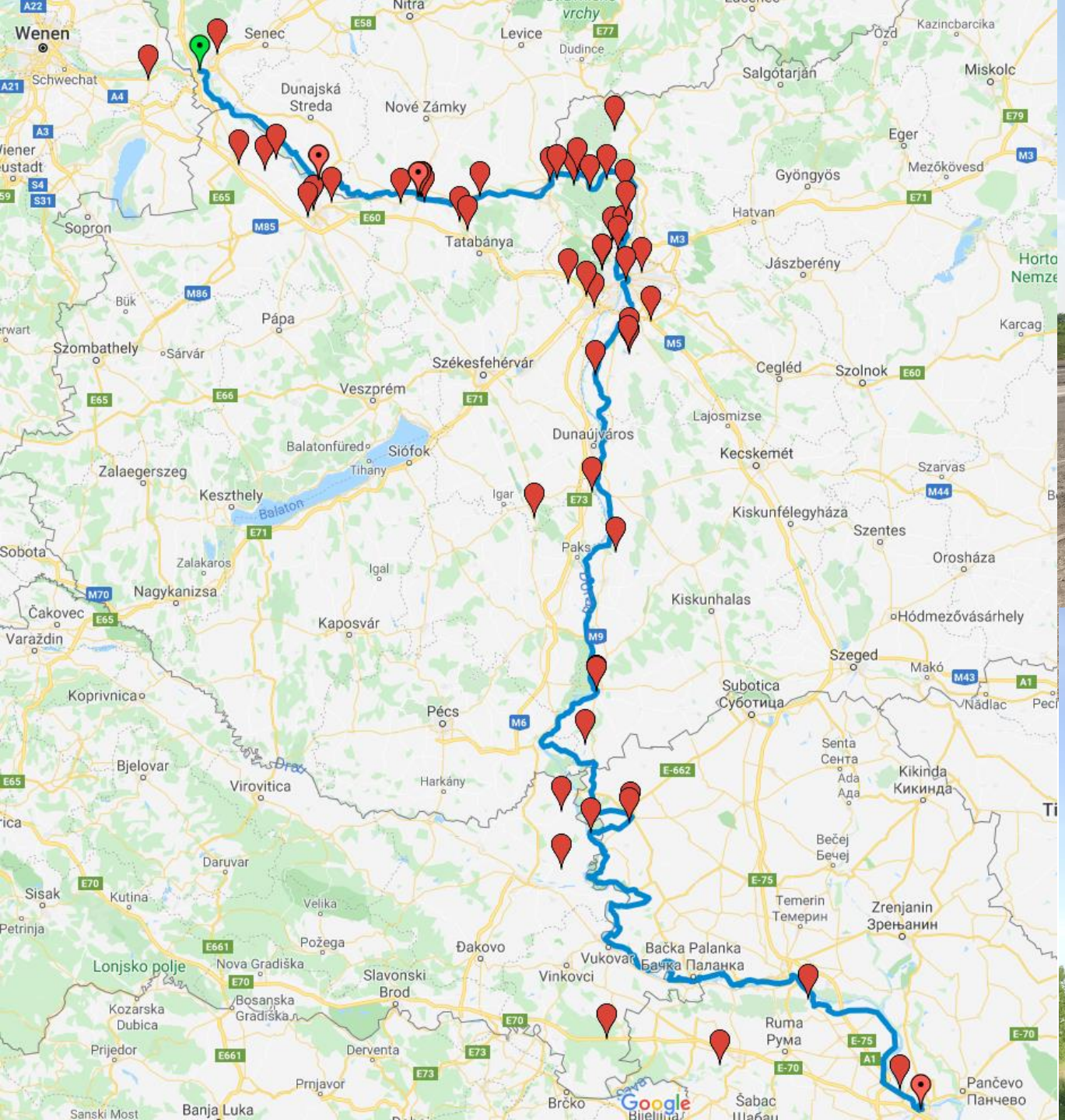
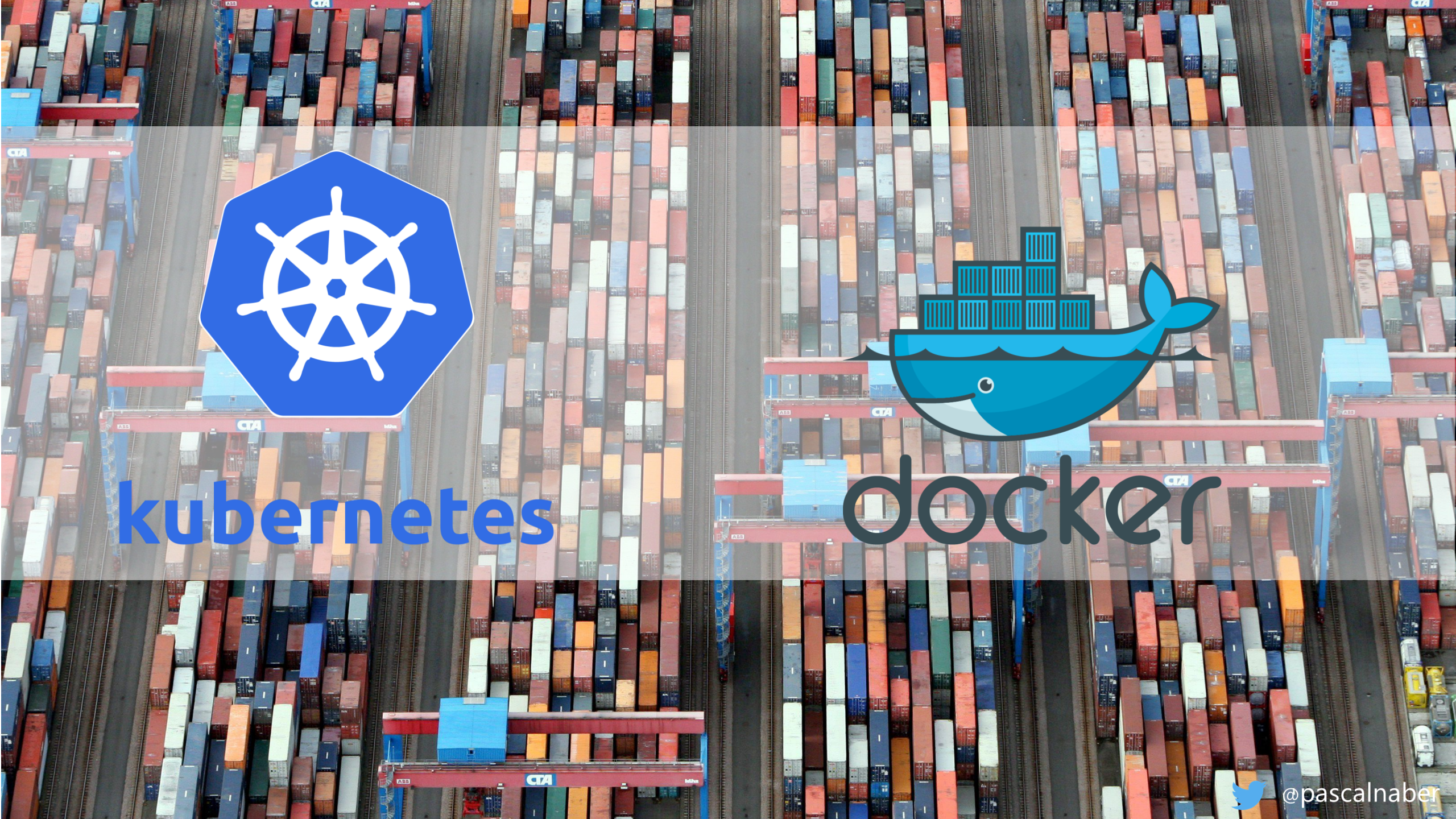

AzureSerbia
Introduction to Kubernetes using AKS
Pascal Naber
MVP
Microsoft
Most Valuable
Professional
Xpirit
MSC HOME TERMINAL
@pascalnaber


Wenen
Schwechat
Senec
Nitra
vrchy
Levice
Dudince
Kazincbarcika
Ózd
Salgótarján
Miskolc
Dunajská Streda
Nové Zámky
Eger
Gyöngyös
Mezőkövesd
Hatvan
Sopron
Tatabánya
Jászberény
Bük
Pápa
Karcag
Szombathely
Sárvár
Székesfehérvár
Cegléd
Szolnok
Veszprém
Lajosmizse
Dunaújváros
Balatonfüred
Siófok
Tihany
Kecskemét
Zalaegerszeg
Szarvas
Keszthely
Balaton
Igar
Kiskunfélegyháza
Szentes
Sobota
Paks
Orosháza
Zalakaros
Igal
Nagykanizsa
Kiskunhalas
Čakovec
Hódmezővásárhely
Varaždin
Kaposvár
Szeged
Makó
Koprivnica
Subotica
Суботица
Nădlac
Pécs
Senta
Сента
Ada
Ада
Kikinda
Кикинда
Bjelovar
Virovitica
Harkány
Drava
Вечеј
Бечеј
Daruvar
Temerin
Темерин
Zrenjanin
Зрењанин
Sisak
Kutina
Petrinja
Velika
Požega
Đakovo
Bačka Palanka
Бачка Паланка
Vukovar
Lonjsko polje
Nova Gradiška
Vinkovci
Slavonski Brod
Bosanska Gradiška
Kozarska Dubica
Ruma
Рума
Prijedor
Derventa
Pančevo
Панчево
Prnjavor
Brčko
Šabac
Шабац
Sanski Most
Banja Luka
Google


Република Србија
Republic of Serbia
Добродошли
Welcome


Welcome to Serbia
We wish you a nice ride and the wind at your back :)
The Danube cycling route in Serbia:
- total distance on main route (Bački Breg-Bregovo): 665km
- web guide: www.danube-info.org
Bački Breg
32
Sombor
64
Bogojevo
87
47
Novi Sad
Bačka Palanka
96
Beograd
69
Kovin
81
Golubac
76
Donji Milanovac
62
Kladovo
55
Negotin
14
Bregovo
gtz


kubernetes
docker
@pascalnaber

WHY I LIKE Containers

@pascalnaber

They are FAST


They are PORTABLE
@pascalnaber

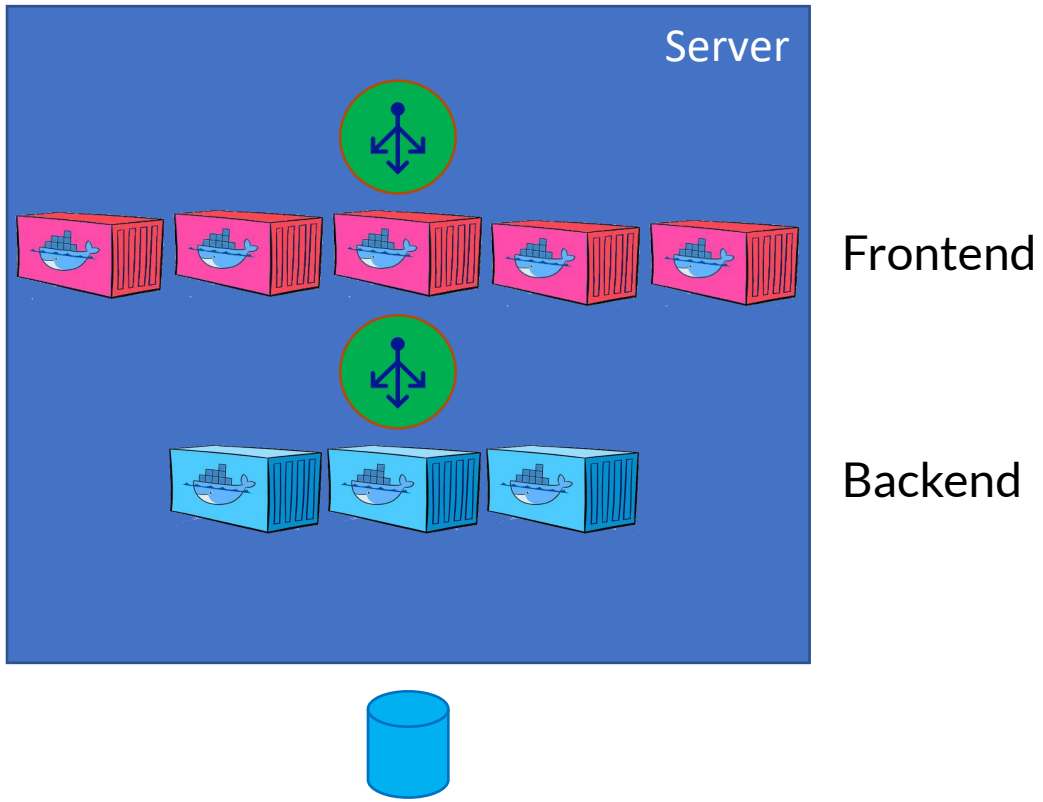
They are ISOLATED

@pascalnaber

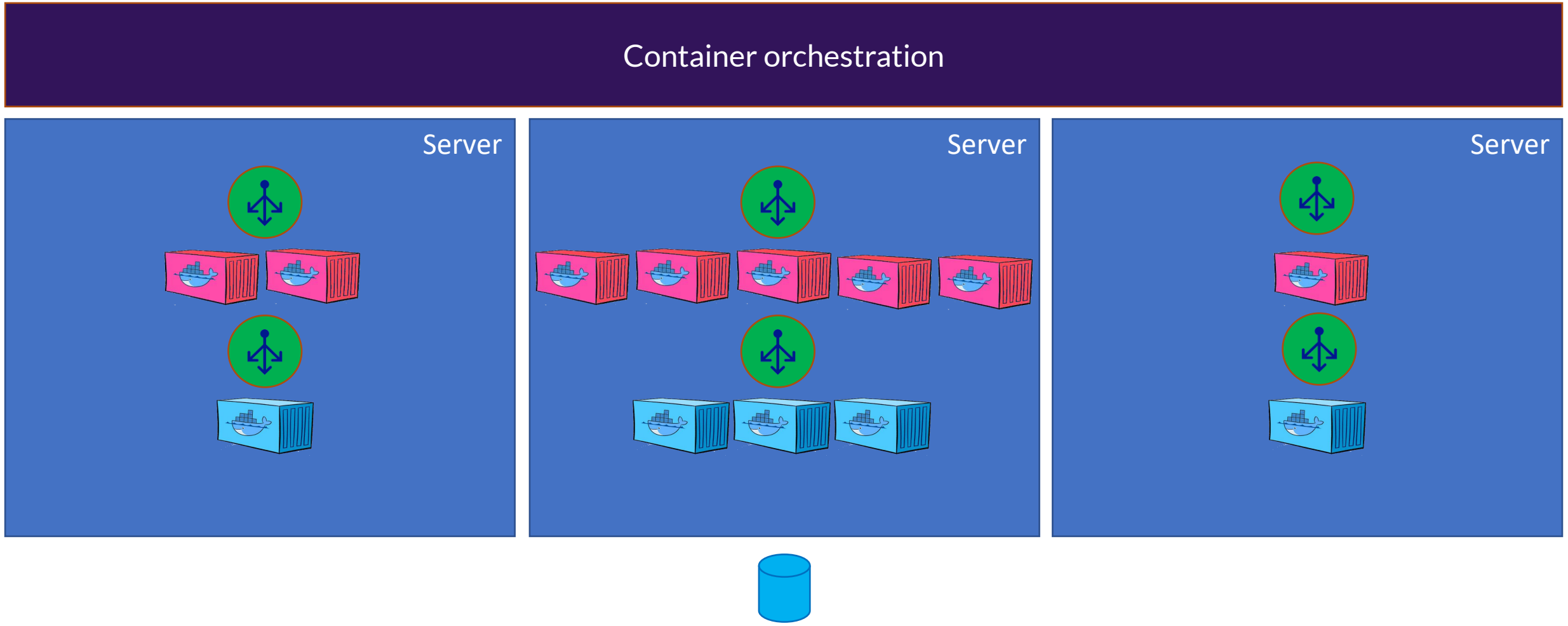
Moving to production

Scaling, load balancing, fault tolerance

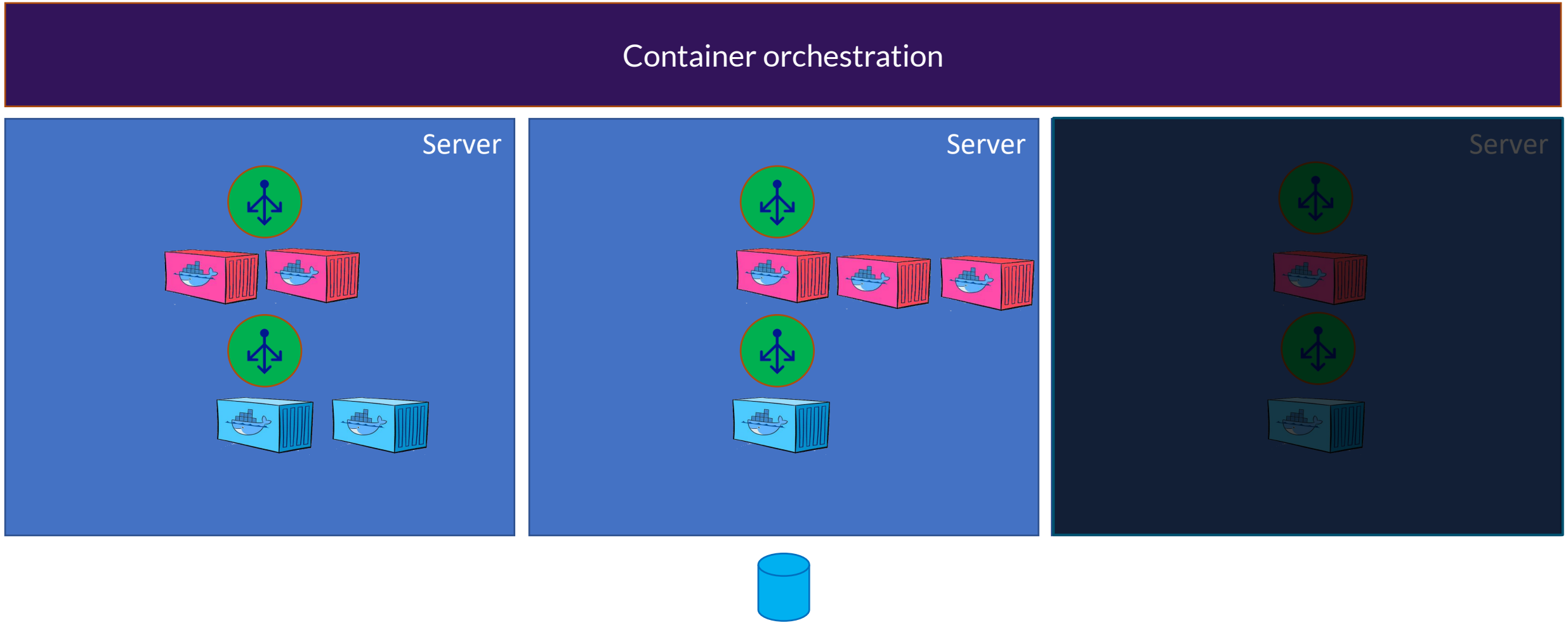
Scaling, Load balancing & Fault tolerance



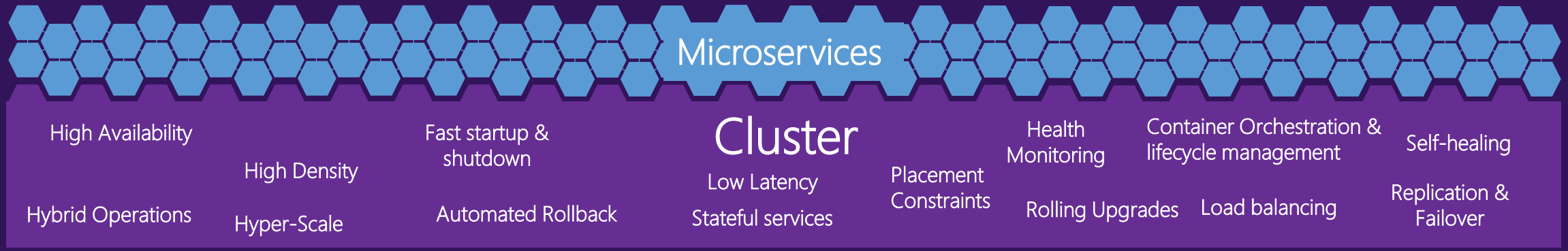
Scaling, Load balancing & Fault tolerance



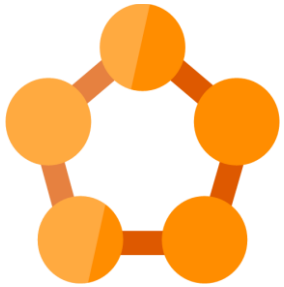
Scaling, Load balancing & Fault tolerance



Container orchestrators

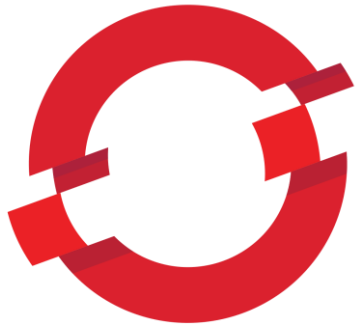


DC/OS



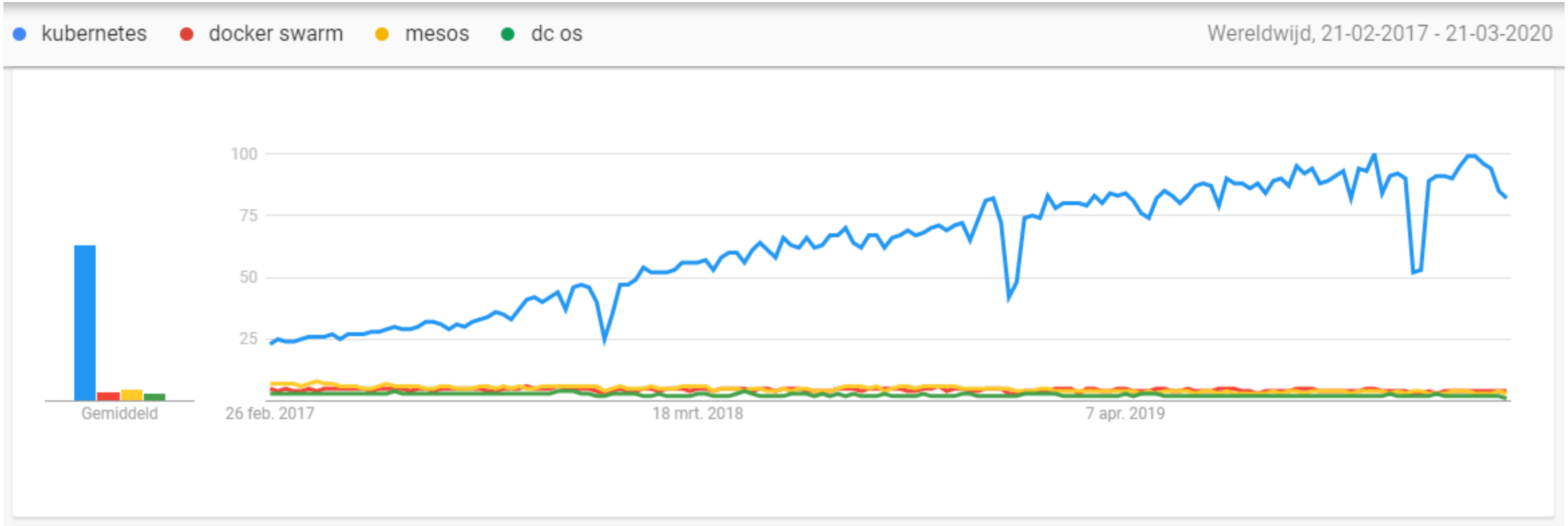
Service Fabric

kubernetes

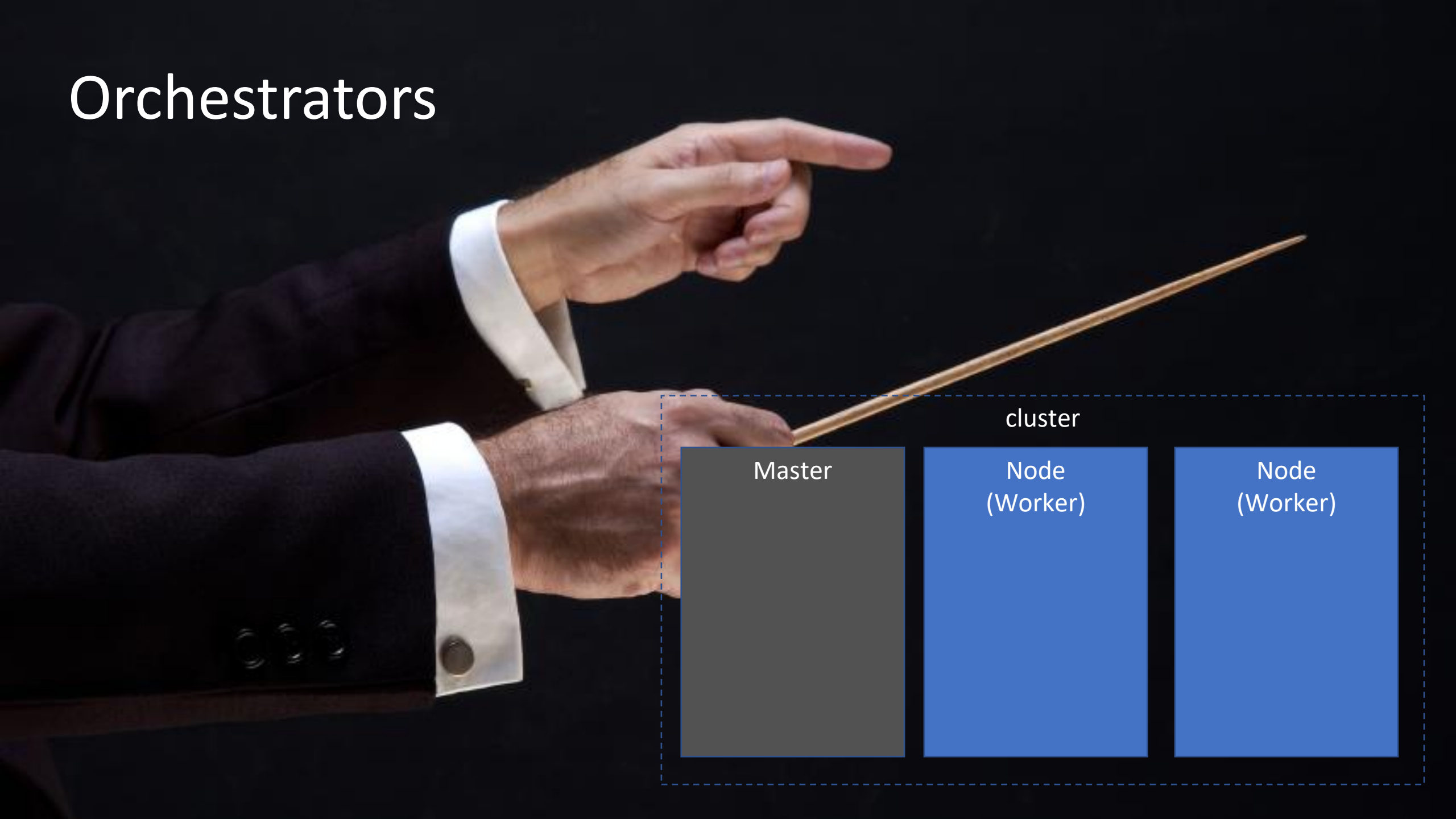


OPENSHIFT

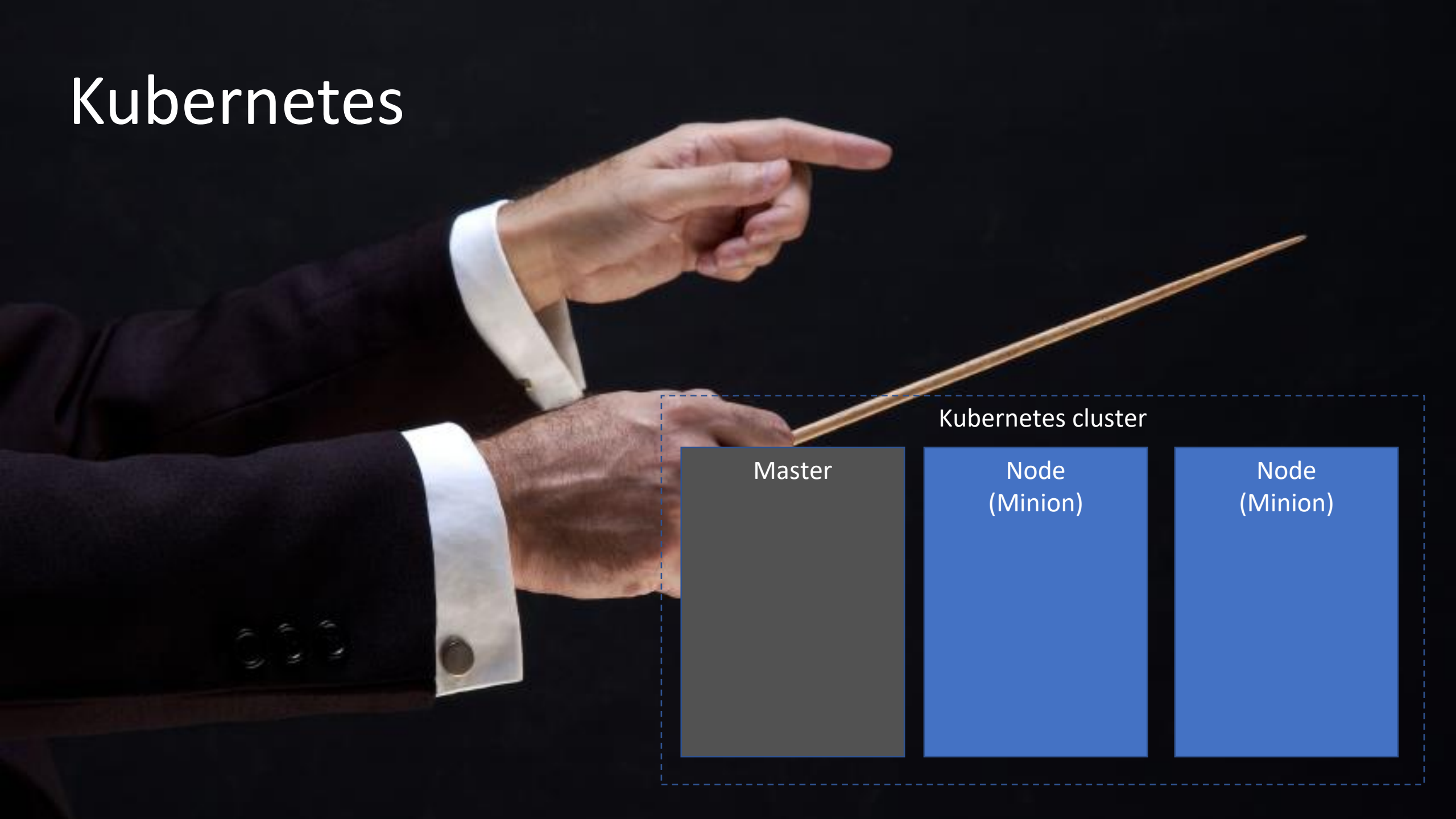
Google Trend



Orchestrators

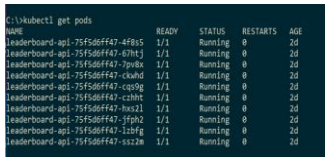


Kubernetes

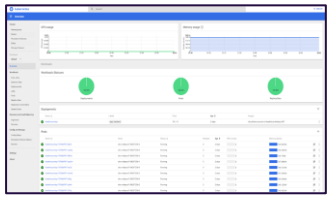


Kubernetes architecture

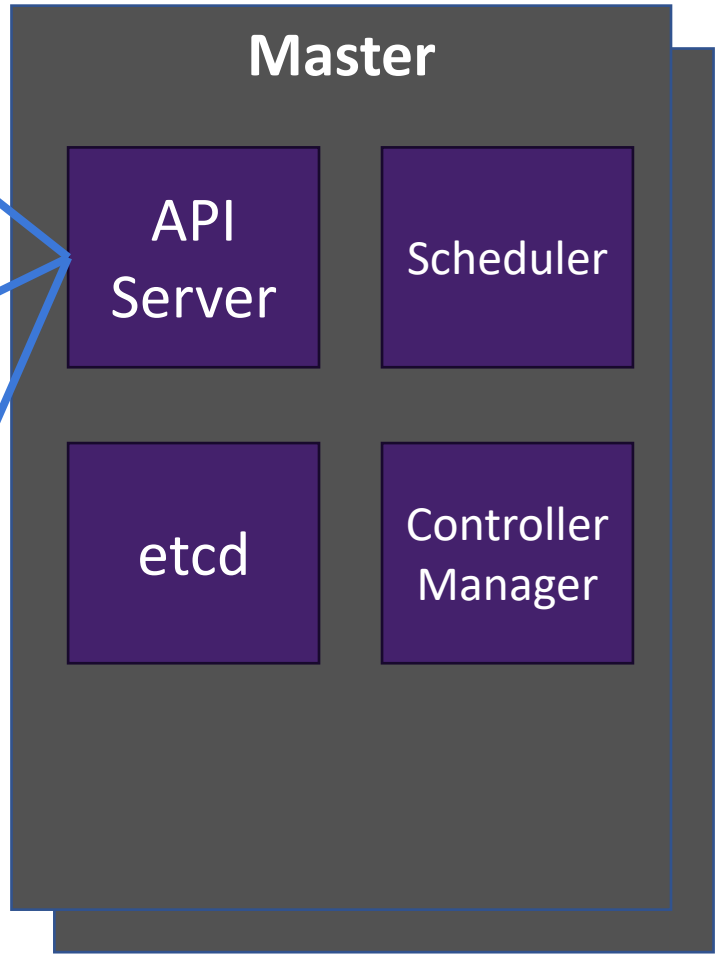
Kubectl
(CLI)

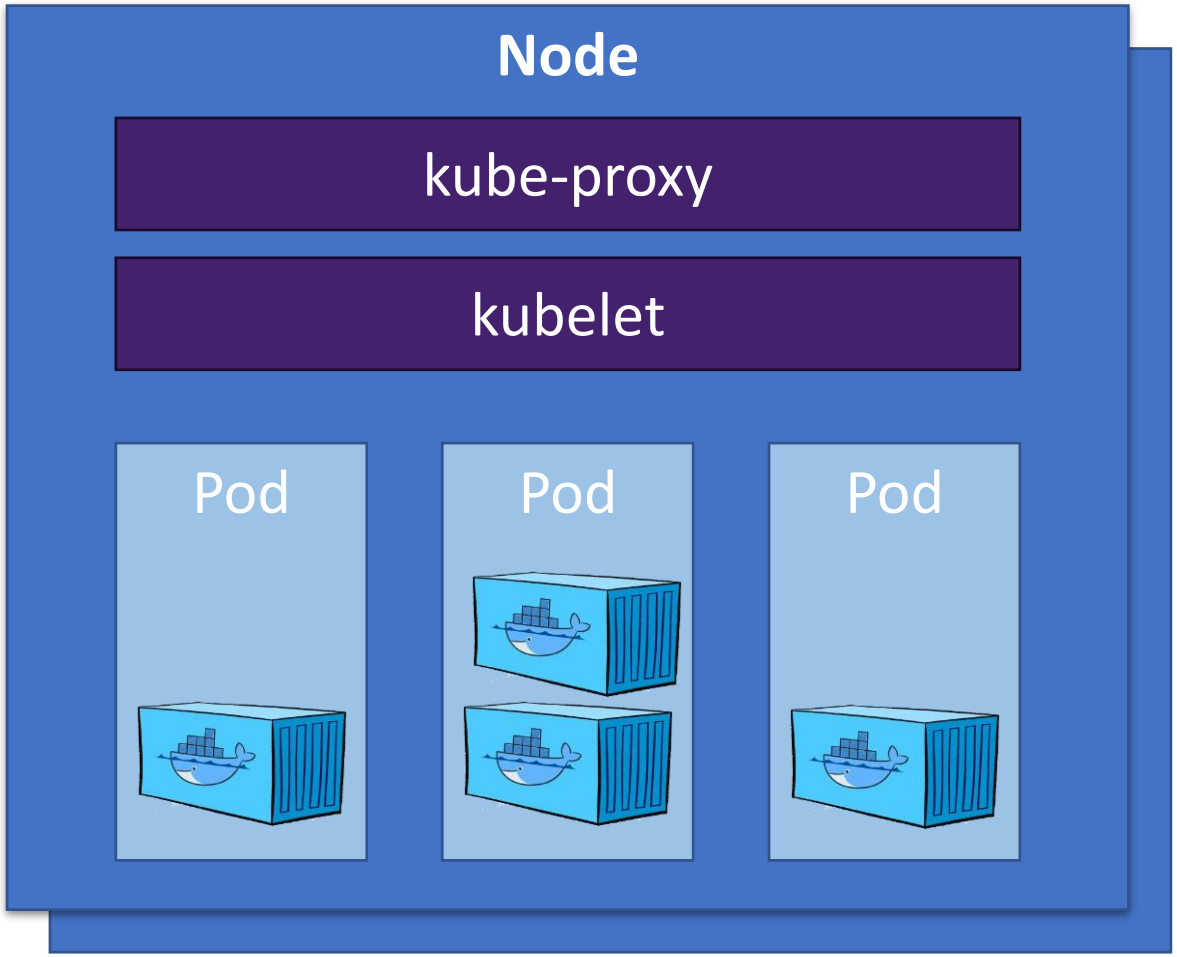


Dashboard
(UI)



REST client
(Code)





History of Kubernetes

First commit

June 2014

September 2014

Announced

Kubernetes v1

July 2015

March 2018

CNCF 1st Graduate

Google

CLOUD NATIVE COMPUTING FOUNDATION

@pascalnaber

Kubernetes

1 2 3 4 5 6 7 8

k8s

OOS

Extensible

Large, rapidly growing ecosystem

Facilitates declarative configuration and automation

Κυβερνήτης -- Greek: meaning helmsman or *pilot*

Hosting of Kubernetes

Local

- Minikube
- Docker for Windows
- Docker for Mac

Raspberry Pi

Private datacenter

Public Cloud

GKE (26-8-2015)

EKS (5-6-2018)

ACS (19-4-2016)
AKS (13-6-2018)

Azure Kubernetes Service (AKS)

AKS

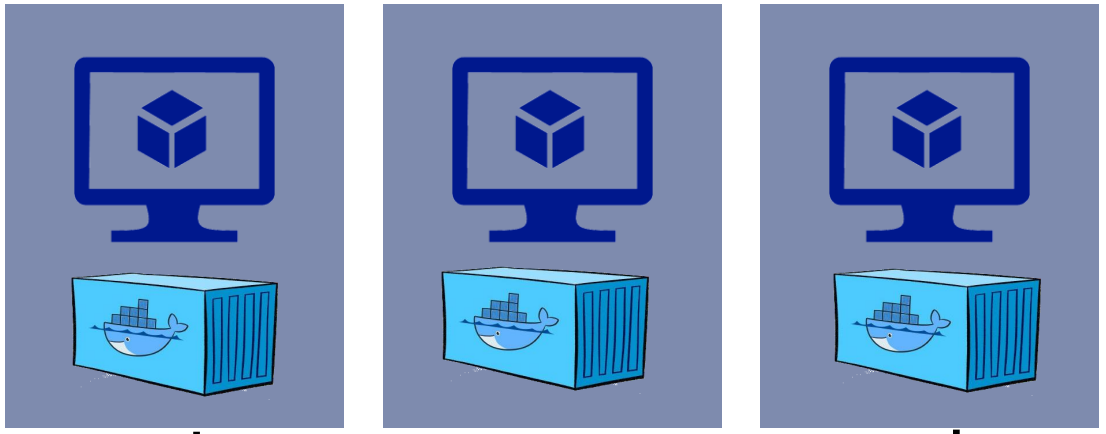
100% managed by Microsoft

€ 0

master master master

IaaS managed by Microsoft

€ … (VM pricing)

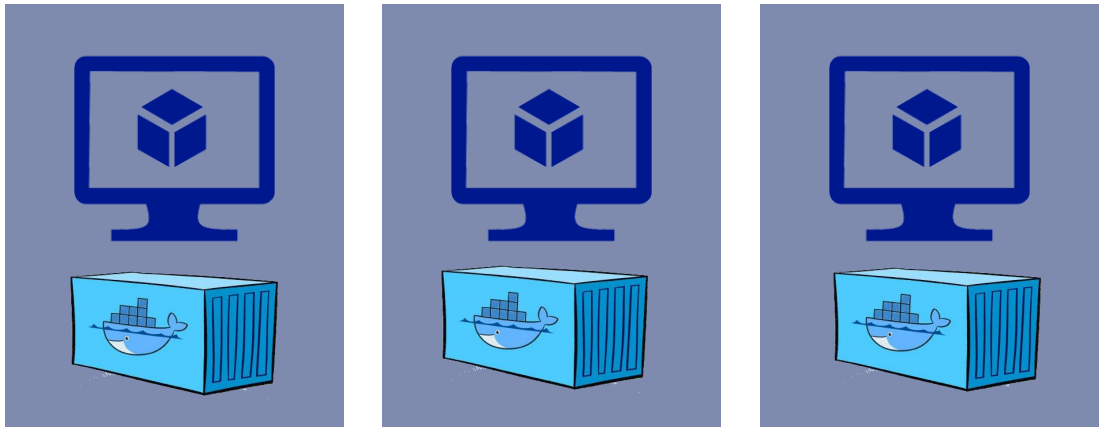


worker worker worker

Nodepools

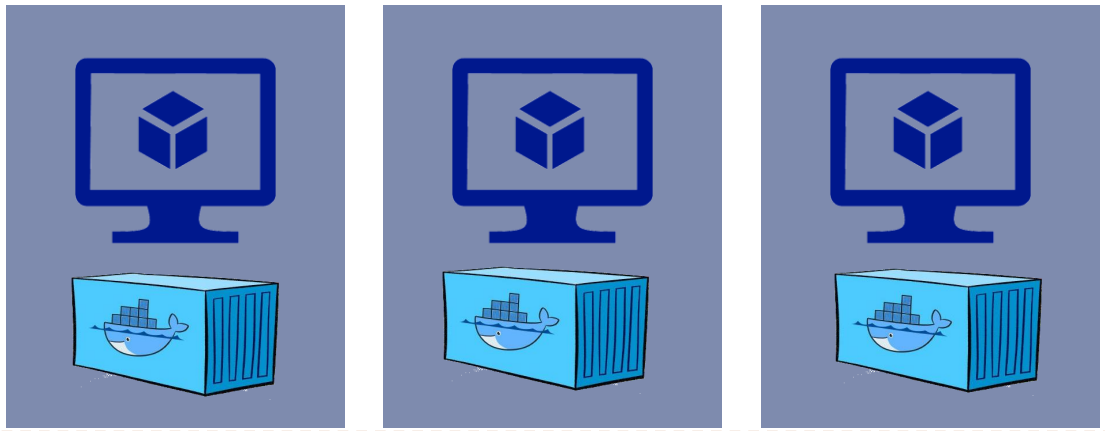
NodePool: Linux | k8s 1.15.7

Standard_DS2_v2



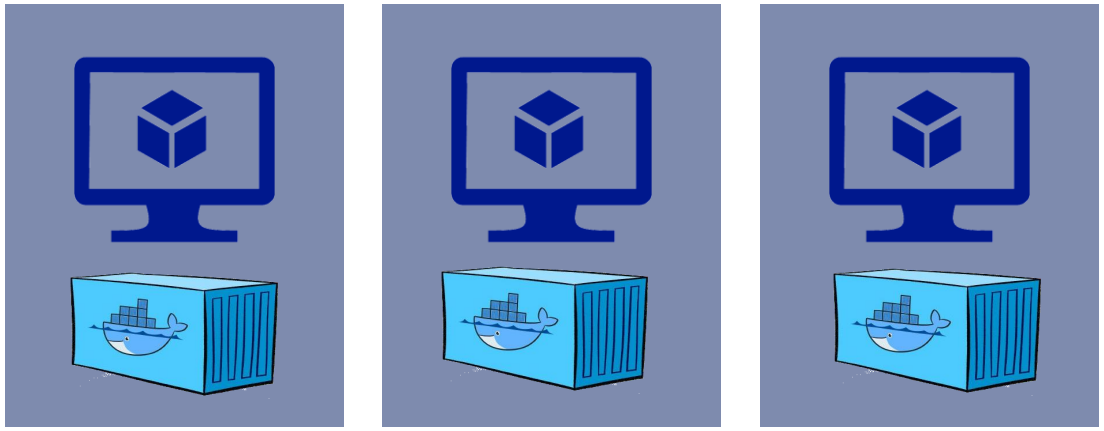
NodePool: Linux | k8s 1.16.7

Standard_NC6 (GPU optimized)



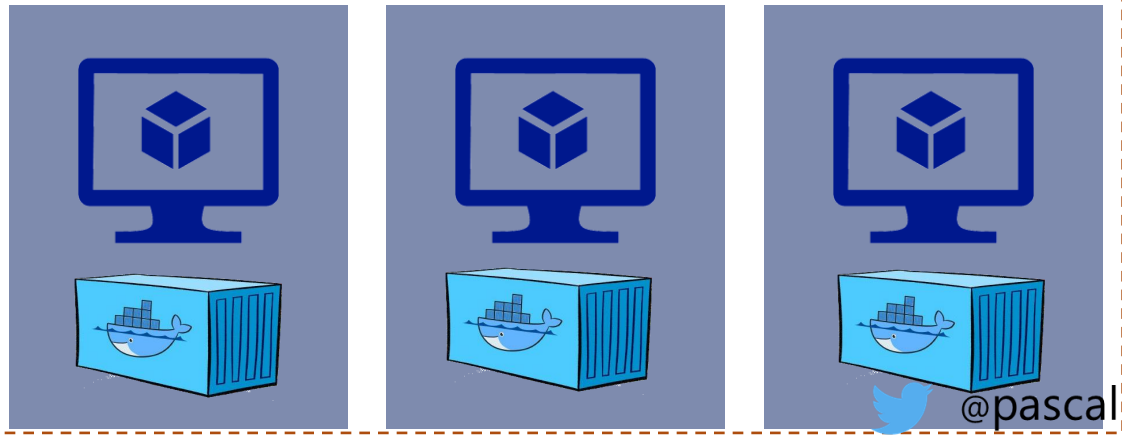
NodePool: Linux | k8s 1.16.7

Standard_DS2_v2

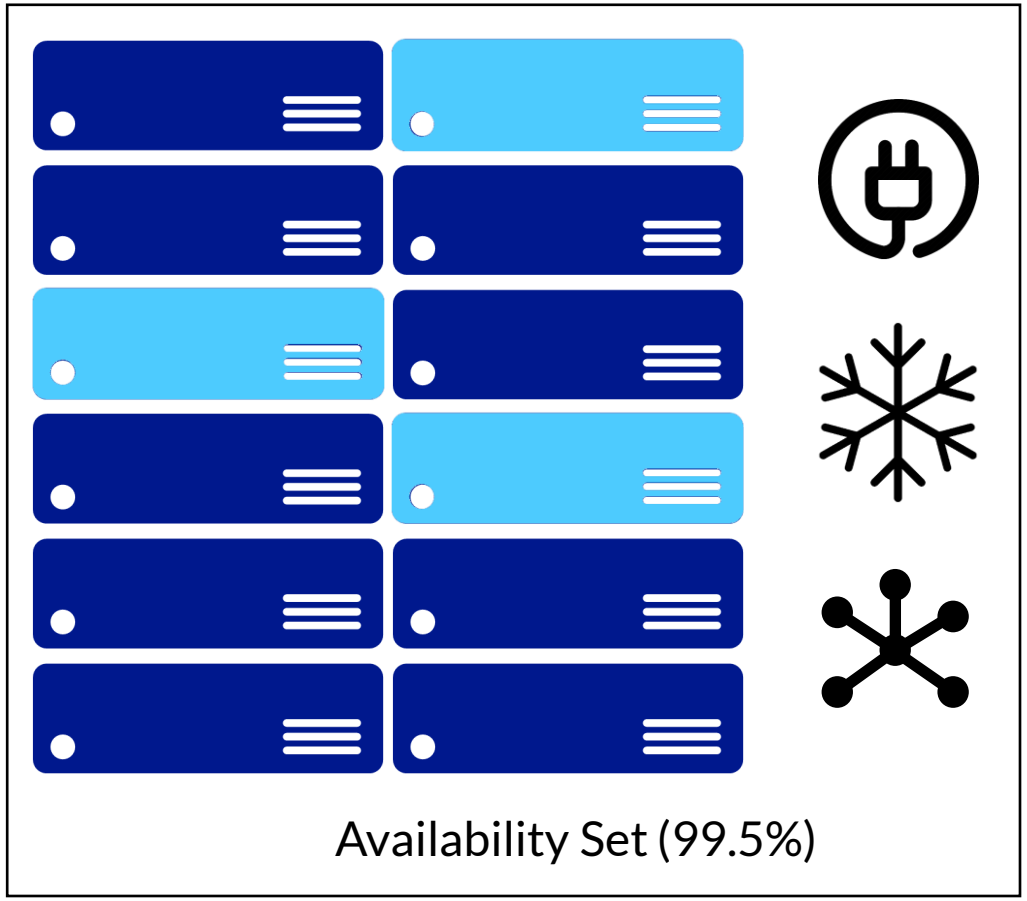


NodePool: Windows | k8s 1.16.7

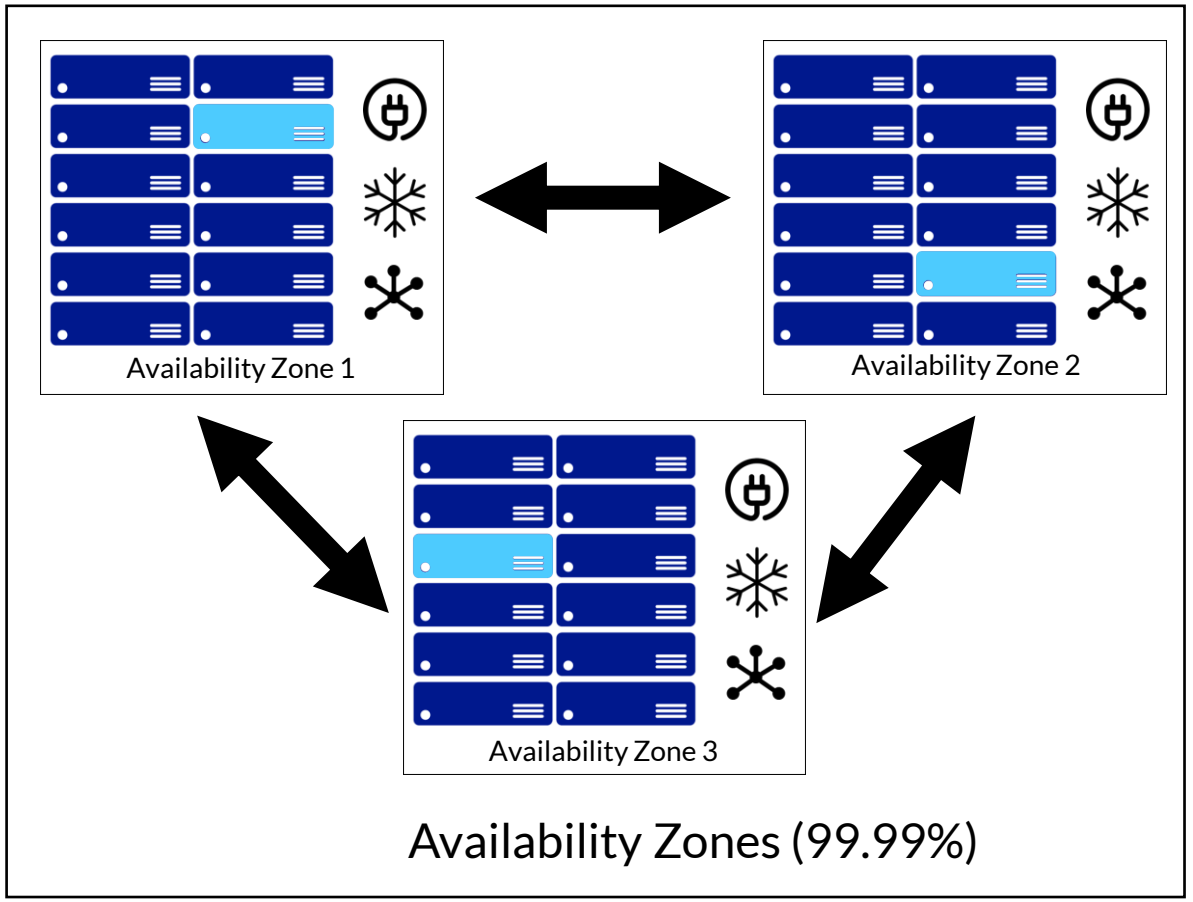
Standard_DS2_v3



Availability zones

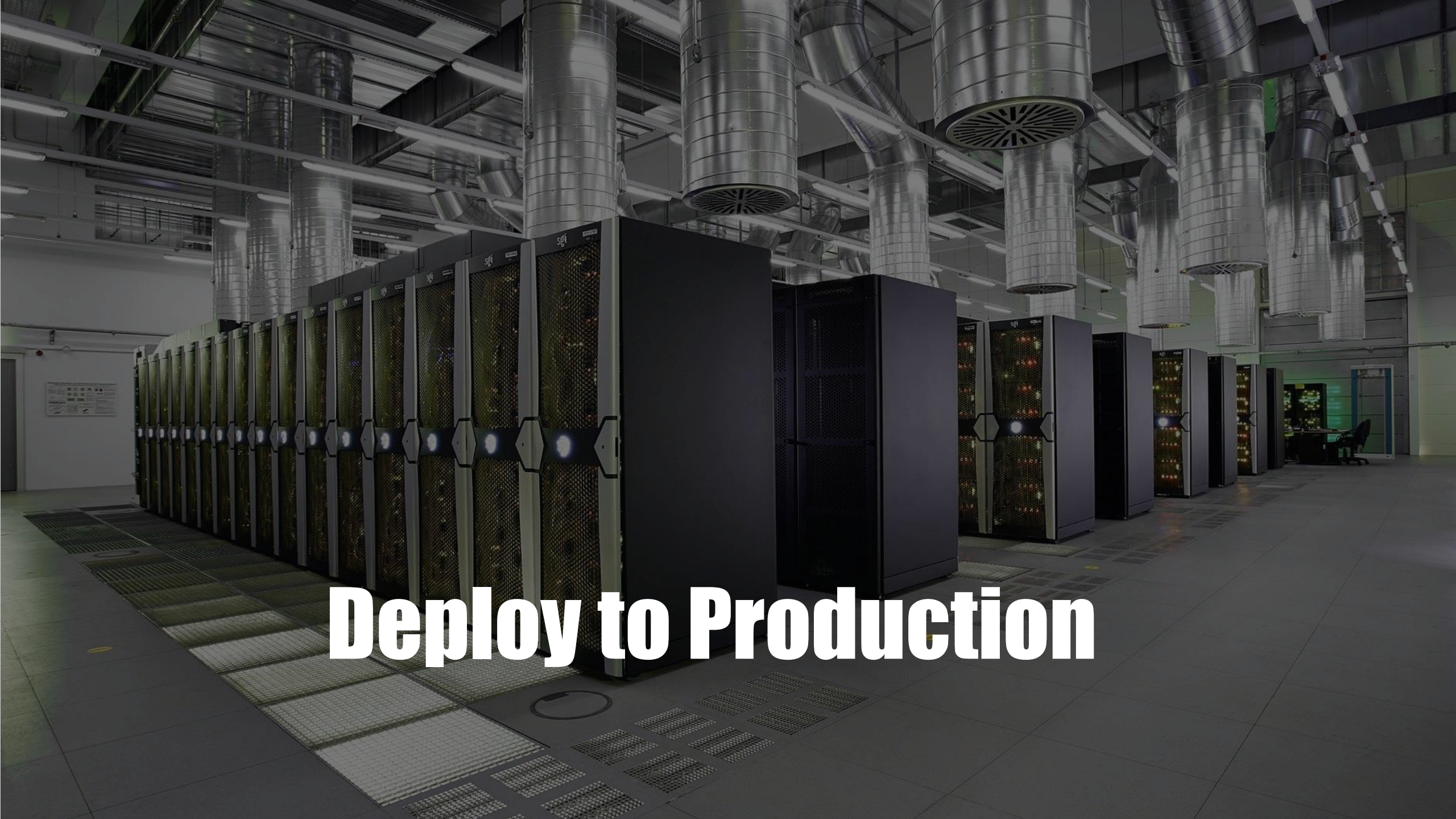


Azure region



€0.009 per GB inbound & outbound traffic

@pascalnaber



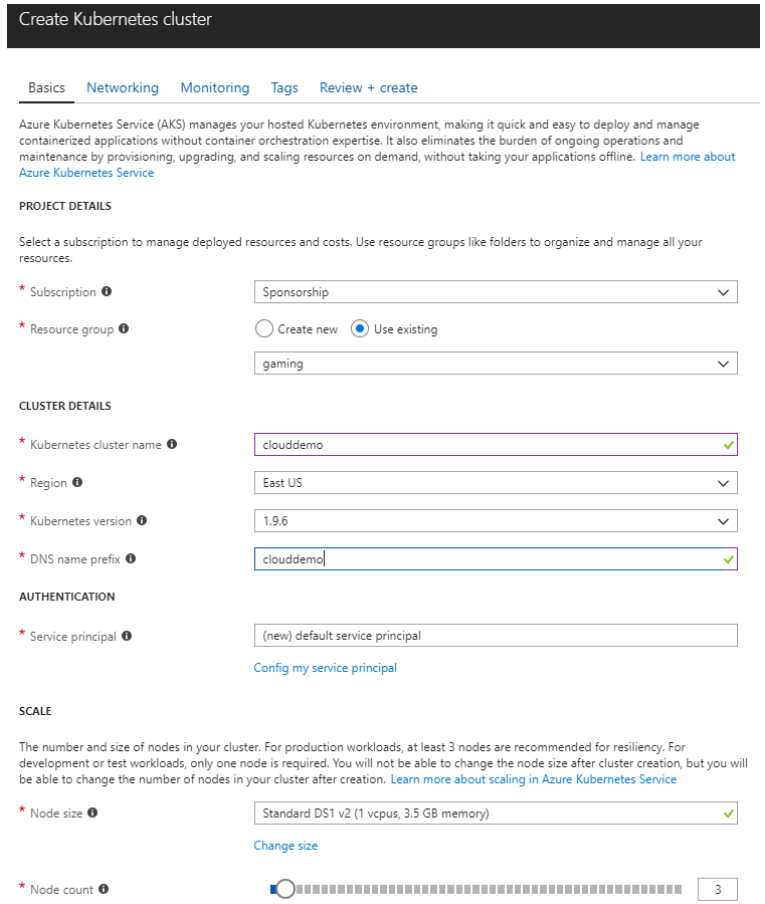
Deploy to Production

Install AKS

ARM

```
"resources": [
    {
        "apiVersion": "2017-08-31",
        "type": "Microsoft.ContainerService/managedClusters",
        "location": "westeurope",
        "name": "[variables('clusterName')]",
        "tags": {
            "department": "[parameters('metadata').department]",
            "projectName": "[parameters('metadata').projectName]",
            "owner": "[parameters('metadata').owner]",
            "environment": "[parameters('metadata').environment]"
        },
        "properties": {
            "kubernetesVersion": "[variables('kubernetesVersion')]",
            "dnsPrefix": "[variables('dnsPrefix')]",
            "agentPoolProfiles": [
                {
                    "name": "agentpool",
                    "osDiskSizeGB": "[variables('osDiskSizeGB')]",
                    "count": "[variables('agentCount')]",
                    "vmSize": "[variables('agentVMSize')]",
                    "osType": "[variables('osType')]",
                    "storageProfile": "ManagedDisks"
                }
            ],
            "linuxProfile": {
                "adminUsername": "[variables('adminUsername')]",
                "ssh": {
                    "publicKeys": [
                        {
                            "keyData": "[variables('sshRSAPublicKey')]"
                        }
                    ]
                }
            },
            "servicePrincipalProfile": {
                "ClientId": "[parameters('servicePrincipalClientId')]",
                "Secret": "[parameters('servicePrincipalClientSecret')]"
            }
        }
    }
],
```

Portal



Ansible

```
- name: Create Azure Kubernetes Service
  hosts: localhost
  connection: local
  vars:
    resource_group: myResourceGroup
    location: eastus
    aks_name: myAKSCluster
    username: azureuser
    ssh_key: "your_ssh_key"
    client_id: "your_client_id"
    client_secret: "your_client_secret"
  tasks:
  - name: Create resource group
    azure_rm_resourcegroup:
      name: "{{ resource_group }}"
      location: "{{ location }}"
  - name: Create a managed Azure Container Ser
    azure_rm_aks:
      name: "{{ aks_name }}"
      location: "{{ location }}"
      resource_group: "{{ resource_group }}"
      dns_prefix: "{{ aks_name }}"
      linux_profile:
        admin_username: "{{ username }}"
        ssh_key: "{{ ssh_key }}"
      service_principal:
        client_id: "{{ client_id }}"
        client_secret: "{{ client_secret }}"
      agent_pool_profiles:
        - name: default
          count: 2
          vm_size: Standard_D2_v2
      tags:
        Environment: Production
```

Terraform

```
resource "azurerm_resource_group" "k8s" {
    name     = "${var.resource_group_name}"
    location = "${var.location}"
}

resource "azurerm_kubernetes_cluster" "k8s" {
    name                = "${var.cluster_name}"
    location            = "${azurerm_resource_group.k8s.location}"
    resource_group_name = "${azurerm_resource_group.k8s.name}"
    dns_prefix          = "${var.dns_prefix}"

    linux_profile {
        admin_username = "ubuntu"

        ssh_key {
        key_data = "${file("${var.ssh_public_key}")}"
        }
    }

    agent_pool_profile {
        name            = "default"
        count           = "${var.agent_count}"
        vm_size         = "Standard_D2"
        os_type         = "Linux"
        os_disk_size_gb = 30
    }

    service_principal {
        client_id     = "${var.client_id}"
        client_secret = "${var.client_secret}"
    }

    tags {
        Environment = "Development"
    }
}
```

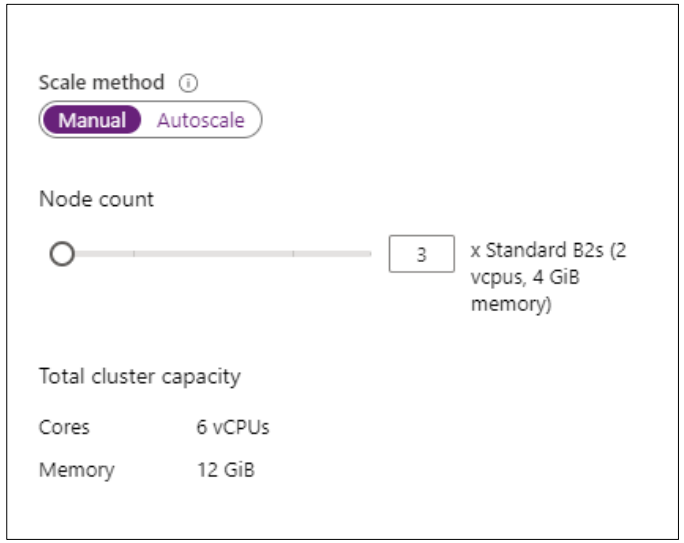
```
## Create Kubernetes cluster
az aks create --resource-group communitylive
              --name aksdemo
              --enable-managed-identity
              --kubernetes-version 1.16.7
              --node-vm-size Standard_DS2_v2
              --node-count 3
              --enable-addons monitoring
              --max-pods $AKS_MAX_PODS_PER_NODE
              --ssh-key-value "C:\repos\pascal\ssh\public.pub"
              --load-balancer-sku standard
              --network-plugin azure
              --vm-set-type VirtualMachineScaleSets
              --vnet-subnet-id $AKSSUBNETID
              --docker-bridge-address $AKS_NETWORKING_DOCKER_BRIDGE_ADDRESS
              --dns-service-ip $AKS_NETWORKING_DNS_SERVICE_IP
              --service-cidr $AKS_NETWORKING_SERVICE_CIDR
              --aad-server-app-id $AADSERVERAPPID
              --aad-server-app-secret $AADSERVERAPPSECRET
              --aad-client-app-id $AADCLIENTAPPID
              --aad-tenant-id $TENANTID
```

Azure resources for AKS

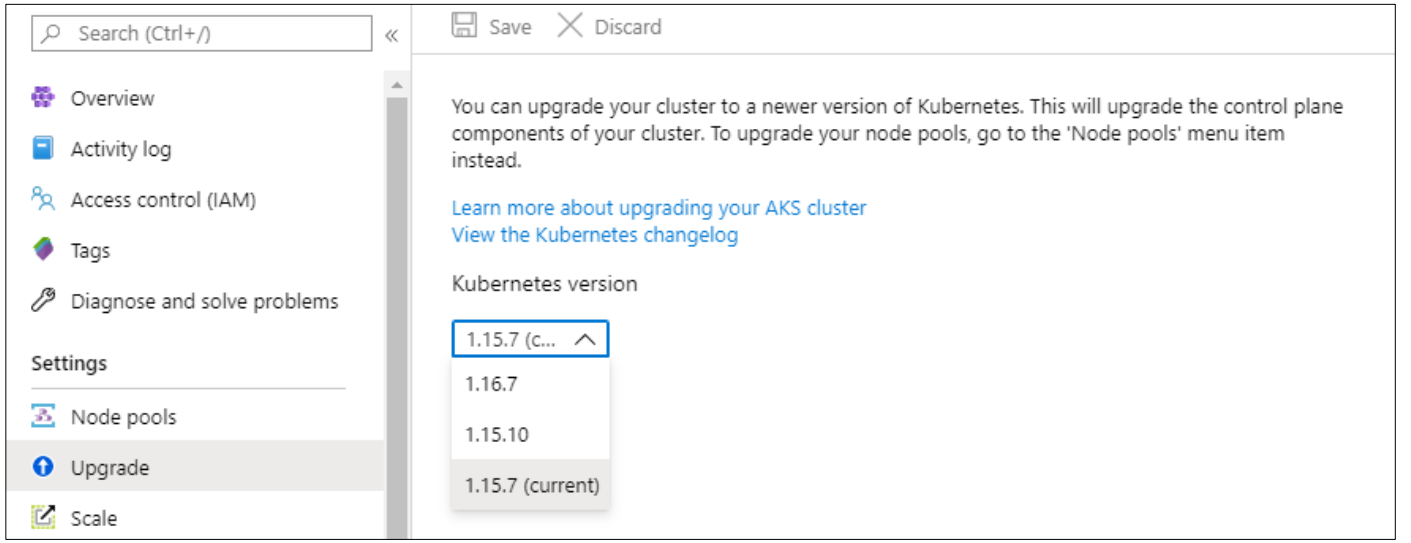
azureserbia

NAME	TYPE	LOCATION
aksdemo	Kubernetes service	East US

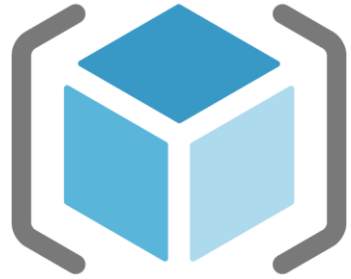
Scale



Upgrade



Azure resources for AKS



azureserbia

NAME	TYPE	LOCATION
aksdemo	Kubernetes service	East US

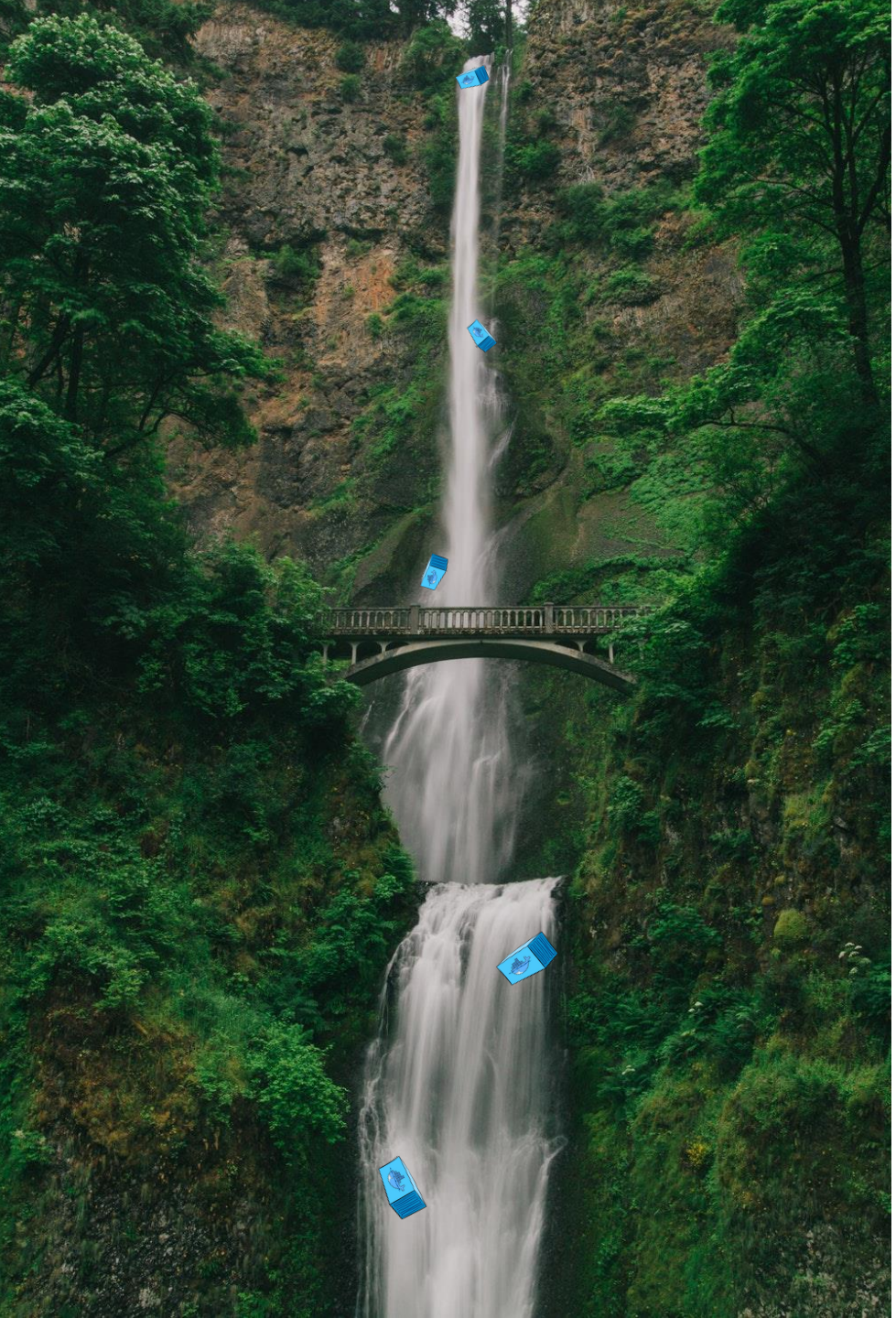
MC_ azureserbia_aksdemo_westeurope

Name	Type	Location
4e71e966-89b3-488f-9c01-1894b33b27d2	Public IP address	West Europe
aks-agentpool-31287680-nsg	Network security group	West Europe
aks-linuxpool-31287680-vmss	Virtual machine scale set	West Europe
aks2go-aks-we-agentpool	Managed Identity	West Europe
kubernetes	Load balancer	West Europe
kubernetes-a1ab23421796744eaaad275b435ee5c2	Public IP address	West Europe
kubernetes-a93232242301947a0ae2463f322bf102	Public IP address	West Europe
omsagent-aks2go-aks-we	Managed Identity	West Europe

Post Install steps

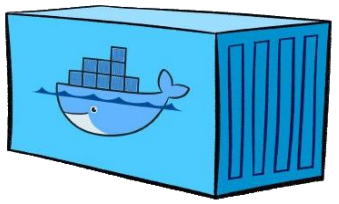
```
## download & install kubectl
az aks install-cli

## download credentials
az aks get-credentials --resource-group communitylive --name aksdemo
```



Deployment flow

1. Create Docker Images



2. Push Docker Images to registry

3. Apply K8s deployments


Kubernetes concepts
@pascalnaber

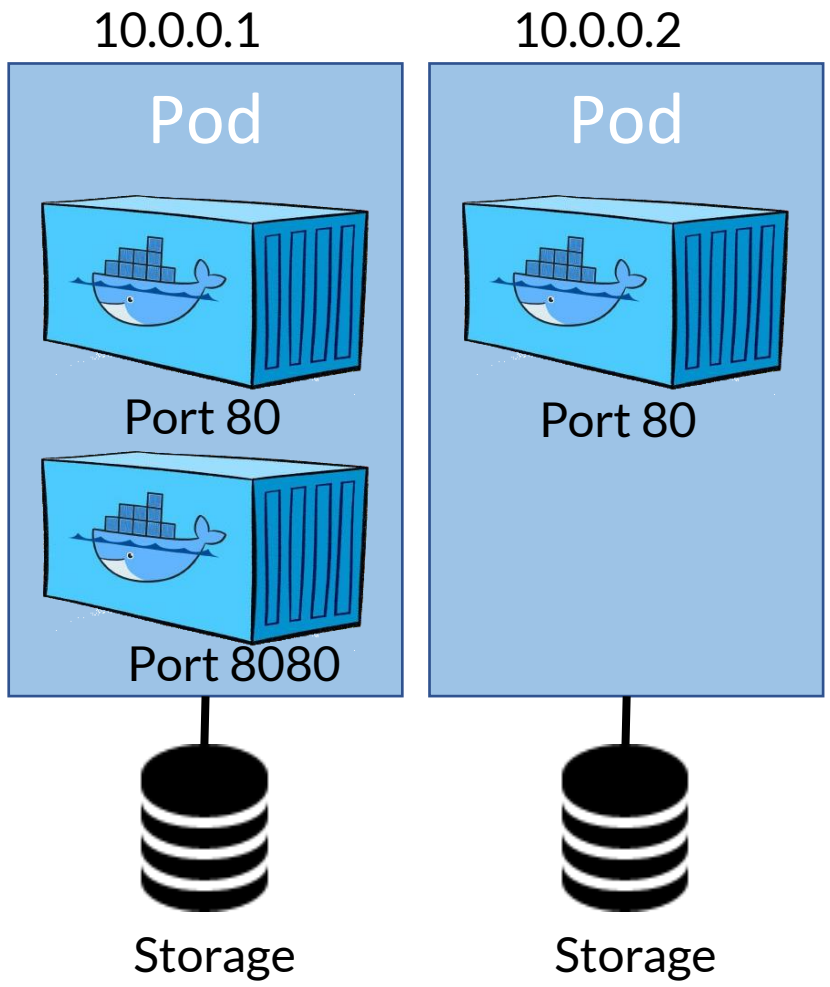
Pod

Group of 1 or more containers

Shared Storage

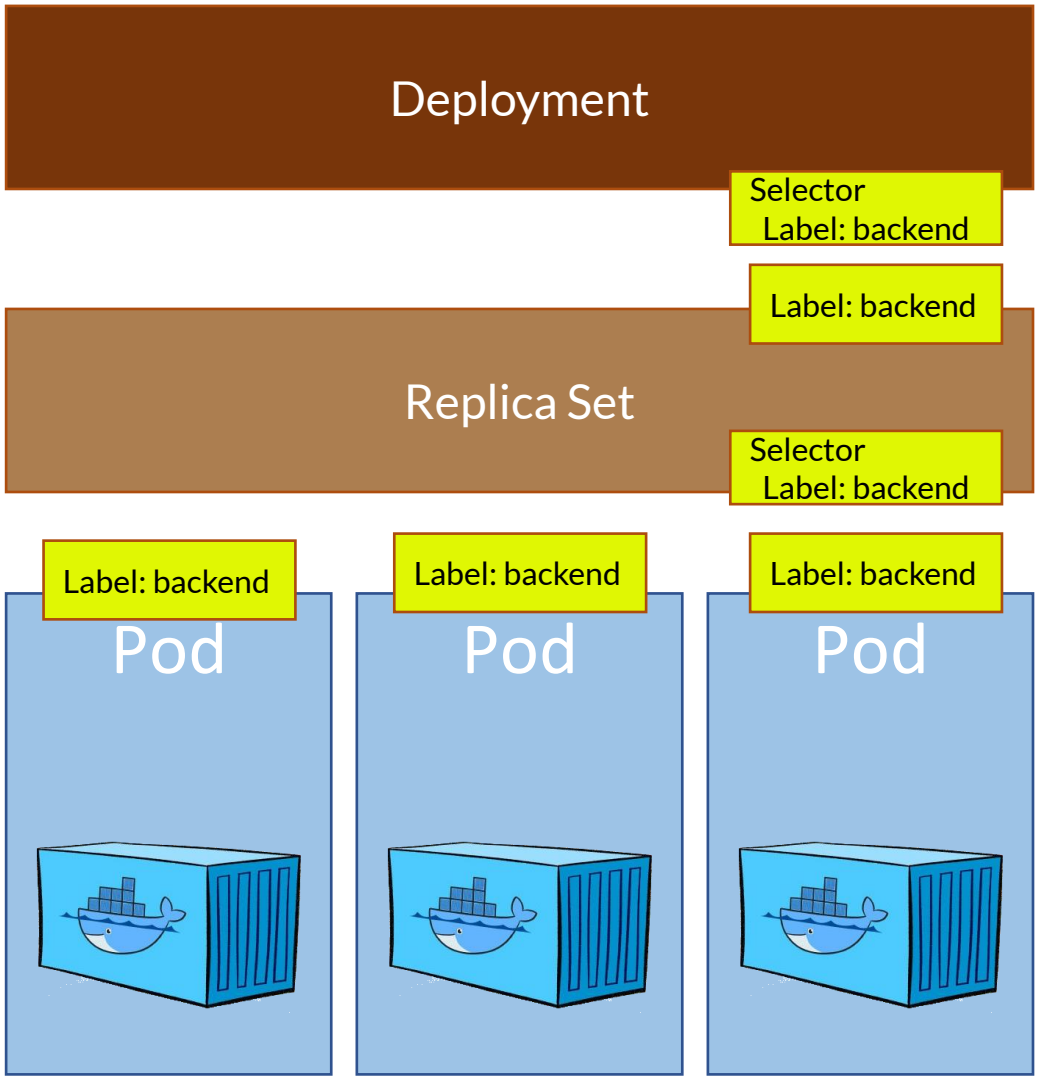
Shared Network

- Same IP-address
- Shared port-range



leaderboard-api.yaml

```yaml
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: leaderboard-api
spec:
  replicas: 3
  template:
    metadata:
      labels:
        app: backend
    spec:
      containers:
      - name: leaderboardwebapi
        image: clouddemo.azurecr.io/leaderboardwebapi:455
        ports:
        - containerPort: 80
```




```
C:> kubectl apply -f leaderboard-api.yaml
```

```yaml
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: leaderboard-api
spec:
  replicas: 10
  minReadySeconds: 5
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxUnavailable: 1
      maxSurge: 1
  template:
    metadata:
      labels:
        app: backend
    spec:
      containers:
      - name: leaderboardwebapi
        image: clouddemo.azurecr.io/leaderboardwebapi:457
      imagePullSecrets:
      - name: clouddemoimages
```

Rolling updates

Zero-downtime deployment

Can be rolled back

```
C:> kubectl apply -f update.yaml
```

Environment variables & Secrets

```yaml
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: leaderboard-api
spec:
  replicas: 3
  template:
    metadata:
      labels:
        app: backend
    spec:
      containers:
      - name: leaderboardwebapi
        image: clouddemo.azurecr.io/leaderboardwebapi:457
        env:
        - name: Serilog__MinimumLevel__Default
          value: "Warning"
        - name: ConnectionStrings__LeaderboardContext
          valueFrom:
            secretKeyRef:
              name: sqlserver
              key: connectionstring
        ports:
        - containerPort: 80
      imagePullSecrets:
      - name: clouddemoimages
```

```yaml
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: leaderboard-api
spec:
  replicas: 3
  template:
    metadata:
      labels:
        app: backend
    spec:
      containers:
      - name: leaderboardwebapi
        image: clouddemo.azurecr.io/leaderboardwebapi:457
        env:
        - name: Serilog__MinimumLevel__Default
          value: "Warning"
        - name: ConnectionStrings__LeaderboardContext
          valueFrom:
            secretKeyRef:
              name: sqlserver
              key: connectionstring
        ports:
        - containerPort: 80
      imagePullSecrets:
      - name: clouddemoimages
```

Environment variables & Secrets

Secrets:

Base64 encoded values

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: sqlserver
type: Opaque
data:
  connectionstring: U2VydmVyPXRjcDpnYW1pb
```

```
C:> kubectl apply –f secret.yaml
```

```
## Create Secret to access Azure Container Registry

kubectl create secret docker-registry clouddemoimages
--docker-server=https://clouddemo.azurecr.io
--docker-username=clouddemo --docker-password=kD98ddl$=
--docker-email=pnaber@xpirit.com
```

```yaml
minReadySeconds: 5
strategy:
  type: RollingUpdate
  rollingUpdate:
    maxUnavailable: 1
    maxSurge: 1
template:
  metadata:
    labels:
      app: backend
  spec:
    containers:
    - name: leaderboardwebapi
      image: clouddemo.azurecr.io/leaderboardwebapi:455
      ports:
      - containerPort: 80
      livenessProbe:
        httpGet:
          path: /health
          port: 80
        initialDelaySeconds: 2
        timeoutSeconds: 1
        periodSeconds: 10
        failureThreshold: 3
      readinessProbe:
        httpGet:
          path: /health
          port: 80
        initialDelaySeconds: 2
        timeoutSeconds: 1
        periodSeconds: 10
        failureThreshold: 3
```

Health checks

livenessProbe

Indicates whether the Container is running

Restart in case of failure

Container:

https://www.app-metrics.io/

HealthChecks library
https://github.com/dotnet-architecture/HealthChecks

Microsoft.AspNetCore.Diagnostics.HealthChecks

readinessProbe

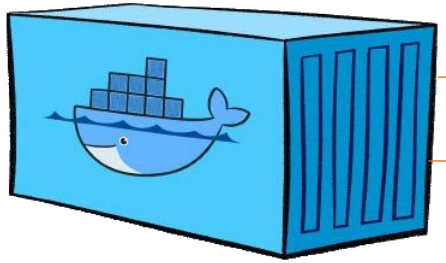
Indicates whether the Container is ready to service requests

No traffic is routed to the Pod

```
C:> kubectl apply –f leaderboard-api.yaml
```

Kubernetes – Self healing

Health endpoints returns != 200?



Every n seconds check

Restarts container

During rolling update deployment

Stop deployment

During container startup

No traffic

```yaml
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: leaderboard-api
spec:
  replicas: 10
  minReadySeconds: 5
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxUnavailable: 1
      maxSurge: 1
  template:
    metadata:
      labels:
        app: backend
    spec:
      containers:
      - name: leaderboardwebapi
        image: clouddemo.azurecr.io/leaderboardwebapi:457
        resources:
          requests:
            cpu: "100m"
            memory: "64Mi"
          limits:
            cpu: "500m"
            memory: "128Mi"
        env:
        - name: Serilog__MinimumLevel__Default
          value: "Warning"
        - name: ConnectionStrings__LeaderboardContext
          valueFrom:
            secretKeyRef:
              name: sqlserver
              key: connectionstring
          path: /health
          port: 80
        initialDelaySeconds: 2
        timeoutSeconds: 1
```

Resource Management

requests: Minimum required resources

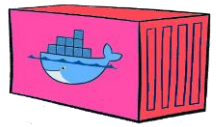
limits: Capped resource usage

100m = 0.1 cpu
500m = 0.5 cpu

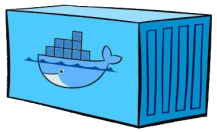
64Mi = 64 MB memory
128Mi = 128 MB memory

```
C:> kubectl apply -f leaderboard-api.yaml
```

Azure Kubernetes Service (AKS) - Scaling



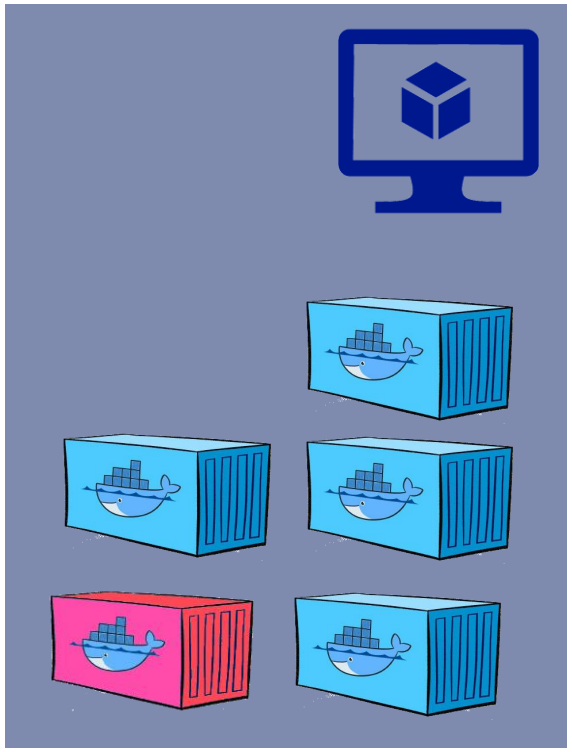
5 replicas



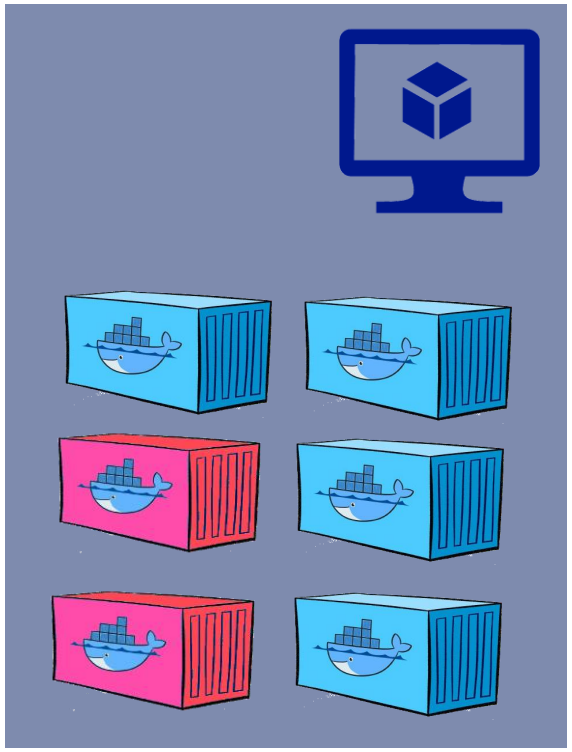
4 replicas

Pod Autoscaler 4-20
> 60% CPU

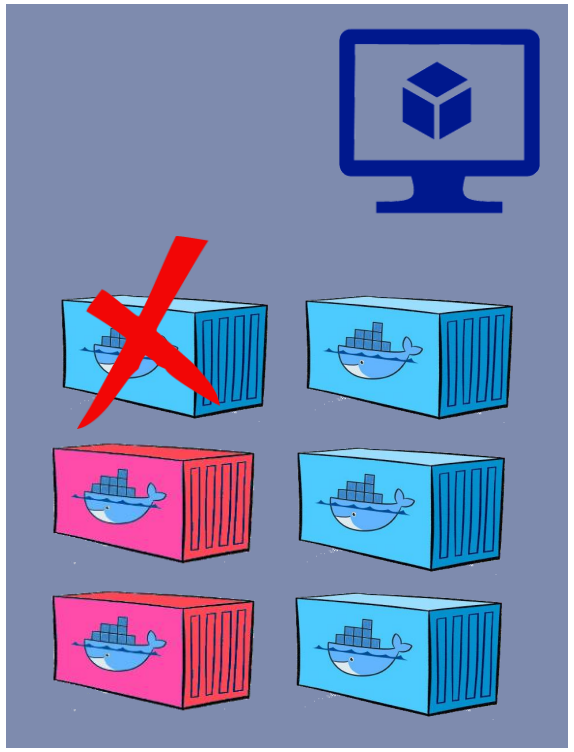
Cluster Autoscaler



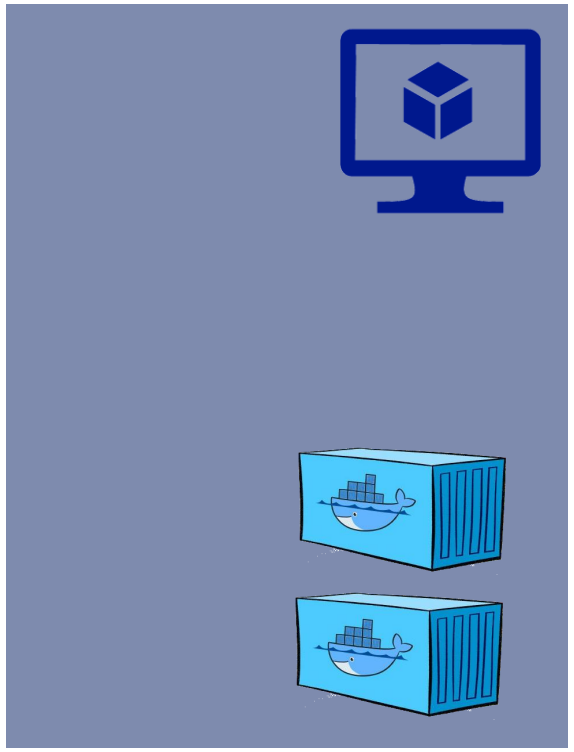
worker



worker



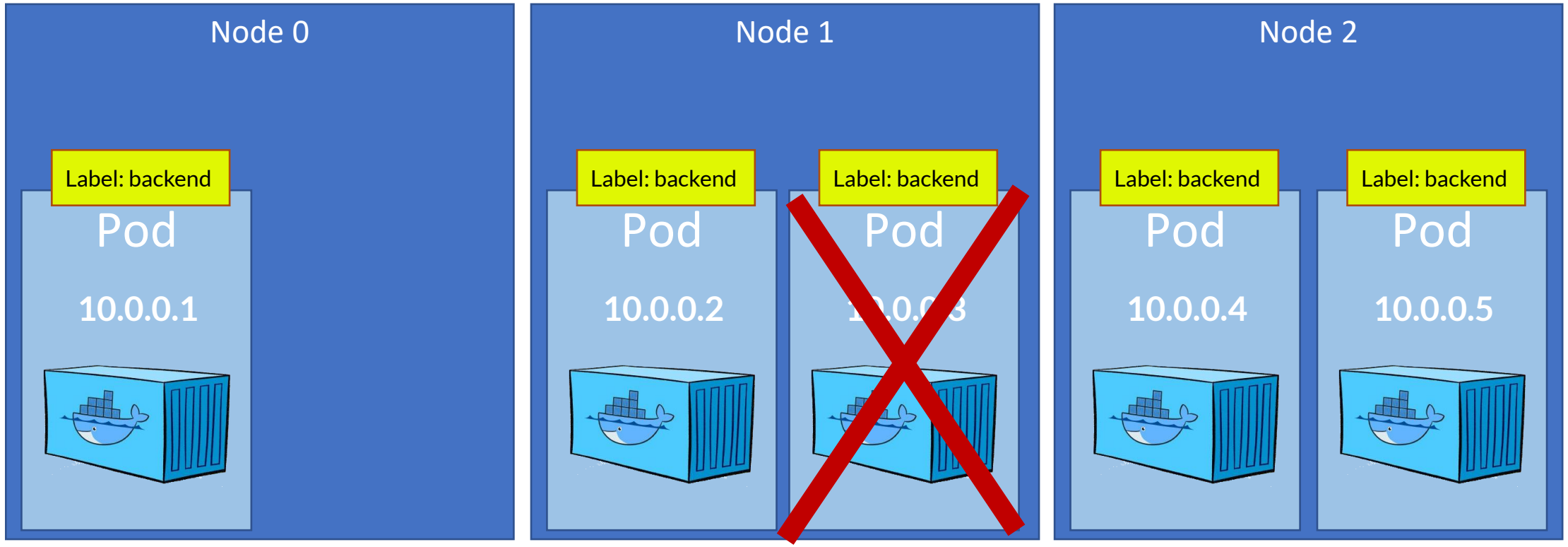
worker



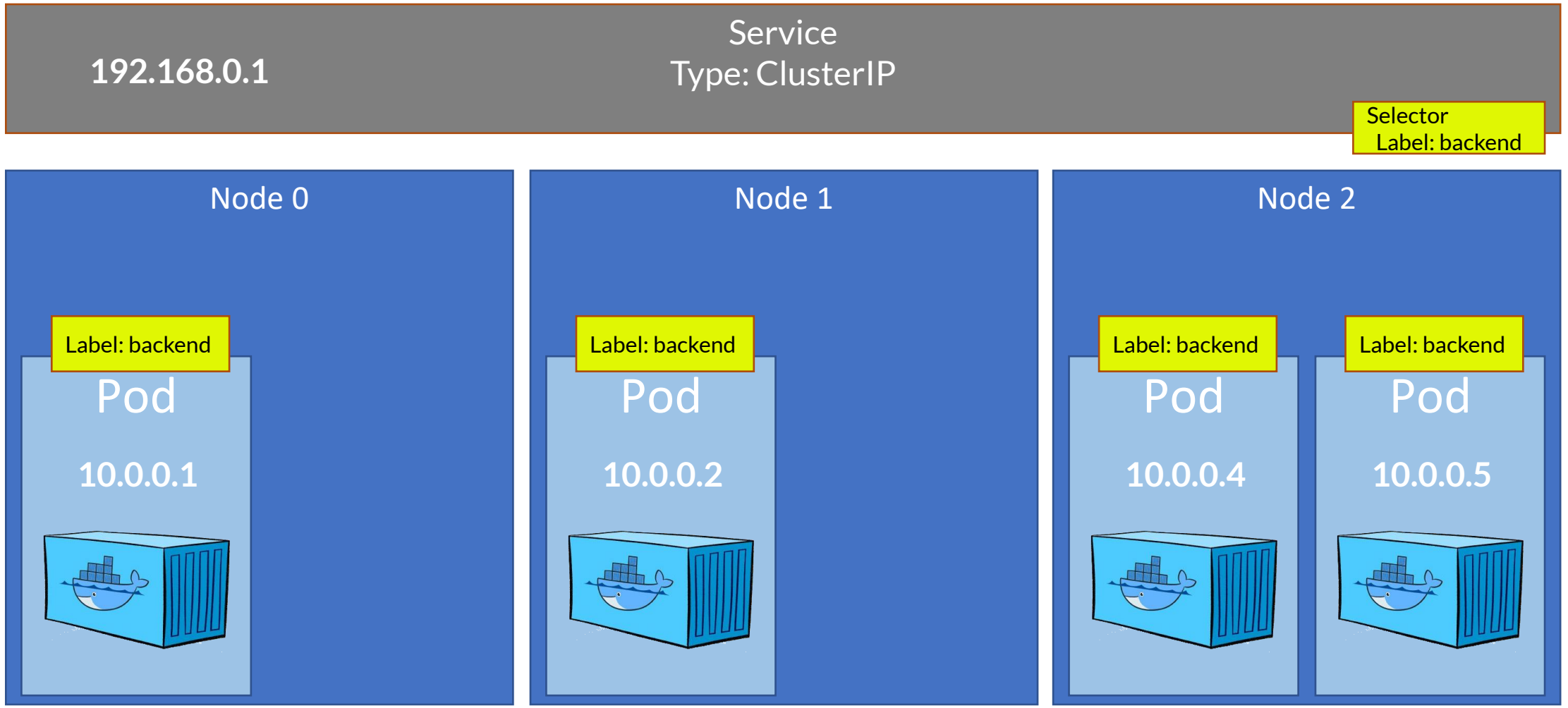
worker


How to access the Pods?
From inside and outside the cluster
@pascalnaber

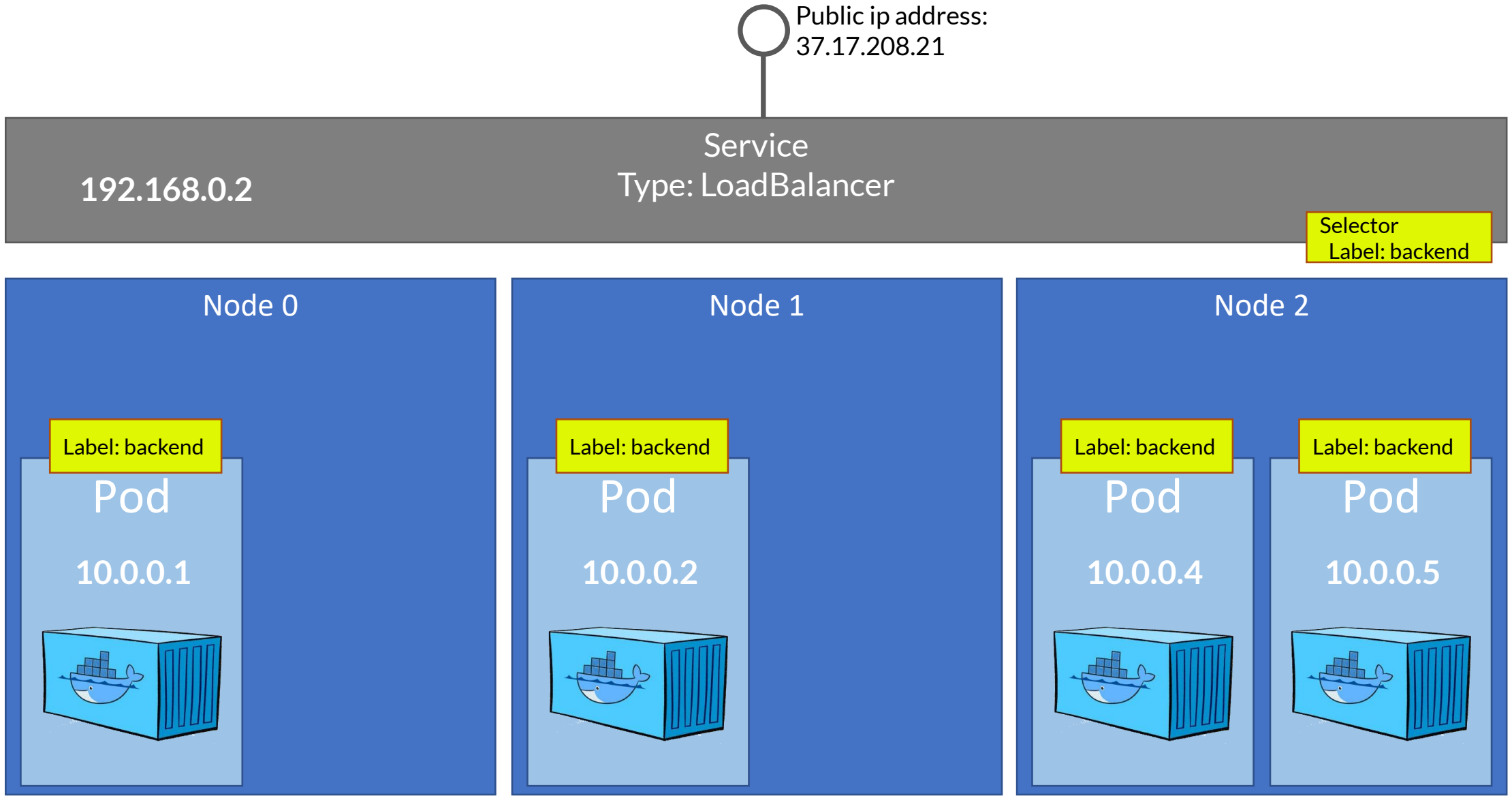
Pods are mortal

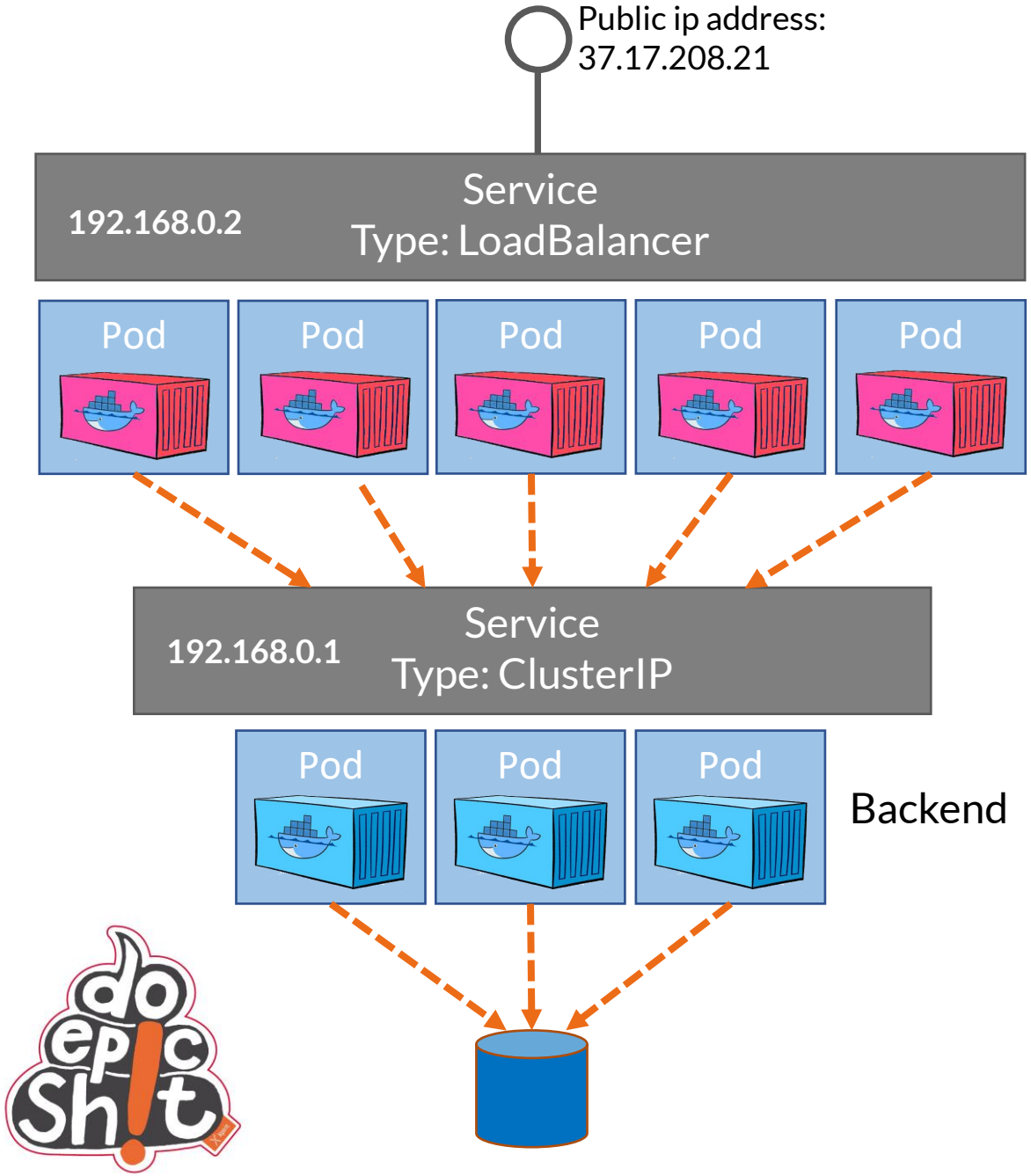


Service


192.168.0.1
Service
Type: ClusterIP
Selector
Label: backend
Node 0
Label: backend
Pod
10.0.0.1
Node 1
Label: backend
Pod
10.0.0.2
Node 2
Label: backend
Pod
10.0.0.4
Label: backend
Pod
10.0.0.5

Service


Public ip address:
37.17.208.21
Service
Type: LoadBalancer
192.168.0.2
Selector
Label: backend
Node 0
Label: backend
Pod
10.0.0.1
Node 1
Label: backend
Pod
10.0.0.2
Node 2
Label: backend
Pod
10.0.0.4
Label: backend
Pod
10.0.0.5




```yaml
apiVersion: v1
kind: Service
metadata:
  name: leaderboardapi-service
  namespace: service
spec:
  ports:
    - port: 80
  selector:
    app: backend
  type: ClusterIP
```

```
C:> kubectl apply -f leaderboard-service.yaml
```

```yaml
spec:
  containers:
  - name: gamingwebapp
    image: clouddemo.azurecr.io/gamingwebapp:455
    env:
    - name: LeaderboardWebApiBaseUrl
      value: "http://leaderboardapi-service"
    ports:
    - containerPort: 80
```

```
C:> kubectl apply -f gamingwebapp.yaml
```

Too many ip-addresses. Now what?

I want to access all my services through the same ip-address

@pascalnaber

Ingress

Public ip address:
37.17.208.21

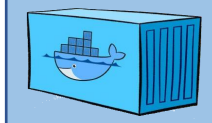
Service
Type: LoadBalancer

Selector
Label: ingresscontroller

quay.io/kubernetes-ingress-controller/
nginx-ingress-controller:0.15.0

Label:
ingresscontroller

Pod



```yaml
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: gamingwebapp-ingresscf
  namespace: ingressdemo
  annotations:
    kubernetes.io/ingress.class: nginx
spec:
  rules:
  - host: gaming.matrixproject.cf
    http:
      paths:
      - backend:
          serviceName: gamingwebapp-service
          servicePort: 80
```

Ingress
host: gaming. community.cf
serviceName: gamingwebapp

Service
Type: ClusterIP

Label: frontend

Pod 10.0.0.1 | Pod 10.0.0.2 | Pod 10.0.0.3

Ingress
host: erp. community.cf
serviceName: erpsvc

Service
Type: ClusterIP

Label: erp

Pod 10.0.0.4 | Pod 10.0.0.5 | Pod 10.0.0.6

Ingress
host: www. community.cf/crm
serviceName: crmsvc

Service
Type: ClusterIP

Label: crm

Pod 10.0.0.7 | Pod 10.0.0.8 | Pod 10.0.0.9


Deployment
Replica set
Pod
Label
Rolling update
Health check
Environment variables
Secret
Resource management
Horizontal Pod Autoscaler
Namespace
Service
Ingress
Annotation
Persistent Volume
Cron Job
Deamon Set
Job
Stateful Set
Config Map
@pascalnaber

Azure Dev Spaces

Ecosystem

Cert manager

+

= Kubeflow

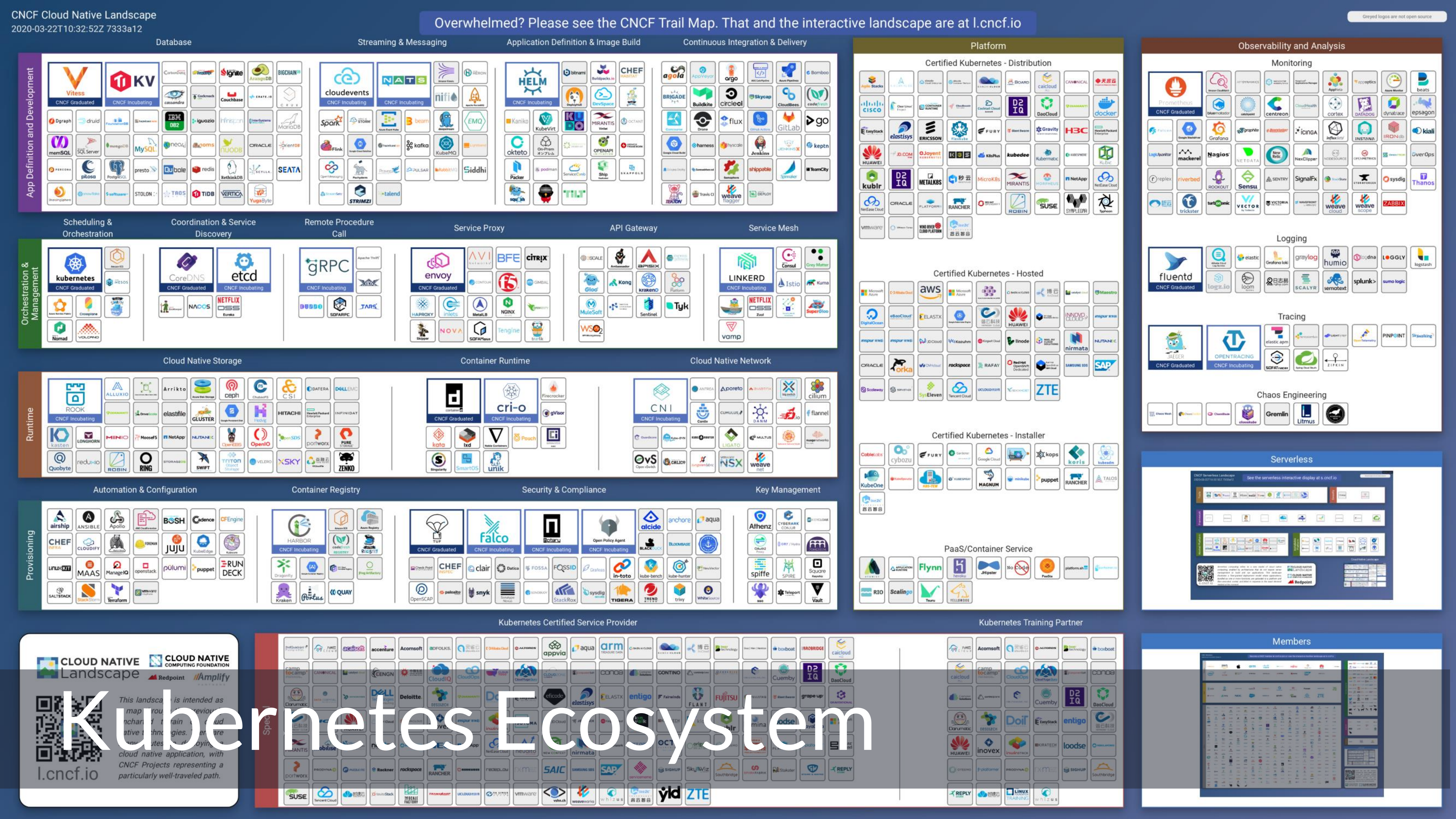
Helm

Helm is a tool for managing packages of pre-configured Kubernetes resources

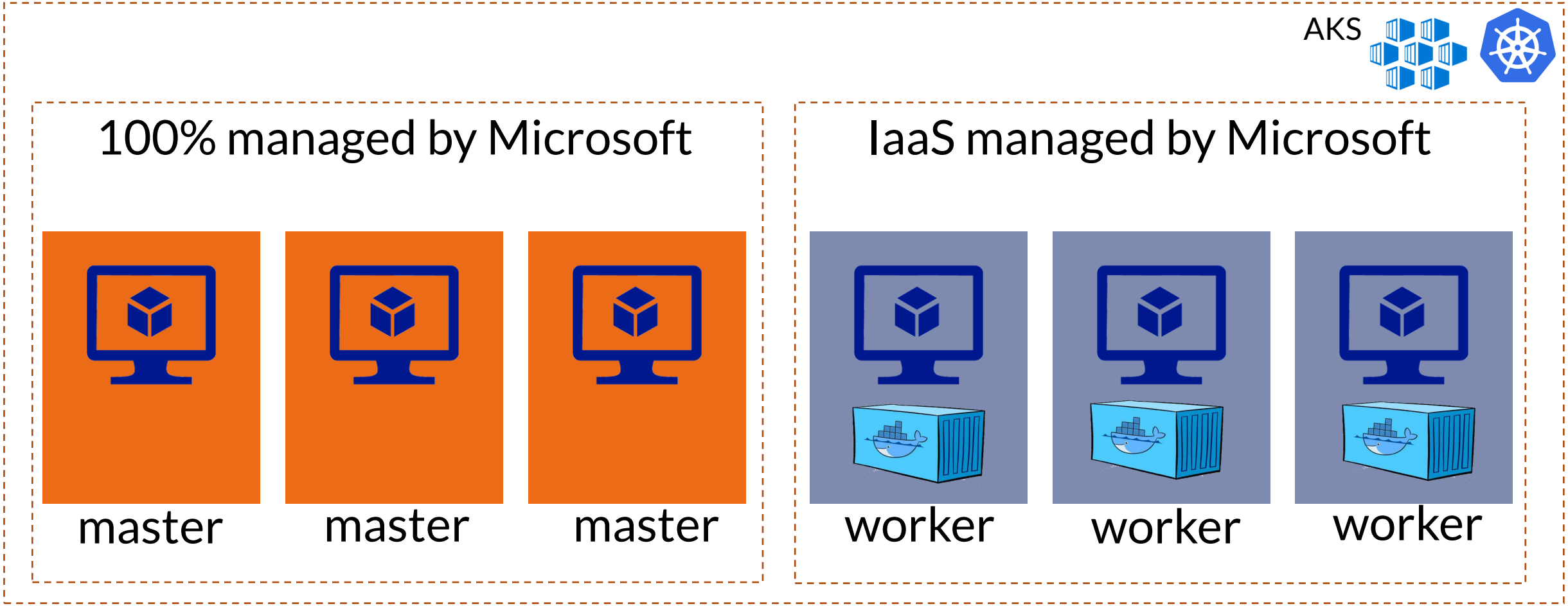
https://github.com/kubernetes/helm

- Artifactory
- Cert-manager
- Consul
- Cassandra
- Couchdb
- Datadog
- Docker-registry
- Drupal
- Elasticsearch
- Ethereum
- fluentd
- Gitlab
- Grafana
- Hadoop
- Heapster
- Jenkins
- Joomla
- Kafka
- Kibana
- Kong
- Kubeless
- Lamp
- Linkerd
- Logstash
- Magento
- Mariadb
- Mongodb
- Mssql-linux
- Mysql
- Neo4j
- Newrelic
- Openvpn
- Postgresql
- Presto
- Prometheus
- Rabbitmq
- Redis
- Selenium
- Sonarqube
- Spinnaker
- Sysdig
- Tensorflow
- Traefik
- Wordpress

Kubernetes Ecosystem



Azure Kubernetes Service (AKS)



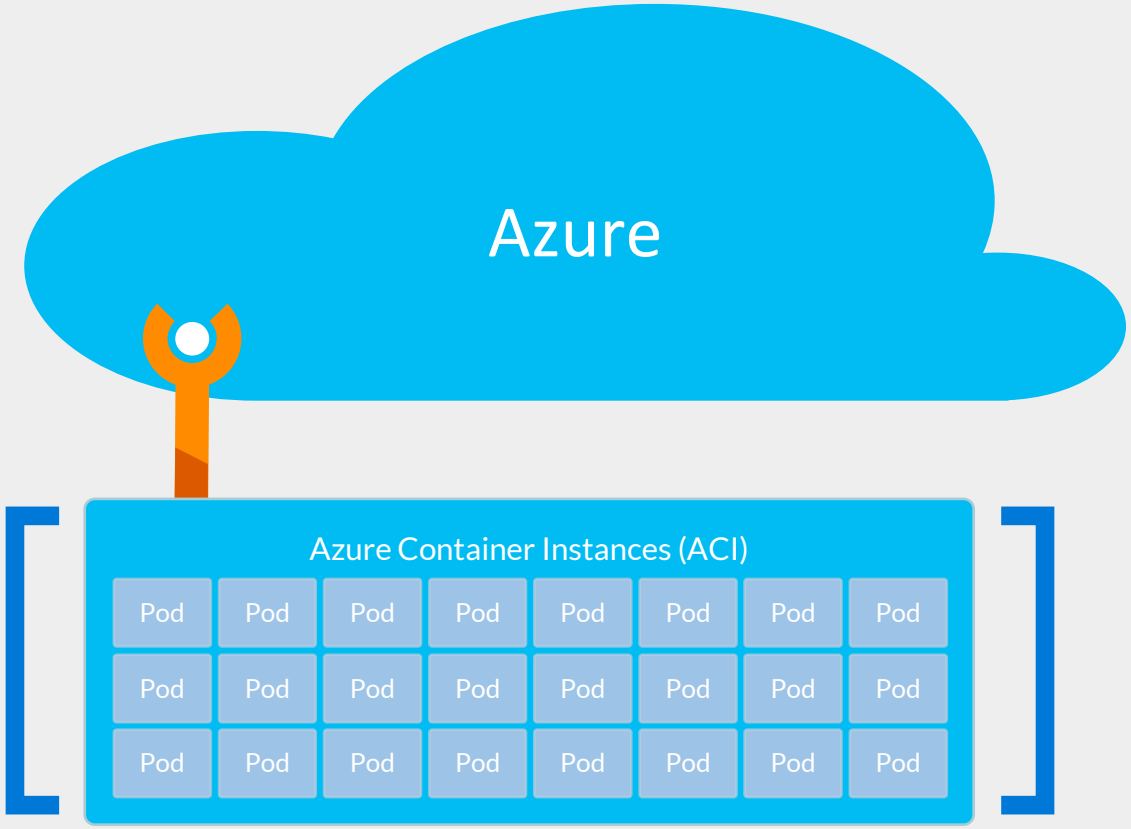
Azure Container Instances (ACI)

Starts in seconds

No VM Management

Billed per second

Linux and Windows containers



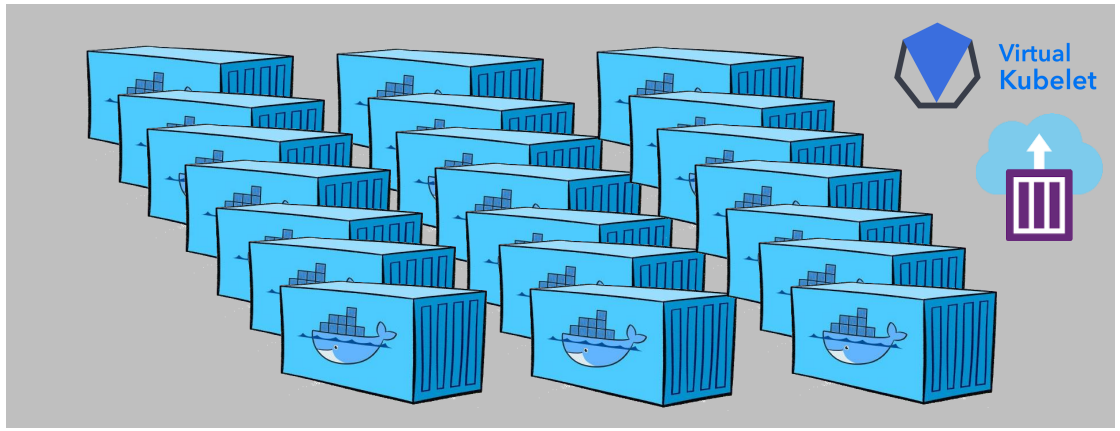
AKS + ACI

AKS

100% managed by Microsoft

master master master

100% managed by Microsoft



worker

@pascalnaber


The future of infrastructure is happening now
kubernetes
@pascalnaber

Wrap up

Docker

Install applications to a container instead of a server

Ship containers everywhere

Kubernetes in the cloud

Deploy your containerized workloads on Kubernetes

Brings a clean separation between Dev and Ops

Provides scaling, rolling updates, high availability

P4I Naber
Coding Azure Architect
Xpirit Netherlands

@pascalnaber
http://pascalnaber.wordpress.com

https://github.com/pascalnaber/azureserbia

Microsoft® Most Valuable Professional

Xpirit