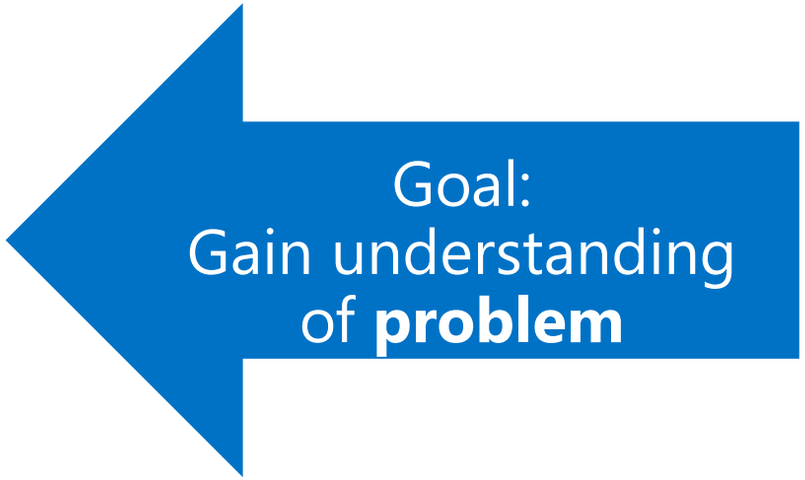

Azure
Kubernetes for Non-Kubernetes People
Daniel Meixner
Microsoft
@DanielMeixner

Agenda

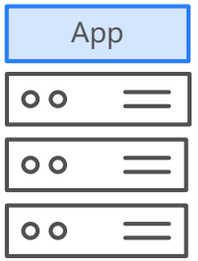
- What are Containers?
- Why do people use containers?
- Did you say Microservice?
- Which challenges are left?

- What is Kubernetes?
- Kubernetes Architecture
- Kubernetes Objects
  - Pod, Service, Tag, Deployments



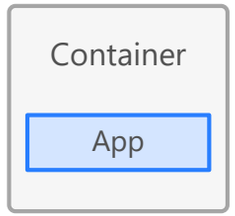
What is a container?

What is a **container**?



Virtual machines

- Virtualize the hardware
- VMs as units of scaling

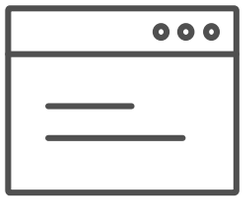


Containers

- Virtualize the operating system
- Applications as units of scaling

Why do people use containers?

What we hear from **developers**



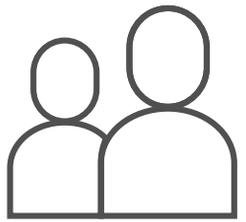
I need to create applications at a competitive rate without worrying about IT

New applications run smoothly on my machine but malfunction on traditional IT servers

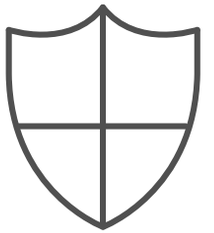
My productivity and application innovation become suspended when I have to wait on IT

What we hear from **IT**

I need to manage servers and maintain compliance with little disruption



I'm unsure of how to integrate unfamiliar applications, and I require help from developers



I'm unable to focus on both server protection and application compliance

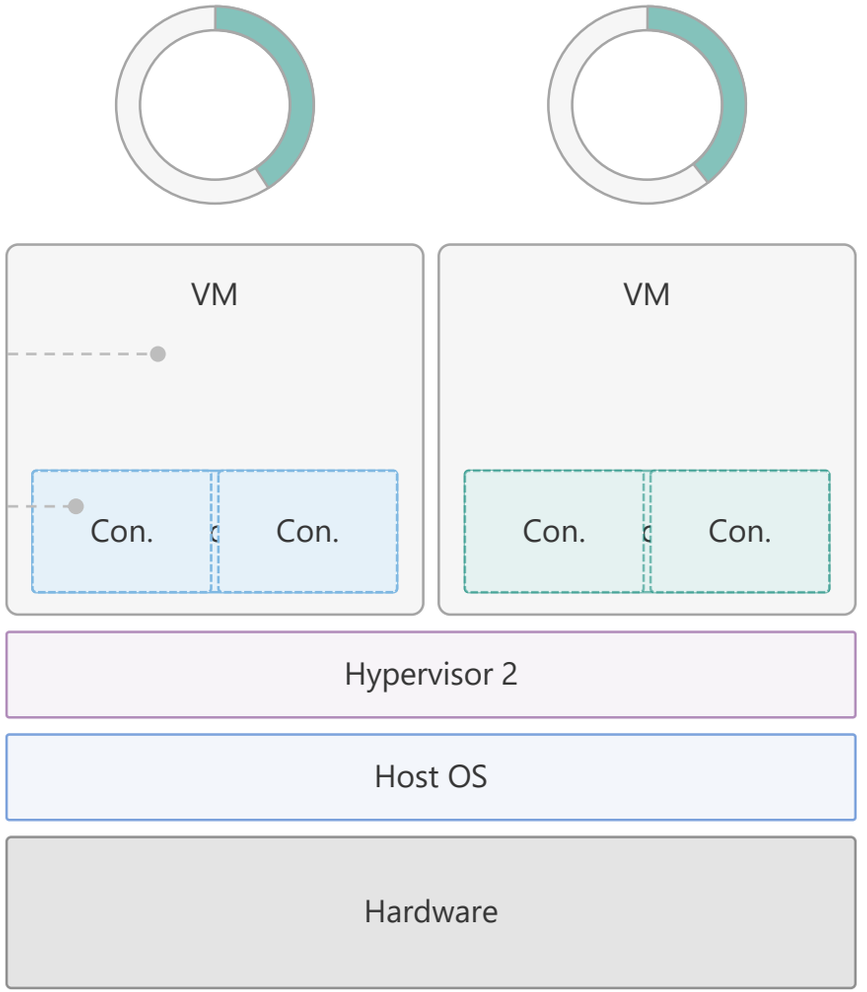
The container **advantage**

Traditional virtualized environment

Low utilization of container resources

Containerization of applications and their dependencies for portability

From dev to production agility across development and operations teams



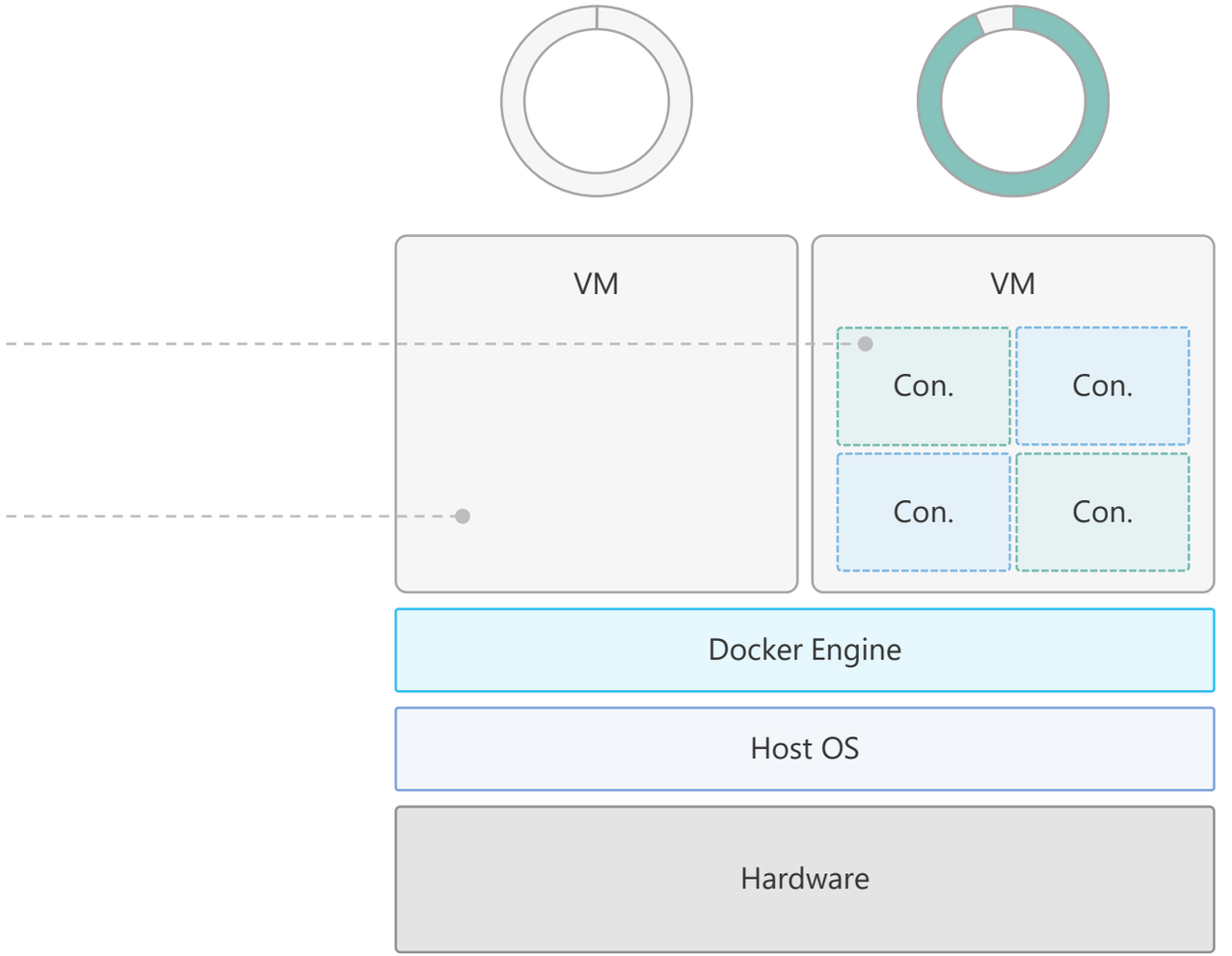
The container **advantage**

Containerized environment

Migrate containers and their dependencies to underutilized VMs for improved density and isolation

Decommission unused resources for efficiency gains and cost savings

Container is lighter weight and faster to scale dynamically



Did you say Microservice?

Microservices ≠ Containers

microservices is an architectural design approach

containers are an implementation detail that often helps

Which challenges are left?

The elements of **orchestration**

Scheduling

Affinity/anti-affinity

Health monitoring

Failover

Scaling

Networking

Service discovery

Coordinated app upgrades

What is Kubernetes?

Kubernetes is …

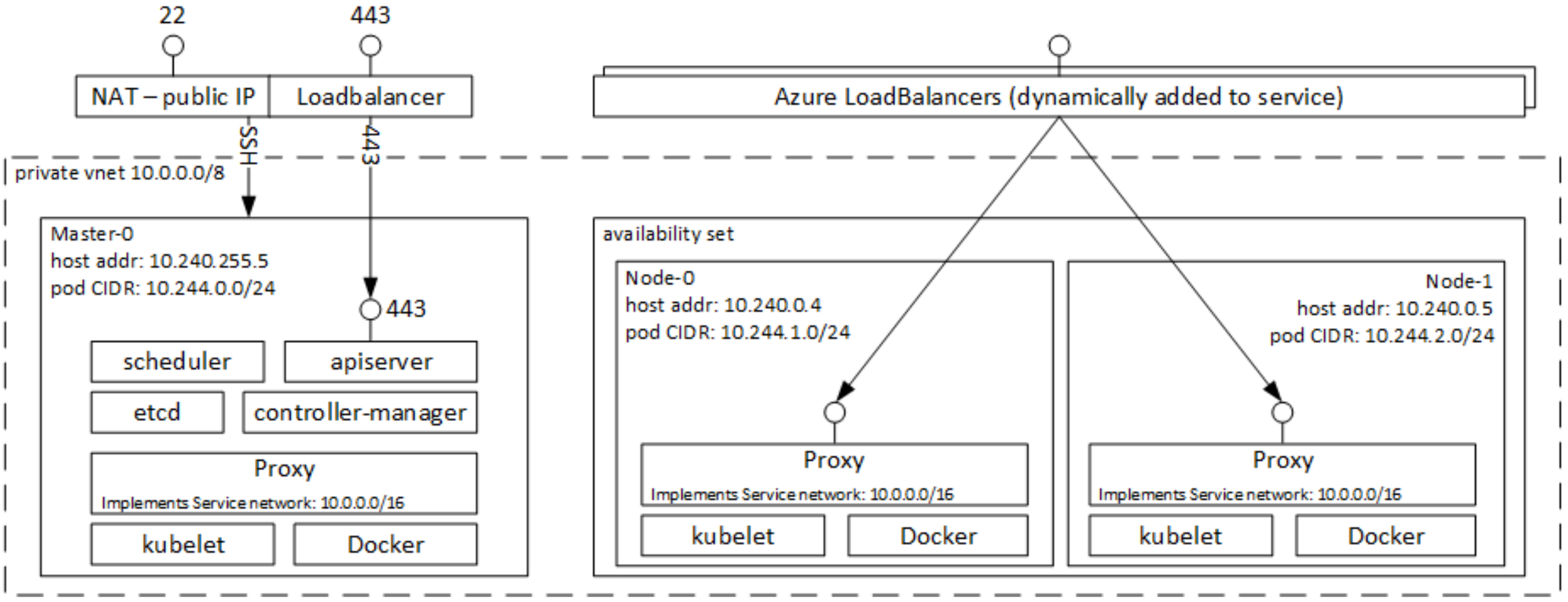
… **multiple pieces** of software
installed & configured
on **multiple machines**
providing orchestration of containers

(This is supposed to make our lifes better.)

Kubernetes löst die Probleme, die wir ohne Container nicht hätten.

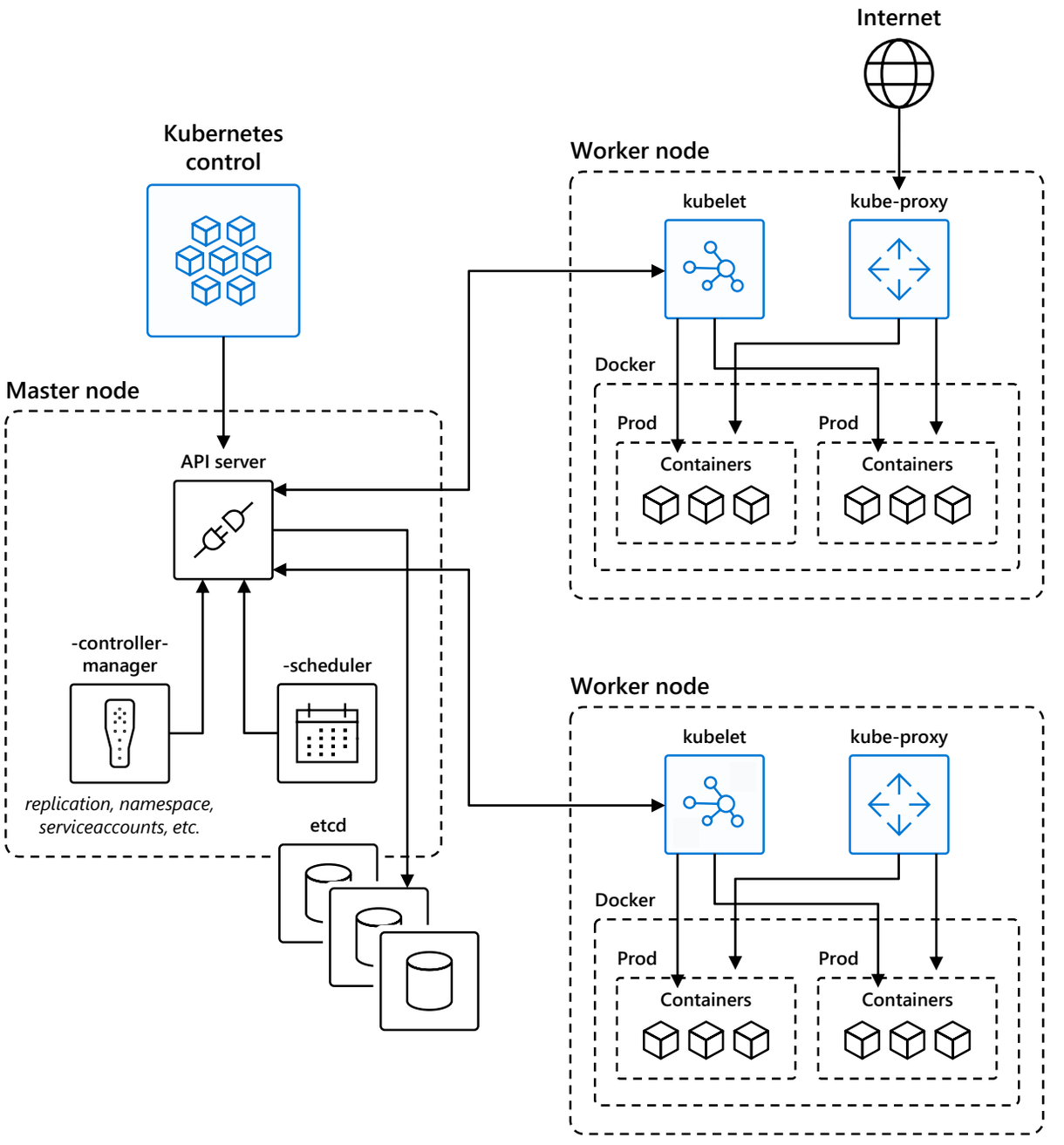
Dennis Zielke, 2017

Kubernetes Infrastrucure
(on Azure)



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



Kubernetes Essentials

Kubectl

- Command Line Interface to talk to master api endpoint

```
dmxonunix@DMXSurface:~$ kubectl
kubectl controls the Kubernetes cluster manager.

Find more information at: https://kubernetes.io/docs/reference/kubectl/overview/

Basic Commands (Beginner):
  create         Create a resource from a file or from stdin.
  expose         Take a replication controller, service, deployment or pod and expose it as a new Kubernetes Service
  run            Run a particular image on the cluster
  set            Set specific features on objects

Basic Commands (Intermediate):
  explain        Documentation of resources
  get            Display one or many resources
  edit           Edit a resource on the server
  delete         Delete resources by filenames, stdin, resources and names, or by resources and label selector
```

Kubernetes Object: Pod

- A Pod defines a **set of containers** that run on a host
- An application-specific “logical host”
- All containers inside a pod share a port-space

- Expect your pod to die anytime.

Definition of objects via *.yaml files

```yaml
apiVersion: "v1"
kind: Pod
metadata:
  name: helloworld
  labels:
    release: V76
    app: helloworld-demo
spec:
  containers:
    - name: aci-helloworld
      image: dzkubereg.azurecr.io/aci-helloworld-ci:latest
      ports:
        - containerPort: 80
      resources:
        requests:
          memory: "128Mi"
          cpu: "500m"
        limits:
          memory: "256Mi"
          cpu: "1000m"
```

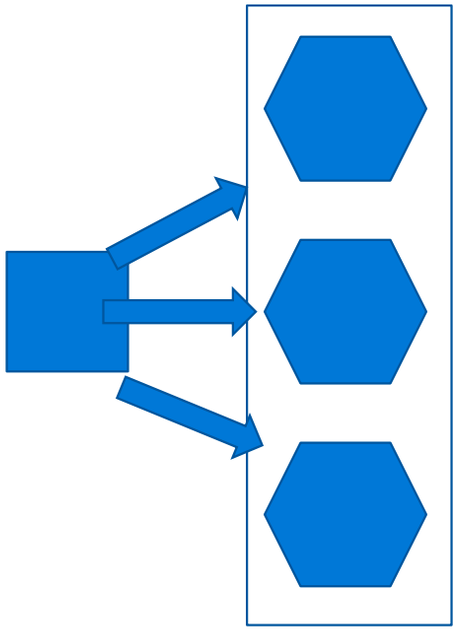
```
dmxonunix@DMXSurface:~$ kubectl apply -f mypod.yaml
```

Kubernetes Object: Service

- Abstraction which defines a logical set of pods

  Mortality of a single pod does **not** affect the availability of a service if the functionality of the service is provided by multiple (and not just one) pods

- Can be used as external interface with public IP



Kubernetes Object: Labels & Selectors

- Labels
  - *Identifying* attributes on kubernetes objects
  - Key/value based

```yaml
labels:
  name: calcbackend-pod
  environment: qa
```

Can be used for manual query and will be used by e.g. services

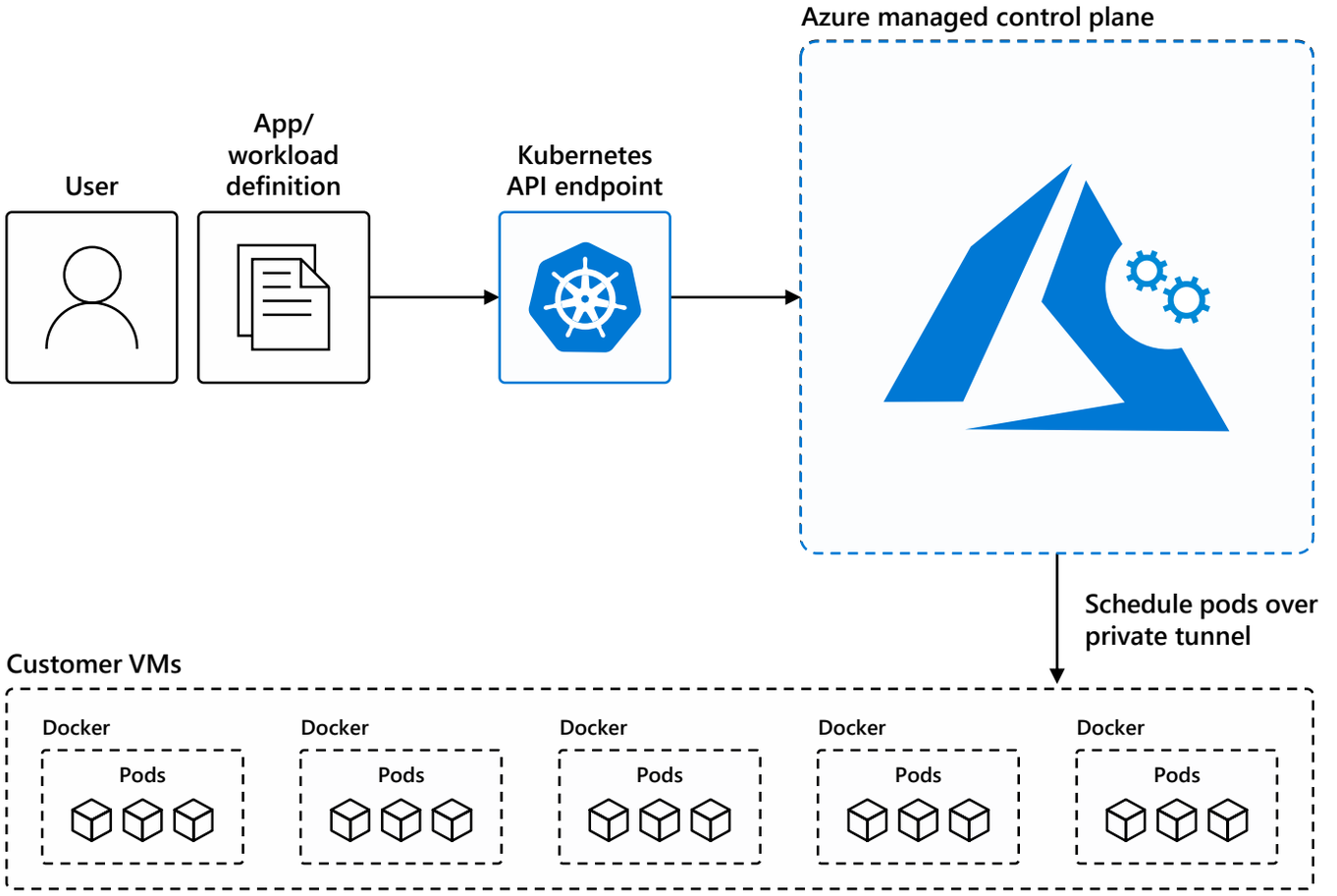
> *kubectl get pods -l environment=production,tier=frontend*

Kubernetes Objects: Deployments & ReplicaSets

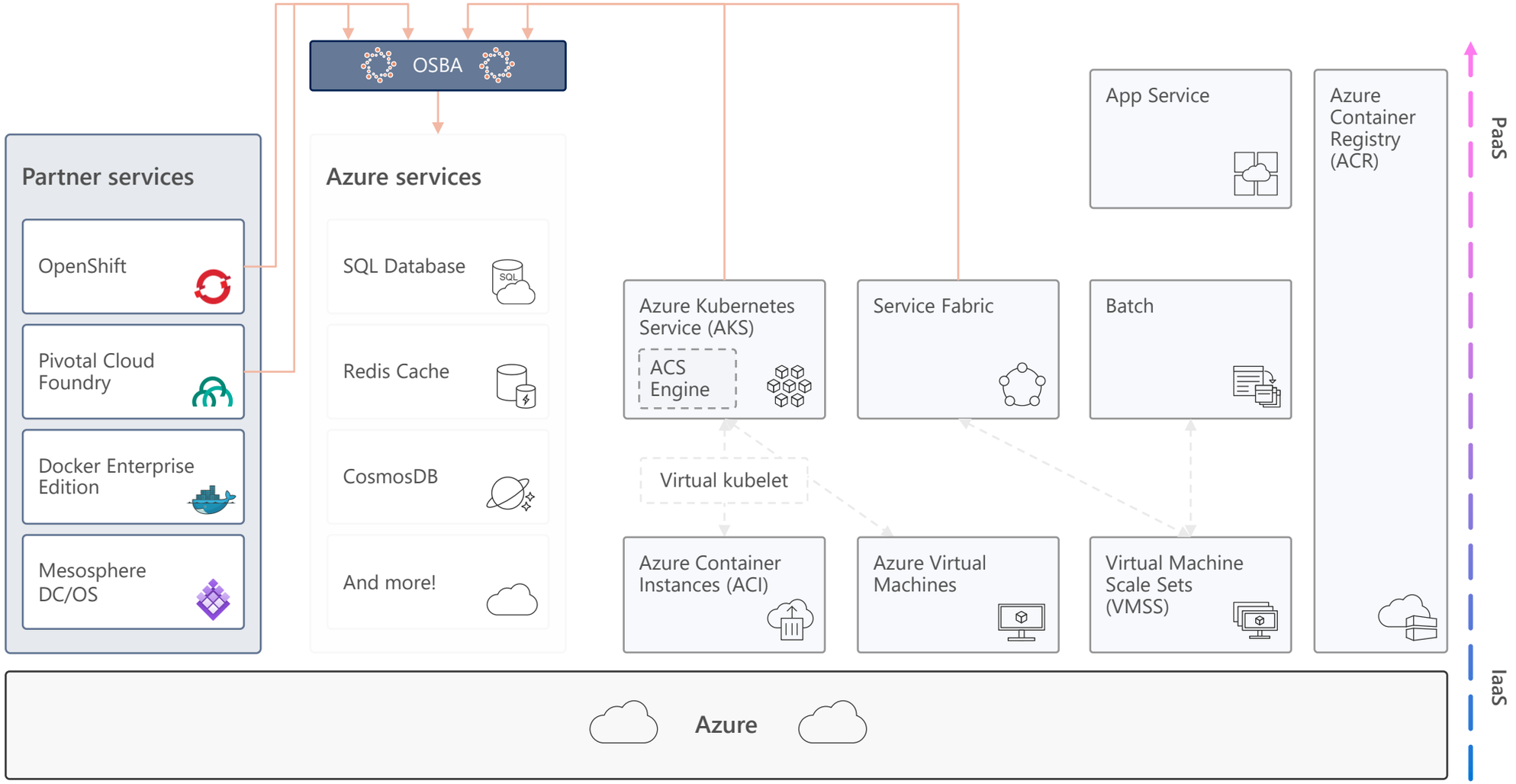
- Desired State Definition for Pods
- Wrap Pods into a ReplicaSet & Deployment to make Kubernetes ensure a specific number of Pods is always available

How managed Kubernetes on Azure works

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



Cloud Integration Open Service Broker



Package Management via Helm

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

Manage complexity

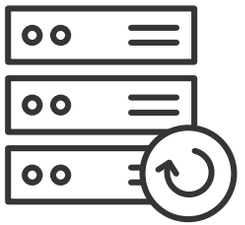
Charts can describe complex apps; provide repeatable app installs, and serve as a single point of authority

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

Microsoft

Microsoft