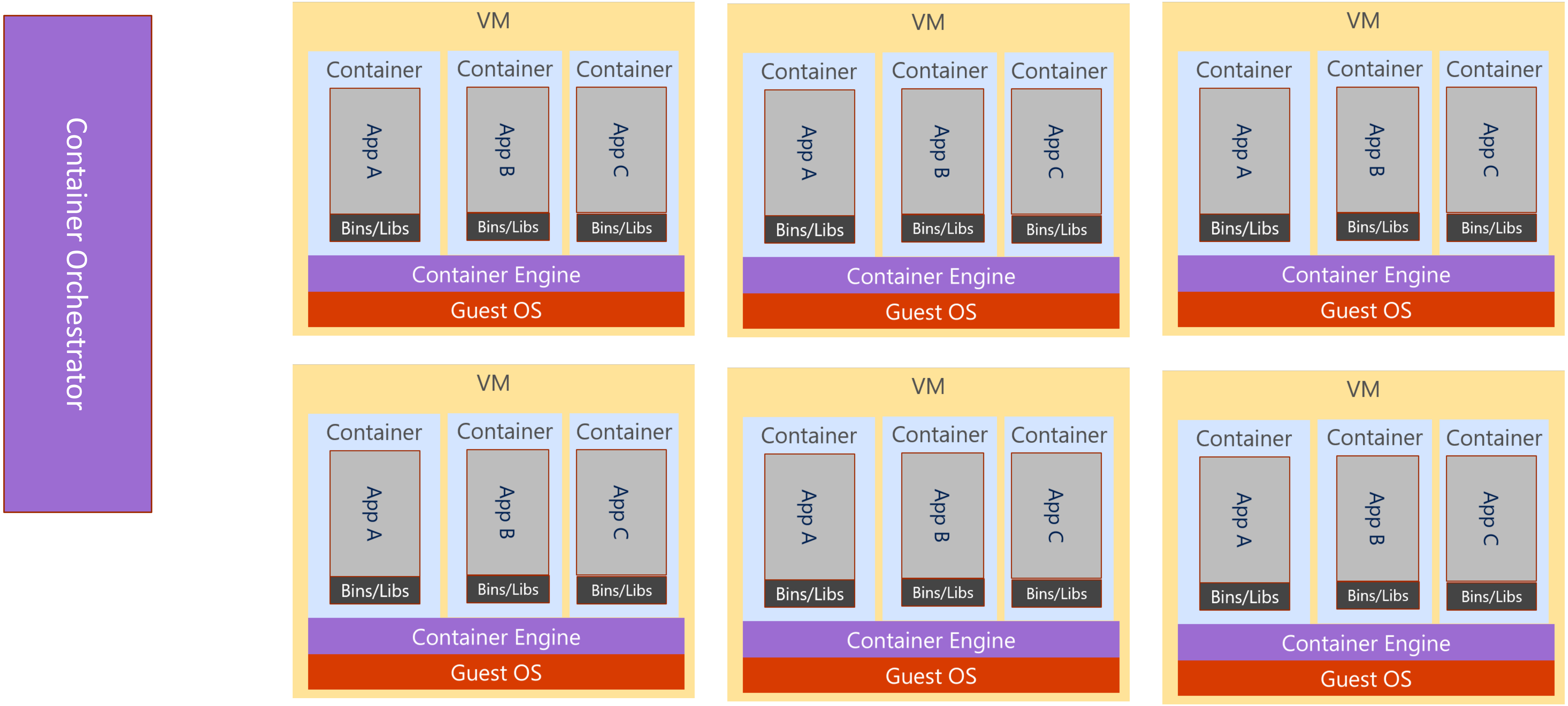


And more… and more… CHAOS!!!

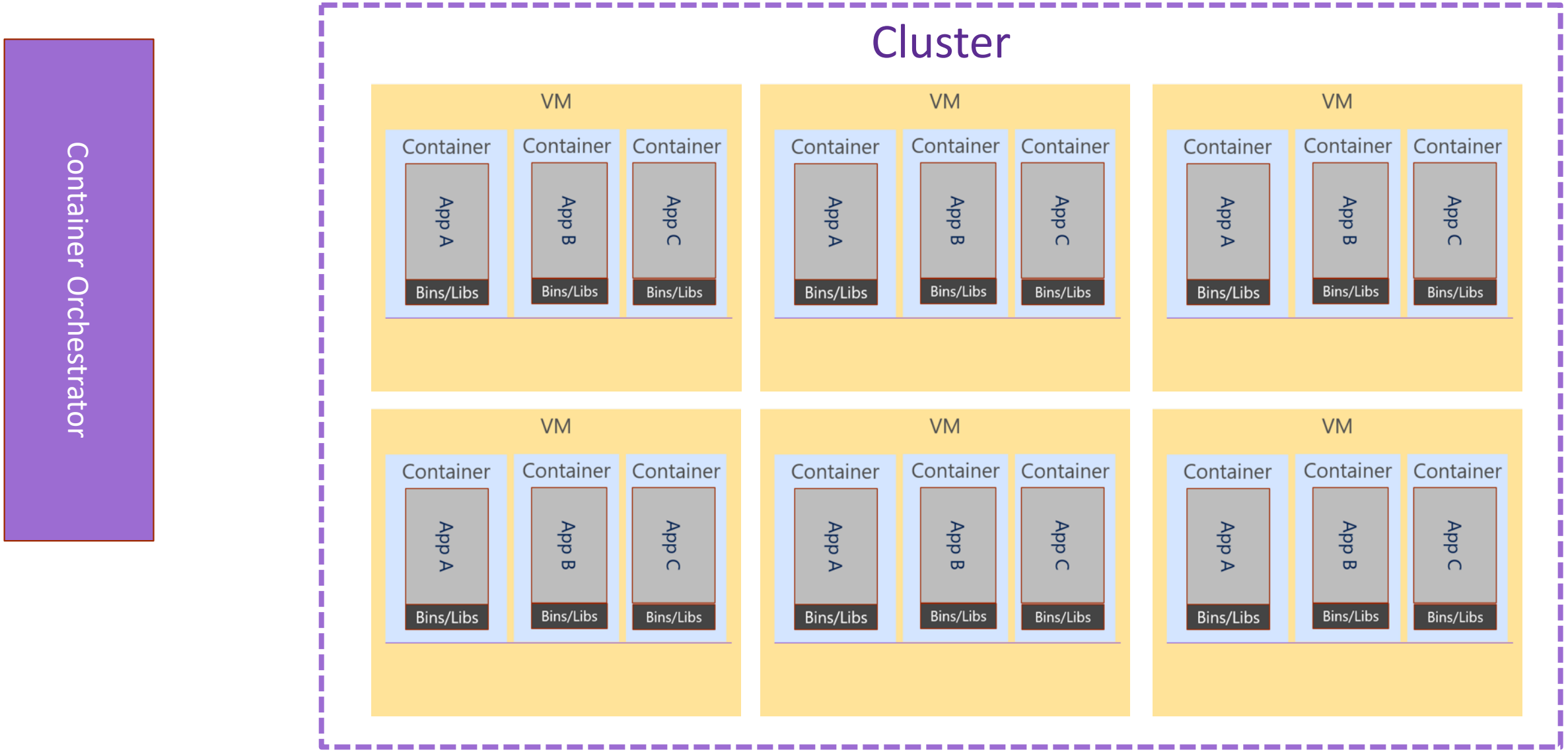


Introducing the container orchestrator!

What is a Container Orchestrator?



What is a Container Orchestrator?



What is a Container Orchestrator?

Container Orchestrator Leaders

What is Kubernetes?

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

Kubernetes comes from the Greek word **κυβερνήτης:**, which means *helmsman* or *ship pilot*, ie: the captainer of a container ship.

Kubernetes has a Master & Cluster of “Nodes”

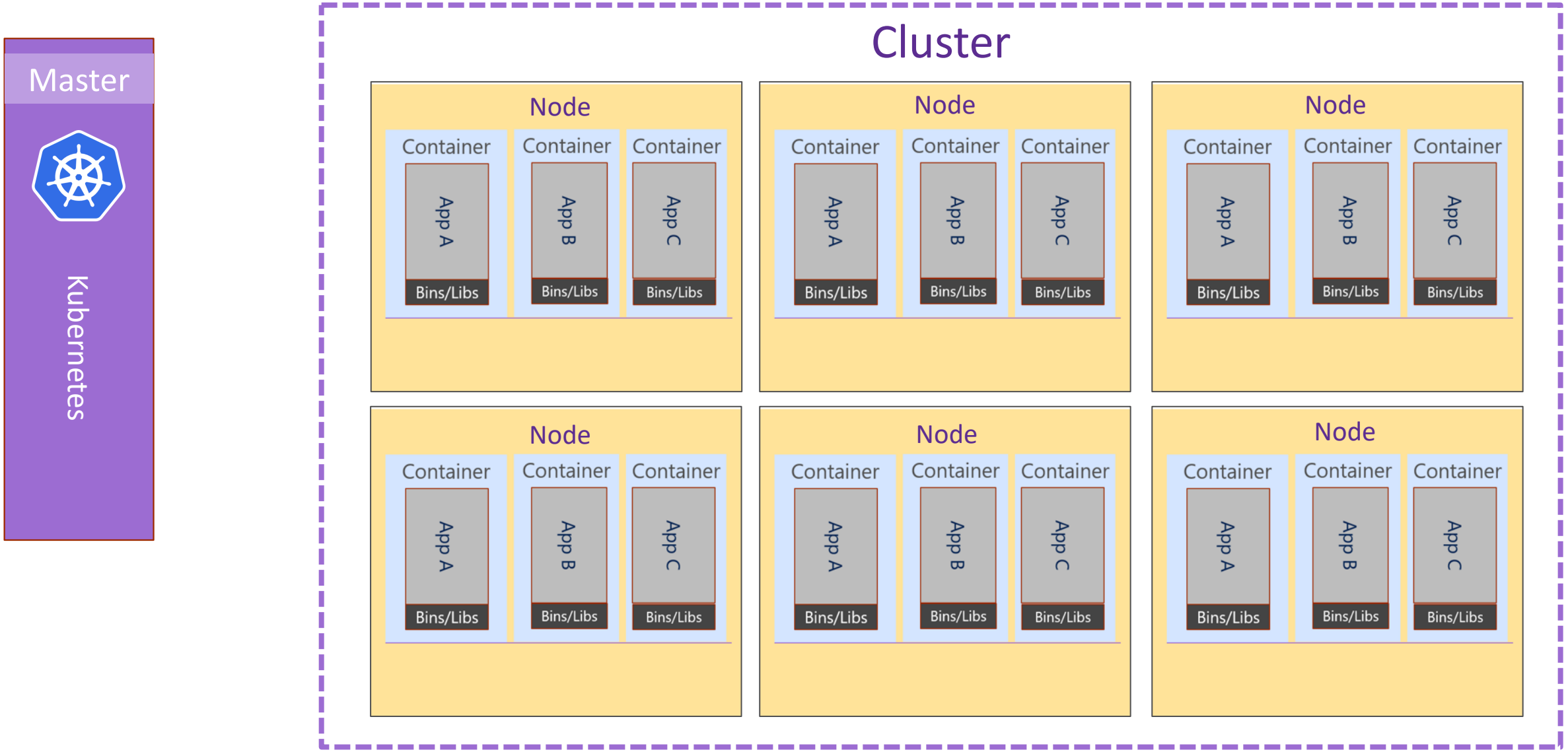
Azure Kubernetes Service (AKS)

AKS reduces the complexity and operational overhead of managing Kubernetes by offloading much of that responsibility to Azure.

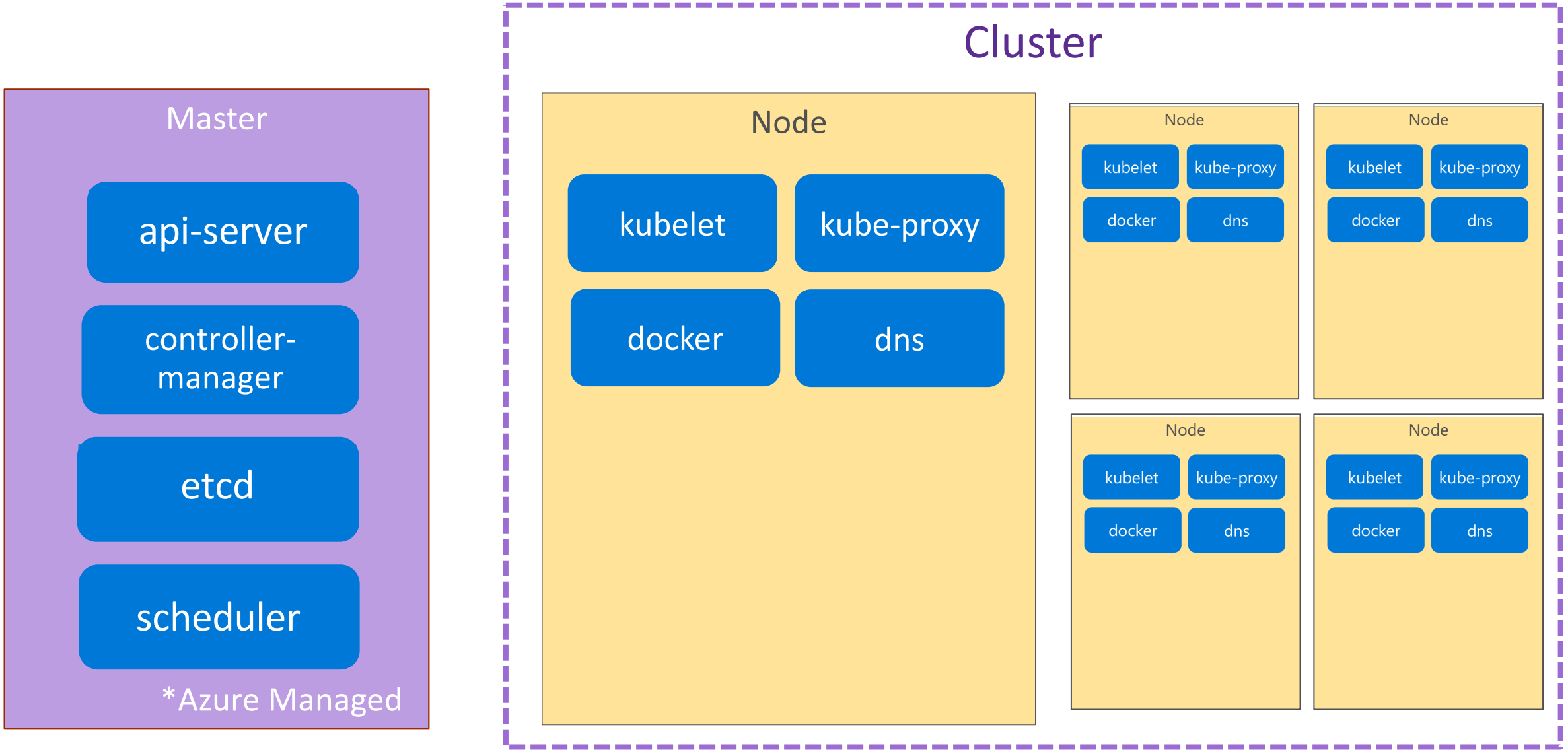
You only pay for the worker nodes within your clusters, not for the master nodes

Kubernetes at 10,000 ft

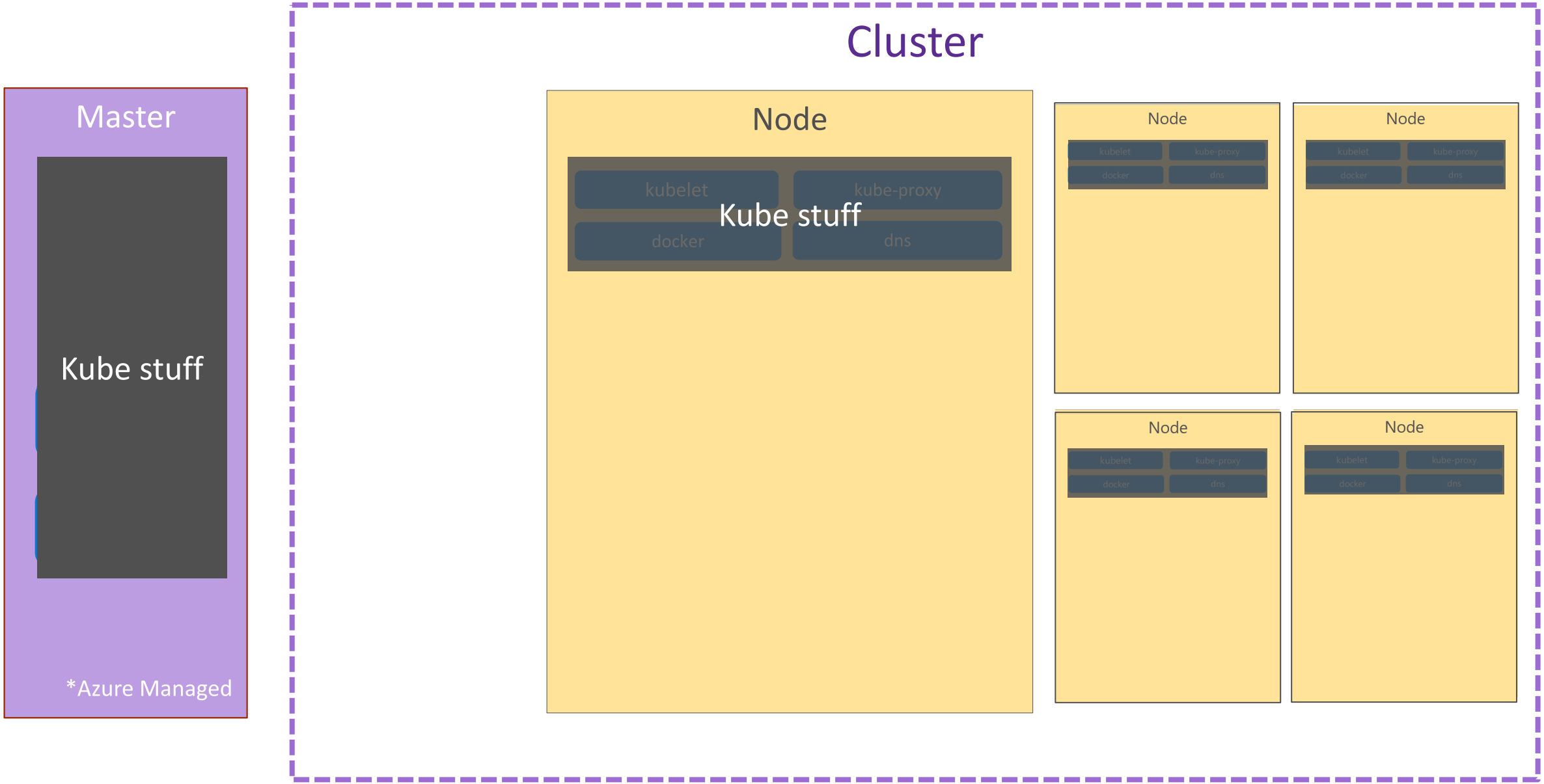
Kubernetes @ 10,000ft



Kubernetes Architecture Components

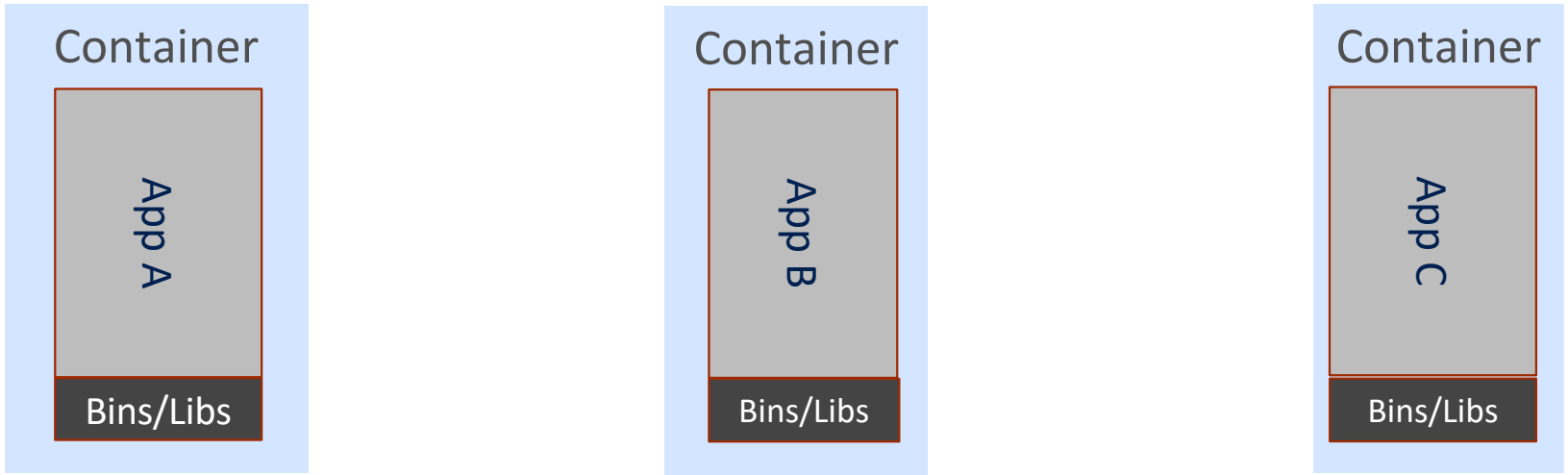


Kubernetes Architecture Components



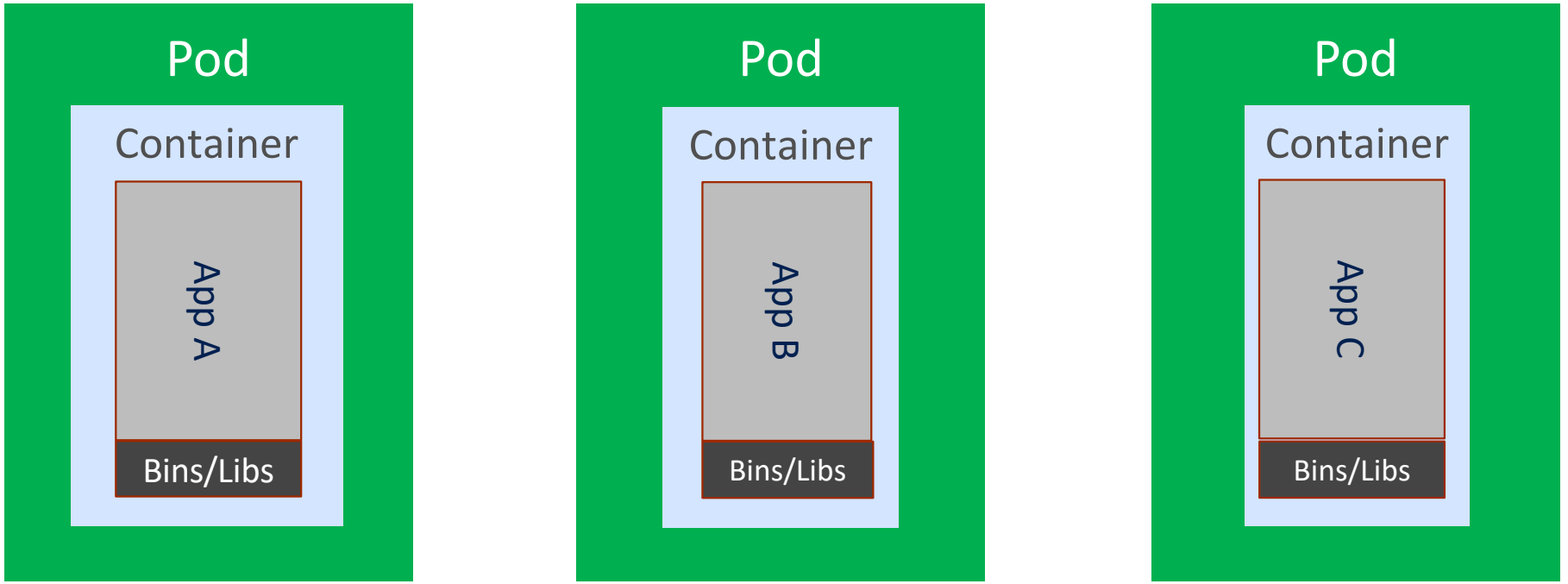
Kubernetes Architecture Components

Where do the Containers go?



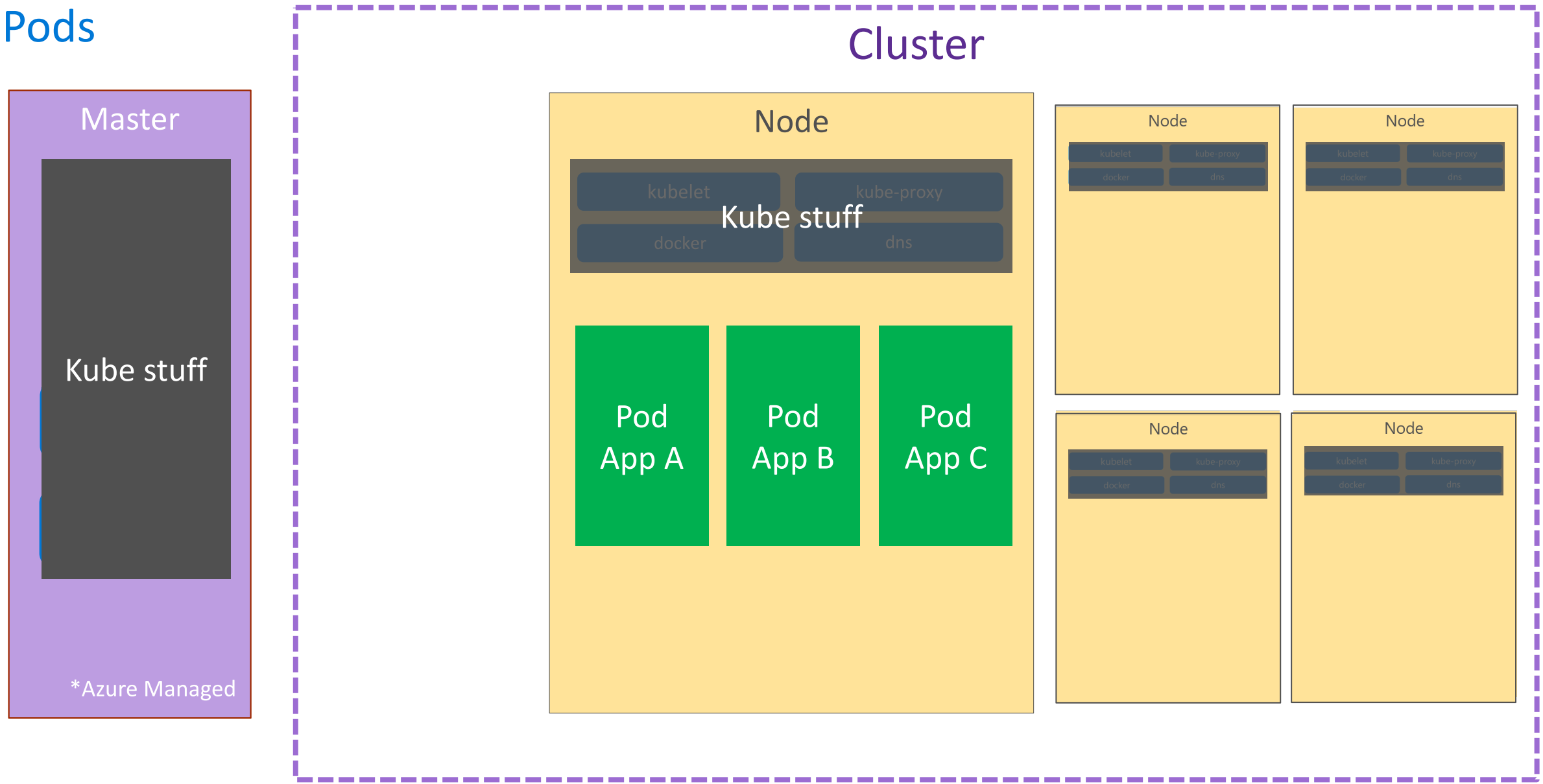
Kubernetes Architecture Components

Introducing…. Pods!

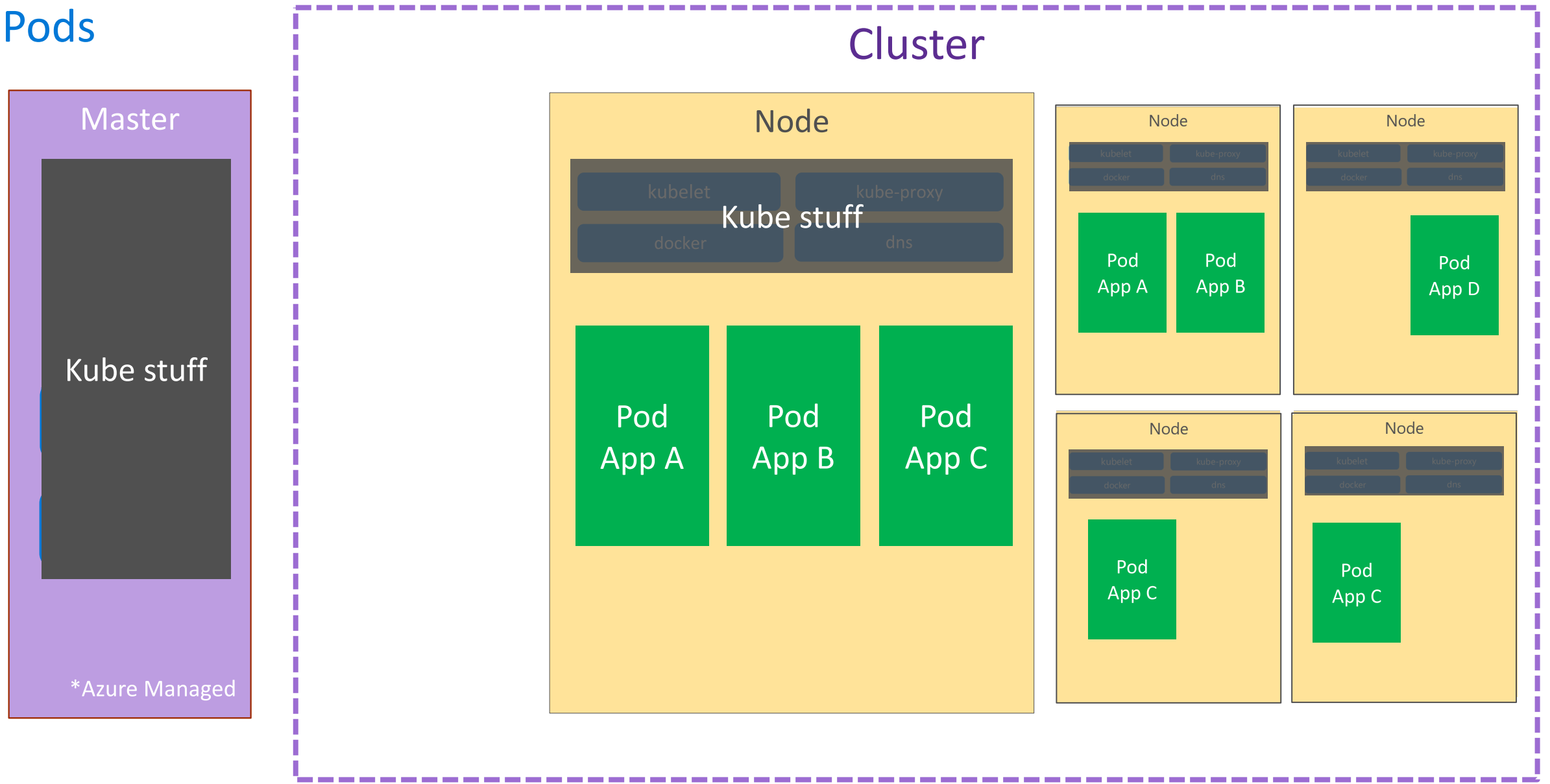


Containers go inside Pods!

Kubernetes Architecture Components

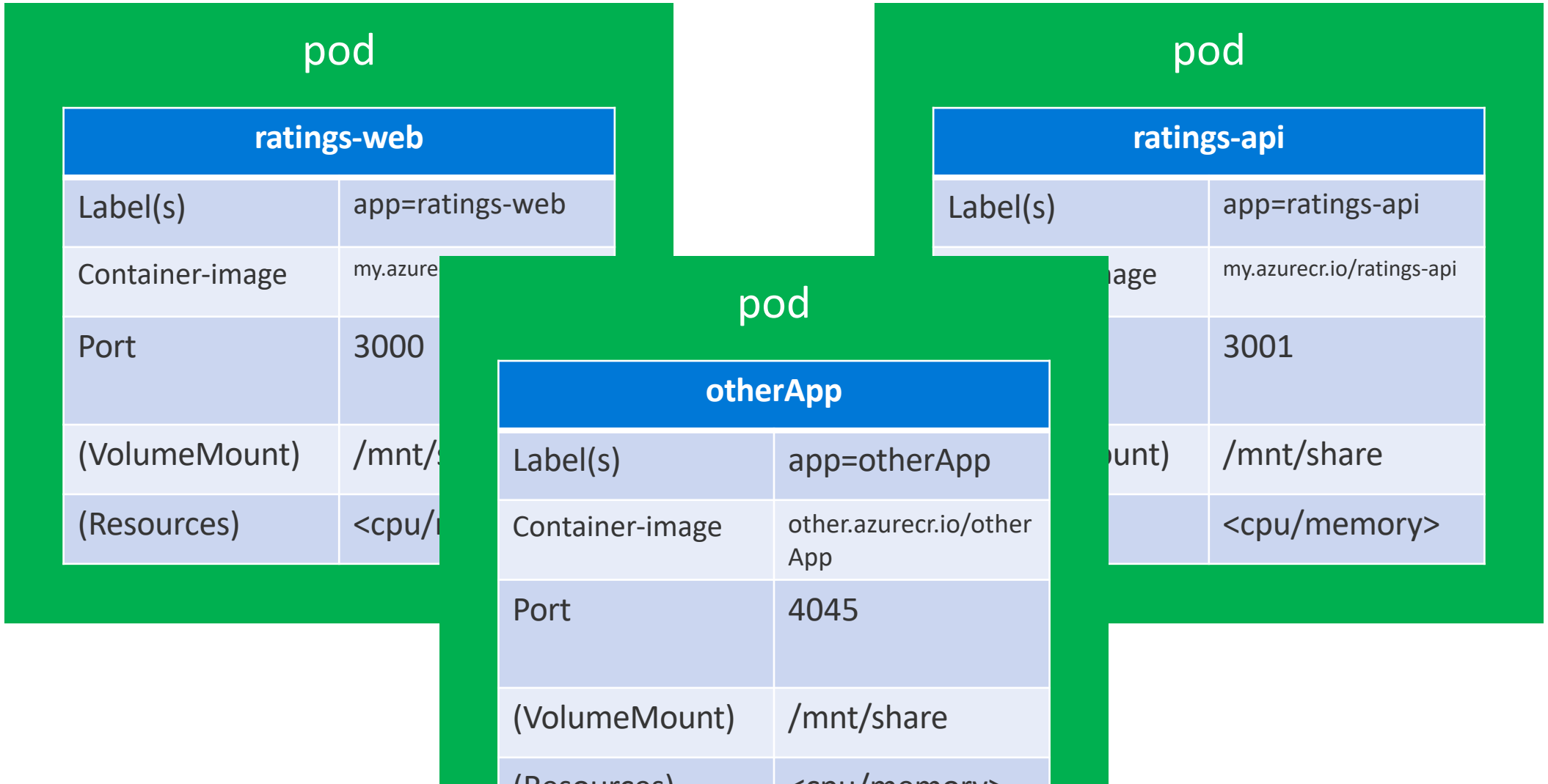


Kubernetes Architecture Components



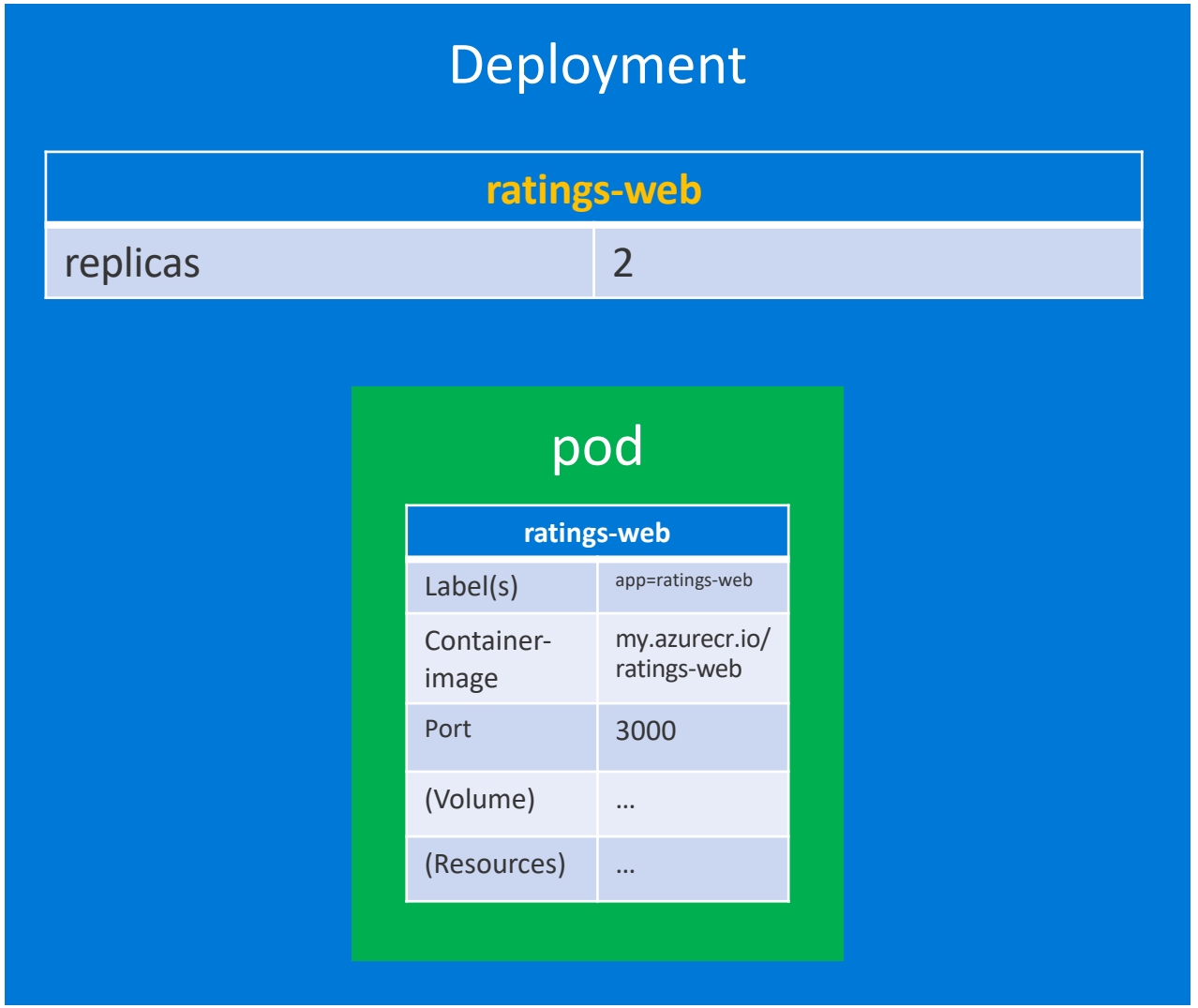
Pods – What are they?

Metadata that describes how the containers are configured.

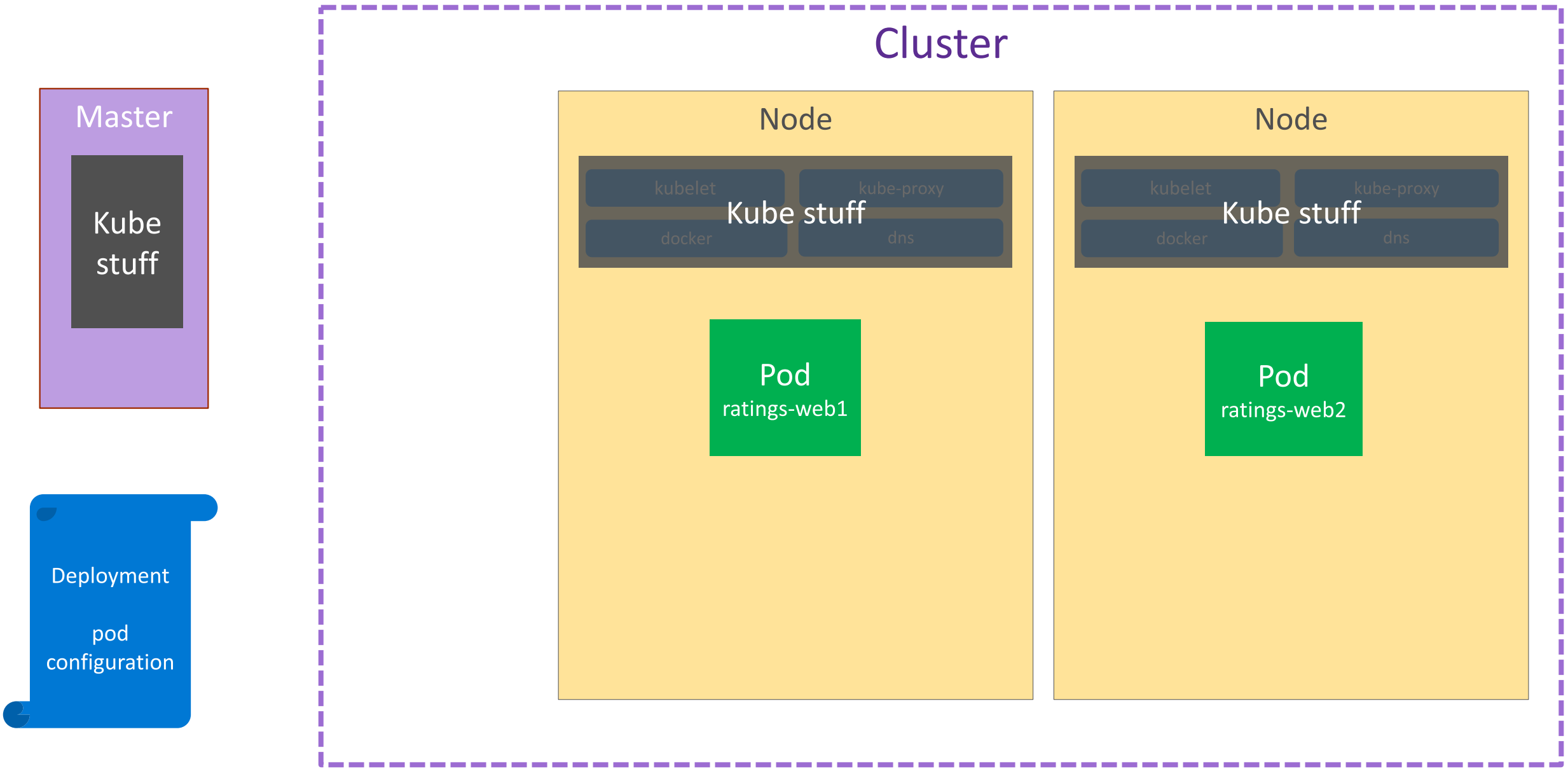


Deployments

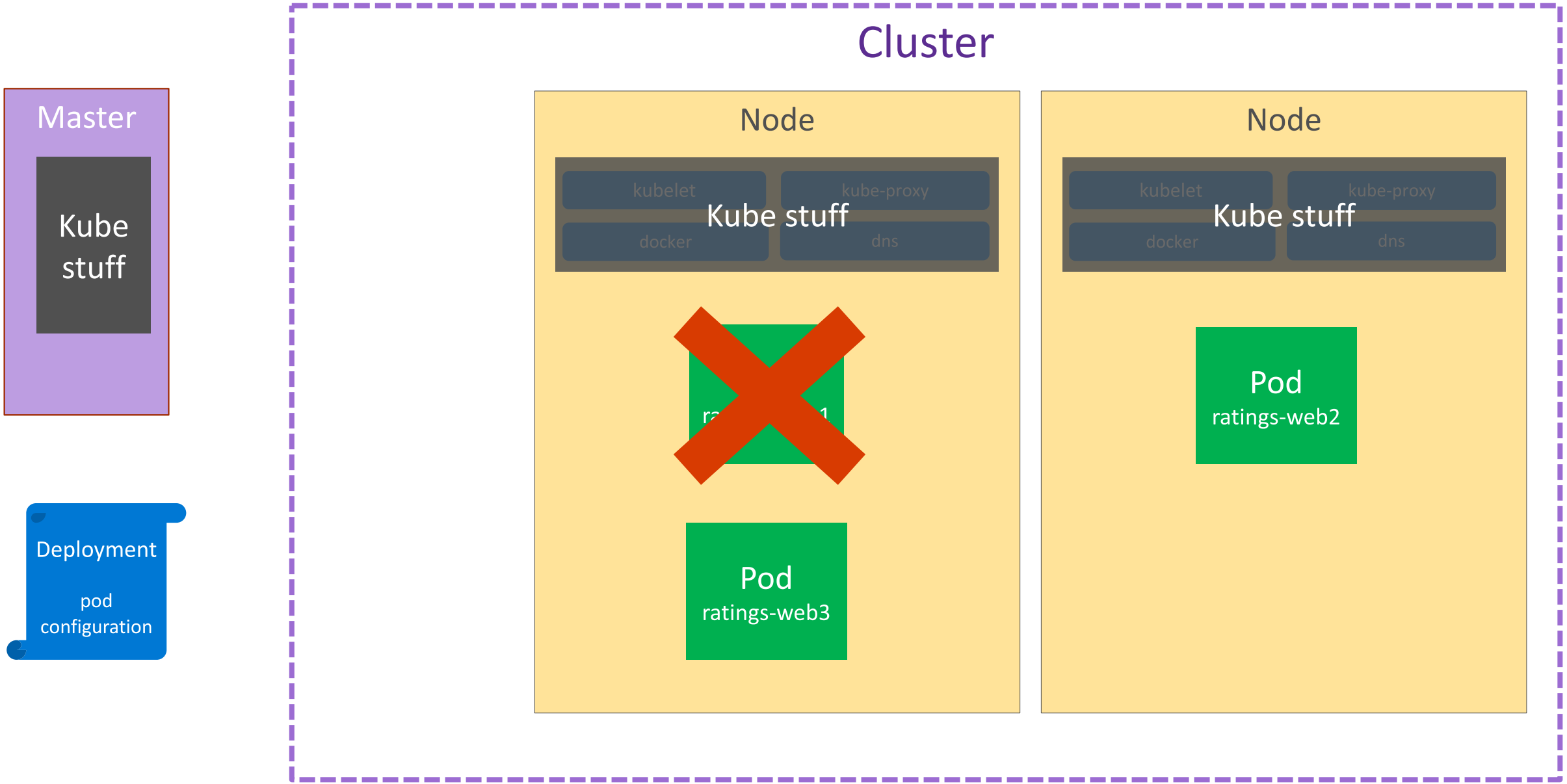
Metadata describes how to deploy a pod



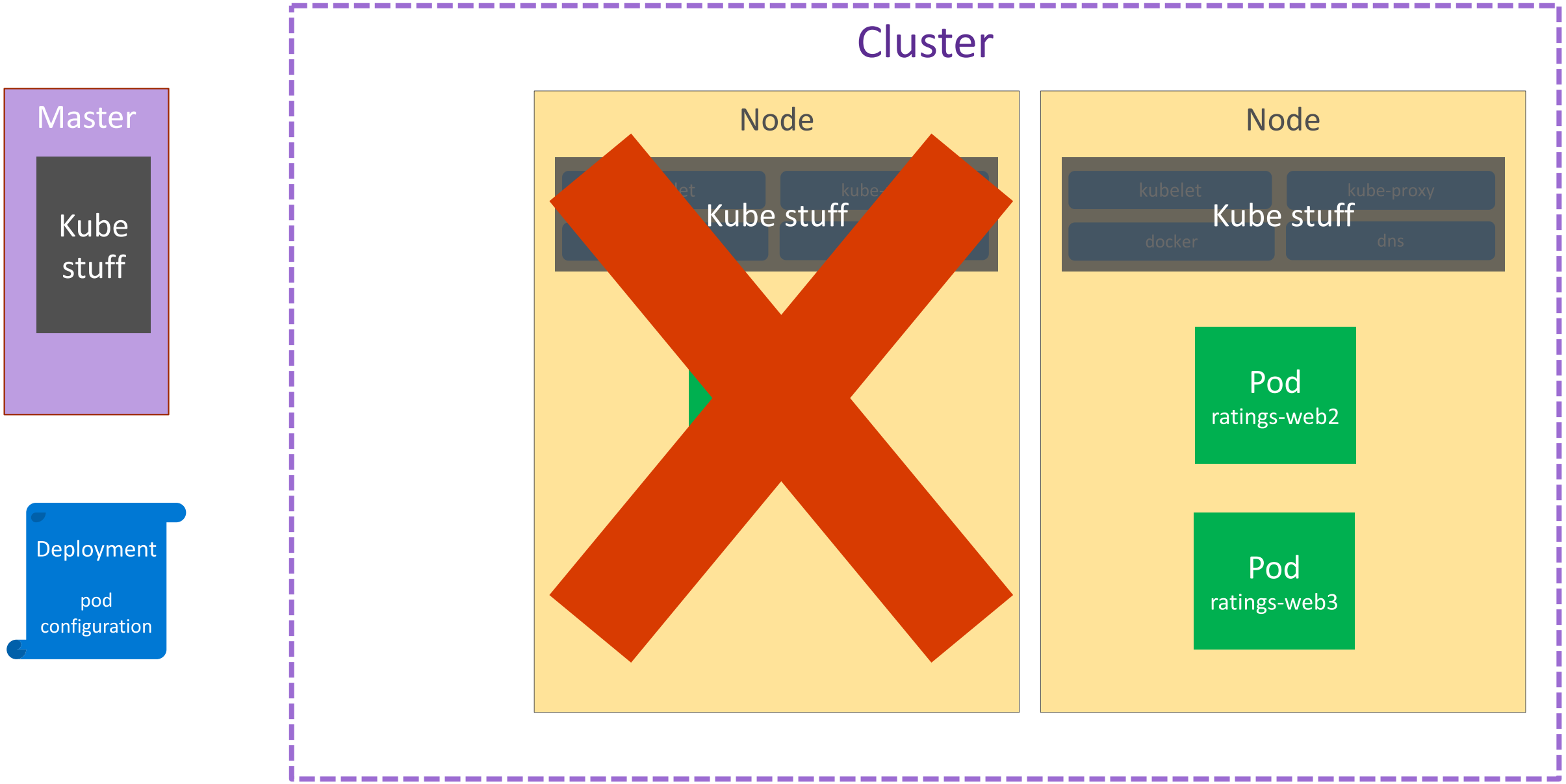
Kubernetes Deployment In Action



Kubernetes Desired State In Action – Example 1

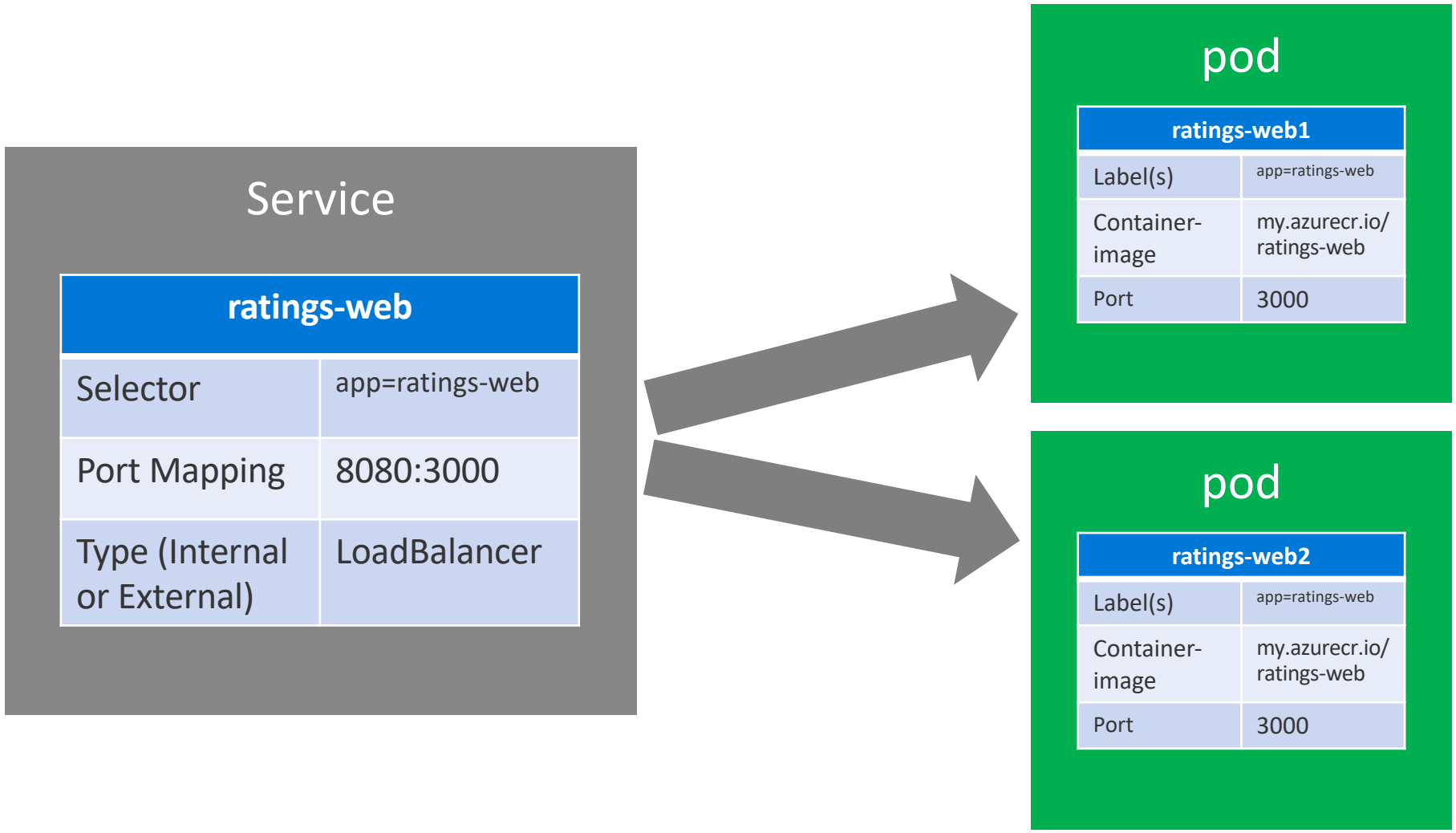


Kubernetes Desired State In Action – Example 2



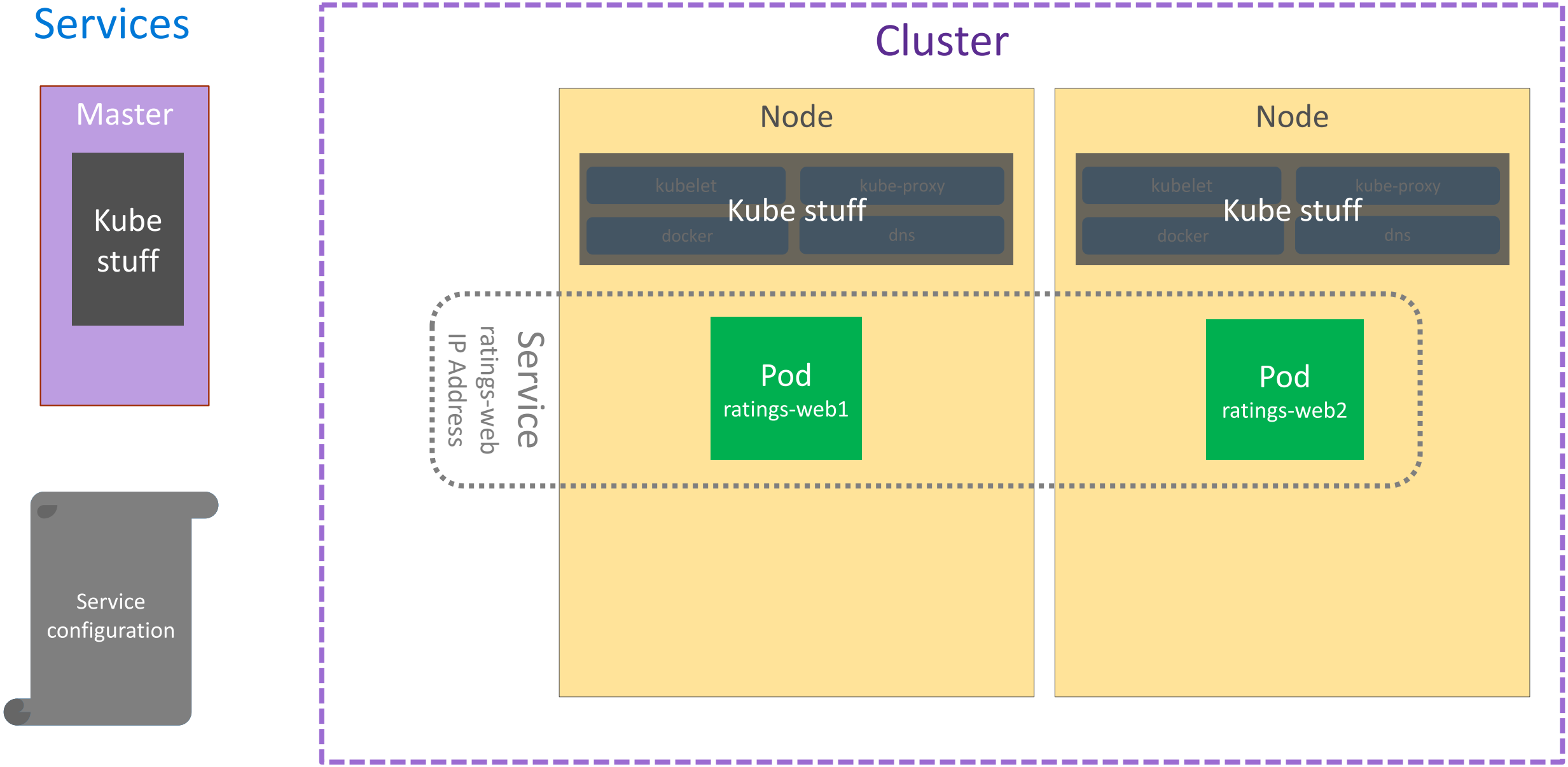
Services

Metadata describes how to reach the pods

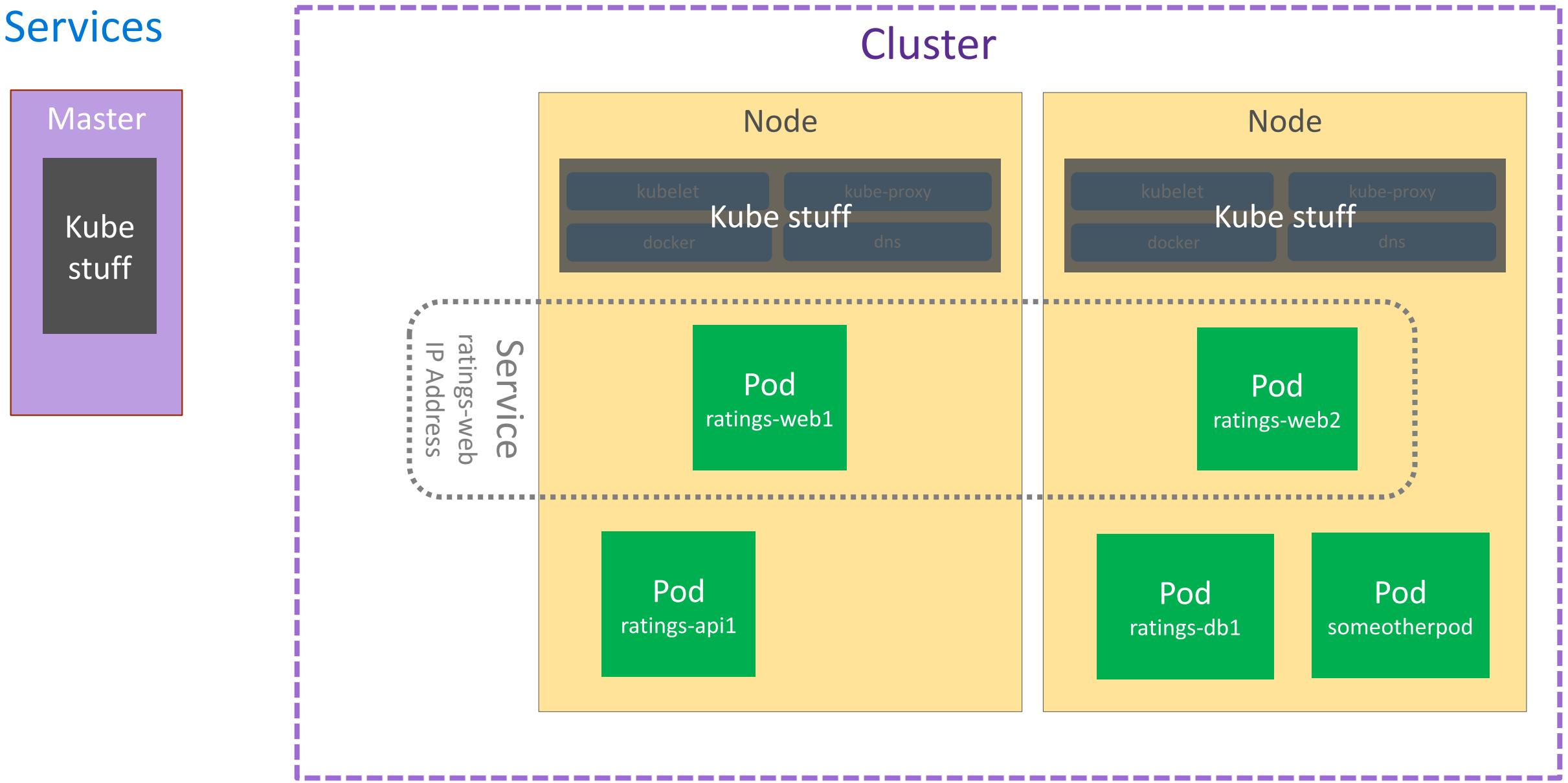


Kubernetes Service Deployment In Action

Services

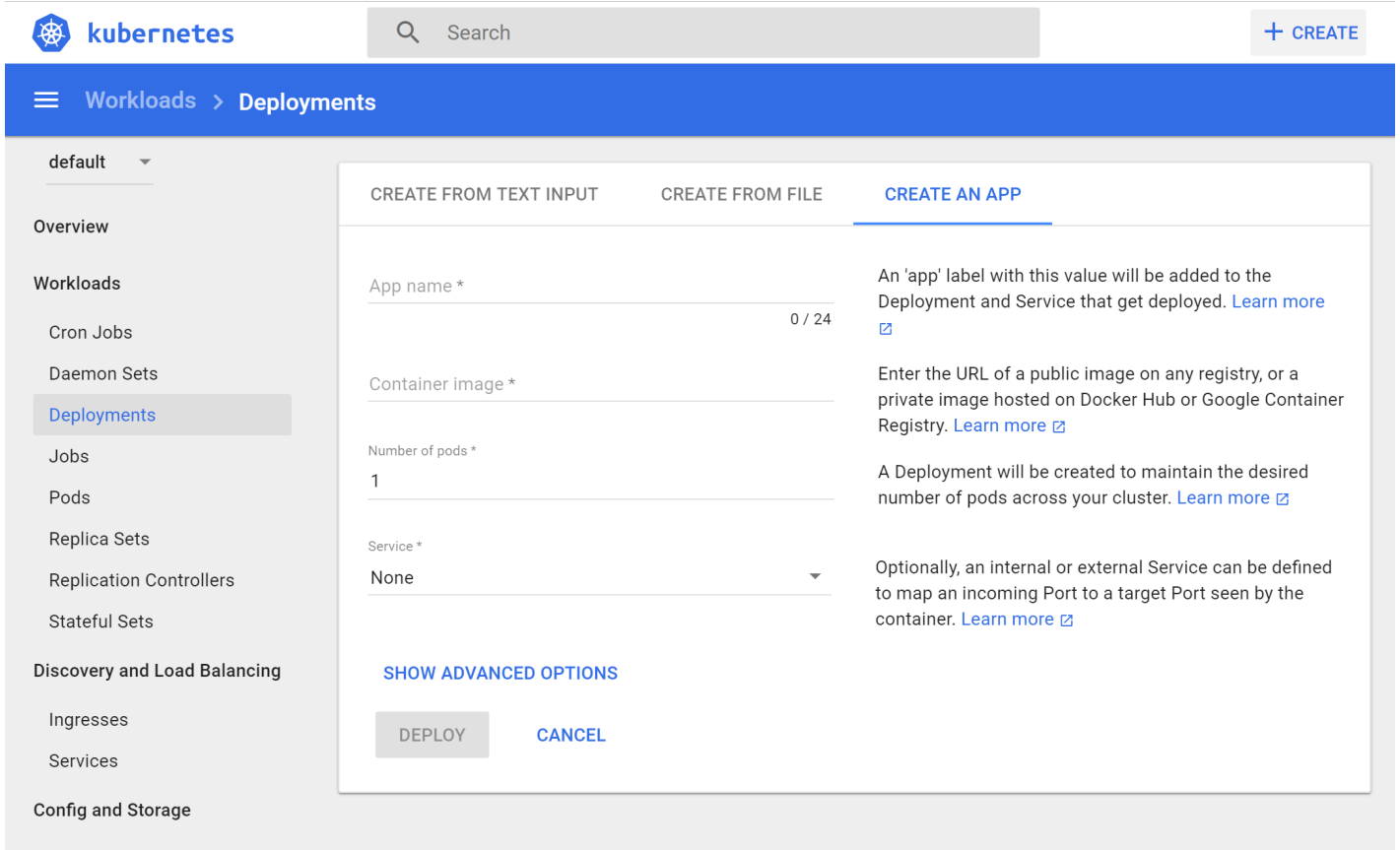


Kubernetes Service Deployment In Action



Deployments in Kubernetes

Manual via Kubernetes Portal

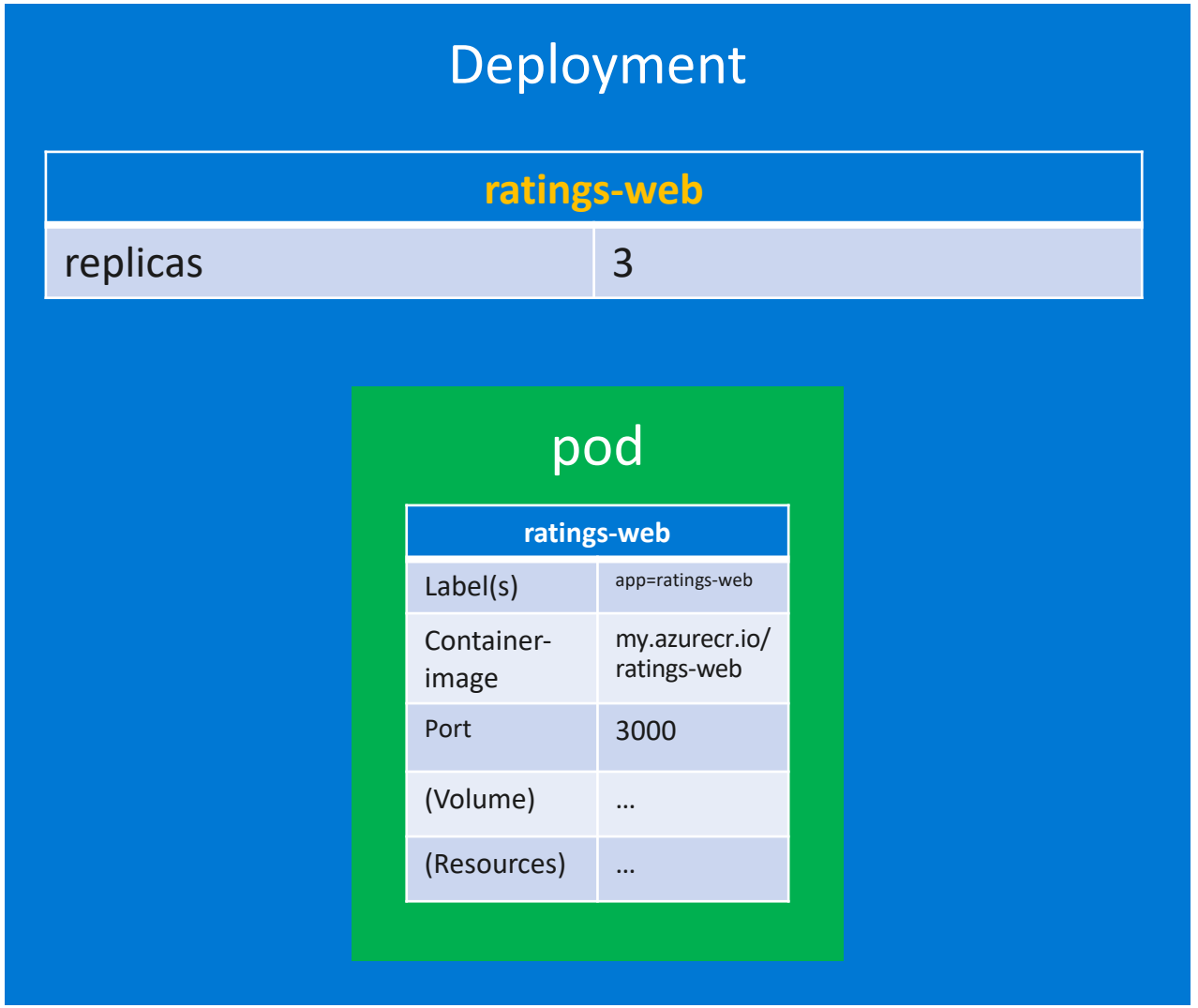


Infra-as-Code a YAML file

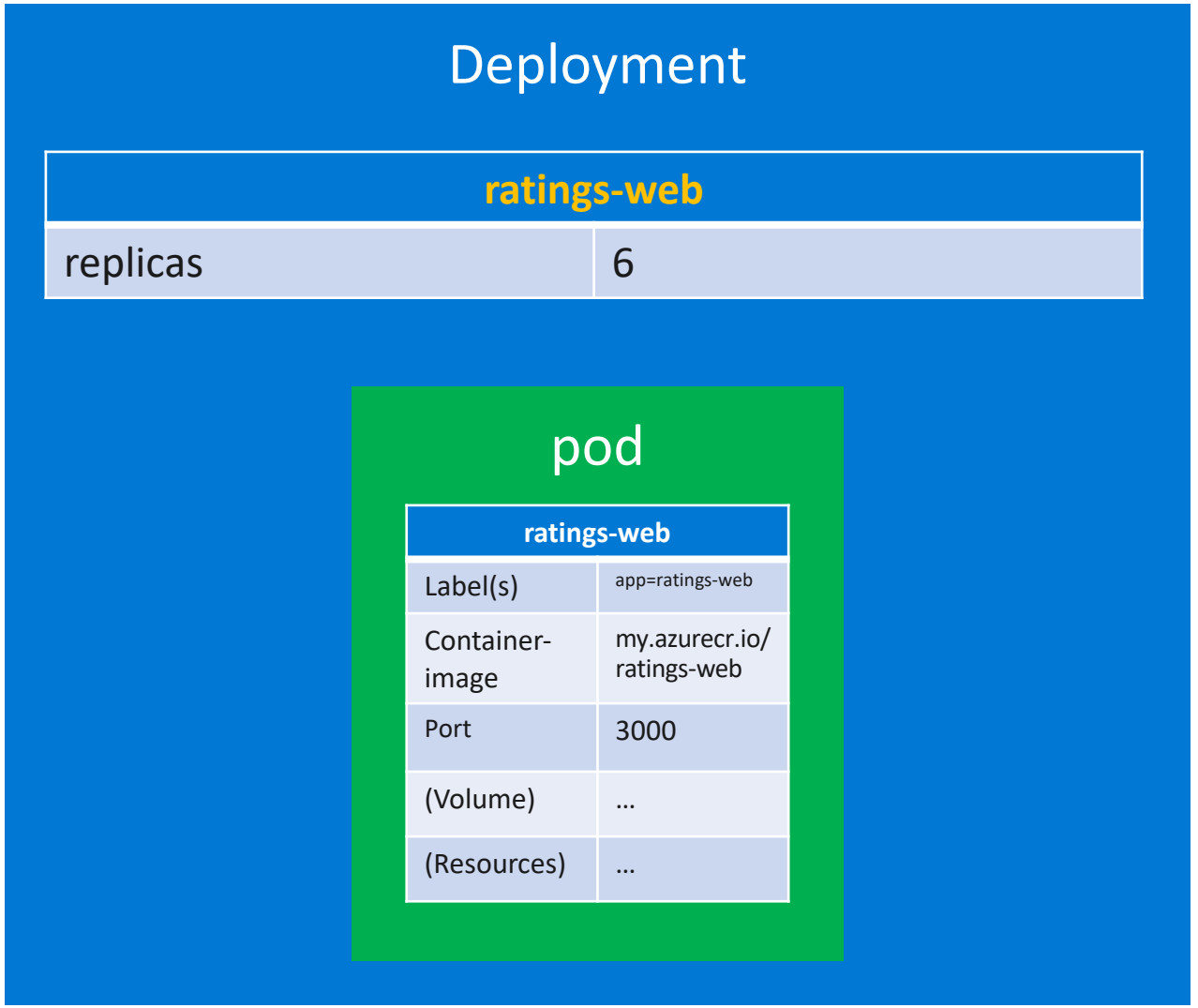
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: content-web
  labels:
    app: content-web
spec:
  replicas: 1
  selector:
    matchLabels:
      app: content-web
  template:
    metadata:
      labels:
        app: content-web
    spec:
      containers:
      - name: content-web
        image: dta2018hack/content-web
        resources:
          requests:
            cpu: 500m
            memory: 128Mi
        ports:
        - containerPort: 3000
        env:
        - name: CONTENT_API_URL
          value: "http://content-api:3001"
```

Scaling & High Availability

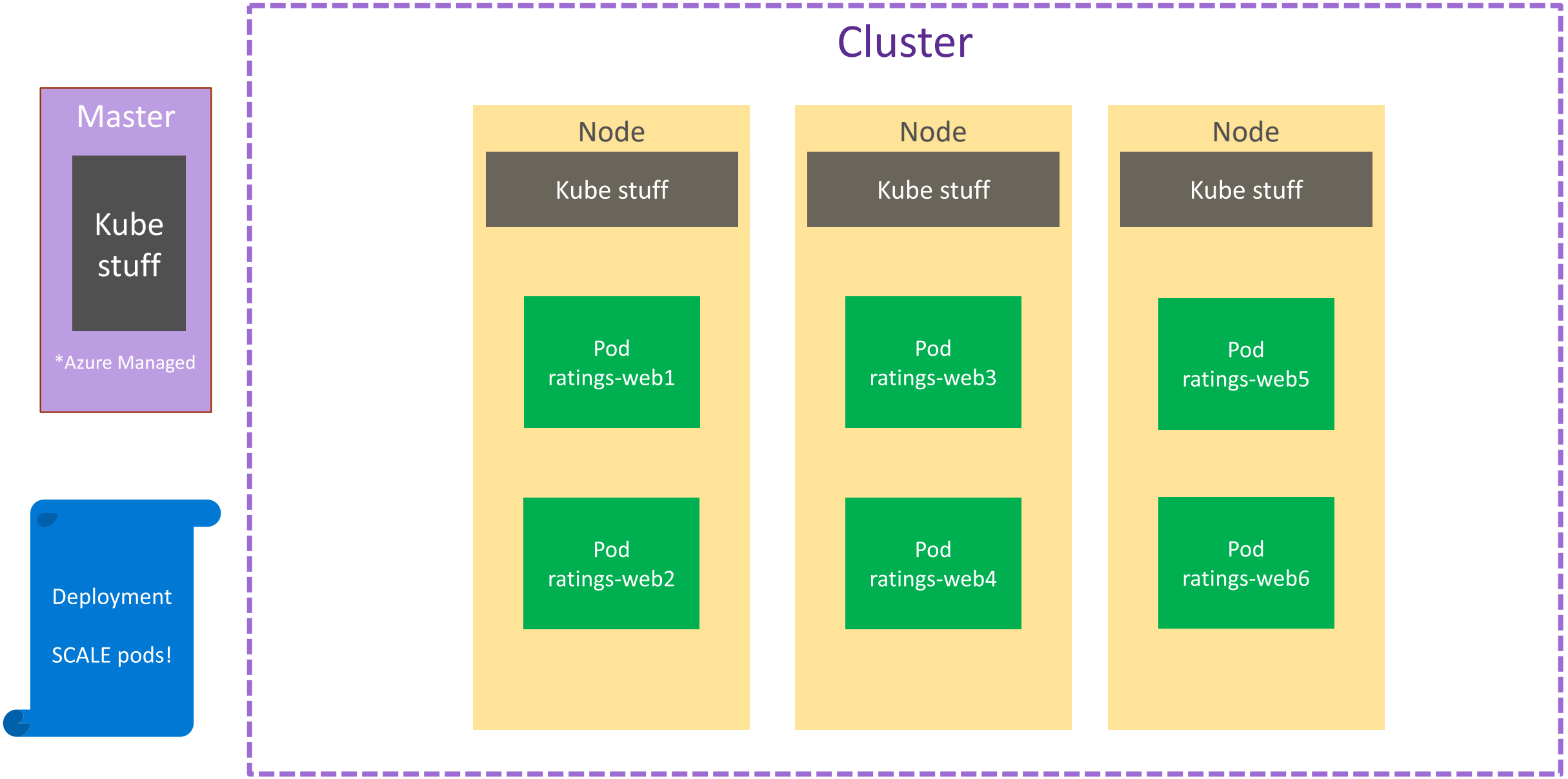
Scaling Pods



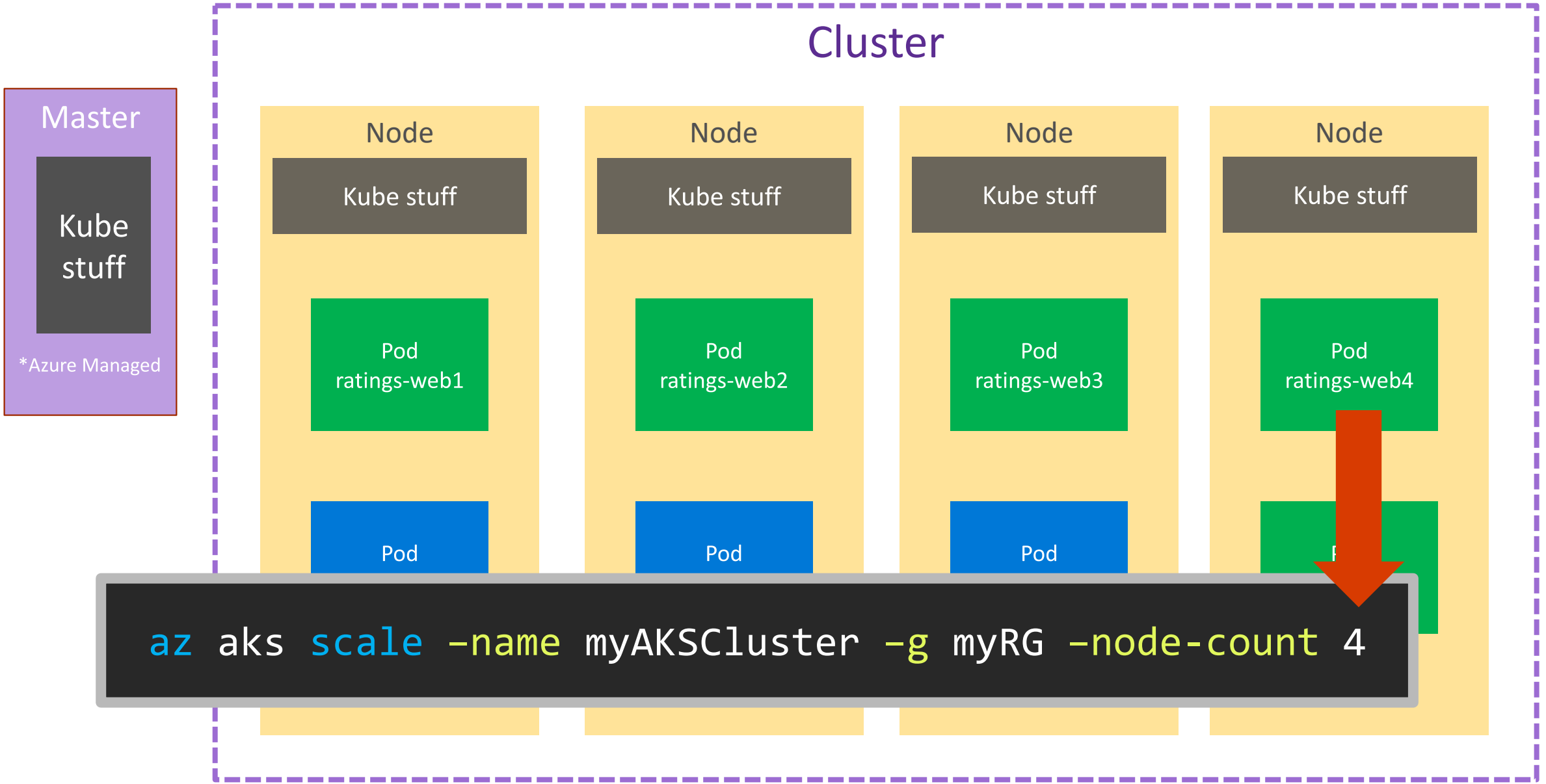
Scaling Pods



Scaling Pods

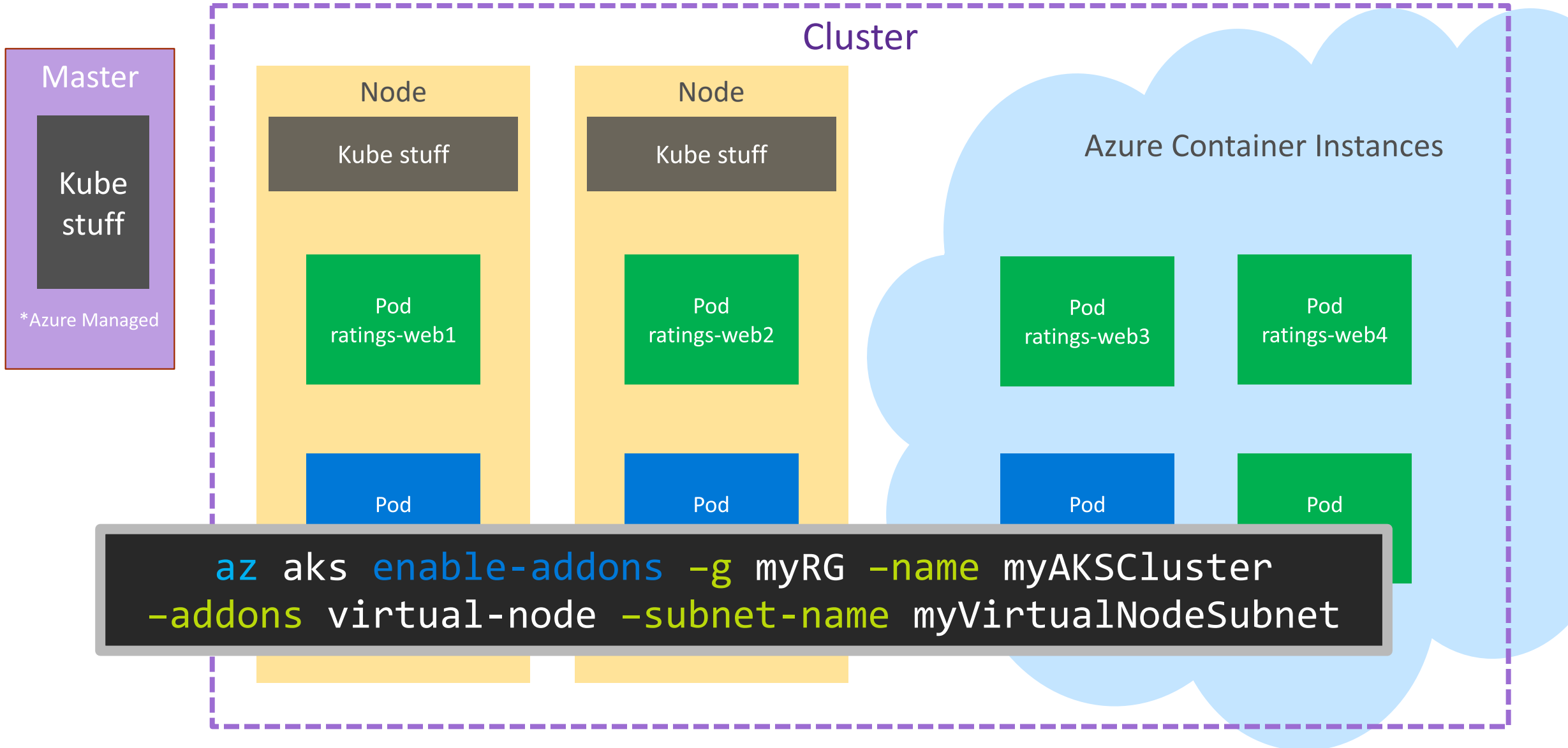


Scaling Nodes



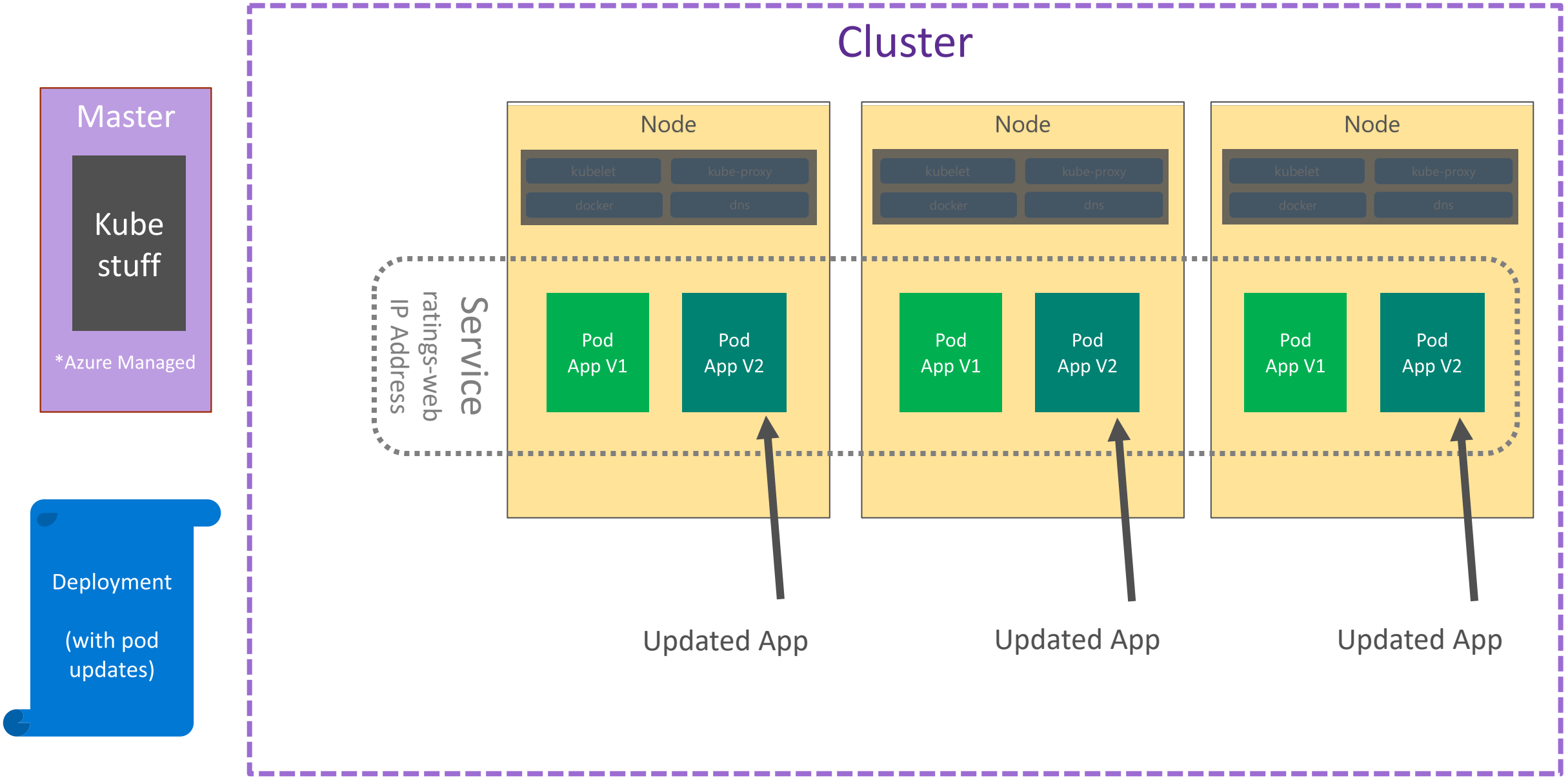
Virtual Nodes

With Azure Container Instances

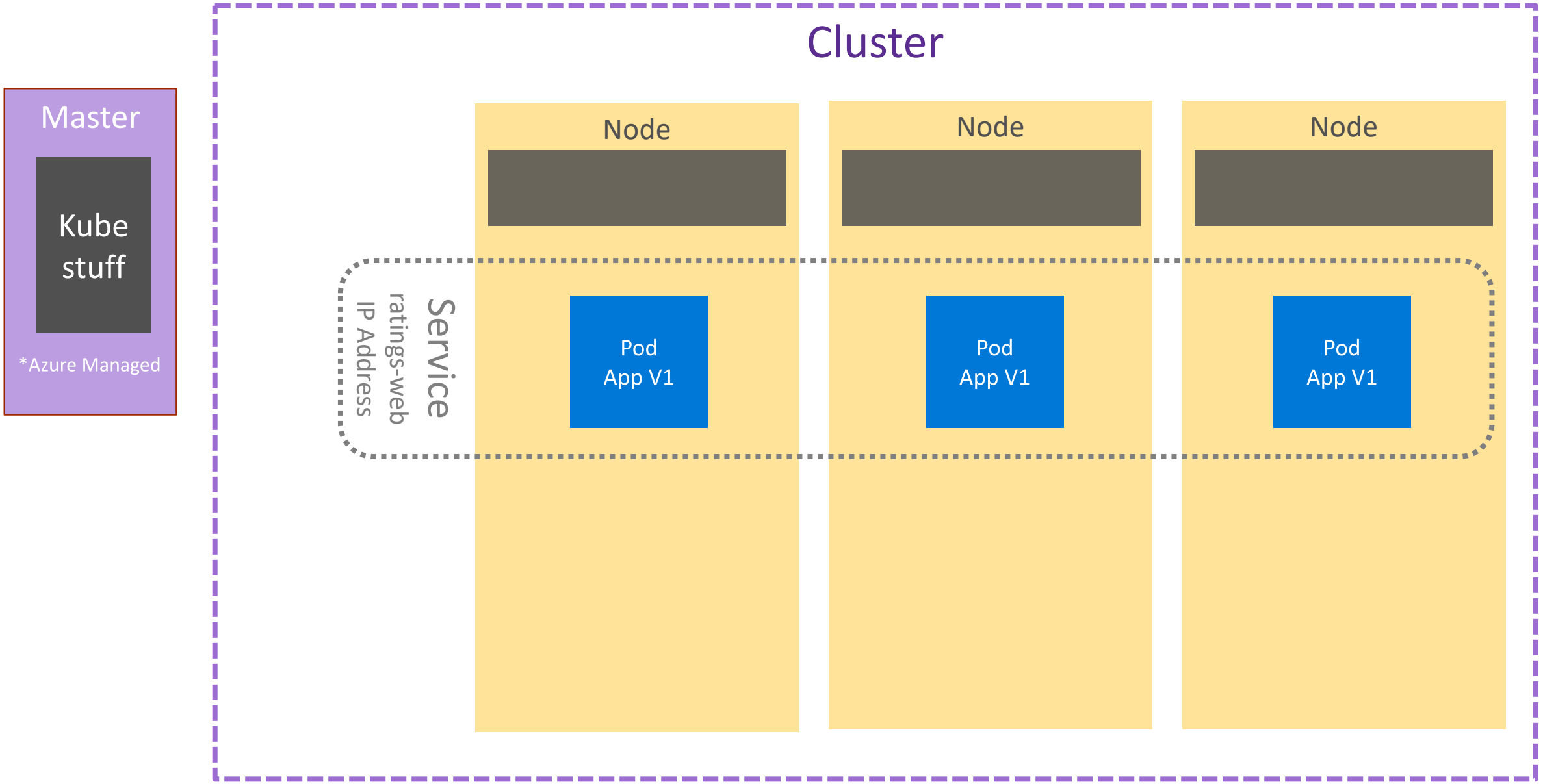


Pod Updates

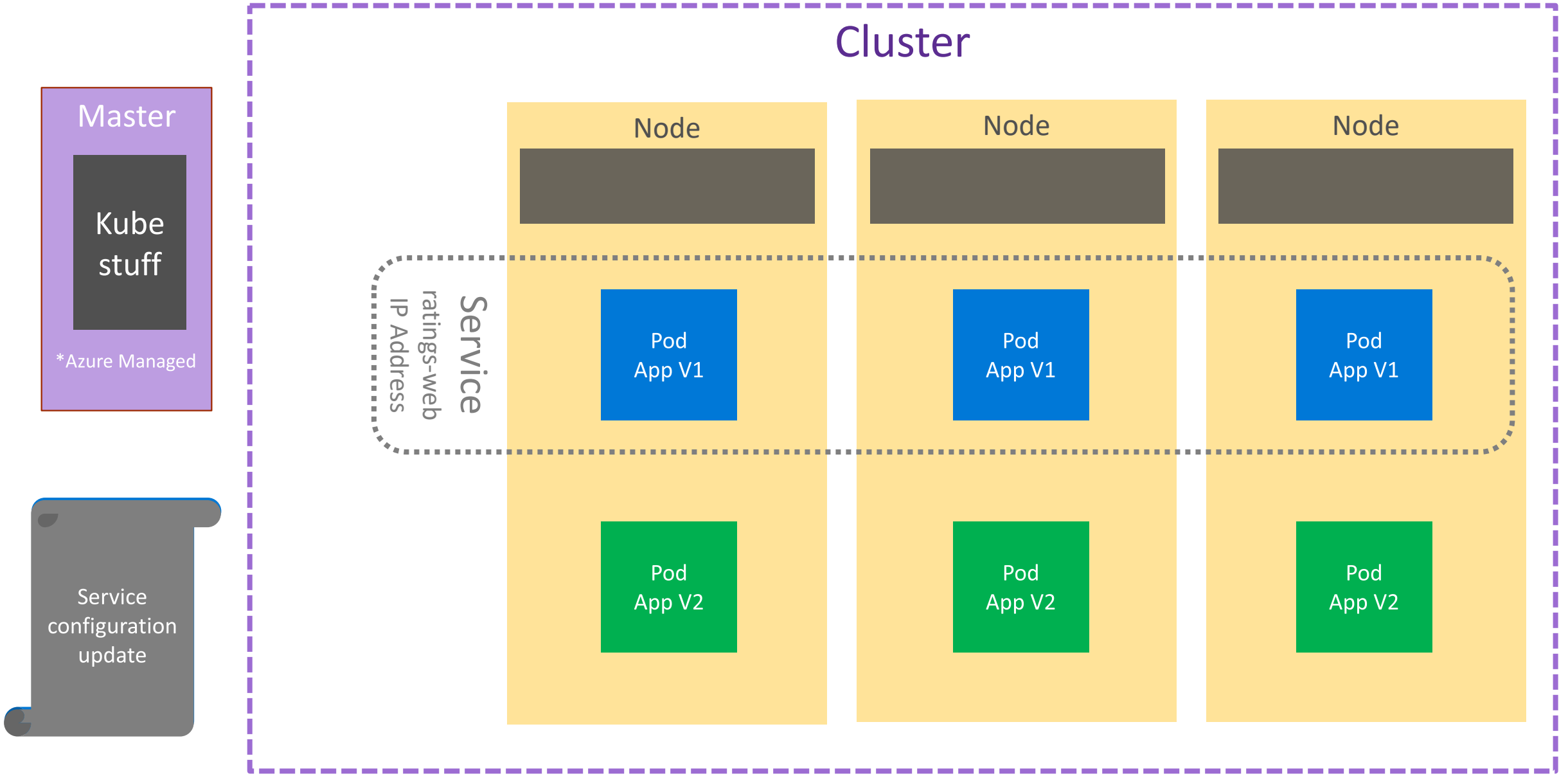
Rolling updates



Blue – the initial deployment



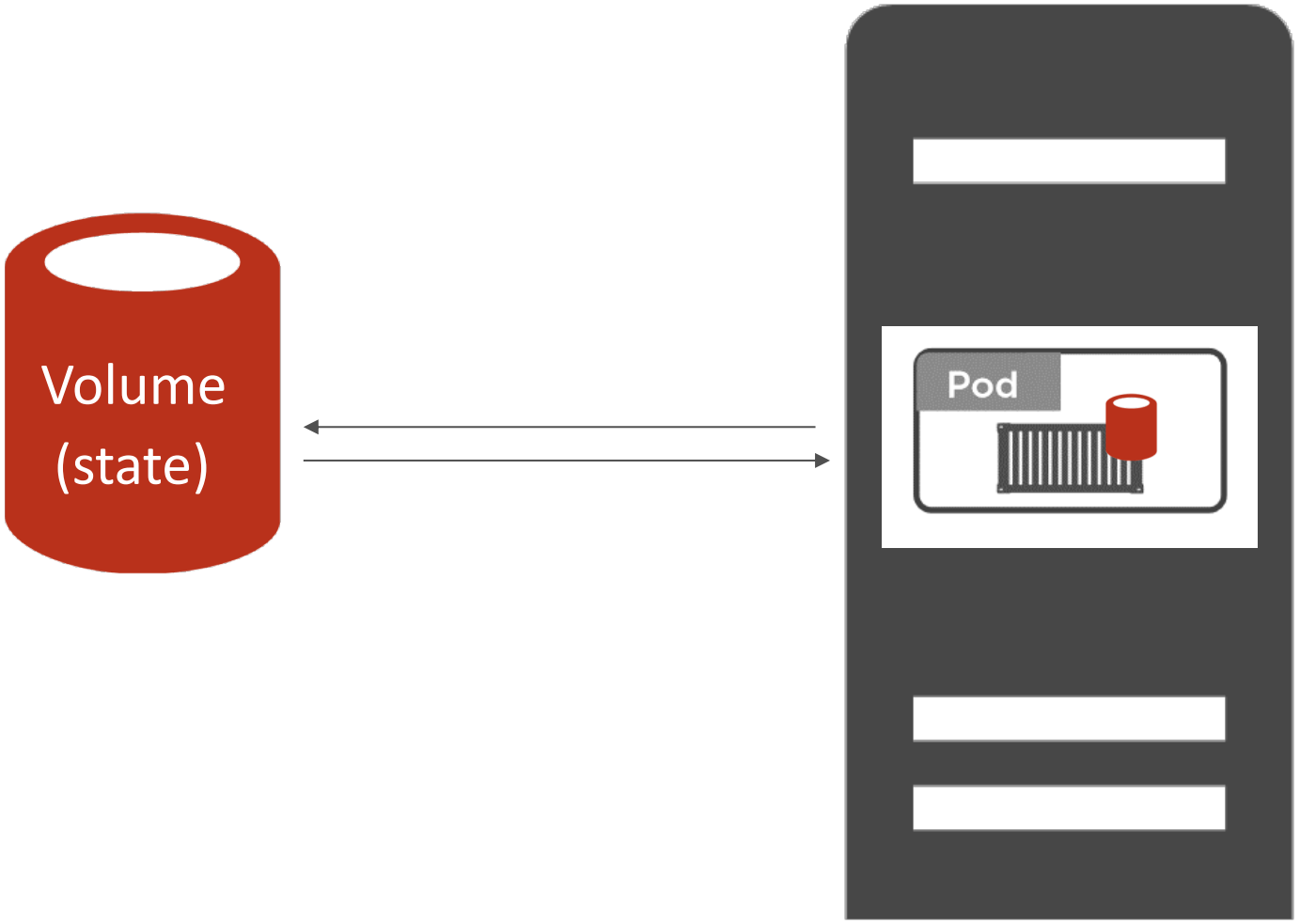
Green – the new deployment



Storage

Volumes in Kubernetes

Why do we need volumes ?



On-disk files in a Container
are
ephemeral

Storage Classes in Kubernetes

Cluster Admin

```
C:\Users\sowmyans>kubectl get sc
NAME                 PROVISIONER                    AGE
default (default)    kubernetes.io/azure-disk       9d
managed-premium      kubernetes.io/azure-disk       9d
```

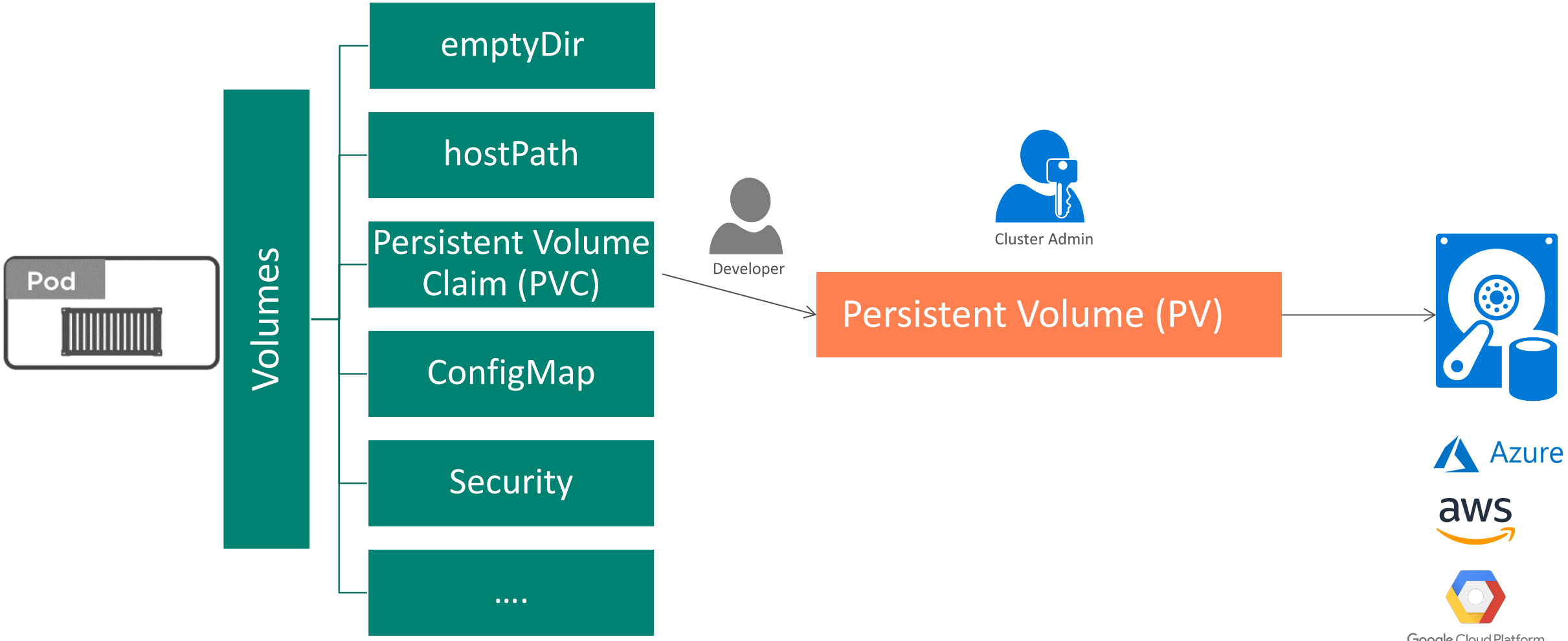
Azure

Google Cloud Platform

aws

Volumes in Kubernetes

PV vs PVC



Volumes in Kubernetes

Dynamic Storage in AKS – Azure Disk

```
C:\Users\sowmyans>kubectl get sc
NAME                 PROVISIONER                AGE
default (default)    kubernetes.io/azure-disk   9d
managed-premium      kubernetes.io/azure-disk   9d
```

```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: azure-managed-disk
spec:
  accessModes:
  - ReadWriteOnce
  storageClassName: managed-premium
  resources:
    requests:
      storage: 5Gi
```

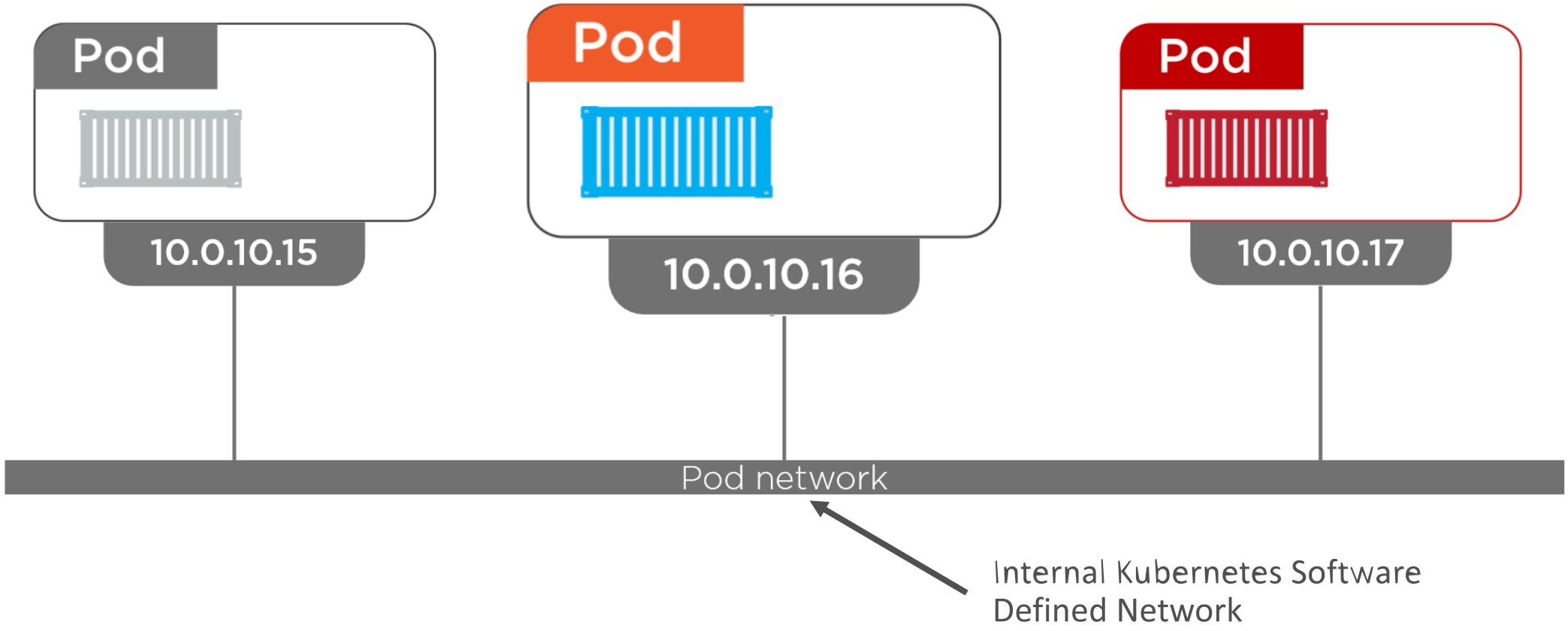


```yaml
kind: Pod
apiVersion: v1
metadata:
  name: mypod
spec:
  containers:
    - name: myfrontend
      image: nginx
      volumeMounts:
      - mountPath: "/mnt/azure"
        name: volume
  volumes:
    - name: volume
      persistentVolumeClaim:
        claimName: azure-managed-disk
```

Networking

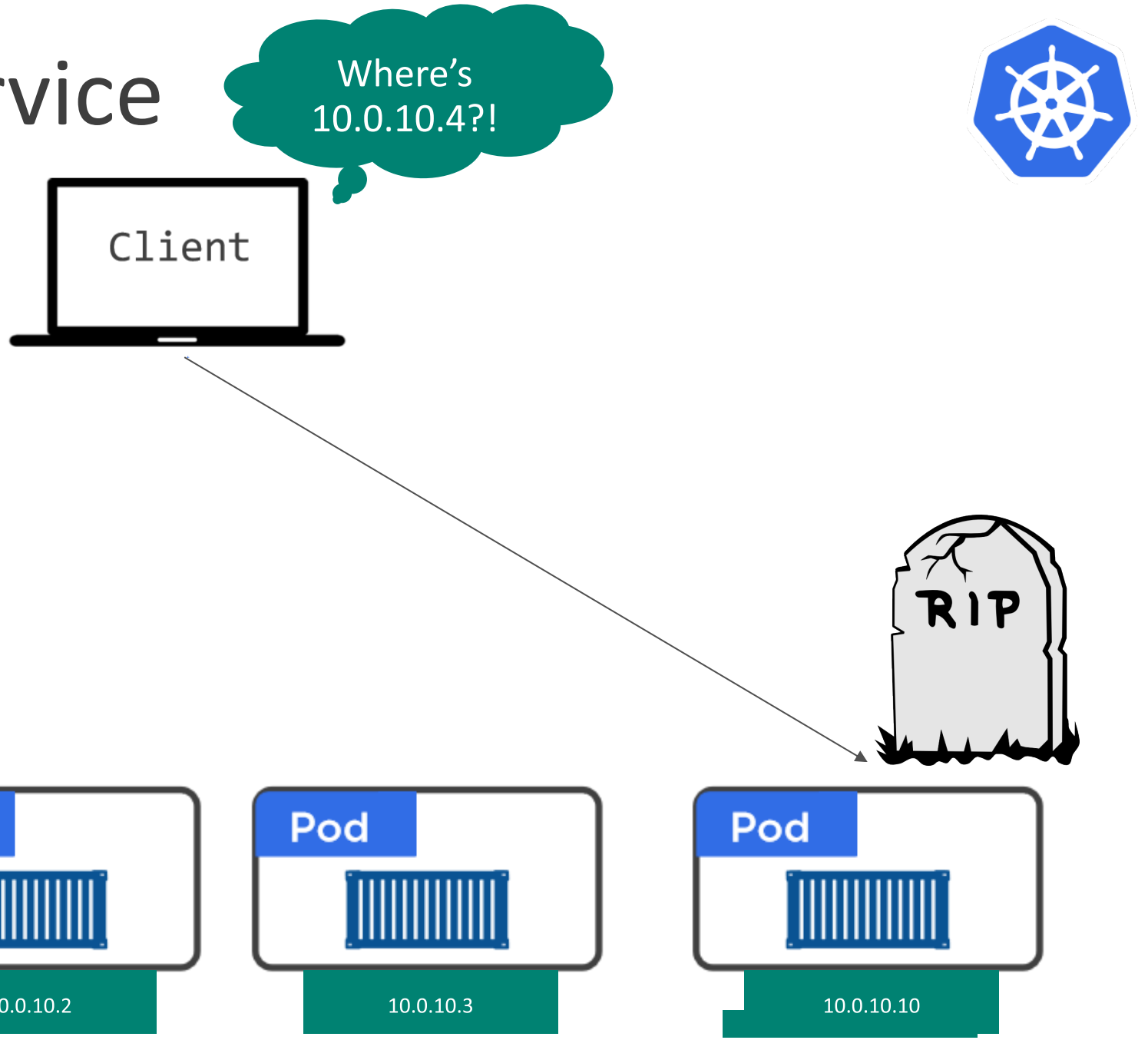
Networking – Pod

Inter-Pod Communication



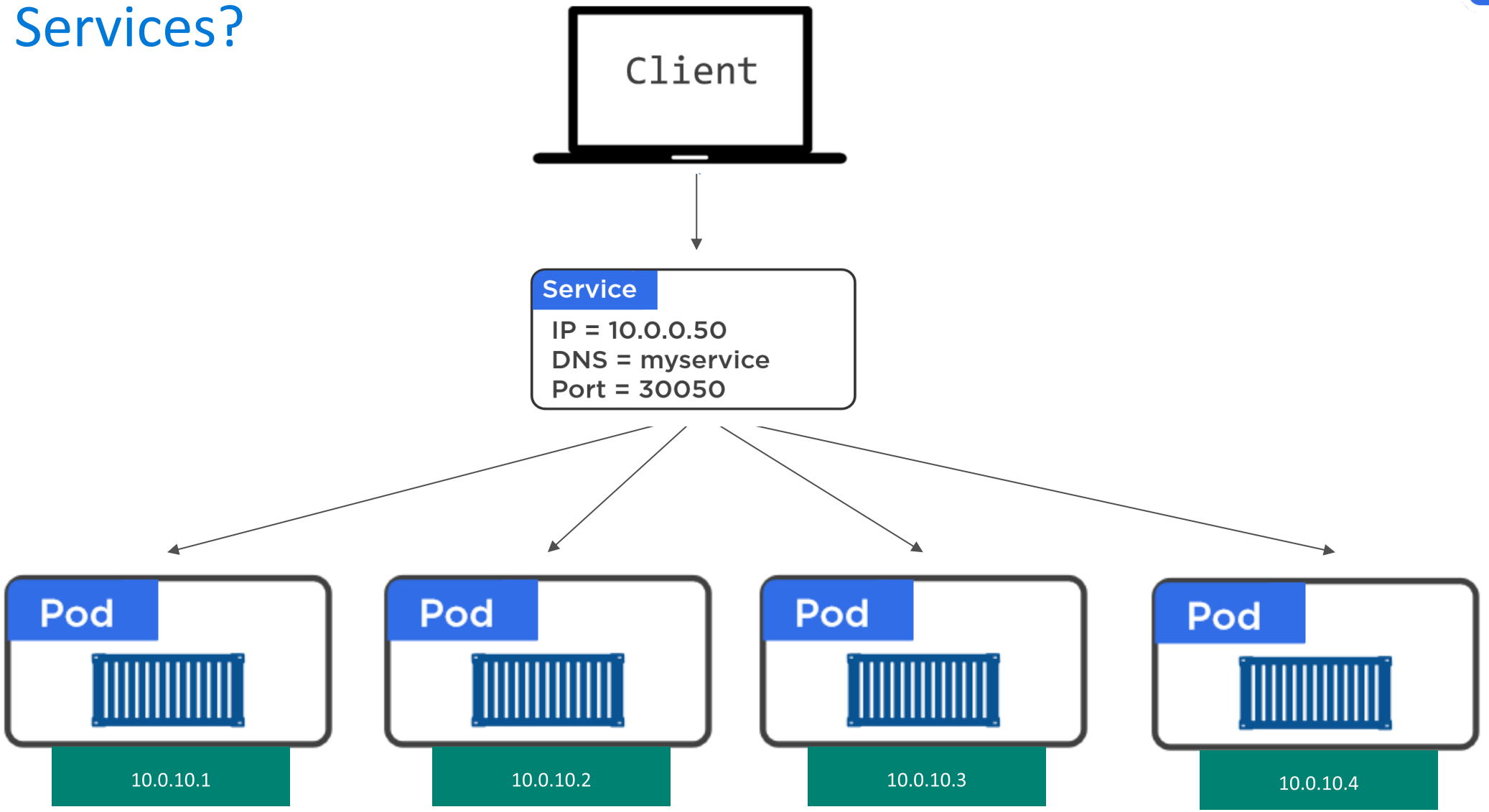
Networking – Service

Why Services?



Networking – Service

Why Services?

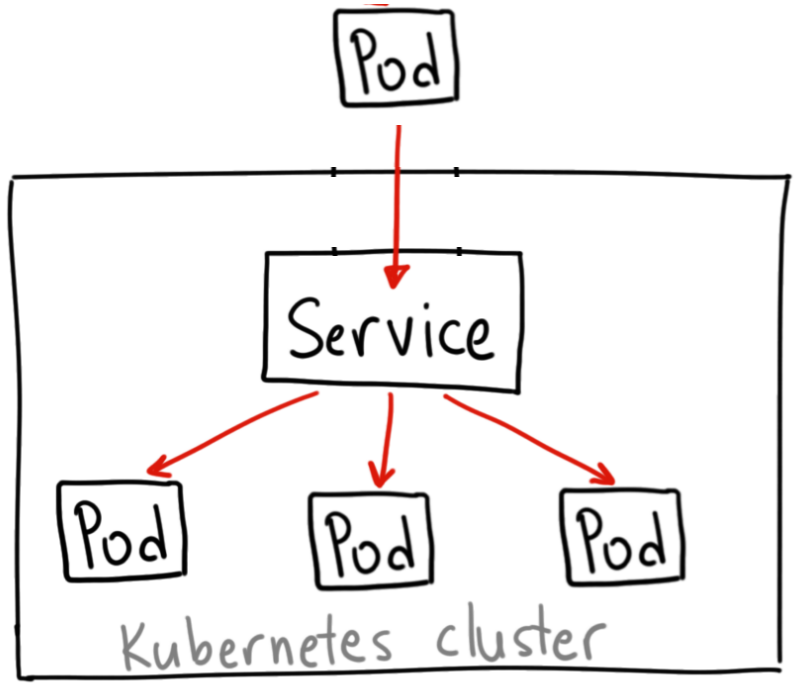


Different service types

- ClusterIP (internal)
- LoadBalancer (external)
- NodePort (external)

Networking – Service : ClusterIP

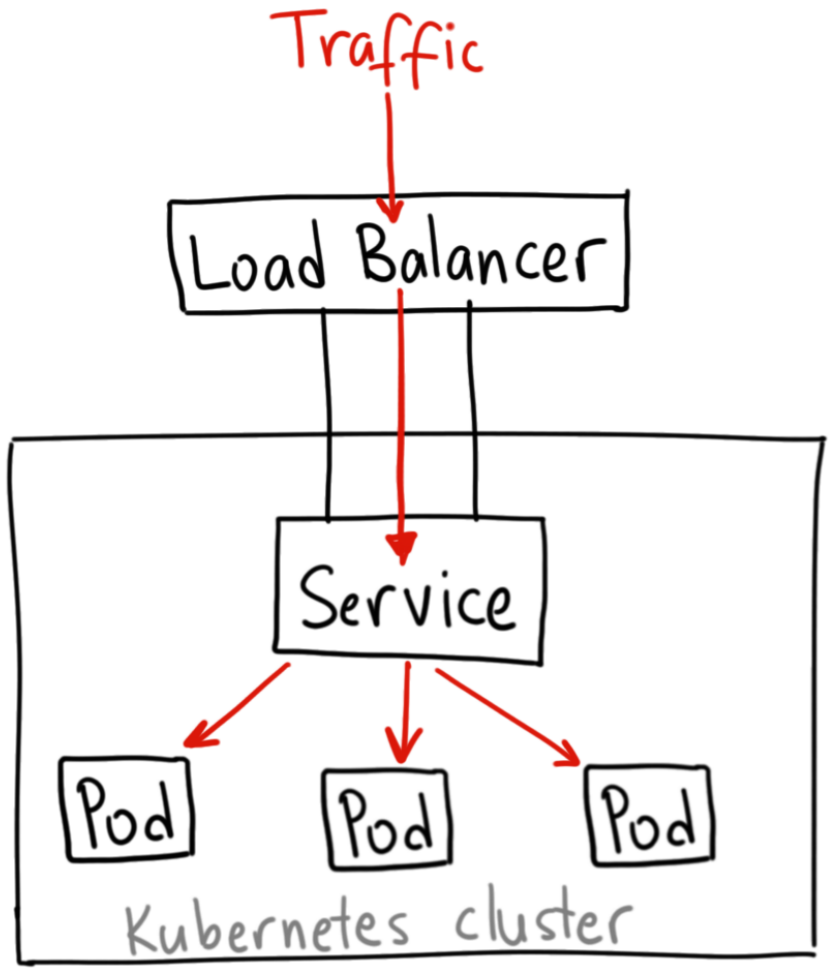
How does ClusterIP Work




```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-internal-service
spec:
  selector:
    app: my-app
  type: ClusterIP
  ports:
  - name: http
    port: 80
    targetPort: 80
    protocol: TCP
```

Networking – Service : LoadBalancer

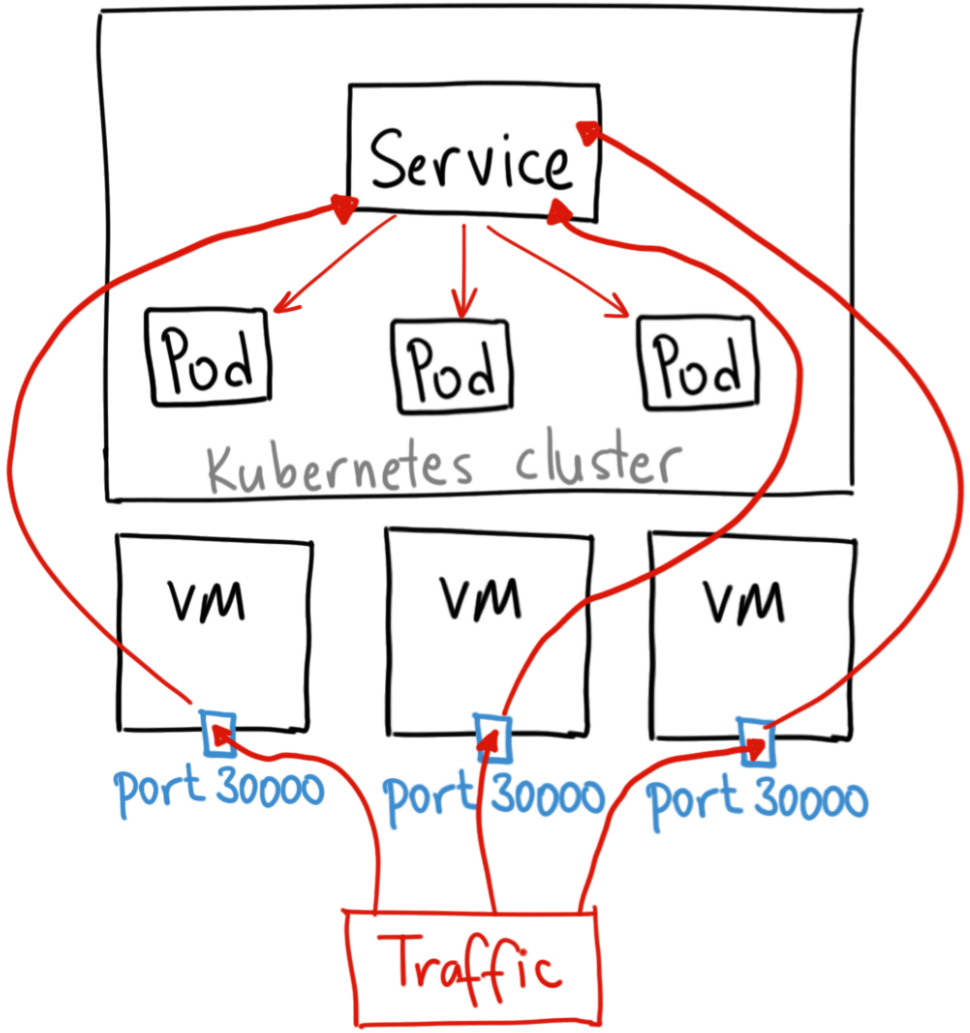
How does LoadBalancer Work




```yaml
apiVersion: v1
kind: Service
metadata:
  labels:
    app: web
  name: web
spec:
  ports:
    - name: web-traffic
      port: 80
      protocol: TCP
      targetPort: 3000
  selector:
    app: web
  sessionAffinity: None
  type: LoadBalancer
```

Networking – Service : NodePort

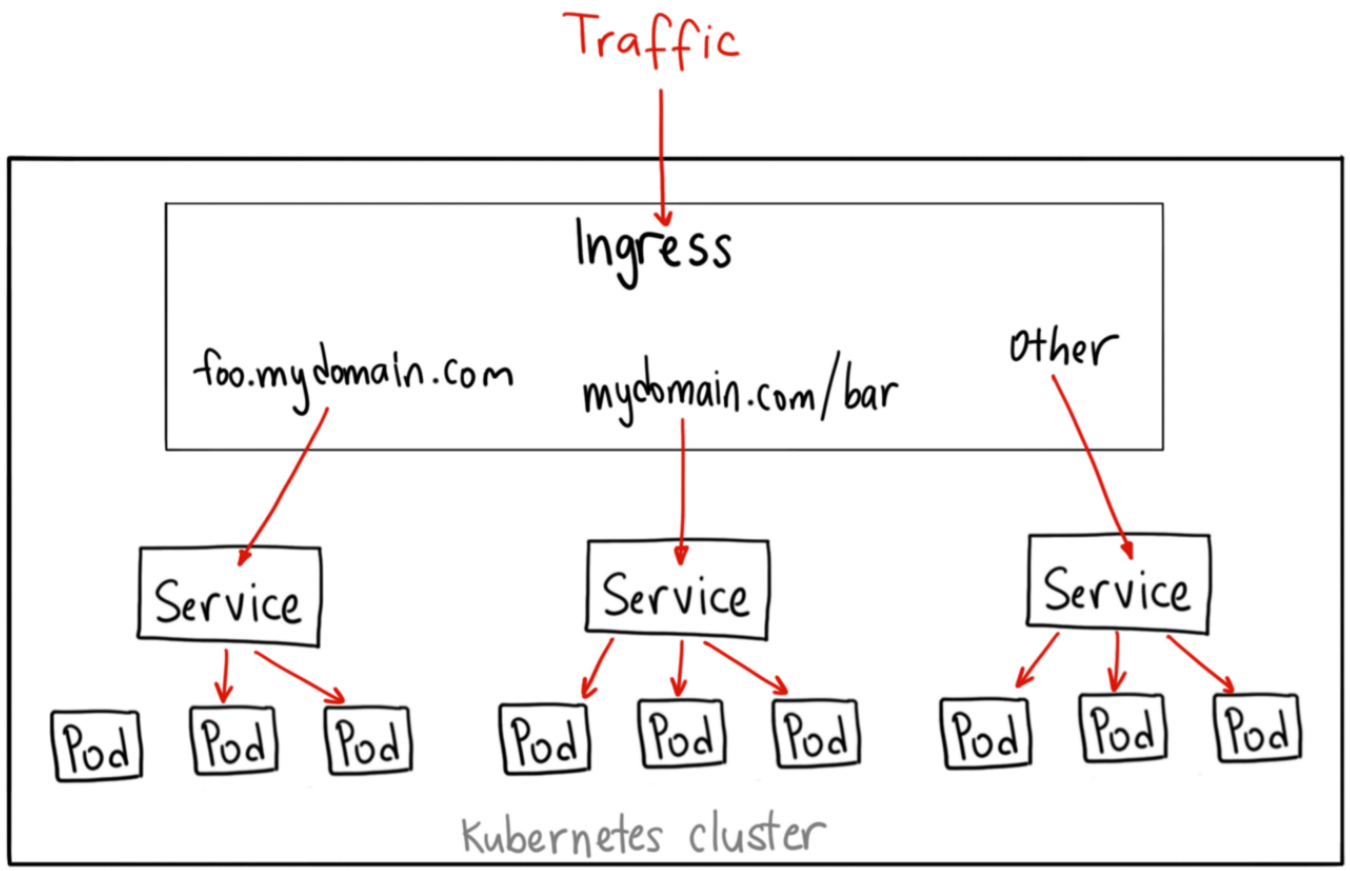
How does NodePort Work




```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-nodeport-service
spec:
  selector:
    app: my-app
  type: NodePort
  ports:
  - name: http
    port: 80
    targetPort: 80
    nodePort: 30036
    protocol: TCP
```

Networking – Ingress

How does Ingress Controller Work




```yaml
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: my-ingress
spec:
  backend:
    serviceName: other
    servicePort: 8080
  rules:
  - host: foo.mydomain.com
    http:
      paths:
      - backend:
          serviceName: foo
          servicePort: 8080
  - host: mydomain.com
    http:
      paths:
      - path: /bar/*
        backend:
          serviceName: bar
          servicePort: 8080
```

Let’s review…

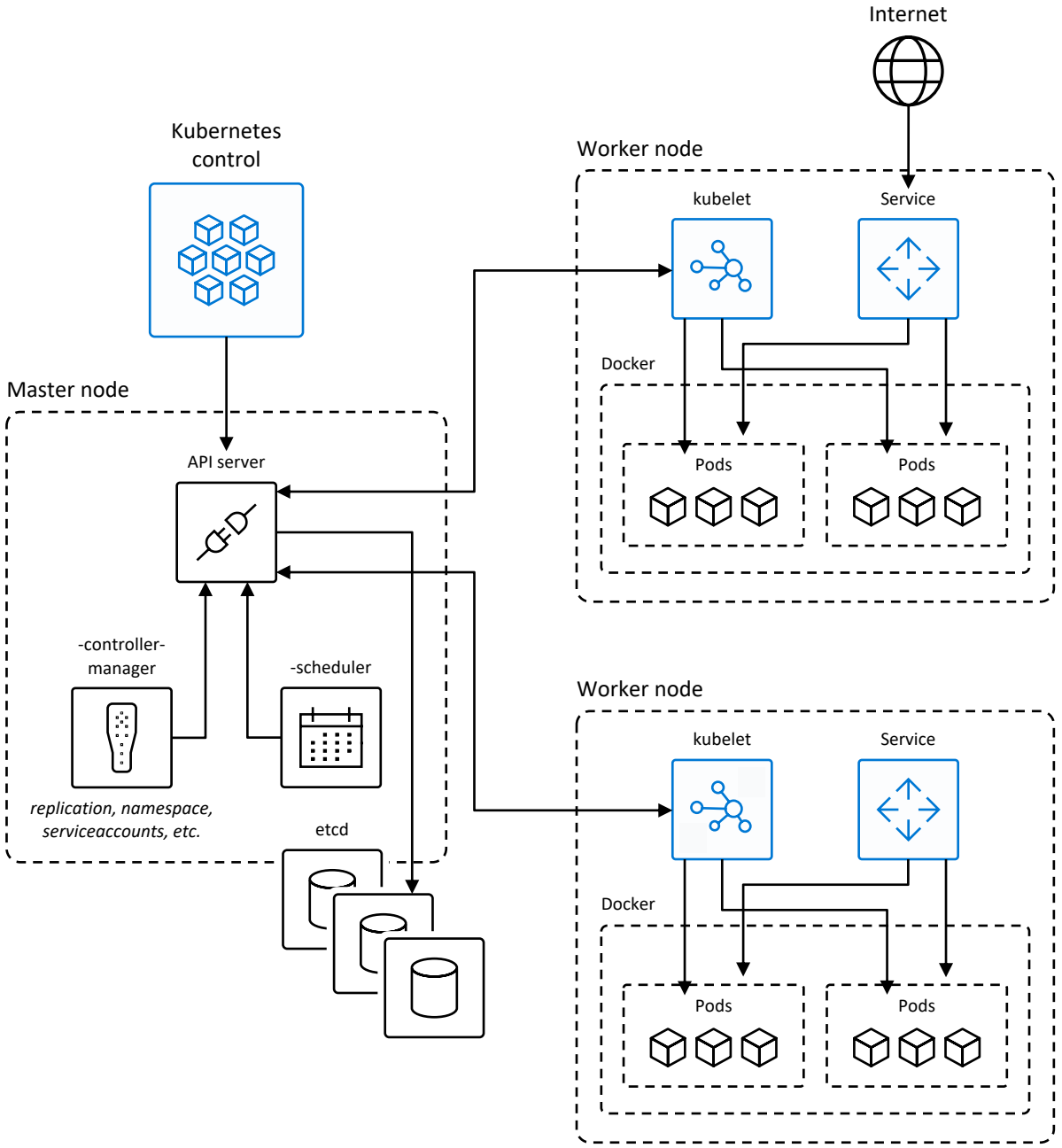
Kubernetes 101

1. Kubernetes users communicate with API server and apply desired state

2. Master nodes actively enforce desired state on worker nodes

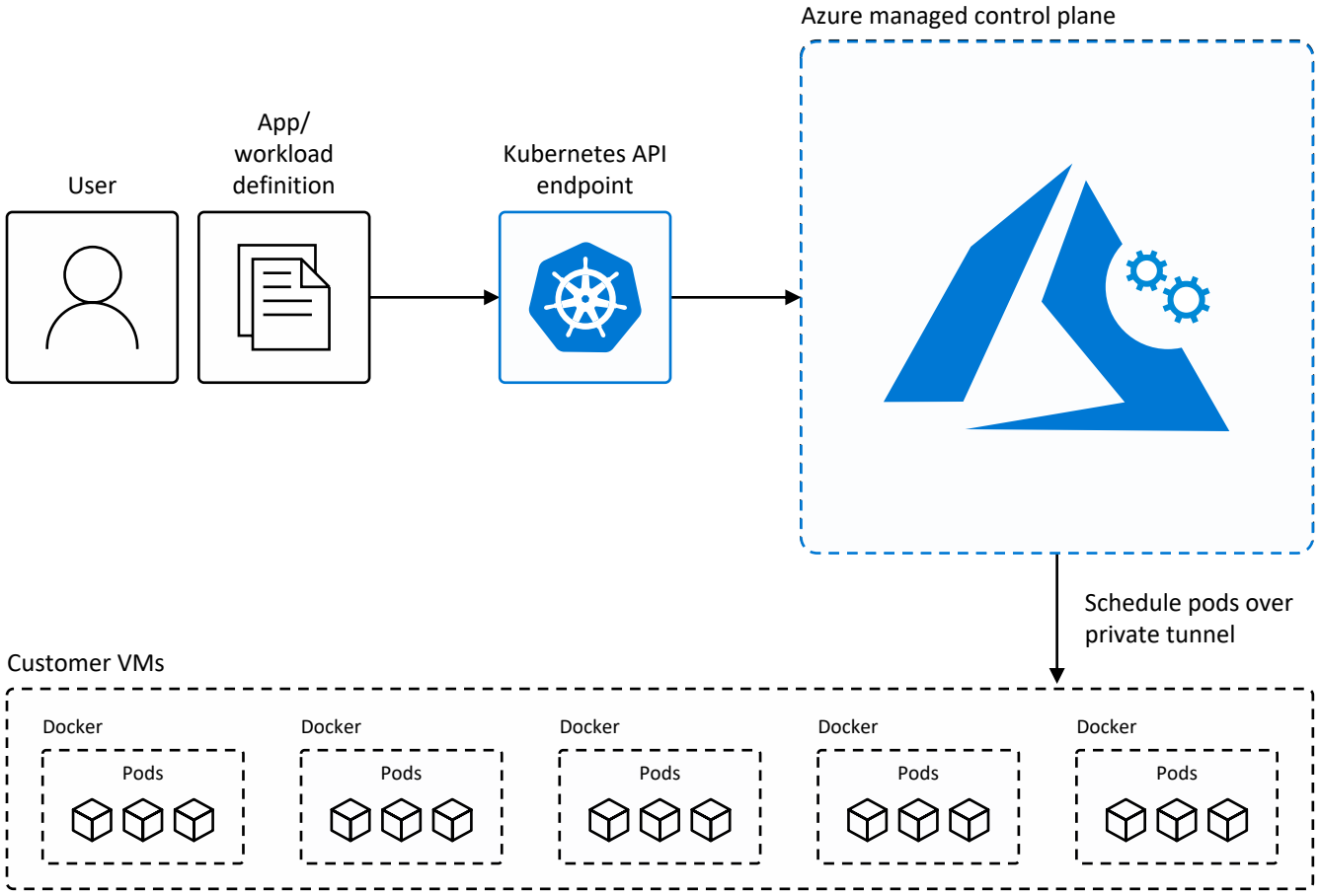
3. Worker nodes support communication between containers

4. Worker nodes support communication from the Internet



How managed Kubernetes on Azure works

- Automated upgrades, patches
- High reliability, availability
- Easy, secure cluster scaling
- Self-healing
- API server monitoring
- At no charge



What if we don’t use a managed service?

Task	The old way	With Azure
Create a cluster	Provision network and VMs Install dozens of system components including etcd Create and install certificates Register agent nodes with control plane	az aks create
Upgrade a cluster	Upgrade your master nodes Cordon/drain and upgrade worker nodes individually	az aks upgrade
Scale a cluster	Provision new VMs Install system components Register nodes with API server	az aks scale

And, we’re done…

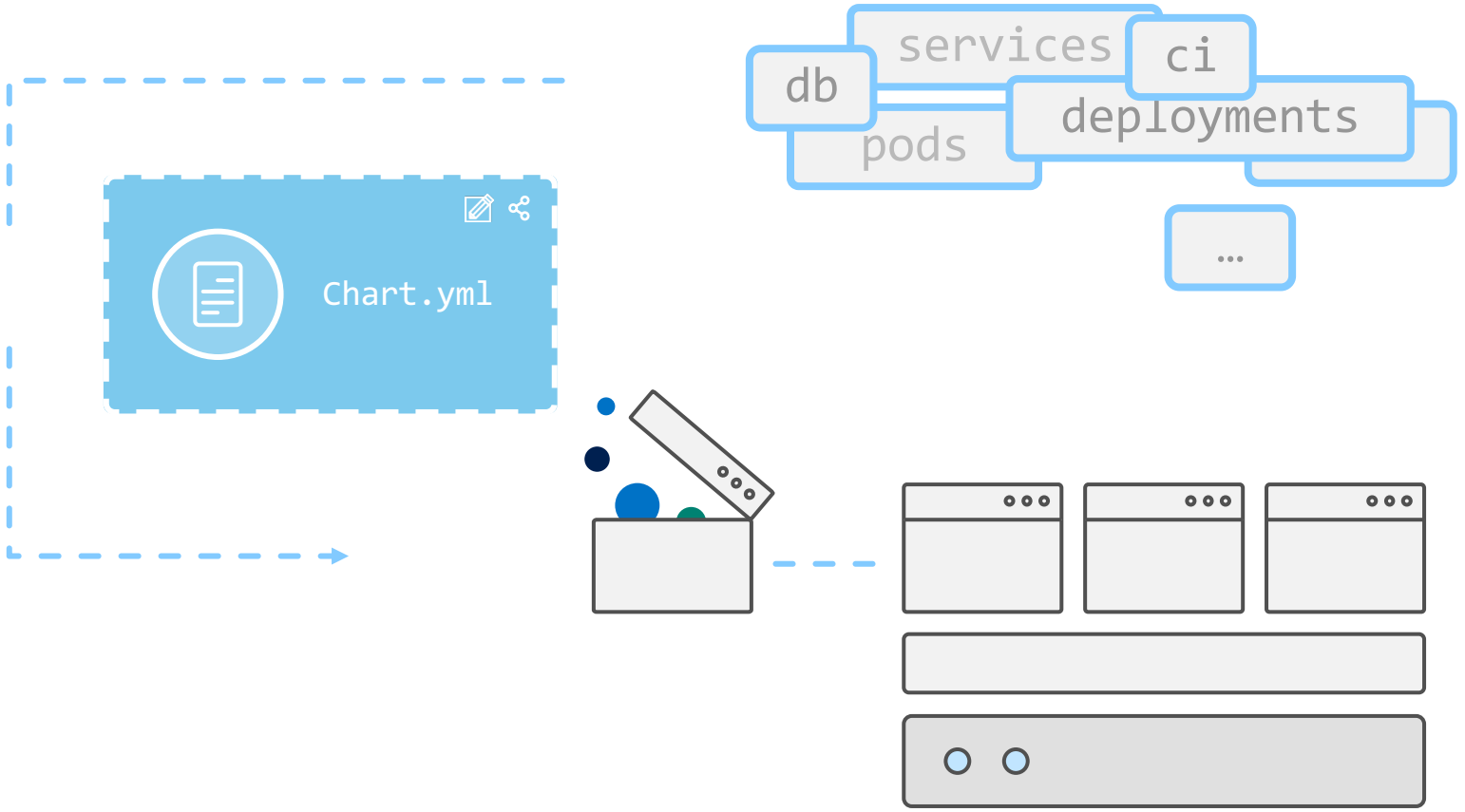
Just kidding!
You’re NEVER done with Kubernetes!

What’s Next?

Operationalizing Kubernetes

Helm

Helm Charts helps you define, install, and upgrade even the most complex Kubernetes application



Helm

The best way to find, share, and use software built for Kubernetes

Manage complexity

Charts can describe complex apps; provide repeatable app installs, and serve as a single point of authority

Easy updates

Take the pain out of updates with in-place upgrades and custom hooks

Simple sharing

Charts are easy to version, share, and host on public or private servers

Rollbacks

Use `helm rollout` to roll back to an older version of a release with ease

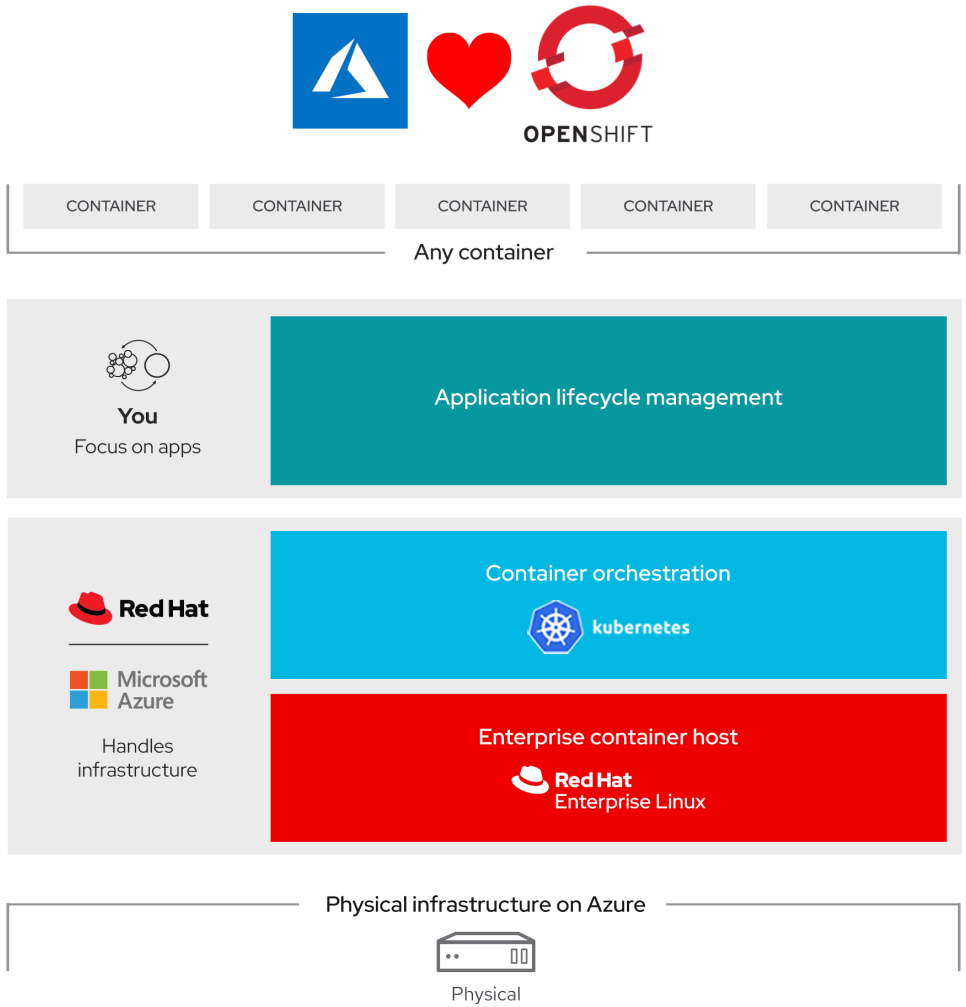
Monitoring

Many Choices

- Kubernetes Dashboard
- Azure Monitor for AKS
- 3rd Party Open Source Tools
  - Prometheus/Grafana
  - Heapster/InfluxDB/Grafana
  - Heapster/ELK Stack
  - Datadog/Dynatrace
  - New Relic

Exciting update: Azure Red Hat OpenShift

A fully managed service of Red Hat OpenShift on Azure, jointly, engineered, operated and supported by Microsoft and Red Hat



Session resources

- Official Kubernetes docs: https://kubernetes.io/docs/home/
- Learning path for k8s: https://aka.ms/learnkubernetes
- AKS hands on: https://aksworkshop.io/
- Helm 3.0: https://helm.sh/blog/helm-3-preview-pt7/
- AKS best practices: https://docs.microsoft.com/en-us/azure/aks/best-practices

Q&A

Thank you!