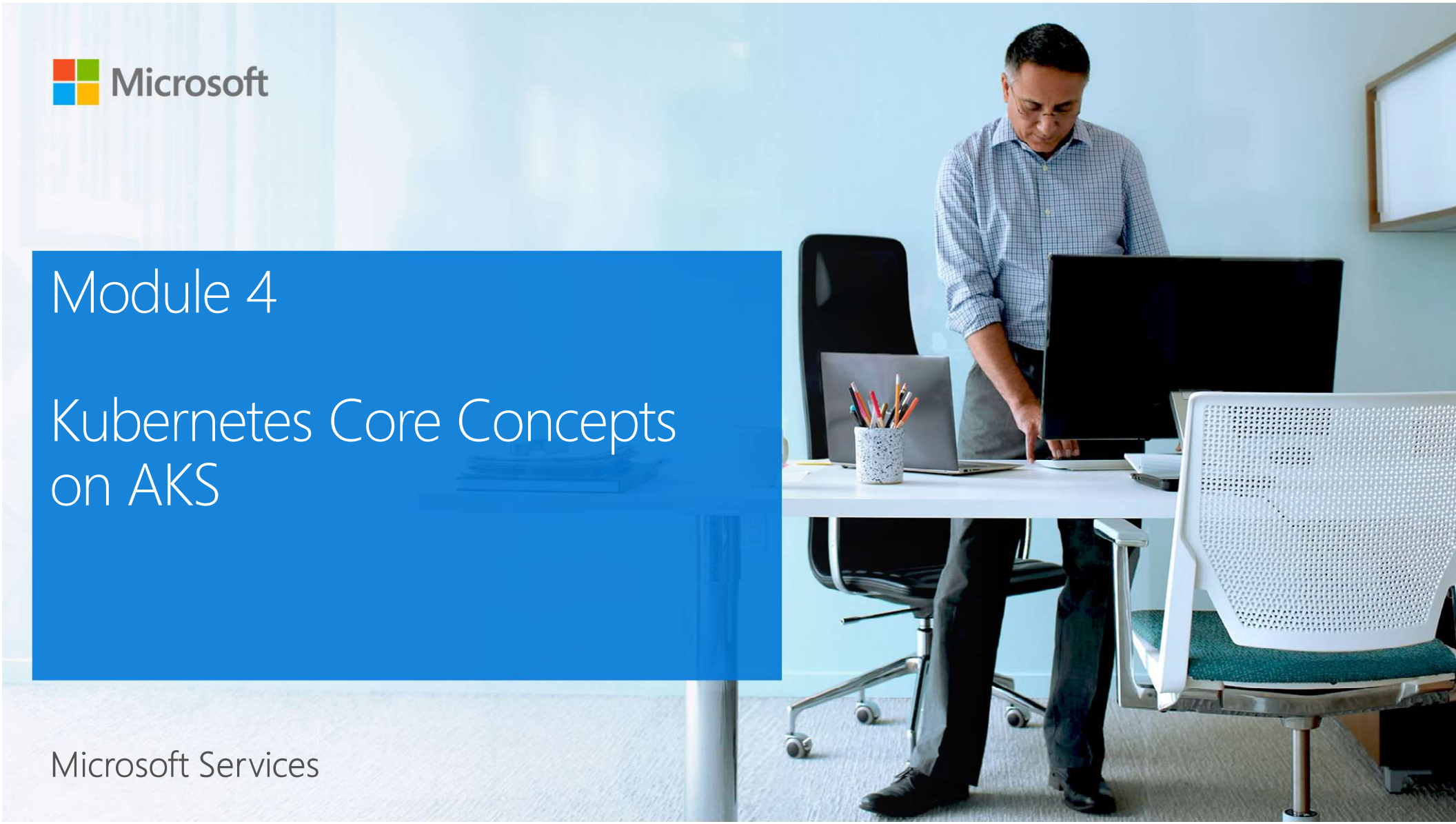
Microsoft

Module 4

Kubernetes Core Concepts on AKS



Microsoft Services

Conditions and Terms of Use

Microsoft Confidential

This training package is proprietary and confidential, and is intended only for uses described in the training materials. Content and software is provided to you under a Non-Disclosure Agreement and cannot be distributed. Copying or disclosing all or any portion of the content and/or software included in such packages is strictly prohibited.

The contents of this package are for informational and training purposes only and are provided "as is" without warranty of any kind, whether express or implied, including but not limited to the implied warranties of merchantability, fitness for a particular purpose, and non-infringement.

Training package content, including URLs and other Internet website references, is subject to change without notice. Because Microsoft must respond to changing market conditions, the content should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication. Unless otherwise noted, the companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, e-mail address, logo, person, place, or event is intended or should be inferred.

Copyright and Trademarks

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

For more information, see **Use of Microsoft Copyrighted Content** at
https://www.microsoft.com/en-us/legal/intellectualproperty/permissions/default.aspx

Microsoft®, Internet Explorer®, Outlook®, SkyDrive®, Windows Vista®, Zune®, Xbox 360®, DirectX®, Windows Server® and Windows® are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. Other Microsoft products mentioned herein may be either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. All other trademarks are property of their respective owners.

How to View This Presentation

To switch to **Notes Page** view:

On the ribbon, click the **View** tab, and then click **Notes Page**

To navigate through notes, use the **Page Up** and **Page Down** keys

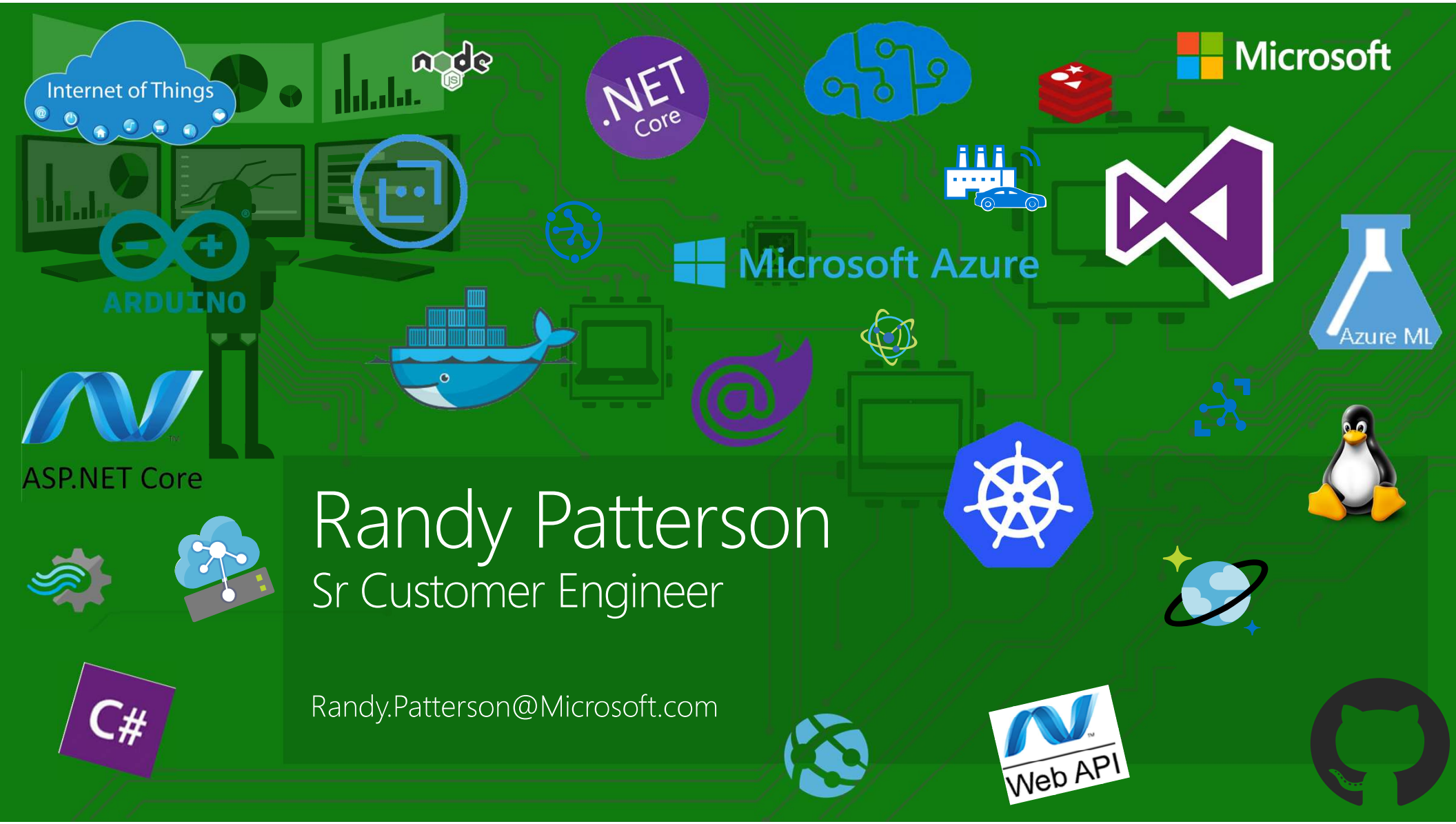
Zoom in or zoom out, if required

In the **Notes Page** view, you can:

Read any supporting text—now or after the delivery

Add notes to your copy of the presentation, if required

Take the presentation files home with you



Randy Patterson

Sr Customer Engineer

Randy.Patterson@Microsoft.com

Setup Labs

http://aka.ms/PremierEducation

Training Key: 4EE6E425C0904A83

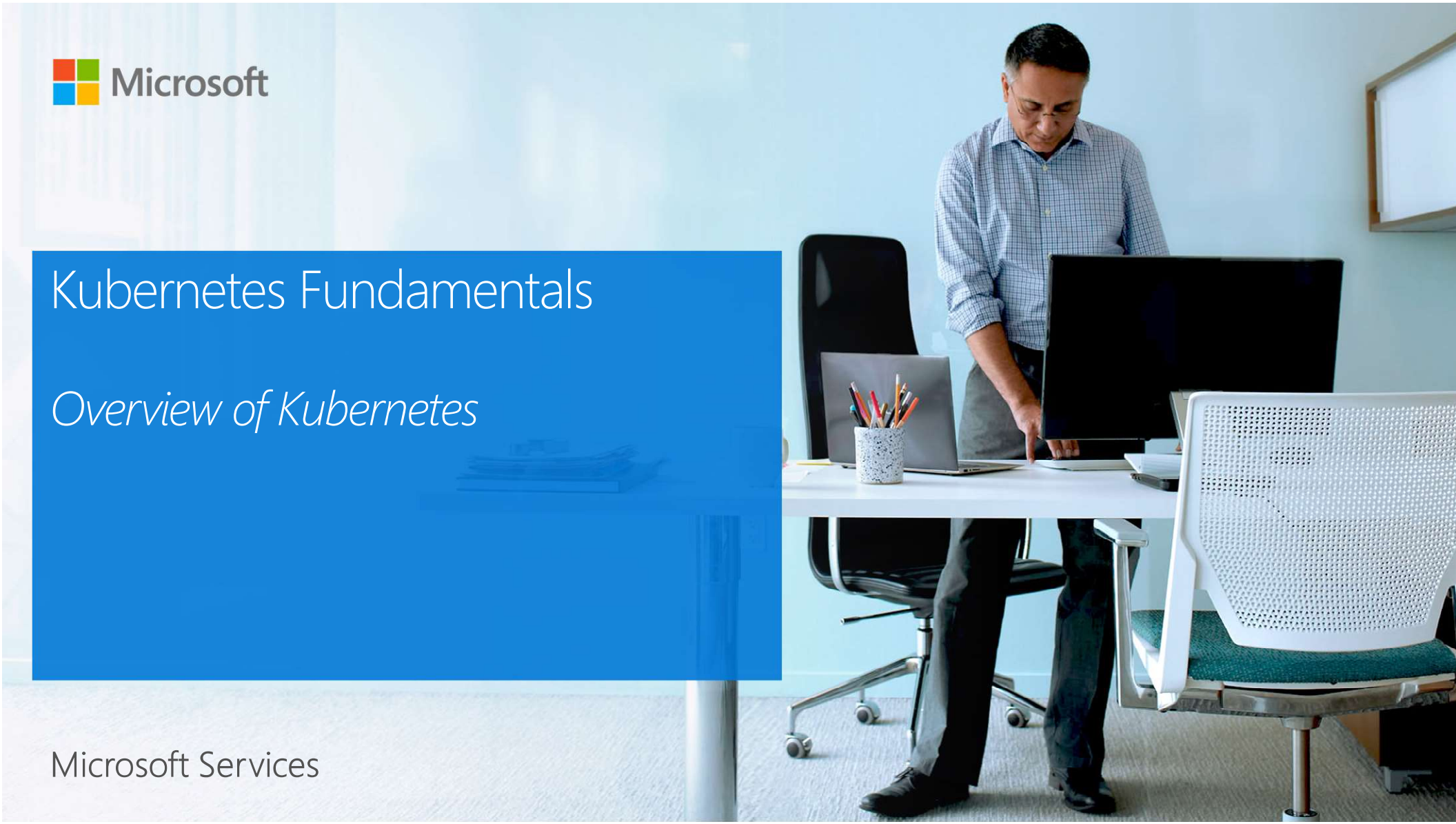
Module Overview

- Kubernetes and AKS Overview
- Pods
- Labels and Selectors
- Services
- Deployment
- ConfigMaps & Secrets
- Volumes
- Scaling and Updating

Microsoft

Kubernetes Fundamentals

Overview of Kubernetes



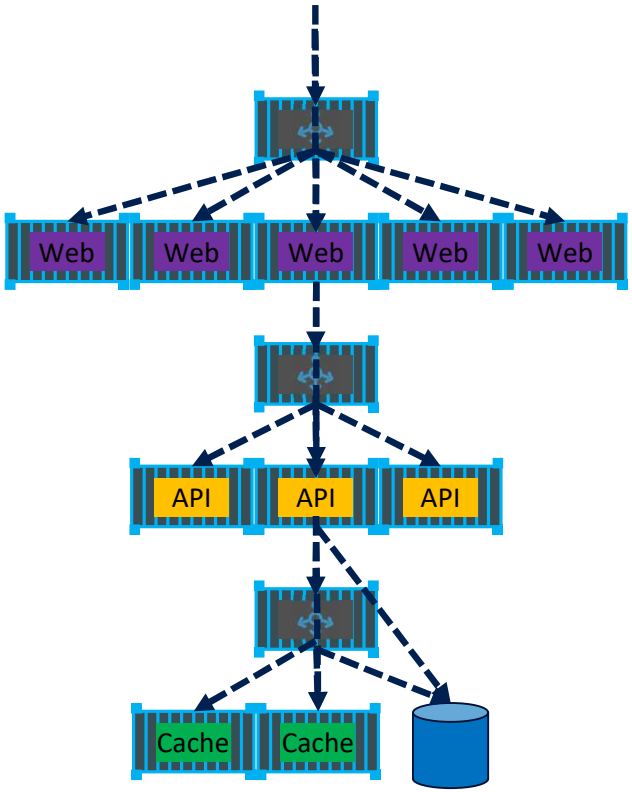
Microsoft Services

Challenges of a containerized world

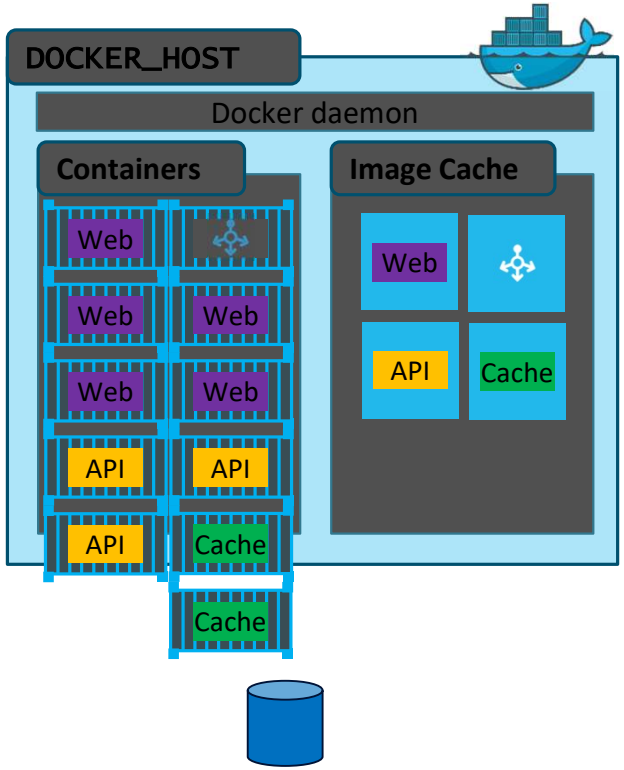
As application development has moved towards a container-based approach, the need to orchestrate and manage the inter-connected resources becomes important

- Load Balancing
  - Distributing traffic across containers at scale
- Naming and Discovery
  - How do containers or groups find one another?
- Logging and Monitoring
  - Keeping track of what containers are doing
- Debugging and Introspection
  - Getting inside running containers
- Networking
  - Differentiating container networks from host networks at scale

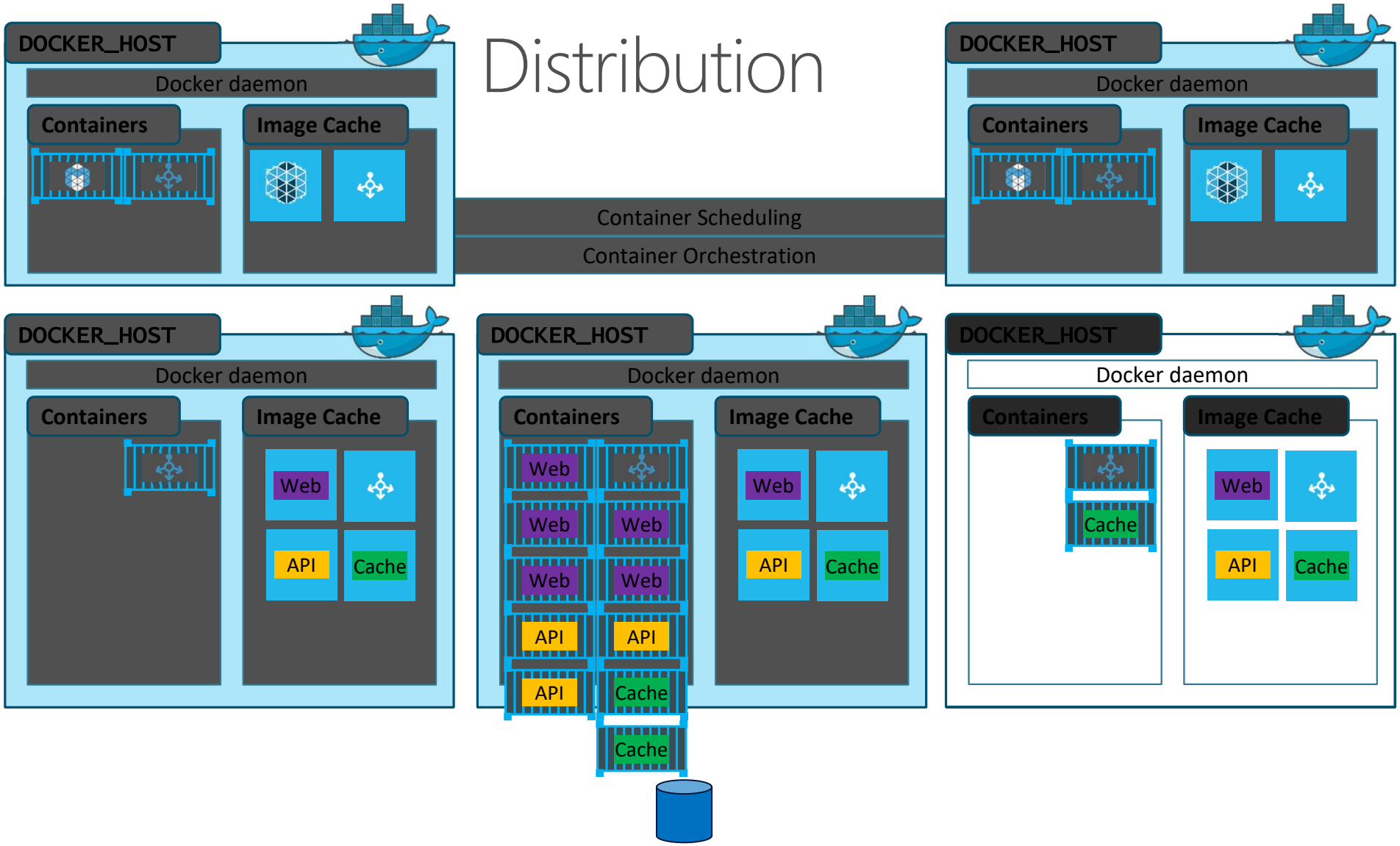
Application Scale



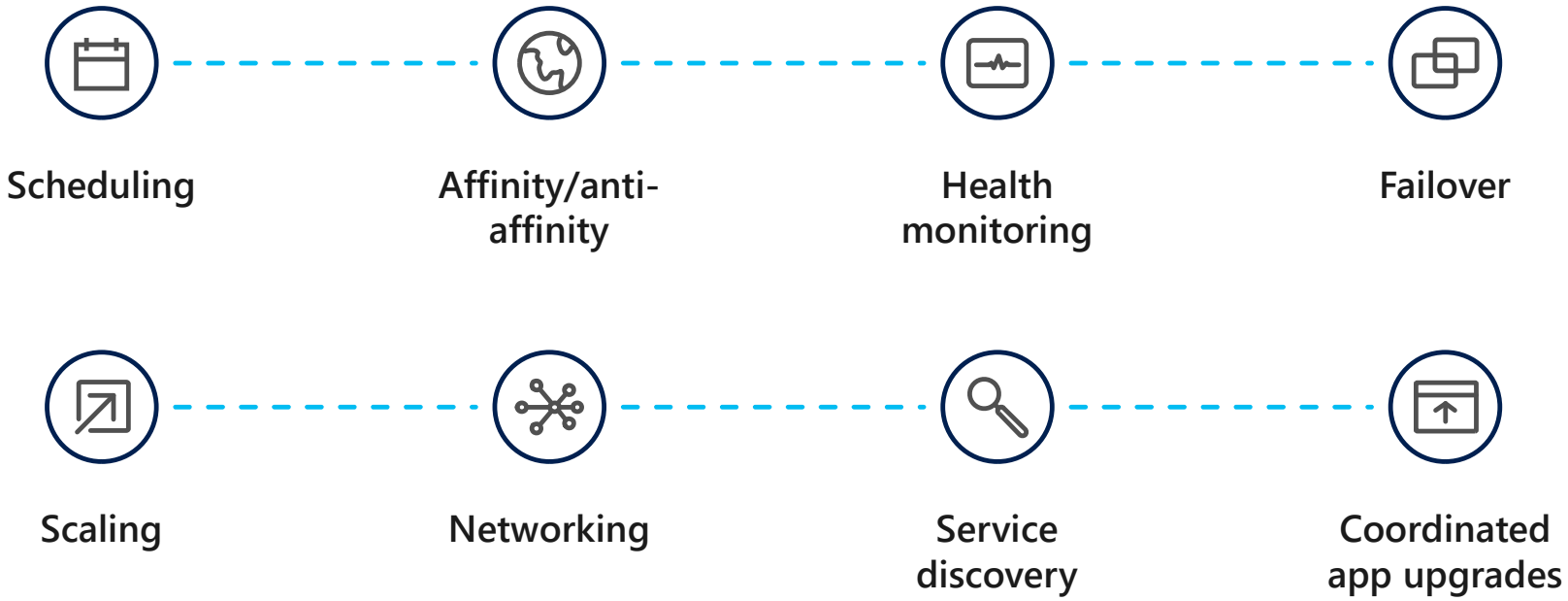
Load Balancing & Fault Tolerance



Distribution



The elements of orchestration



Clustering versus orchestration

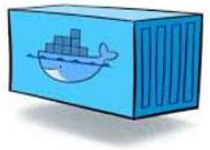
Clustering

- Grouping “hosts”—either VMs or bare metal—and networking them together
- A cluster should feel like a single resource rather than a group of disparate machines

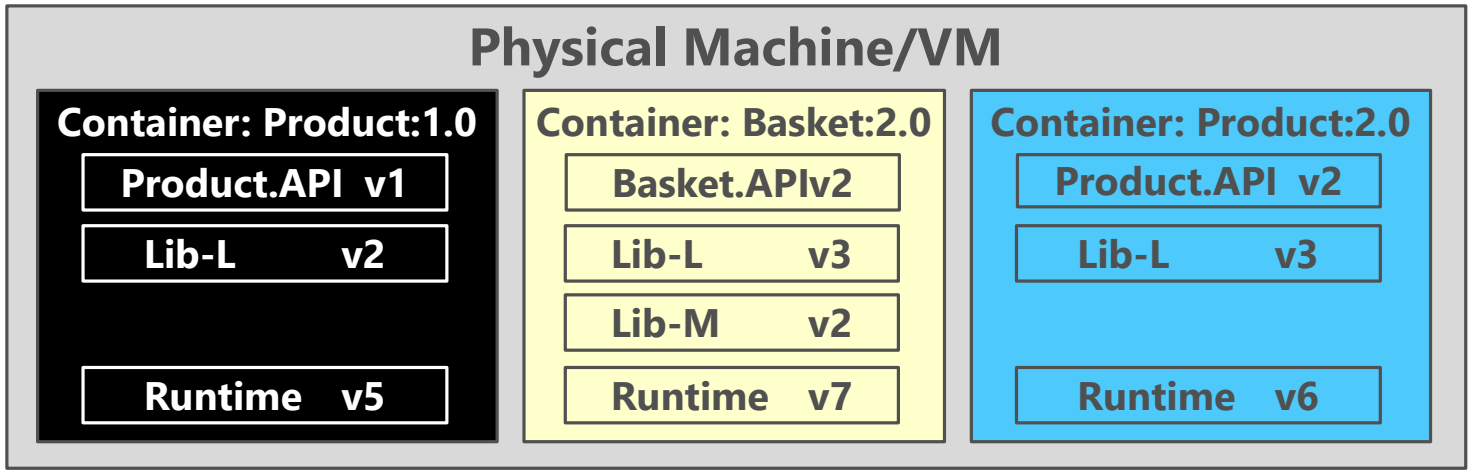
Orchestration

- Managing and monitoring of the workloads running in your cluster
- Starting containers on appropriate hosts and connecting them
- May also include support for scaling, automatic failover, and node rebalancing

What is a Container?



- Portable unit of deployment
- Packs application code and dependencies together into single unit
- Virtualization without the need of a full virtual machine
  - Slice up the OS to run multiple apps on a single OS
  - Each container runs in isolated memory, but shares the kernel of underlying host
- Typically run one service per container (container and app share lifecycle)

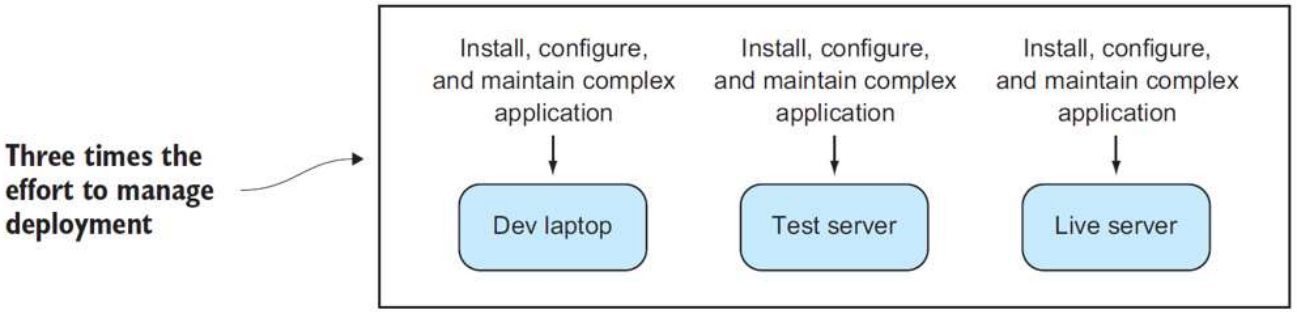


What is Docker?

- Docker is the company driving the software container movement
  - In a short time, it has become the de facto standard for packaging, deploying and running distributed and cloud native platforms
- Docker is a technology stack…
  - An open platform that enables you to “build, ship, and run any app, anywhere”
  - A container format
  - A set of tools for creating and running application in containers
  - Includes open source and (for-purchase) enterprise offerings
- Docker has become a standard for solving one of the costliest aspects of software: deployment

What is Docker?

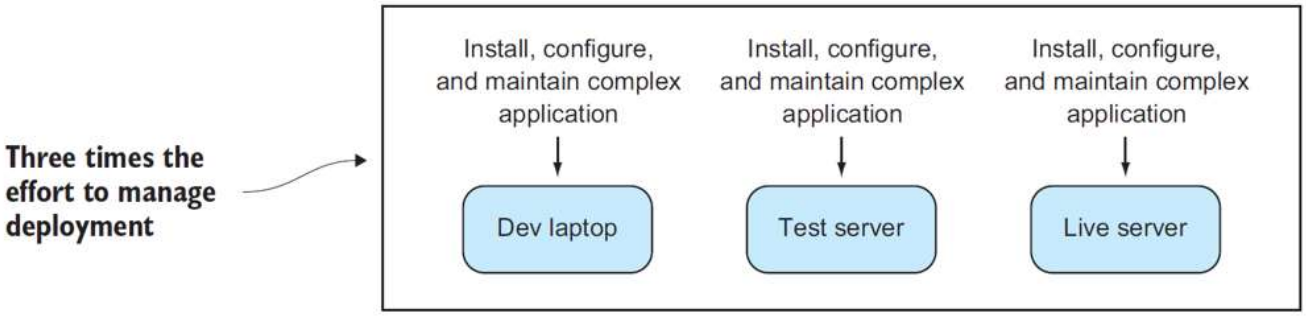
Life Before Docker



Expensive Environmental Issues

- **Missing dependencies**
- **Versioning issues**
- **Incorrect configurations**
- **Outdate runtimes**

Life With Docker



Expensive E tal Issues

- **Missing de cies**
- **Versioning**
- **Incorrec rations**
- **Outdat**

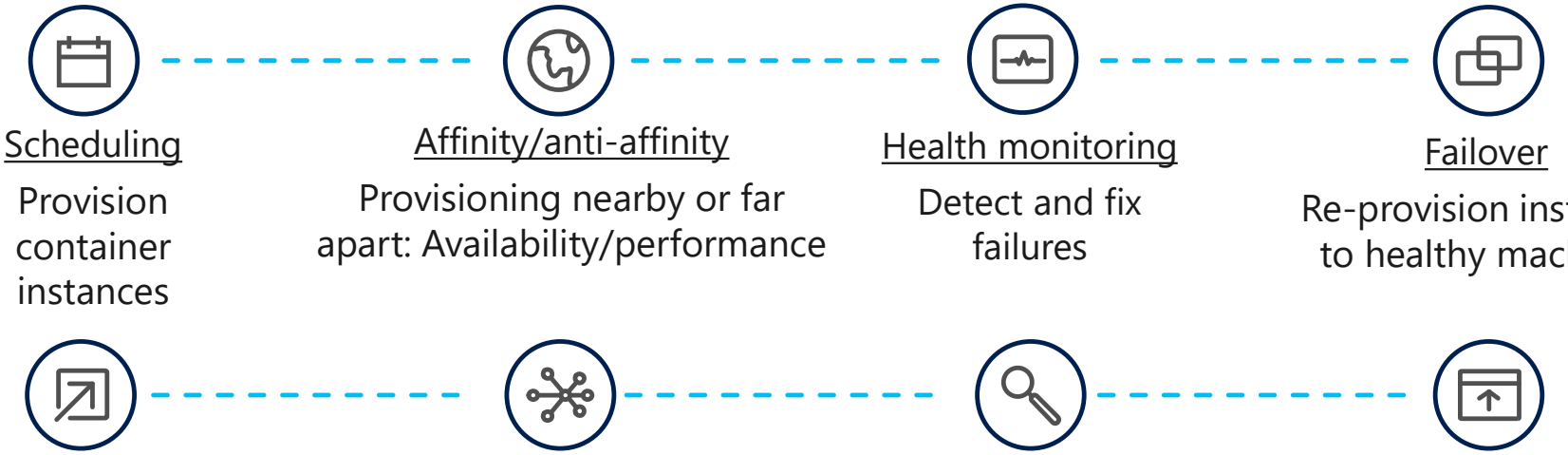
What Problems Do Containers Solve?

- Guarantees consistency across dev, test and prod environments – everything is self-contained
  - Provides portability – works across all environments
- Increases Productivity
  - Less time setting up environments/debugging environment-specific issues
- Smaller footprint than VMs
  - Increase server consolidation and density per host
- Isolation
  - Each container has separate slice of OS, CPU and memory without affecting other containers
- Performant and quick start-up

However….

Large containerized workloads require *orchestration*…

- Management/automation across the multiple hosts inside a cluster



Scheduling
Provision container instances

Affinity/anti-affinity
Provisioning nearby or far apart: Availability/performance

Health monitoring
Detect and fix failures

Failover
Re-provision instances to healthy machines

Scaling
Add/remove instances to meet demand

Networking
Networking overlay for container communication

Service discovery
Enable containers to locate each other

Coordinated app upgrades
Avoid downtime and automatically rollback

Kubernetes (K8s)

What exactly is Kubernetes?

- Greek term for "Ship Master"
- Has become the de facto standard container-orchestration system
- Is becoming the *operating system* for the *cloud native world*
- Automates deployment, scaling, and operational concerns of containerized workloads across clusters of VM hosts
- Originated from Google and donated to Cloud Native Computing Foundation (CNCF) in 2015
- Microsoft's Brendan Burns was a co-creator!

But, a major challenge to provision and manage yourself!

Azure Kubernetes Service (AKS)

Fully-managed Kubernetes platform hosted in Azure as a PaaS service

Deeply integrated with Azure dev tools and services

Abstracts the complexity and operational overhead of managing Kubernetes

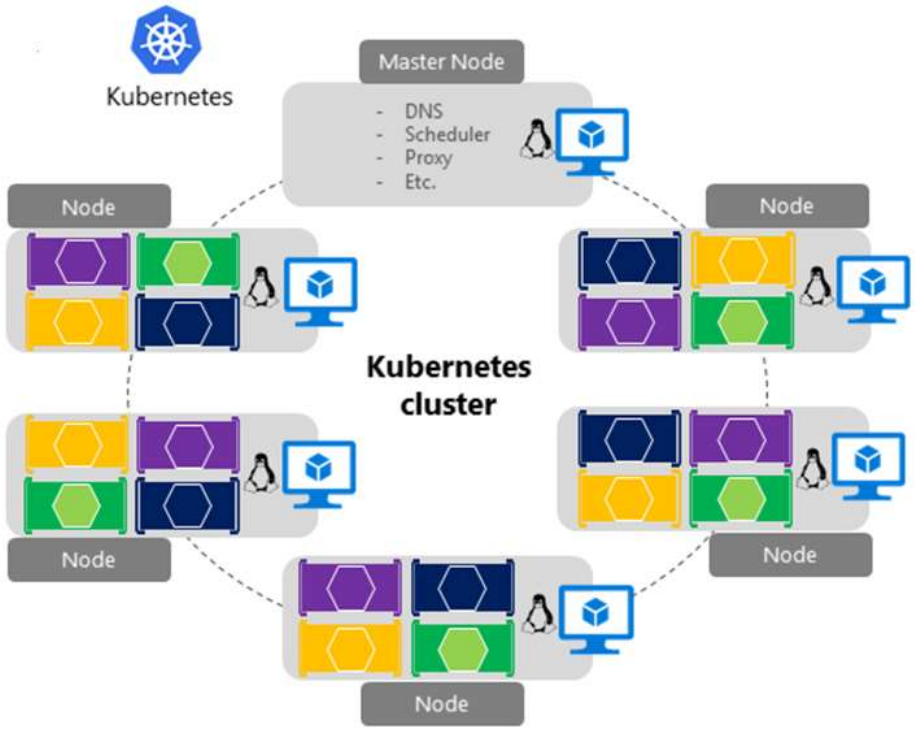
- You see Kubernetes as a managed service in the portal
- Azure sees the 1700 line configuration file

At no charge…

- Automated upgrades, patches
- High reliability, availability
- Automatic scaling
- Self-healing
- Monitoring

What is a Cluster?

- Orchestrators support cluster-based technology
  - A pool (federation) of virtual machines (nodes)
  - Forms highly-available environment
  - Start small with a few machines and can scale out to hundreds
  - Fully elastic with no single point of failure
  - Each node is aware of other nodes
  - Nodes can be heterogeneous, some small, some large



How Can You Provision an AKS Cluster?

- Azure Portal
- Azure CLI (Command Line Interface)
- Azure ARM Script

Demo: Cluster

Provision AKS Cluster

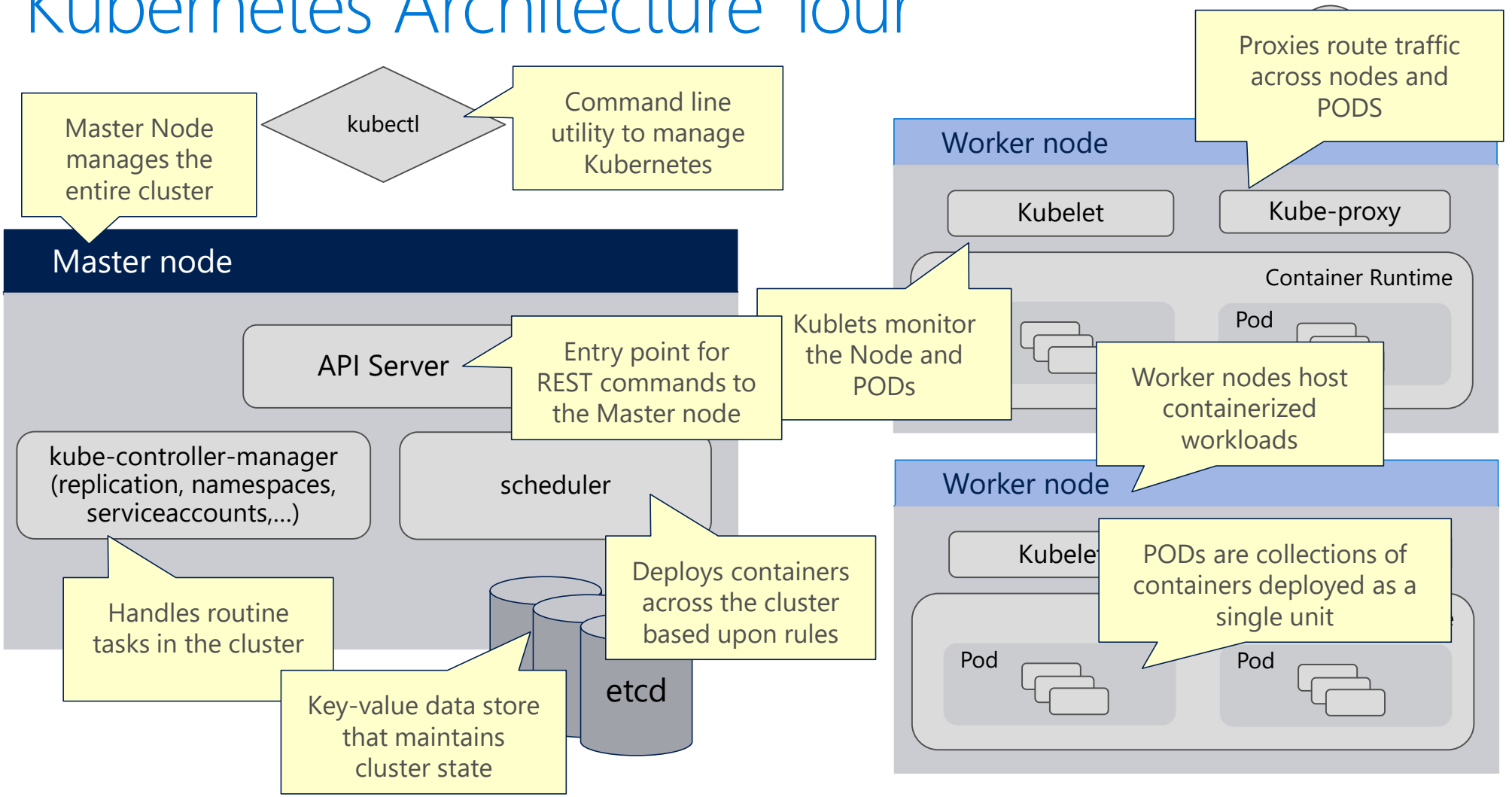
How Do You Manage AKS?

- *Kubectl* is the official command line utility needed to interact with the Kubernetes cluster and resources running upon it

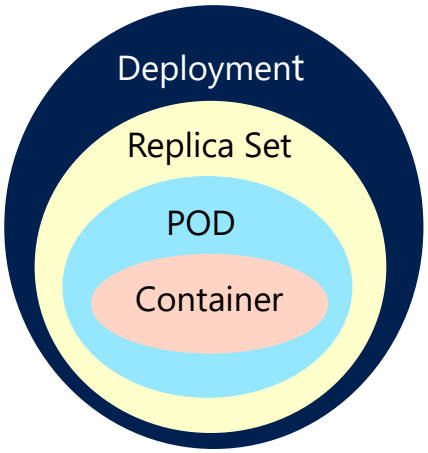
```
kubectl <command> <kubernetes object type> <object name> <flags>
```

- Kubectl connects to the cluster and calls into Kube Master APIServer
- Type of commands include…
  - Generic commands – manage Kubernetes objects
  - Cluster management commands – manage the cluster
  - Troubleshooting commands
  - Deployment commands – deployment and scaling
  - Setting commands – configuration values

Kubernetes Architecture Tour



What are Pods?



- The is smallest building block in Kubernetes…
  - A collection of co-located containers and volumes
  - Running in the same execution environment
  - Managed as a single atomic unit
- You never directly run a container, instead you run a POD
- Apps running in a POD share the same IP, port and communicate using native interprocess communication channels
- Pods are immutable - if a change is made to a pod definition, a new pod is created, and the old pod is deleted
- PODs are identified using labels (key-value pairs)…
  - app=voting-app; tier= frontend

Slide 26

JO9 I would say that they are tagged/flagged with labels but not really identified. This is there is their name to identify them

Julien Oudot, 2/25/2019

JO14 Actually, I read the K8s documentation and they also say "identify", so you are probably right :)

Julien Oudot, 3/6/2019

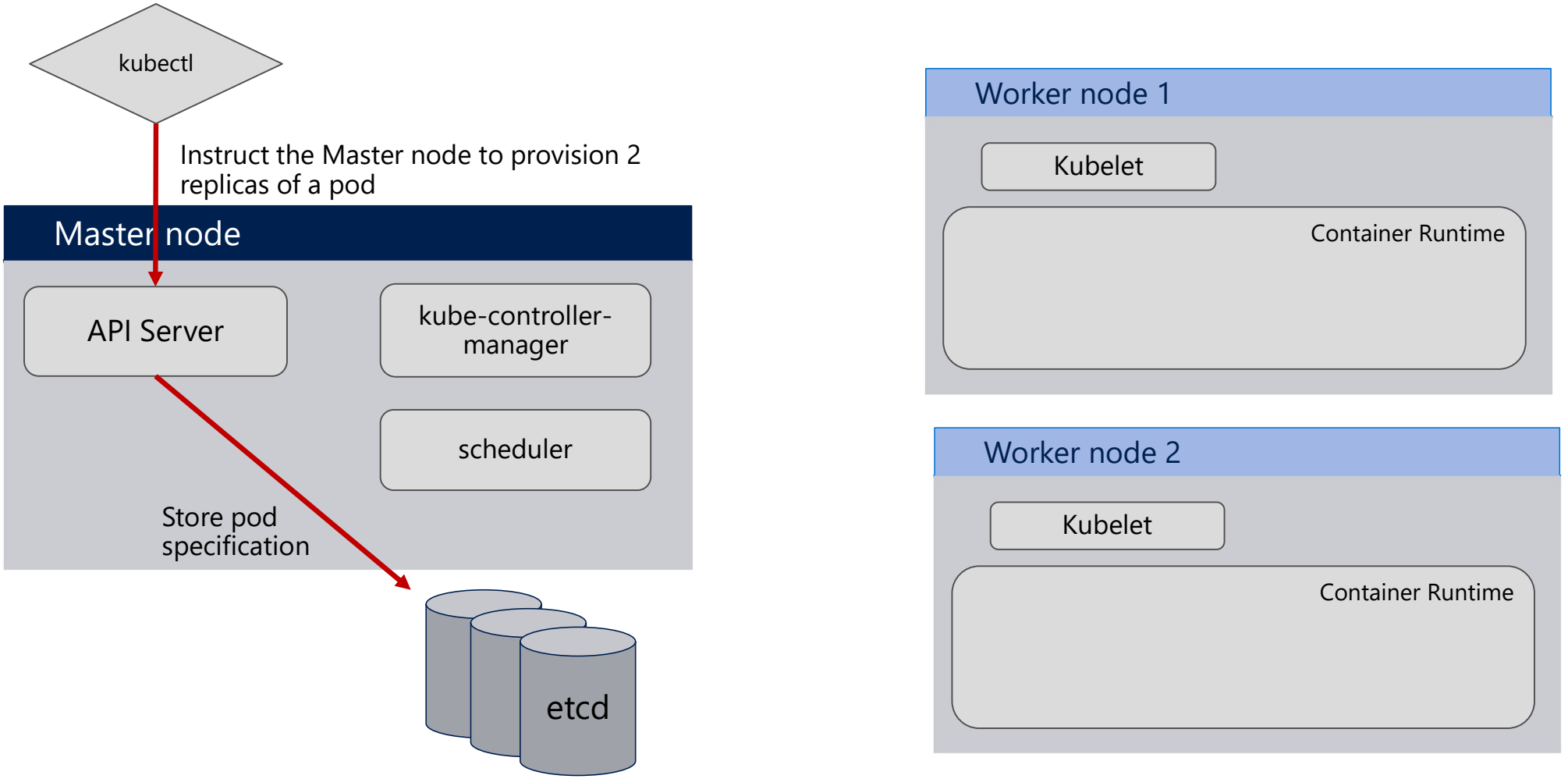
Declarative Configuration

- PODs are defined in a POD manifest: A readable, declarative text-file
- Kubernetes itself thrives on *declarative configuration*...
  - Capture the desired state of a Kubernetes object in a configuration
  - Submit that configuration to a service that takes actions to ensure the desired state becomes the actual state
  - Provides for a more manageable, dynamic and reliable system
- Contrast with *imperative configuration* where you explicitly instruct the system what to do, typically by issuing a series of commands

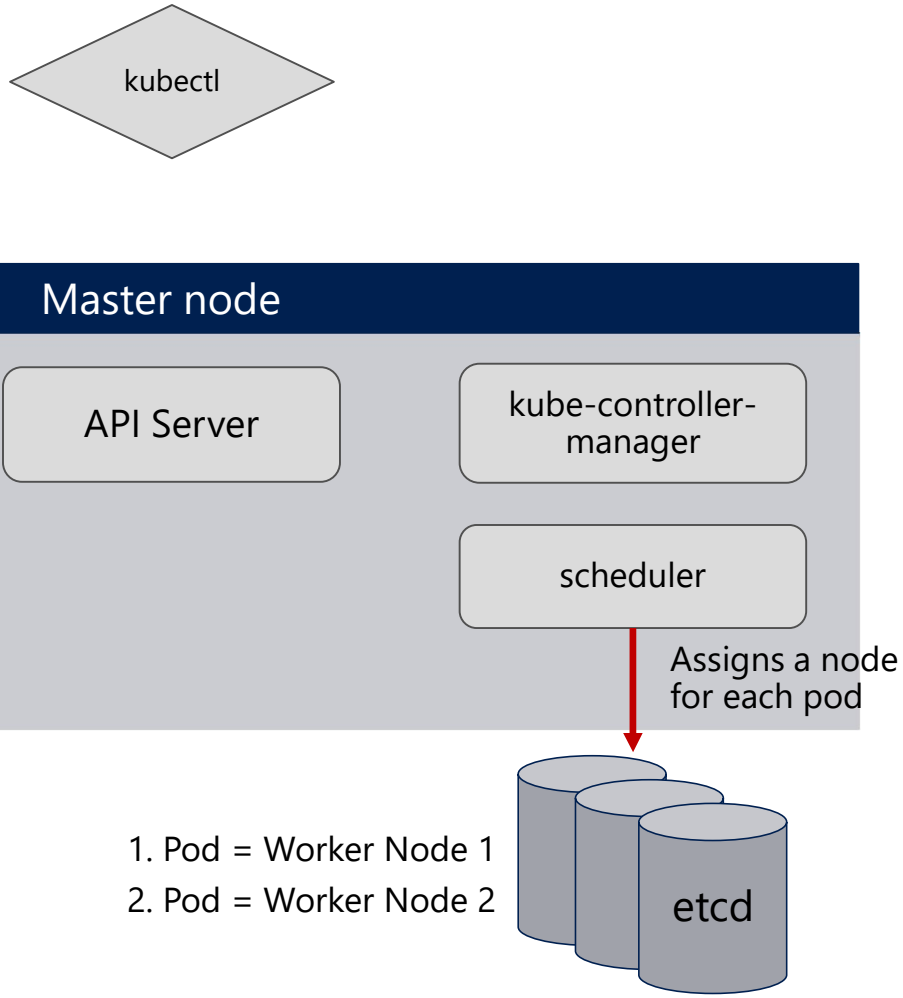
The POD Lifecycle

- Let's say you want to provision a container in Kubernetes
- From the Kubectl console, you…
  - Make a POD request to an API server using a POD definition (YAML) file
  - The API server saves the configuration data to the persistent storage (ETCD store)
  - The scheduler finds the unscheduled POD and schedules it to an available node
  - The Kubelet sees the POD scheduled and fires up Docker
  - Docker runs the container
- The Kubelet manages objects on the worker nodes
- The entire lifecycle state of the POD is stored in the ETCD store

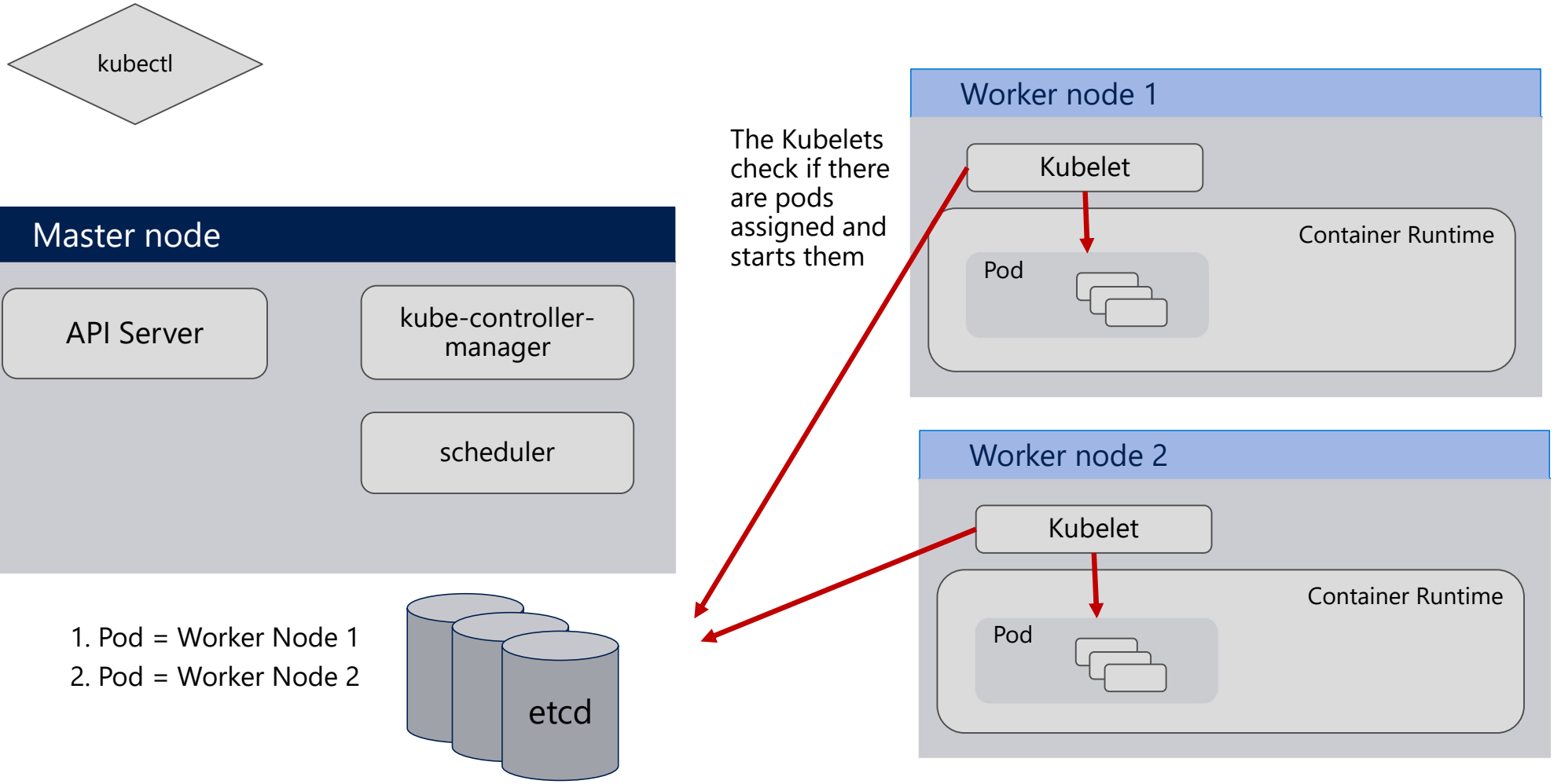
How Pods Work



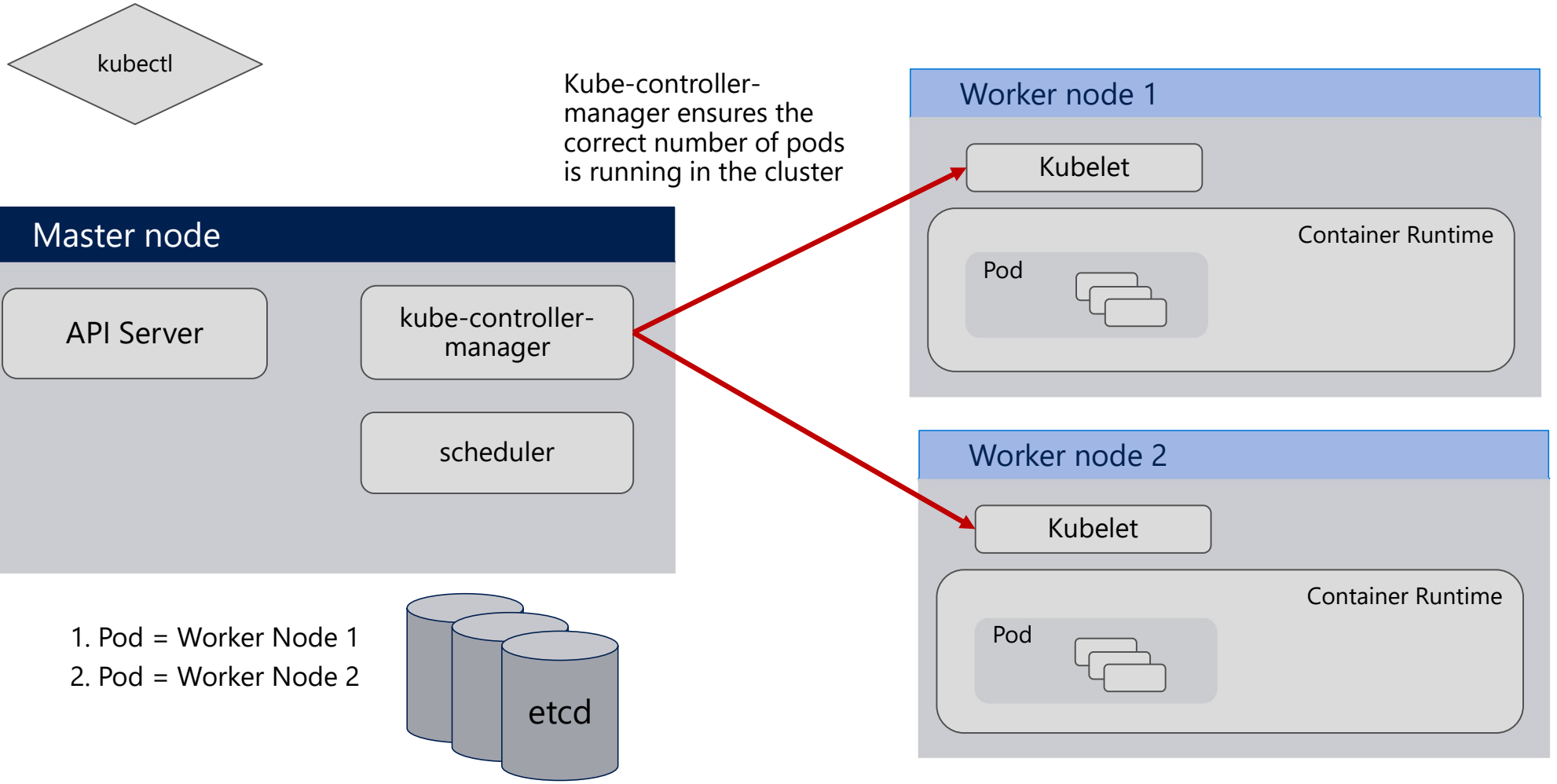
How Pods Work



How Pods Work



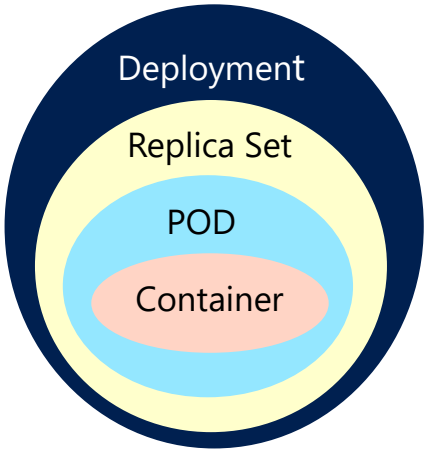
How Pods Work



Demo: Pods

Creating a POD

Labels & Selectors



- A declarative way for which to identify Kubernetes objects
- Labels…
  - Simple key-value pair
  - Mechanism to attach arbitrary but meaningful metadata to an object
  - Ways for things in Kubernetes to find other things in Kubernetes
  - A Kubernetes object can have zero to many labels
- Selectors…
  - Mechanism to filter for labels that match certain criteria or logic

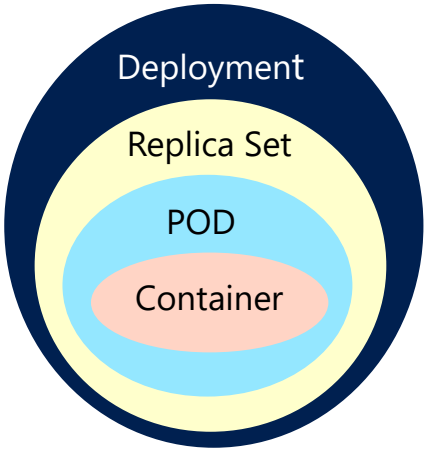
```
"labels" : {
   "tier" : "staging",
   "type" : "redis"
}
```

```
tier = staging
type != nginx
```

Demo: Labels

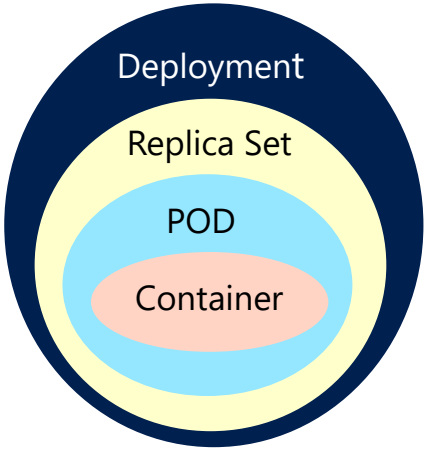
Assign a Label to a POD

What are Replica Sets?



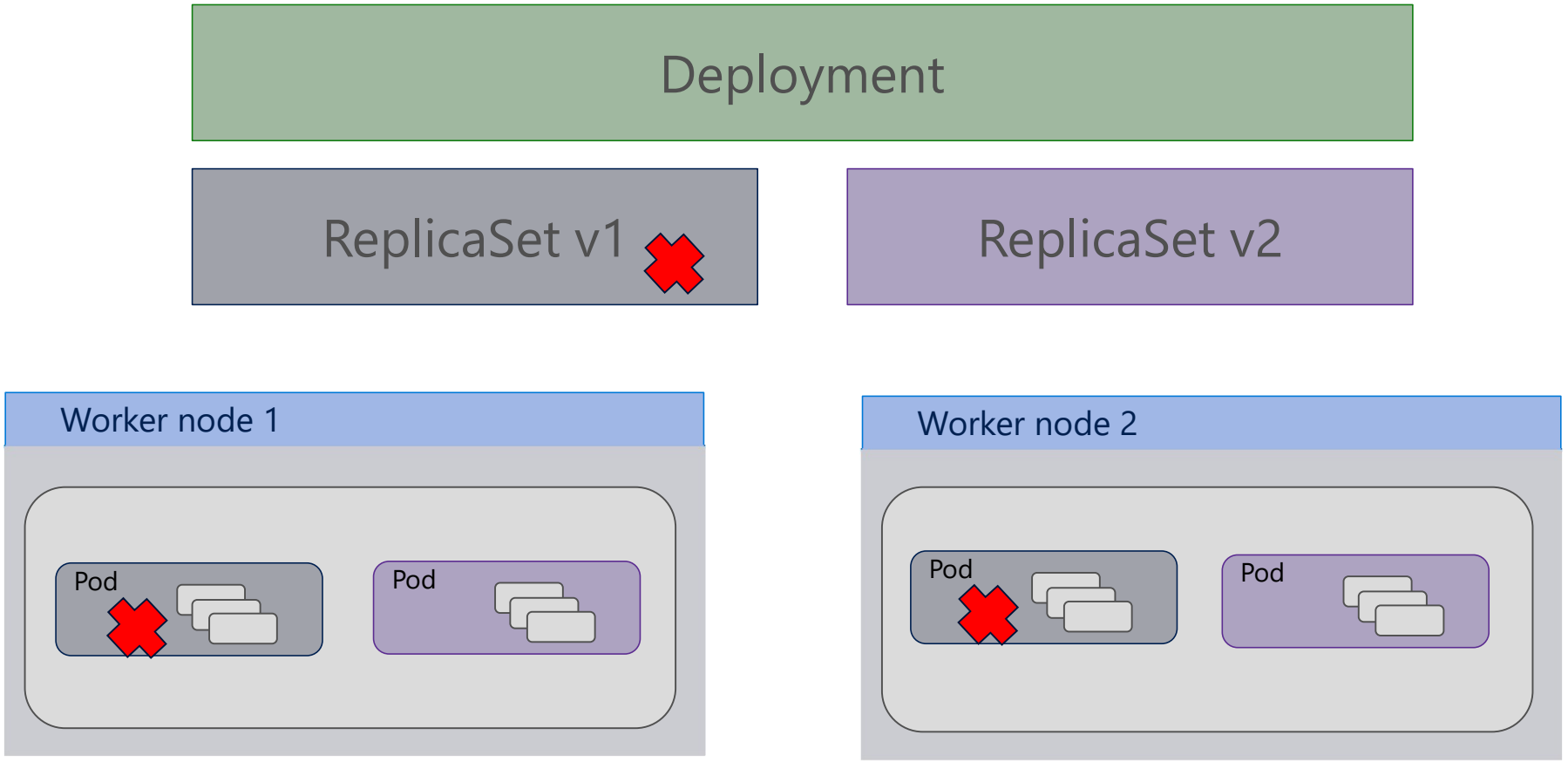
- A POD is essentially a one-off singleton instance
- *Replica Sets* are a Kubernetes object that manage PODs
  - Redundancy –allow for failure
  - Scale –allow for more requests to be processed
- They monitor the cluster and ensure the desired number of PODs are correctly running
  - If no PODs are provisioned, the Replica Set Controller will schedule them
  - If actual count drops below the desired, the controller will schedule replacements
  - If you exceed the desired count, the controller would destroy them
- Replica sets are created by and managed through Kubernetes Deployment objects

What are Deployments?



- A Kubernetes object that you use to control the running state of PODs and replicas…
  - Instruct the Kubernetes Deployment controller how to create, scale and update instances of your services
  - Provide fine-grained control over how and when a new pod version is rolled out as well as rolled back to a previous state
- The Kubernetes Deployment Controller reads the deployment and schedules the service instances onto individual nodes in the cluster
- Deployment objects abstract provisioning operations…
  - To delete a set of pods, simply delete the deployment that controls them
  - To update a set of pods, edit the deployment definition

How Deployment Works

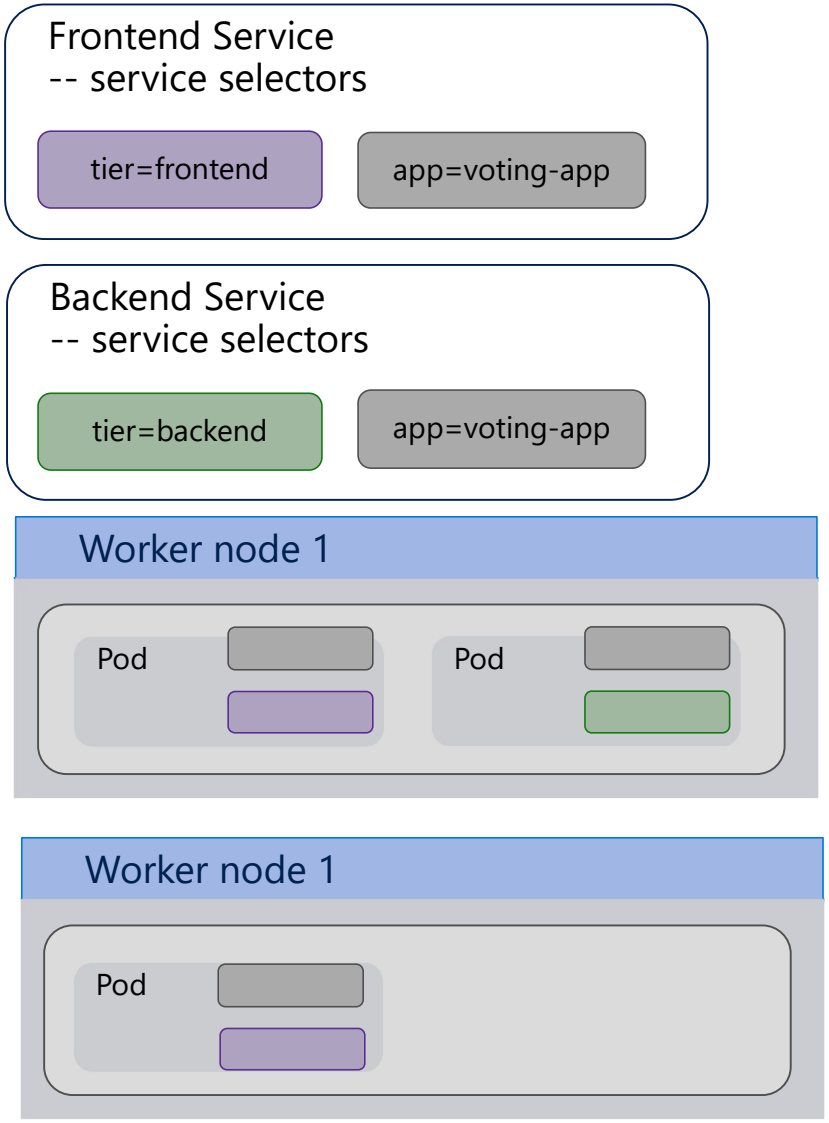


Demo:
Deployments and Replica Sets

Show Replica Sets and Deployments

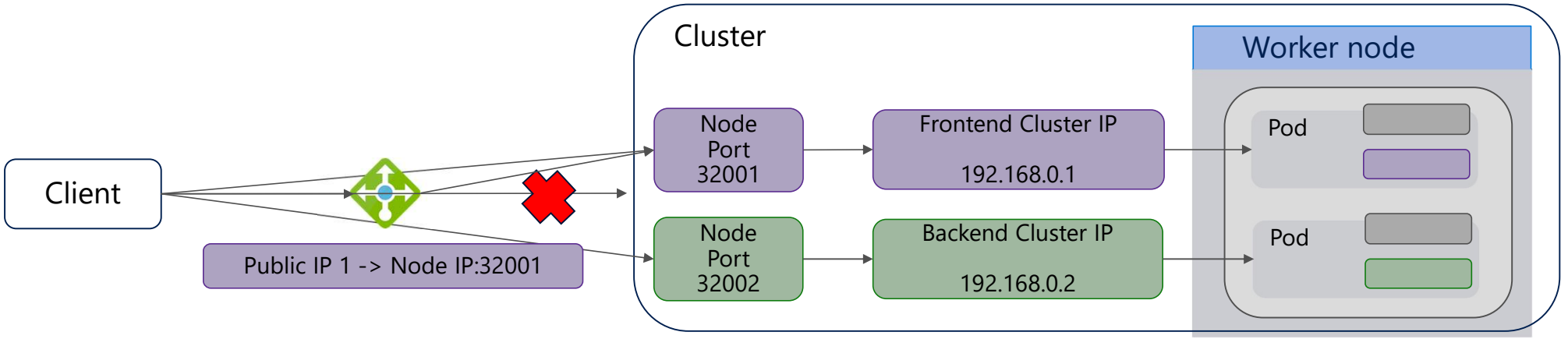
What is a Service?

- An abstraction that defines a logical set of loosely-coupled PODs and a policy by which to access them
  - Defined with a YAML markup file
  - Use “selectors” to define which pods to include
- Load balance traffic to PODs
  - Expose a frontend service to a public load balancer using selectors to bind to a specific POD
    - labels app=voting-app
    - tier=frontend

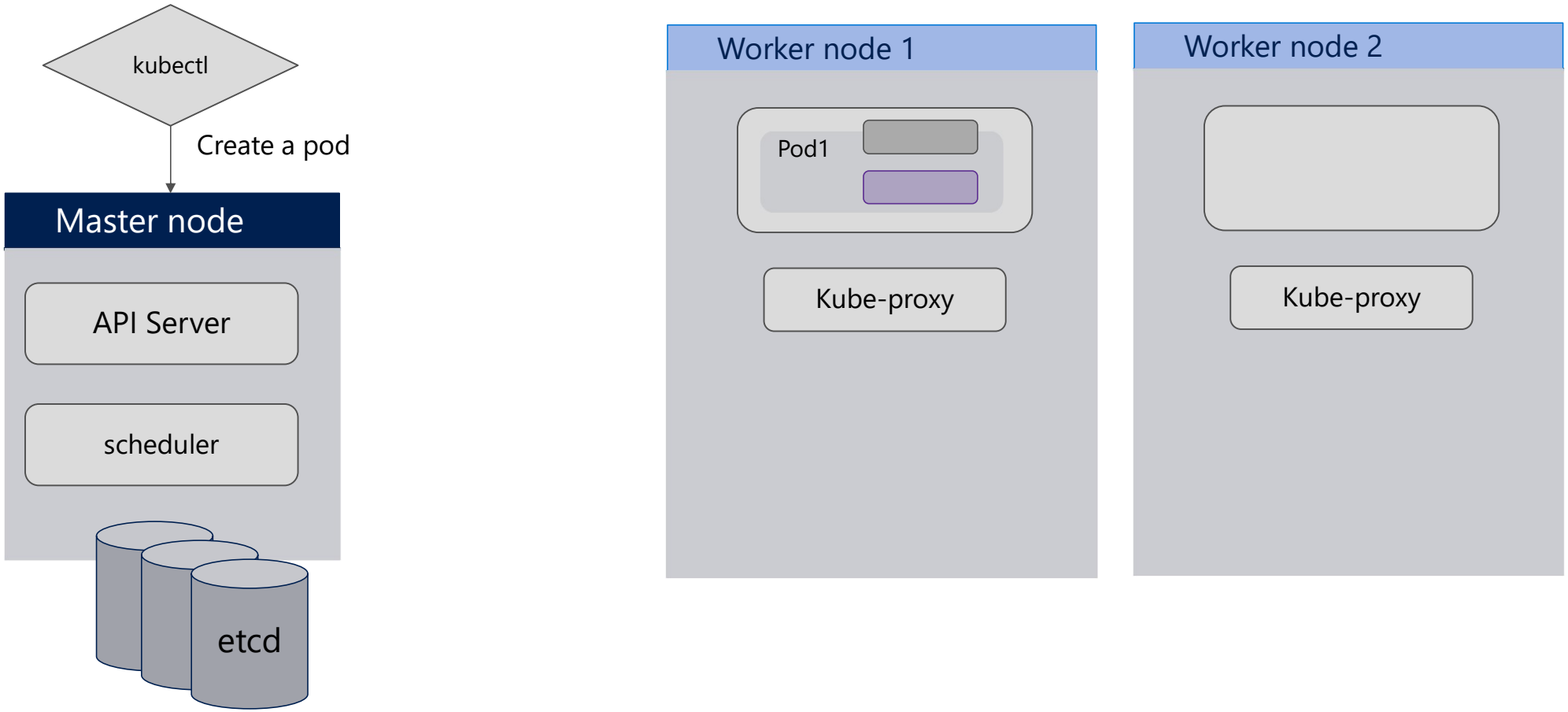


Service Object Types

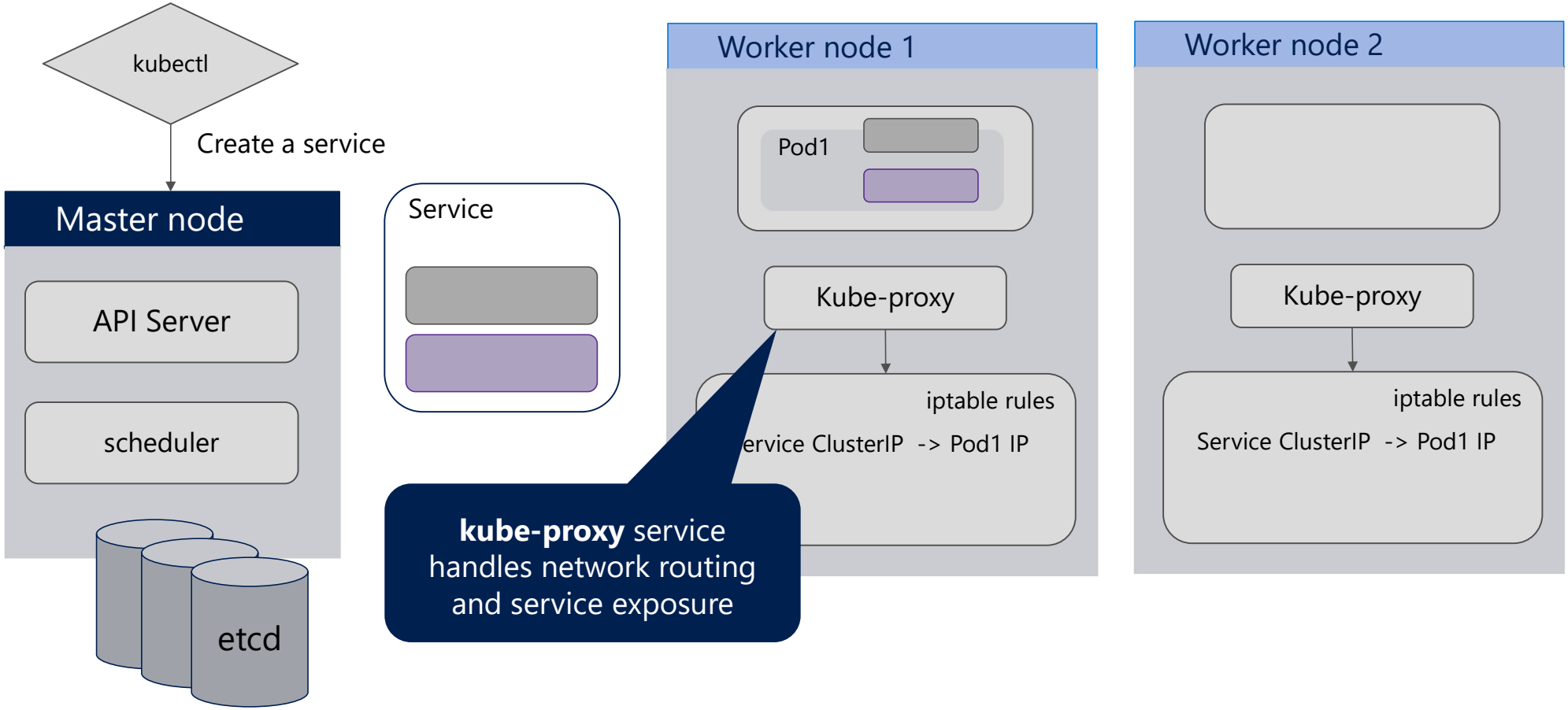
- PODs are not exposed outside of the immediate cluster without a Service object – they allow PODs to receive traffic
- There are different types of services that expose your pod in different ways
  - *ClusterIP*: Provides a single IP internal to the cluster to represent a set of pods
  - *NodePort*: Reserves a specified port on the node to represent a set of pods
  - *LoadBalancer*: Creates a public-facing, load-balanced IP address



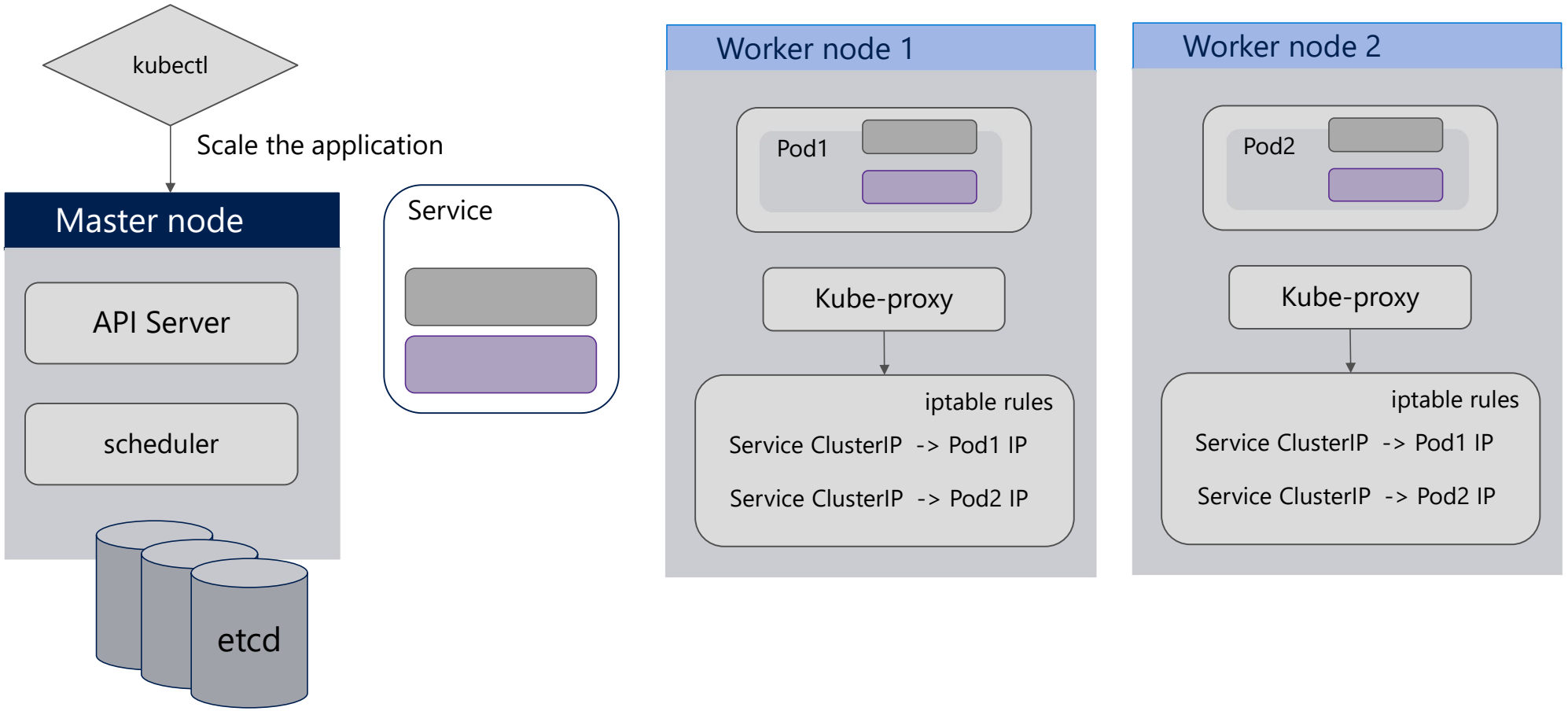
How Services Work



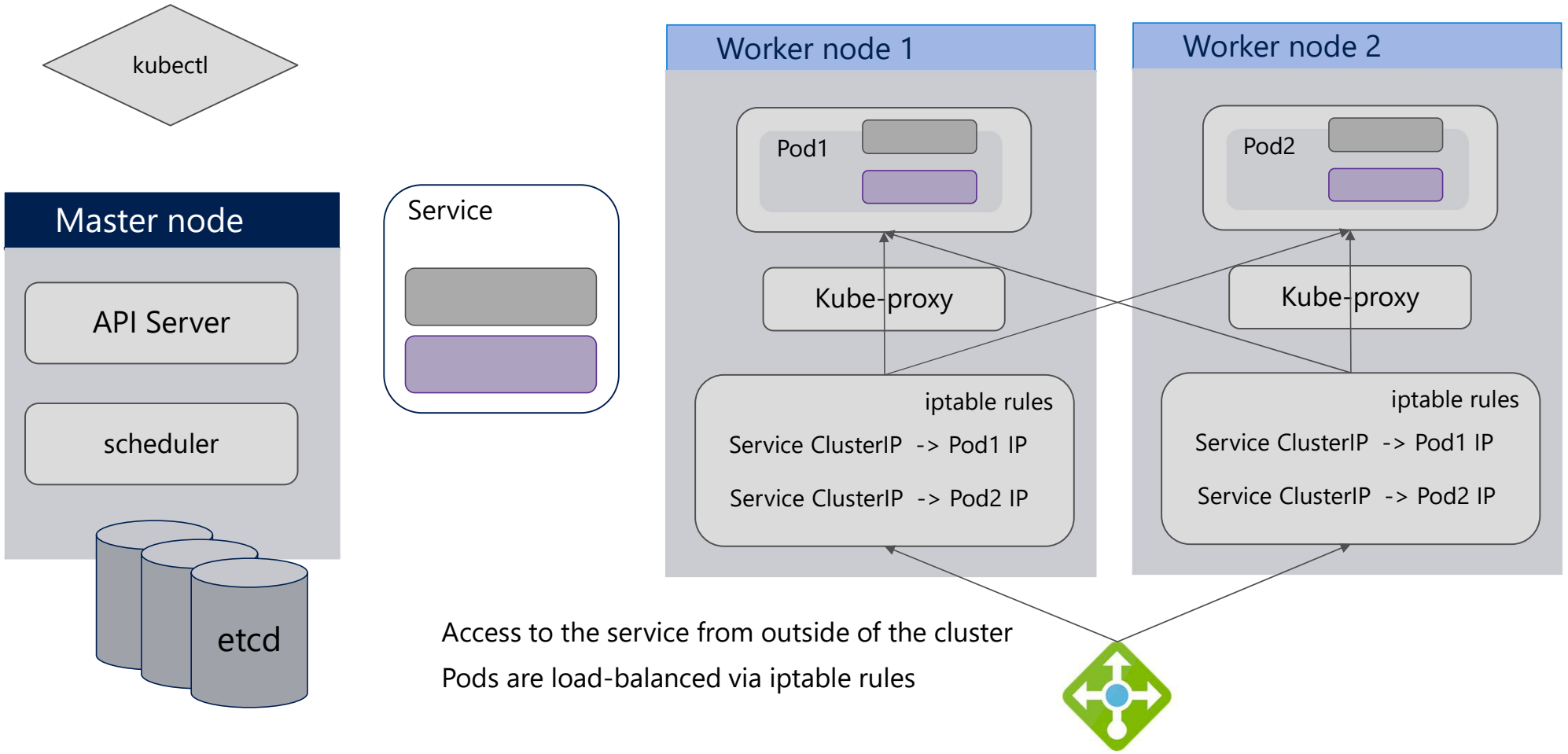
How Services Work



How Services Work



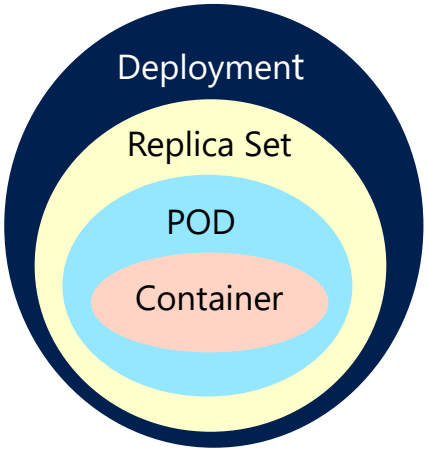
How Services Work



Demo: Services

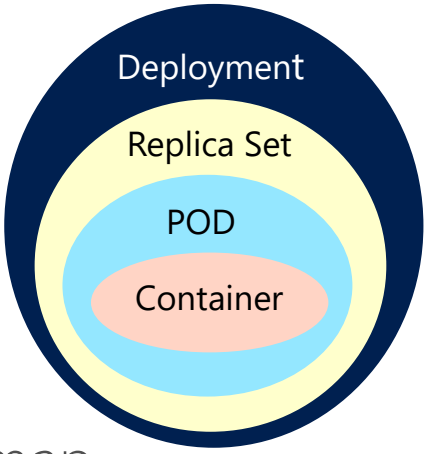
Kubernetes Services

What are ConfigMaps & Secrets?



- Expose configuration information to pods as environment variables
- Assigned at runtime as key-value pairs or contained in file
- Enables the same image to be re-used across environments
- Secrets contain sensitive information that is encoded
  - Example: Connection Strings, Passwords
- ConfigMaps contain non-sensitive information
  - Example: The name of a backend component

What are Volumes?



- Enable persistent data outside of a POD
- Deleting or restarting a POD destroys any persistent data stored in the containers file system
- Follows same concepts as in traditional containers- volumes map that data to the underlying host
- There are different types of volumes:
  - Volume- mounts a volume on the underlying node
  - Persistent Volume- defines a static, external volume (e.g. Azure file, Azure disk)
  - Persistent Volume Claim (PVC)- reserves a set of memory from a Persistent Volume

Demo:

Volumes and Secrets

Microsoft

Kubernetes Fundamentals

Networking

Microsoft Services

Basic networking

Uses kubenet network plugin and has the following features

Nodes and Pods are placed on different IP subnets

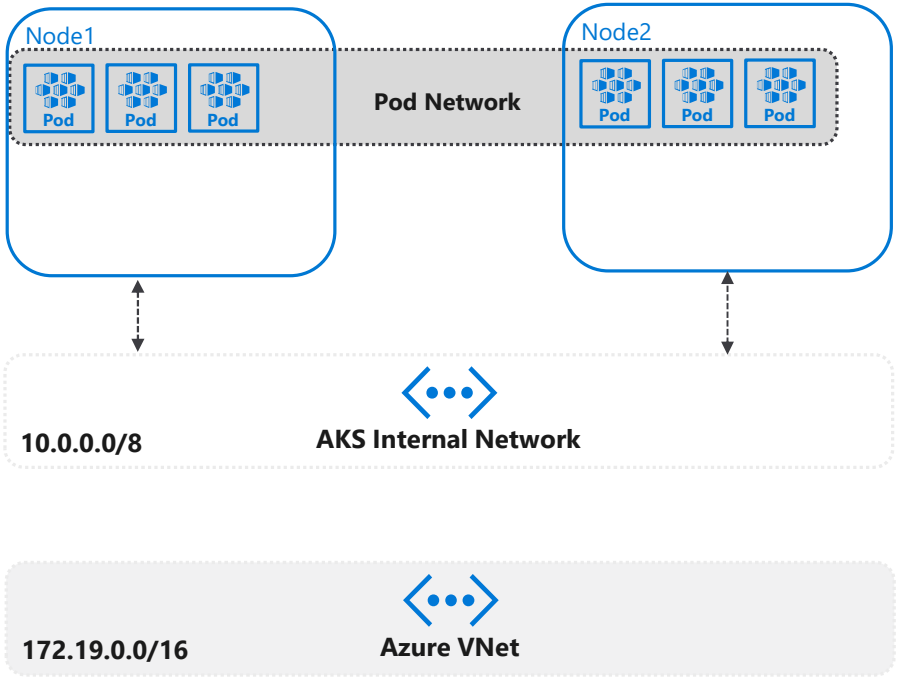
User Defined Routing and IP Forwarding is for connectivity between Pods across Nodes.

Drawbacks

2 different IP CIDRs to manage

Performance impact

Peering or On-Premise connectivity is hard to achieve



Advanced networking

Uses the Azure CNI (Container Networking Interface)

CNI is a vendor-neutral protocol, used by container runtimes to make requests to Networking Providers

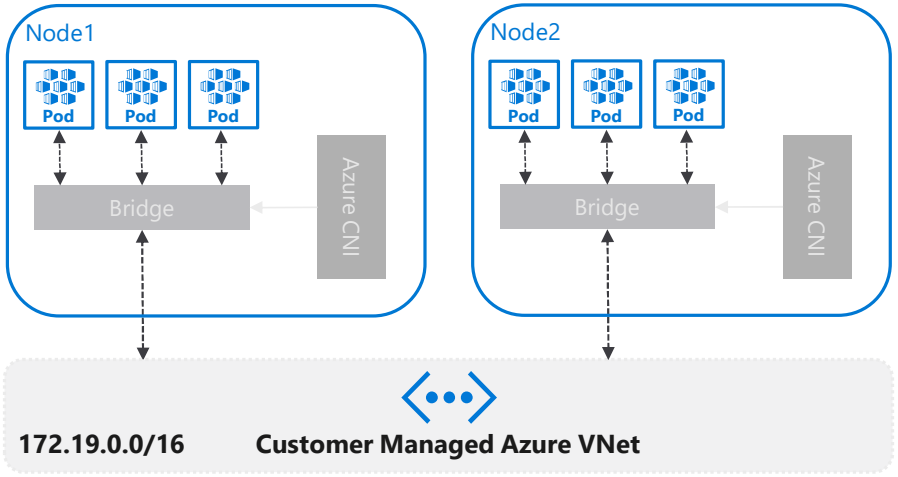
Azure CNI is an implementation which allows you to integrate Kubernetes with your VNET

Advantages

Single IP CIDR to manage

Better Performance

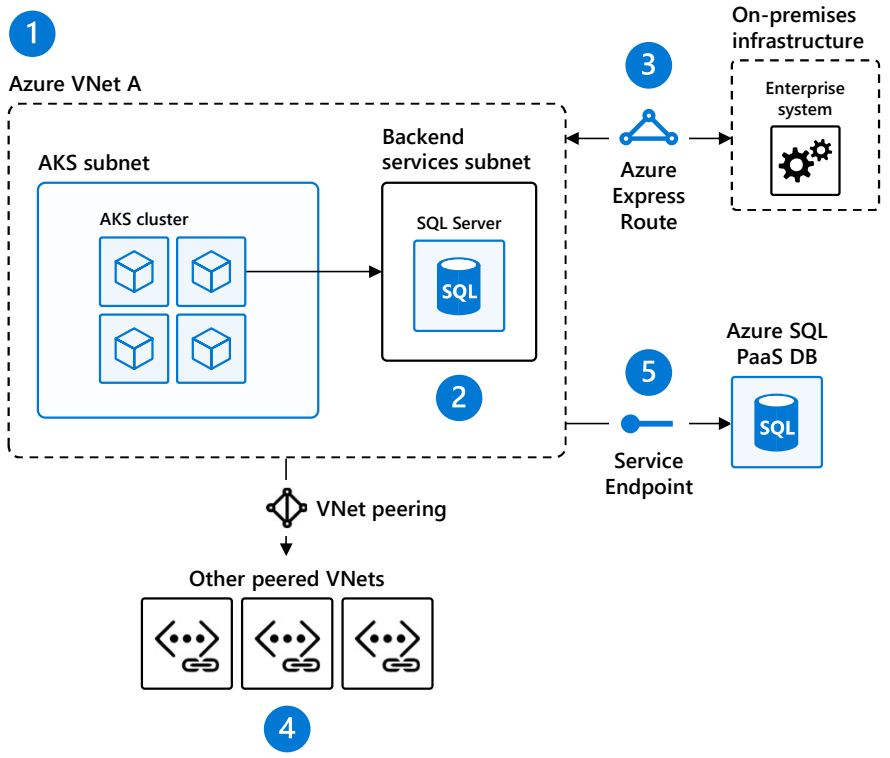
Peering and On-Premise connectivity is out of the box



Scenarios enabled by Advanced Networking

1. Uses Azure subnet for both your containers and cluster VMs
2. Allows for connectivity to existing Azure services in the same VNet
3. Use Express Route to connect to on-premises infrastructure
4. Use VNet peering to connect to other VNets
5. Connect AKS cluster securely and privately to other Azure resources using VNet endpoints

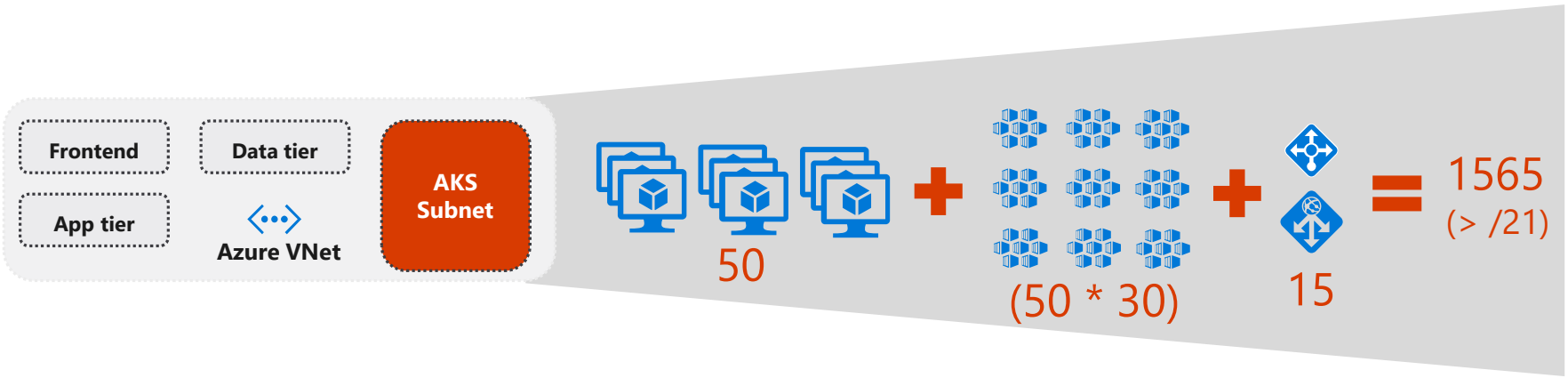
AKS VNet integration works seamlessly with your existing network infrastructure



Advanced networking: Prerequisites

- The virtual network for the AKS cluster must allow outbound internet connectivity.
- Don't create more than one AKS cluster in the same subnet.
- AKS clusters may not use 169.254.0.0/16, 172.30.0.0/16, or 172.31.0.0/16 for the Kubernetes service address range.
- The service principal used by the AKS cluster must have at least Network Contributor permissions on the subnet within your virtual network. If you wish to define a custom role instead of using the built-in Network Contributor role, the following permissions are required:
  - Microsoft.Network/virtualNetworks/subnets/join/action
  - Microsoft.Network/virtualNetworks/subnets/read

Advanced networking: Planning IP addressing for your cluster



Additional subnets

Kubernetes service address range

- Non-overlapping with other subnets in your network (does not need to be routable) and not 169.254.0.0/16, 172.30.0.0/16 and 172.31.0.0/16.
- Smaller than /12

Docker bridge address

- Non-overlapping with AKS Subnet

Service, Ingress and Ingress Controllers

Service

Logical set of Pods and a policy by which to access them.
The set of Pods targeted by a Service is (usually) determined by a Label Selector.

Services provide important features that are standardized across the cluster: load-balancing (L4), service discovery between applications, and features to support zero-downtime application deployments.

Ingress

A Kubernetes API that manages external access to the services in the cluster

- Supports HTTP and HTTPs
- Path and Subdomain based routing
- SSL Termination
- Save on Public IPs

Ingress Controller

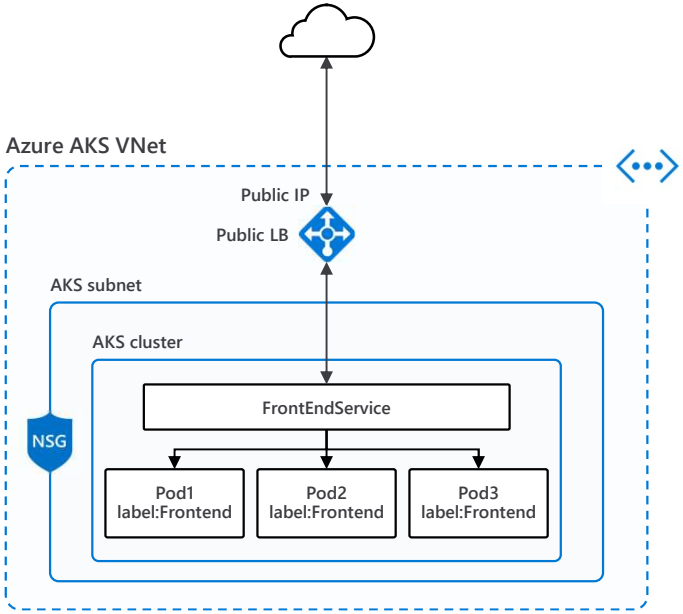
A daemon, deployed as a Kubernetes Pod, that watches the Ingress Endpoint for updates. Its job is to satisfy requests for ingresses.

Some popular Ingress Controllers include nginx, Traefik and HAProxy.

Public Service

- Service Type LoadBalancer
- Basic Layer4 Load Balancing (TCP/UDP)
- Each service as assigned an IP on the ALB

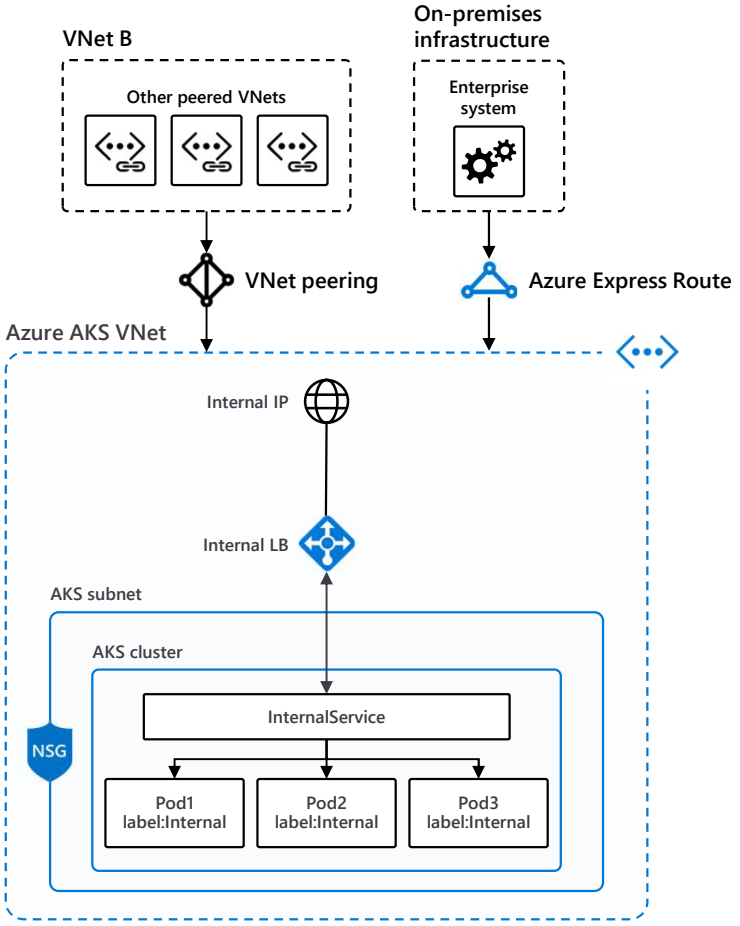
```
apiVersion: v1
kind: Service
metadata:
  name: frontendservice
spec:
  type: LoadBalancer
  ports:
  - port: 80
  selector:
    app: frontend
```



Internal Service

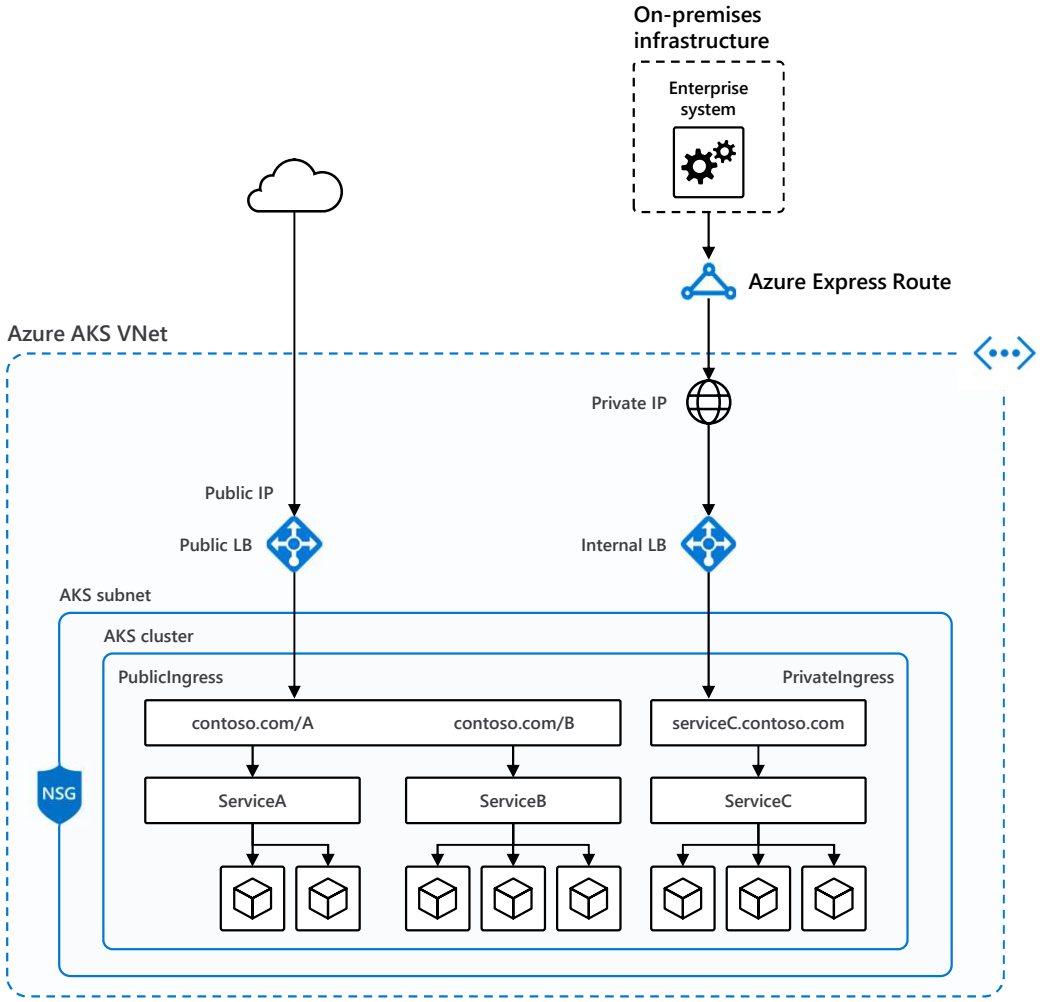
- Used for internal services that should be accessed by other VNETs or On-Premise only

```
apiVersion: v1
kind: Service
metadata:
  name: internalservice
  annotations:
    service.beta.kubernetes.io/azure-load-balancer-internal: "true"
spec:
  type: LoadBalancer
  ports:
  - port: 80
  selector:
    app: internal
```

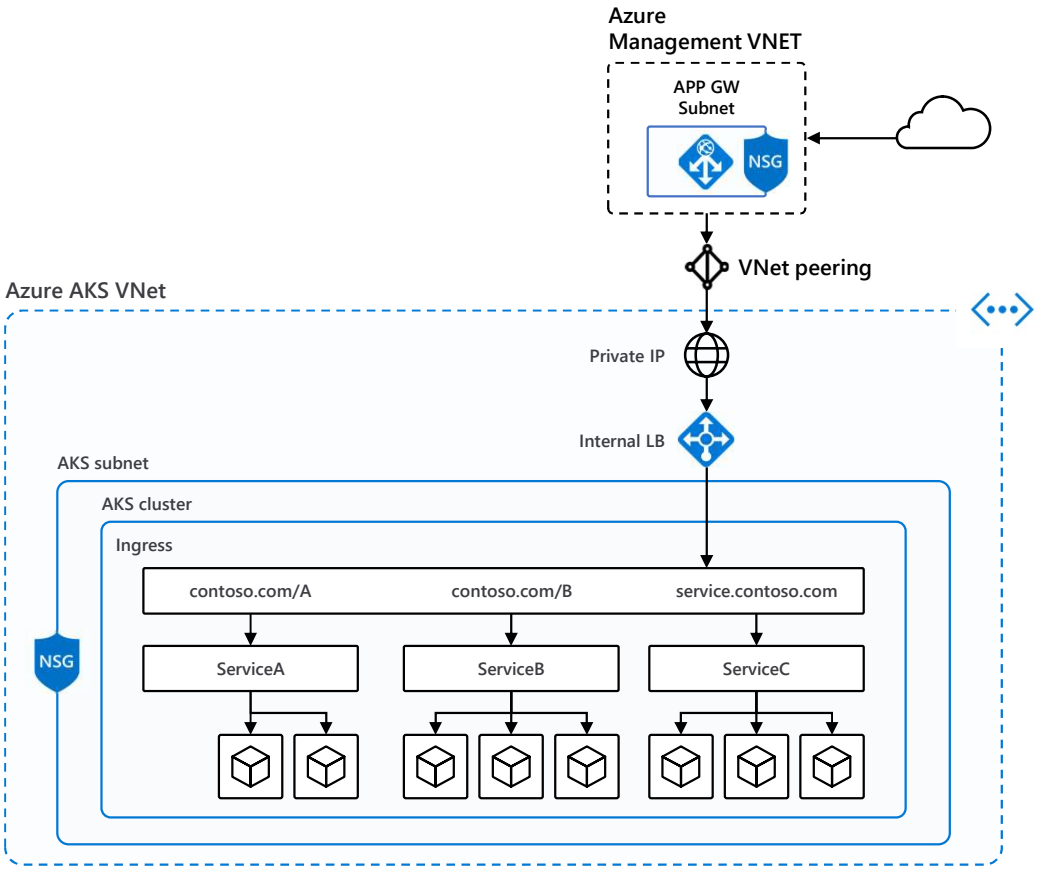


Ingress

```
kind: Ingress
metadata:
 name: contoso-ingress
   annotations: kubernetes.io/ingress.class: PublicIngress
spec:
 tls:
 - hosts:
   - contoso.com
   secretName: contoso-secret
 rules:
   - host: contoso.com
     http:
      paths:
      - path: /a
        backend:
         serviceName: servicea
         servicePort: 80
      - path: /b
        backend:
         serviceName: serviceb
         servicePort: 80
```



Securing Kubernetes Services with WAF

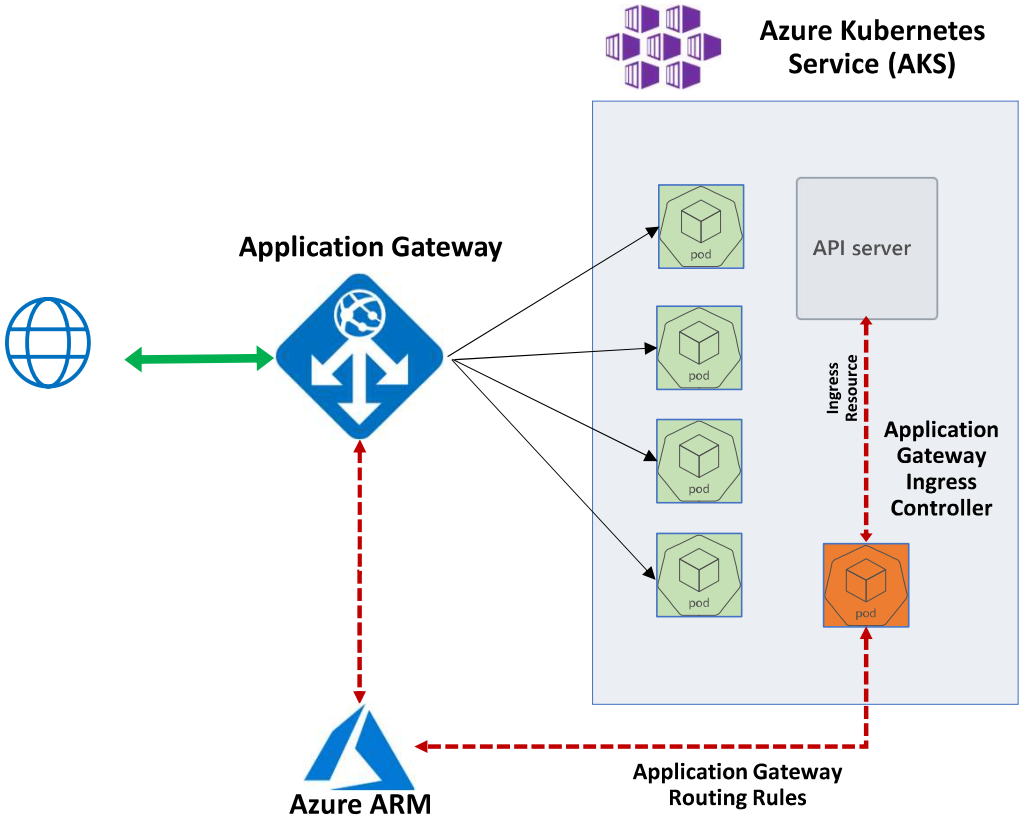


Application Gateway Ingress Controller

Preview

- Attach Application Gateways to AKS Clusters
- Load Balance from the Internet to pods
- Supports features of k8s ingress resource – TLS, multi-site and path-based routing
- Pod-AAD for ARM authentication

https://github.com/Azure/application-gateway-kubernetes-ingress



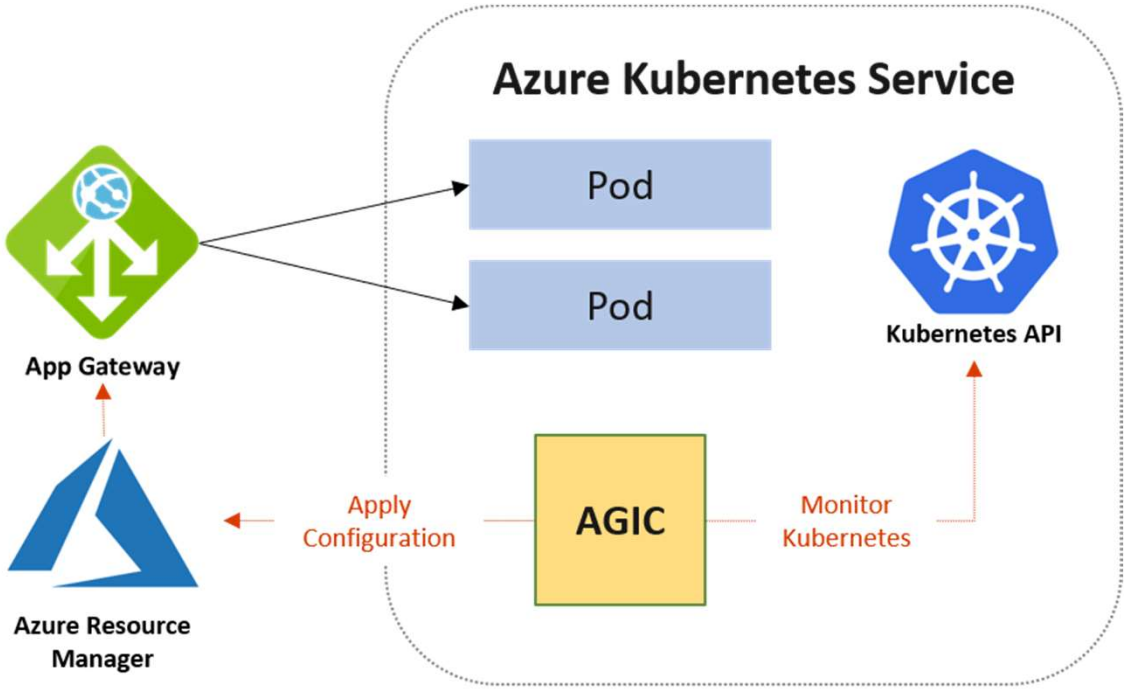
Preview

Application Gateway Ingress Controller

- Attach Application Gateways to AKS Clusters
- Load Balance from the Internet directly to pods

Features

- URL routing
- Cookie-based affinity
- TLS termination
- End-to-end TLS
- Support for public, private, and hybrid web sites
- Integrated web application firewall

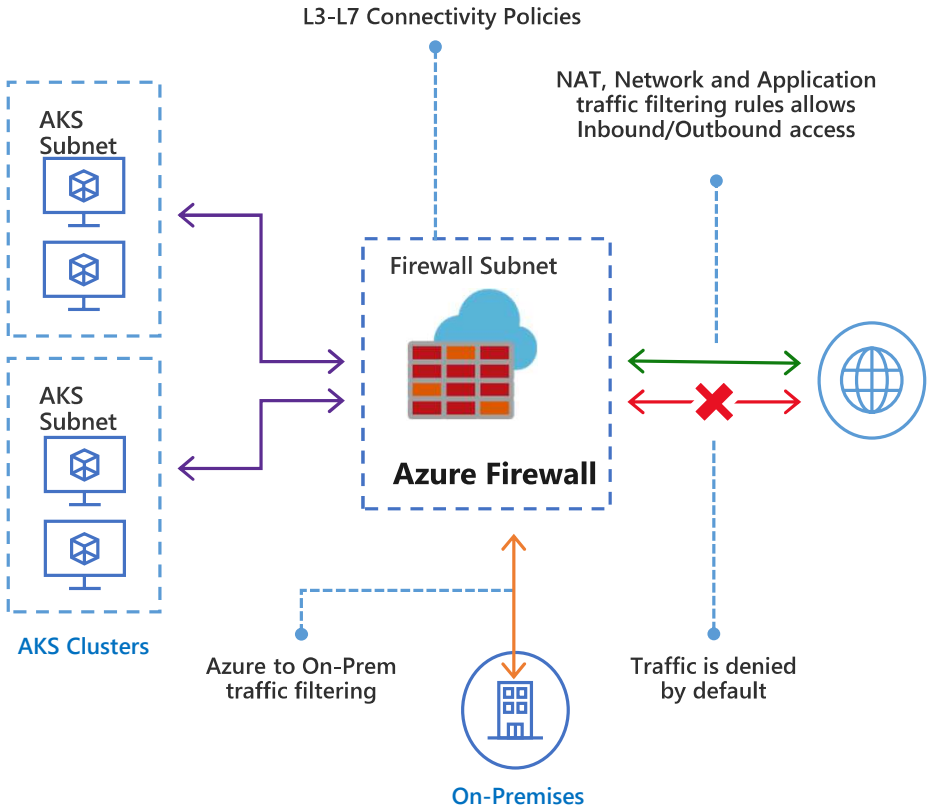


North/South Traffic in AKS with Azure Firewall

Cloud native stateful Firewall as a service

A first among public cloud providers

- Target FQDN
  - *.azmk8s.io – http, https (Eg. *eastus.azmk8s.io)
  - k8s.gcr.io – http, https
  - storage.googleapis.com - http, https
  - *auth.docker.io – http, https
  - *cloudflare.docker.io – http, https
  - *registry-1.docker.io – http, https
- NAT Rules
  - TCP Port 22 to AKS Destination Addresses for the Region



Network policy

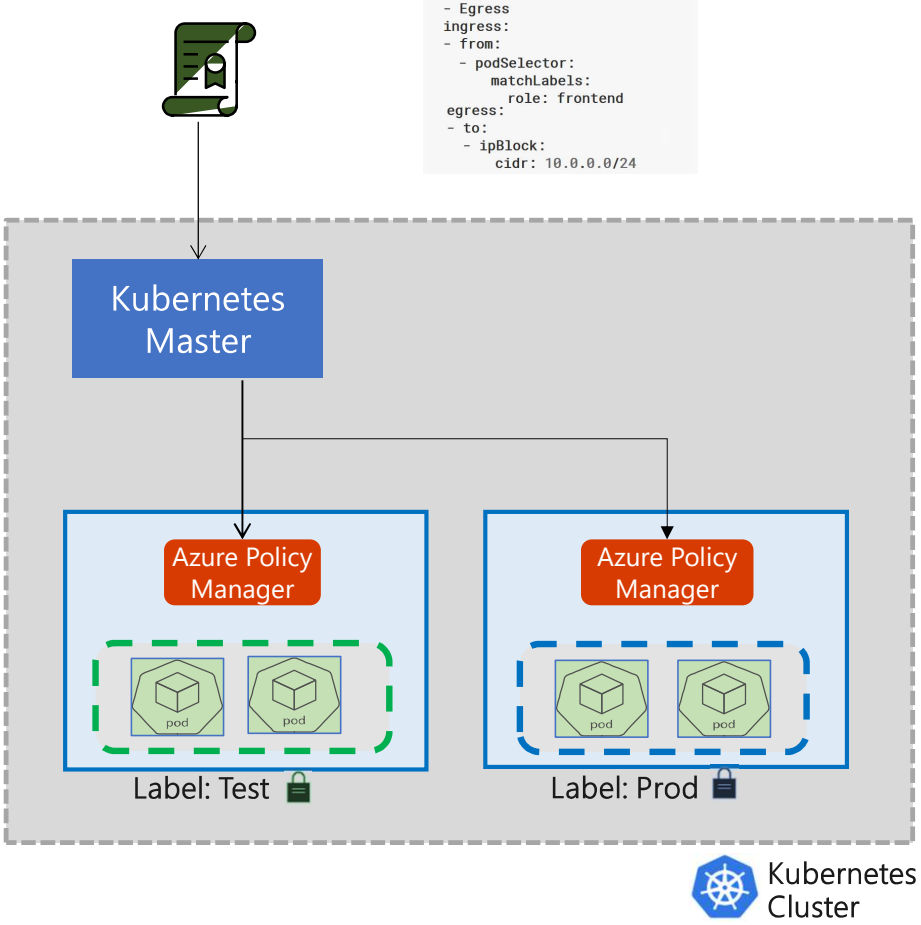
- All Pods are non-isolated by defaults.
- Flat network. All pods can talk to other pods
- Accept traffic from anyone
- Multi stage/zone project this could expose security risks.
  - 3 tier webapp.
  - Front end could technically talk to DB

Azure Kubernetes Network Policies

- Provides micro-segmentation for containers – like NSGs for VMs
- Label-based selection of Pods
- Policy resource yaml file specifies Ingress and Egress policies
  - Policies defined for a label
- Works in conjunction with Azure CNI
- Supports Linux hosts
- Supported in *aks-engine*
  - Set `networkPolicy` setting to `azure` in cluster definition file
- Supported on AKS in **Preview**

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: test-network-policy
spec:
  podSelector:
    matchLabels:
      role: db
  policyTypes:
  - Ingress
  - Egress
  ingress:
  - from:
    - podSelector:
        matchLabels:
          role: frontend
  egress:
  - to:
    - ipBlock:
        cidr: 10.0.0.0/24
```

```
kubectl apply –f
policy.yaml
```



Network policy

- Pod Selector
- PolicyTypes
  - Ingress, egress

Ingress

- namespaceSelector
- podSelector
- ipBlock

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: test-network-policy
  namespace: default
spec:
  podSelector:
    matchLabels:
      role: db
  policyTypes:
  - Ingress
  - Egress
  ingress:
  - from:
    - ipBlock:
        cidr: 172.17.0.0/16
        except:
        - 172.17.1.0/24
    - namespaceSelector:
        matchLabels:
          project: myproject
    - podSelector:
        matchLabels:
          role: frontend
    ports:
    - protocol: TCP
      port: 6379
  egress:
  - to:
    - ipBlock:
        cidr: 10.0.0.0/24
    ports:
    - protocol: TCP
      port: 5978
```

Network policy

- Recommend to create default policies that apply to all pods
  - Default deny all ingress traffic
  - Default deny all egress traffic.
  - Allow Specific Traffic (Ingress & Egress)

Network policy

- Network Policy is provided by CNI implementation
- Current support of Network policy
  - Azure Network Policy (Supported in AKS)
  - Calico (Supported in AKS)
  - Cilium
  - KubeRouter (Works in AKS)
  - Romana
  - WeaveNet

When to use Azure NSG vs. Kubernetes Network policy

- Azure Network Security Groups
  - Use it to filter North-South traffic, that is, traffic entering and leaving your cluster subnet

- Kubernetes Network Policies
  - Use it to filter East-West traffic, that is, traffic between pods in your cluster

Summary

- Use AKS Advanced networking for seamless integration with your VNET
- Use Ingress and Ingress controllers for HTTP and HTTPs services
- Use Azure Application Gateway or any other alternative from the Azure Market place to secure your services using a WAF
- Use Bastion Hosts to access your nodes when needed

Microsoft

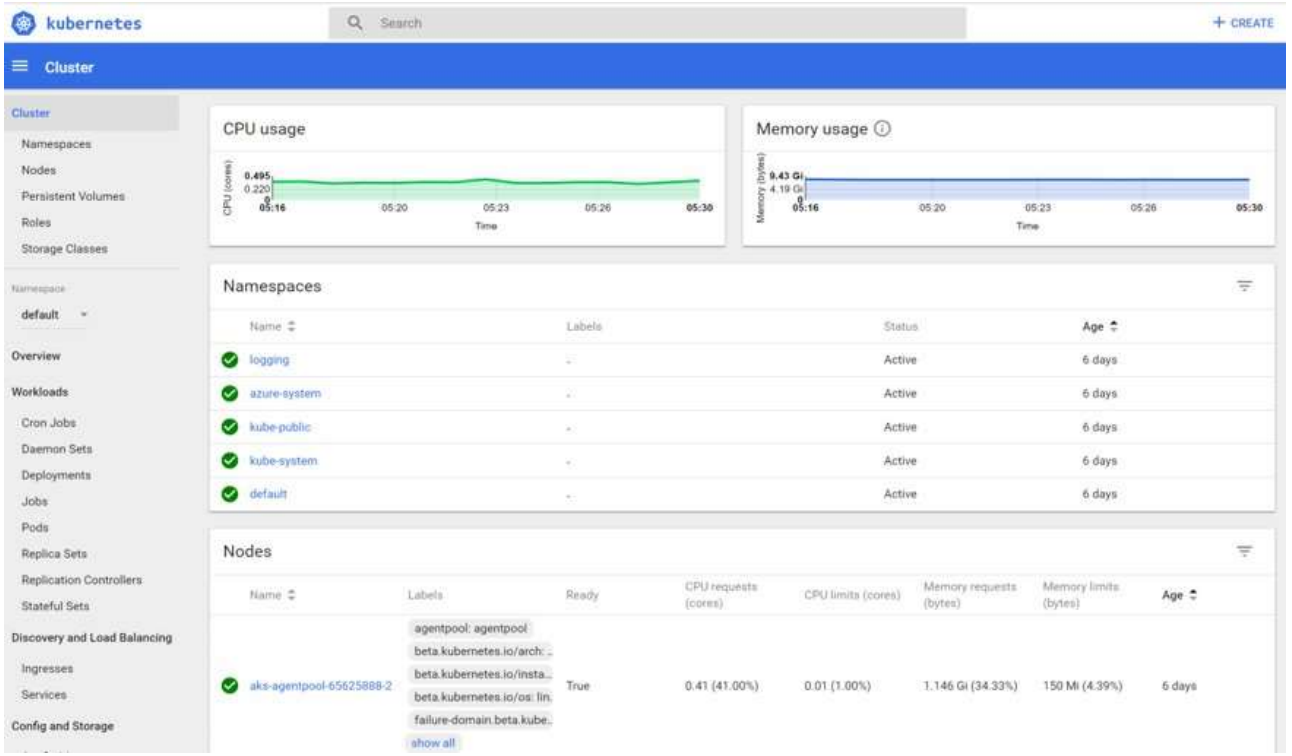
Kubernetes Fundamentals

Kubernetes Dashboard

Microsoft Services

Kubernetes Dashboard

- General purpose, web-based UI for Kubernetes clusters
- Allows users to manage applications running in the cluster and troubleshoot them, as well as manage the cluster itself
- Very basic means of deploying and interacting with those resources



Deploy the Kubernetes Dashboard

To deploy Dashboard, execute the following command:

kubectl apply -f https://raw.githubusercontent.com/kubernetes/dashboard/v2.0.0-beta1/aio/deploy/recommended.yaml

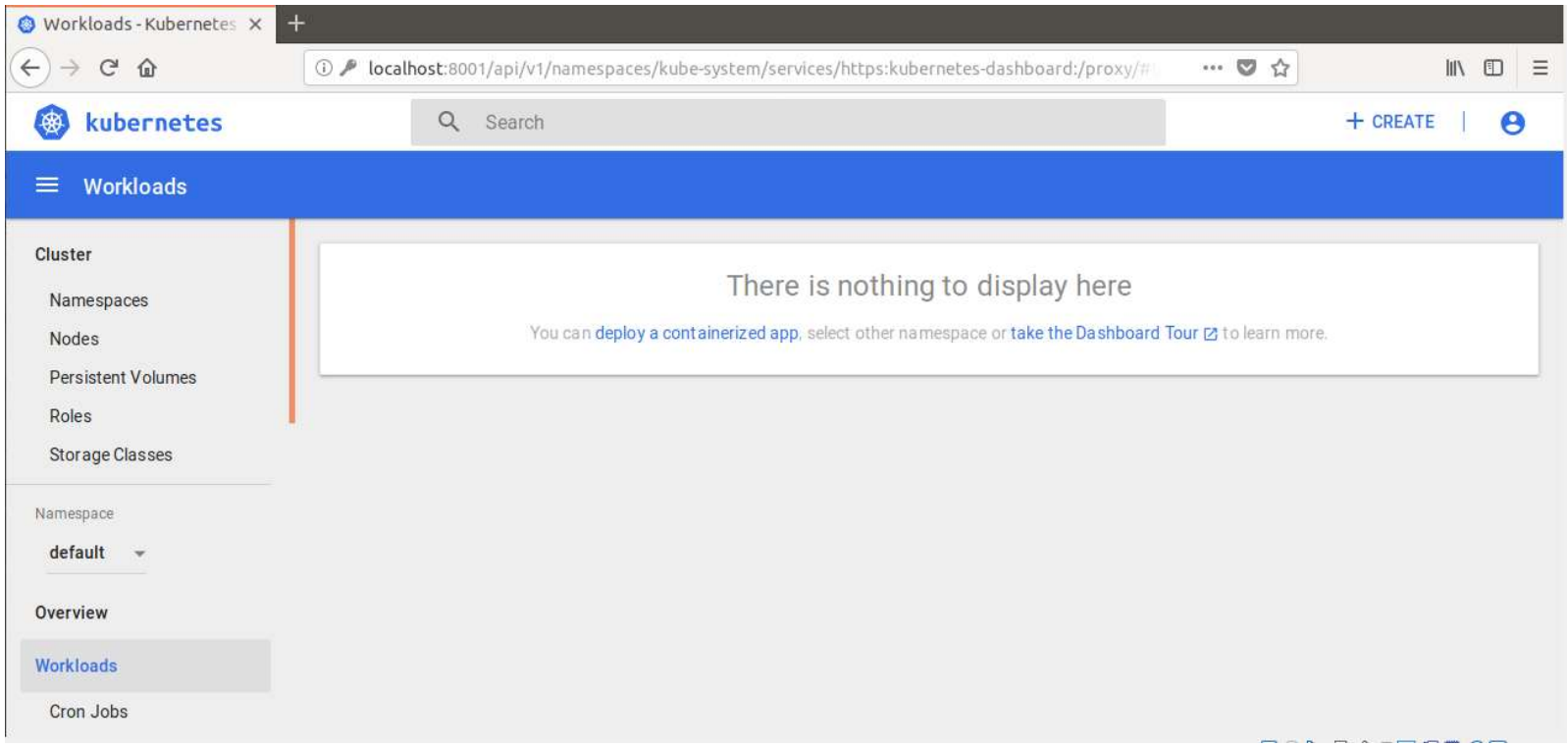
To access Dashboard from your local workstation, you must create a secure channel to your Kubernetes cluster by running the following command:

Kubectl proxy

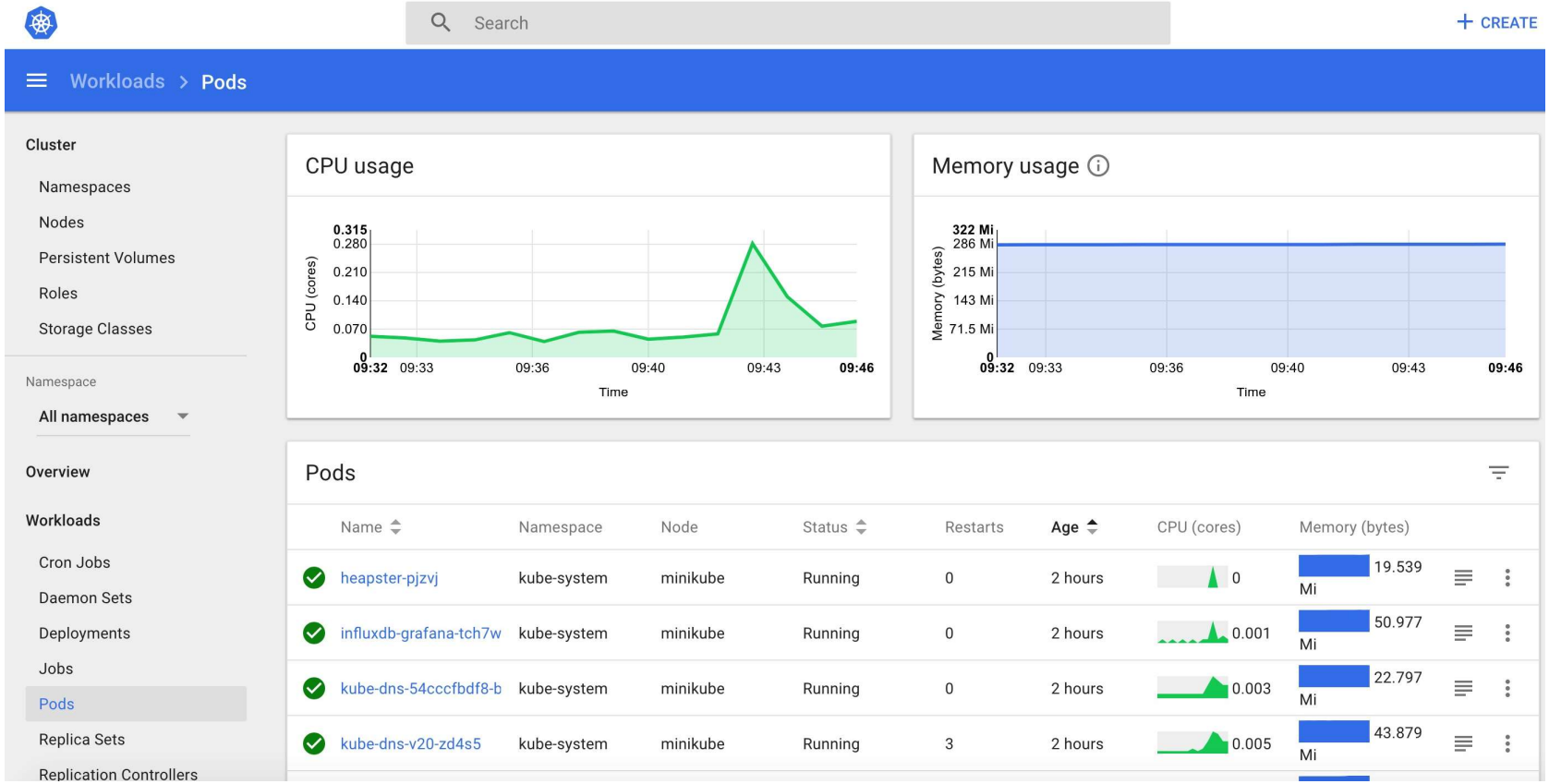
Kubectl will make Dashboard available at

http://localhost:8001/api/v1/namespaces/kubernetes-dashboard/services/https:kubernetes-dashboard:/proxy/

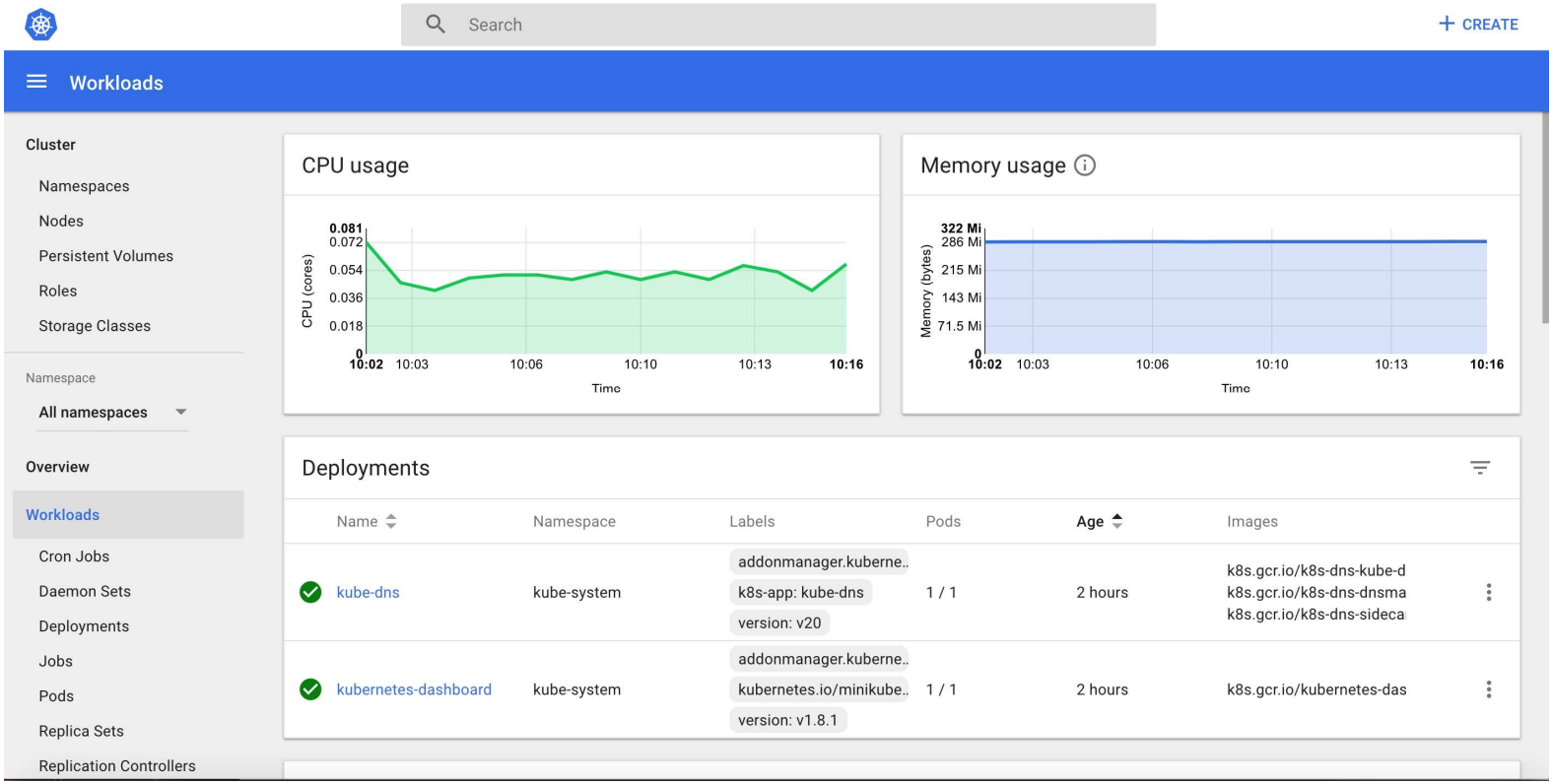
Kubernetes Dashboard



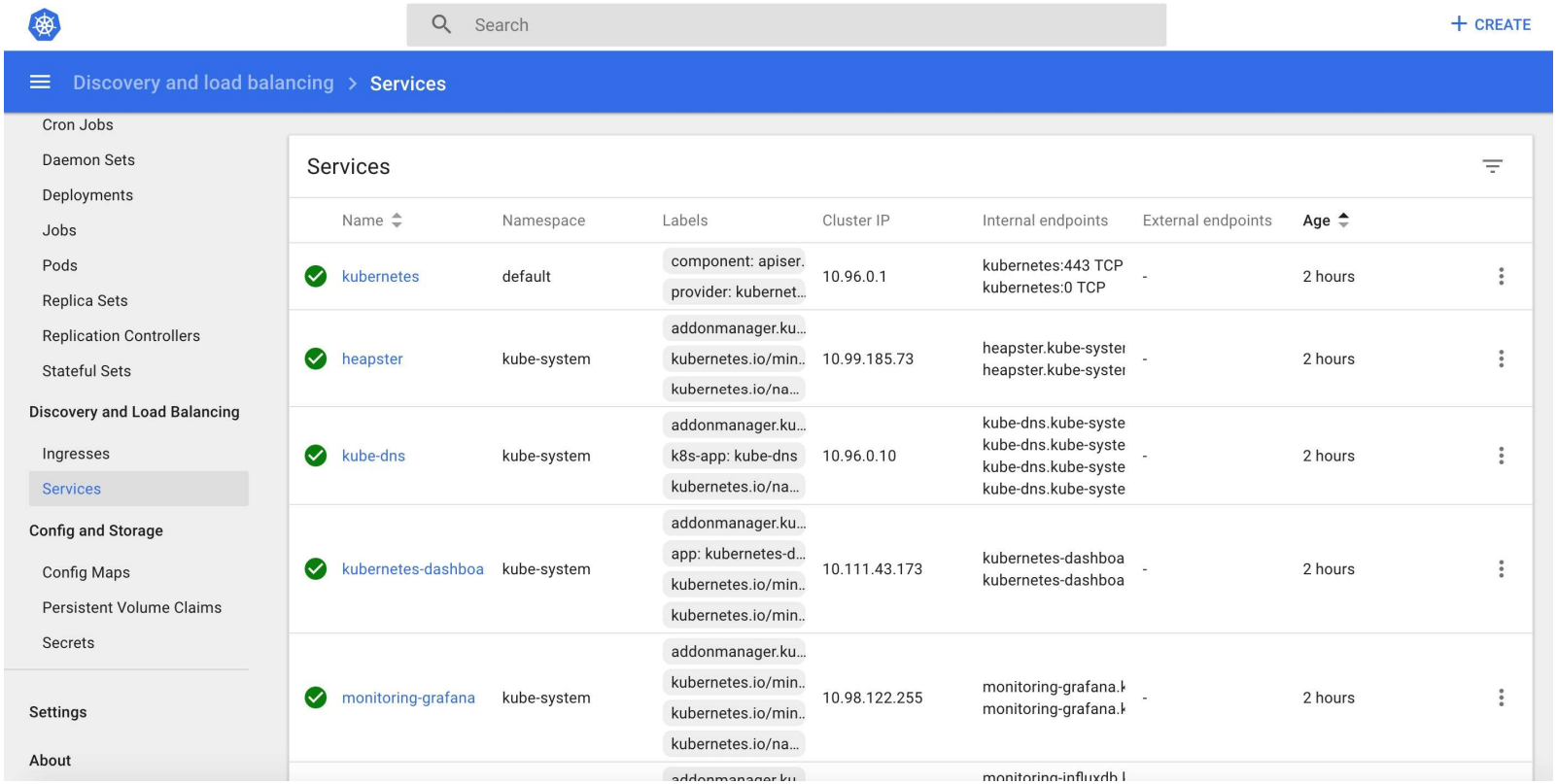
Kubernetes Dashboard



Kubernetes Dashboard



Kubernetes Dashboard



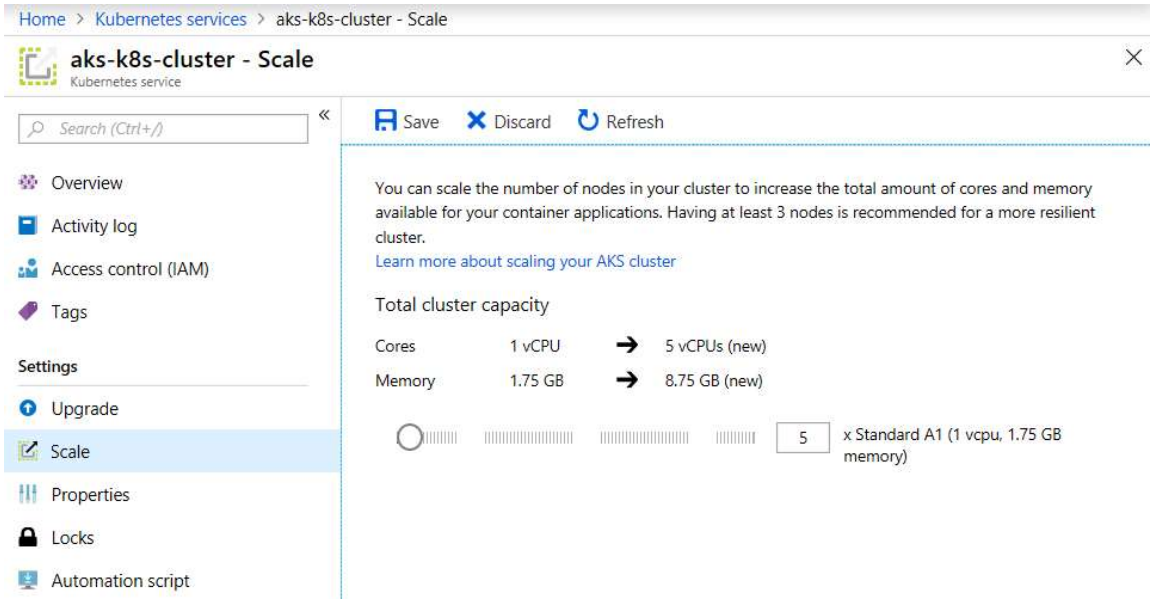
Demonstration: *Dashboard*

Create a Pod

AKS – Scaling Nodes

- Perform manually…
  - From the AZ CLI
  - From the Portal
  - Adding a node takes several minutes to complete

```
az aks scale --resource-group=myResourceGroup --name=myAKSCluster --node-count 3
```



AKS – Autoscale Nodes

- Cluster Autoscale (preview) allows your cluster to grow to meet demand based on constraints that you set
  - The Autoscaler scans the cluster periodically to check for pending pods or empty nodes and increases the size if possible.
  - By default, the Autoscaler scans for pending pods every 10 seconds and removes a node if it's unneeded for more than 10 minutes.
  - Autoscaler will not scale down the node running a pod

AKS - Scaling Pods

- Manually scale pods: You can change the replicas count (--replicas) to increase or decrease the number of pods.

```
aks scale --resource-group=myResourceGroup --name=myAKSCluster --node-count 3
```

- Autoscale pods: Kubernetes supports horizontal pod autoscaling to adjust the number of pods in a deployment depending on CPU utilization or other select metrics.

```
kubectl scale --replicas=5 deployment/azure-vote-front
```

- Autoscale requires Metrics Server to provide resource utilization to Kubernetes

AKS - Autoscale Pods

- Pods must have CPU requests and limits defined
- Kubectl autoscale command takes parameters to let autoscale increase/decrease the pods.
- Example…
  - Configure with Kubectl…

```
kubectl autoscale deployment azure-vote-front --cpu-percent=50 --min=3 --max=10
```

  - When CPU utilization exceeds 50%, the autoscaler increases the pods up to a maximum of 10 instances.

```
resources:
  requests:
     cpu: 250m
  limits:
     cpu: 500m
```

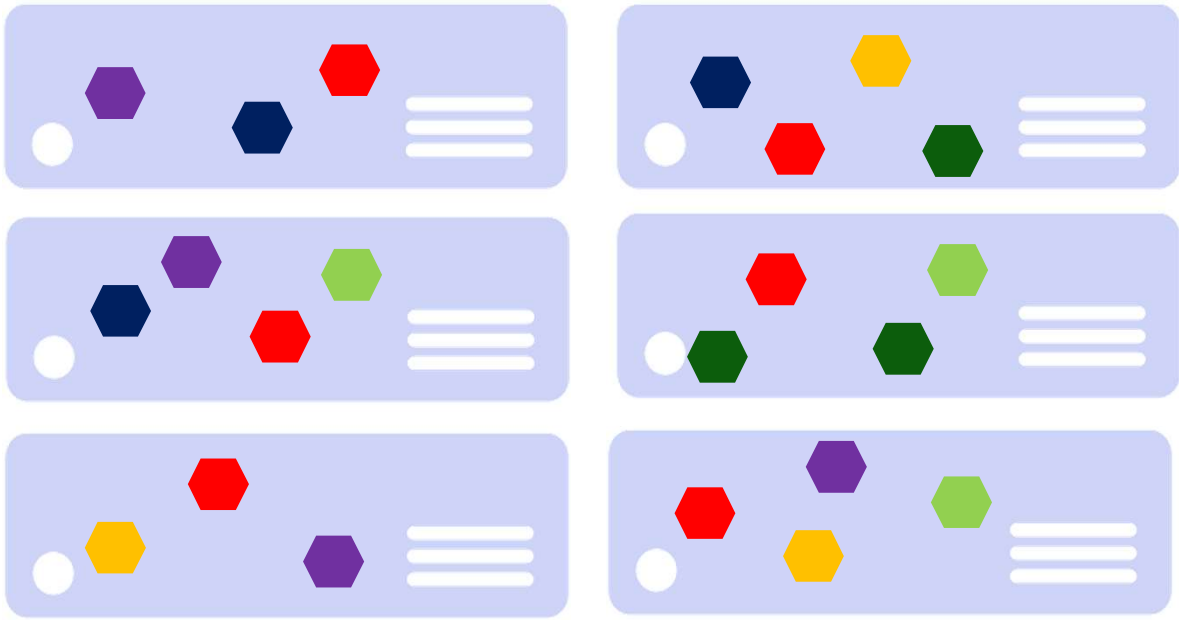
CPU Requests Limit Set in YAML file

Kubernetes – Scaling Pods

Can be done for a Deployment or ReplicaSet

Scale out

Scale in



Kubernetes Cluster VMs

Demo: Scaling

AKS Node and
POD Scaling

AKS – Upgrade Kubernetes

Upgrade can be performed with minimum disruption

```
az aks get-upgrades --name myAKSCluster --resource-group myResourceGroup --output table
```

- Show Kubernetes releases available for upgrade

```
MasterVersion    NodePoolVersion    Upgrades
---------------  -----------------  -------------------
1.8.10           1.8.10             1.9.1, 1.9.2, 1.9.6
```

```
az aks upgrade --name myAKSCluster --resource-group myResourceGroup --kubernetes-version 1.9.6
```

- Add new node with new version to the cluster and update nodes one by one (cordon and drain process)

```
az aks show --name myAKSCluster --resource-group myResourceGroup --output table
```

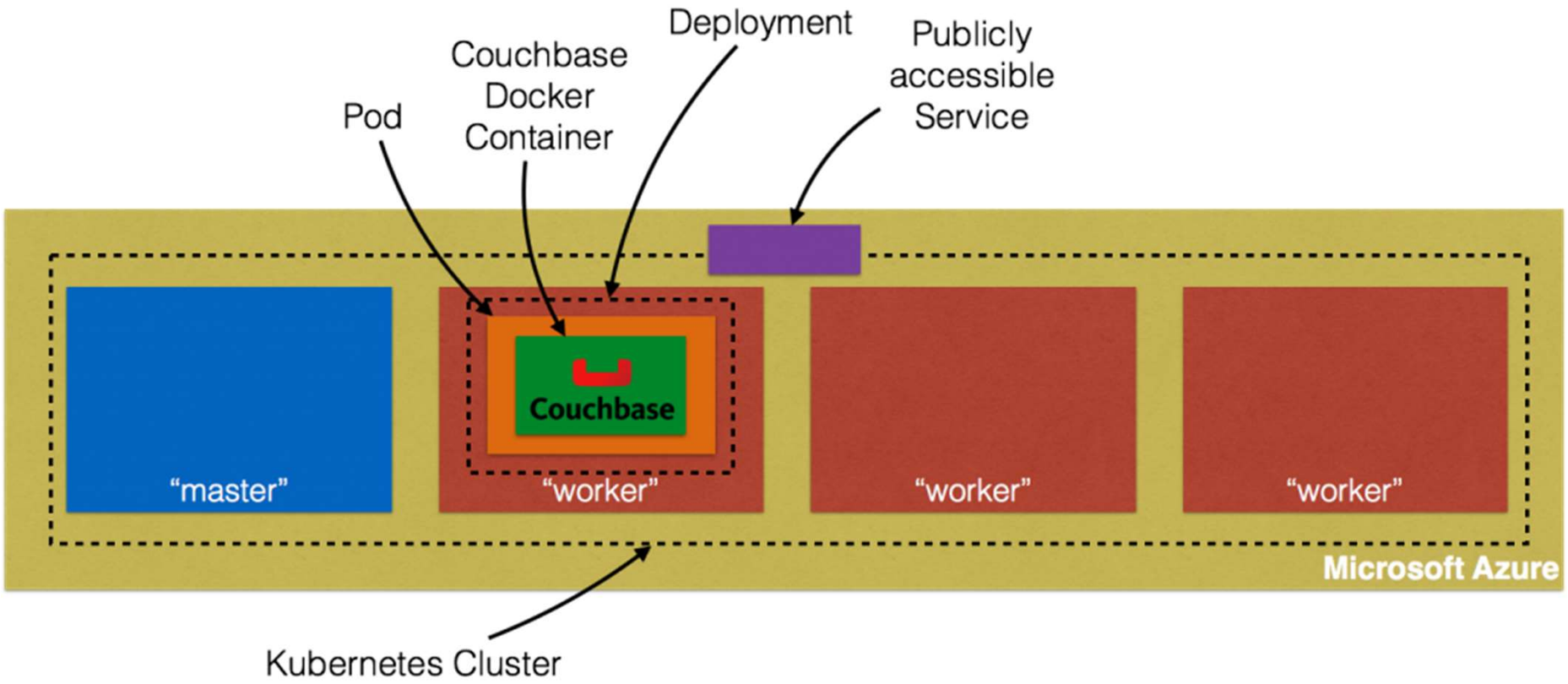
- Show upgrade result

```
Name          Location    ResourceGroup    KubernetesVersion    ProvisioningState
------------  ----------  ---------------  -------------------  -------------------
myAKSCluster  eastus      myResourceGroup  1.9.6                Succeeded
```

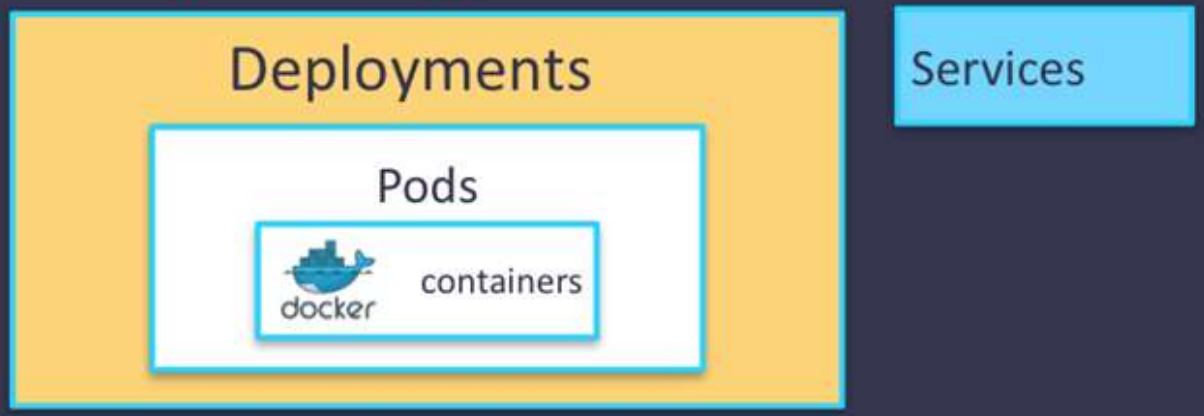
Demo: Upgrades

Updating Kubernetes Cluster

Putting It All Together



Recap: all you need to know



Container Image	Docker container image, contains your application code in an isolated environment.
Pod	A set of containers, sharing network namespace and local volumes, co-scheduled on one machine. Mortal. Has pod IP. Has labels.
Deployment	Specify how many replicas of a pod should run in a cluster. Then ensures that many are running across the cluster. Has labels.
Service	Names things in DNS. Gets virtual IP. Two types: ClusterIP for internal services, NodePort for publishing to outside. Routes based on labels.

Knowledge Check

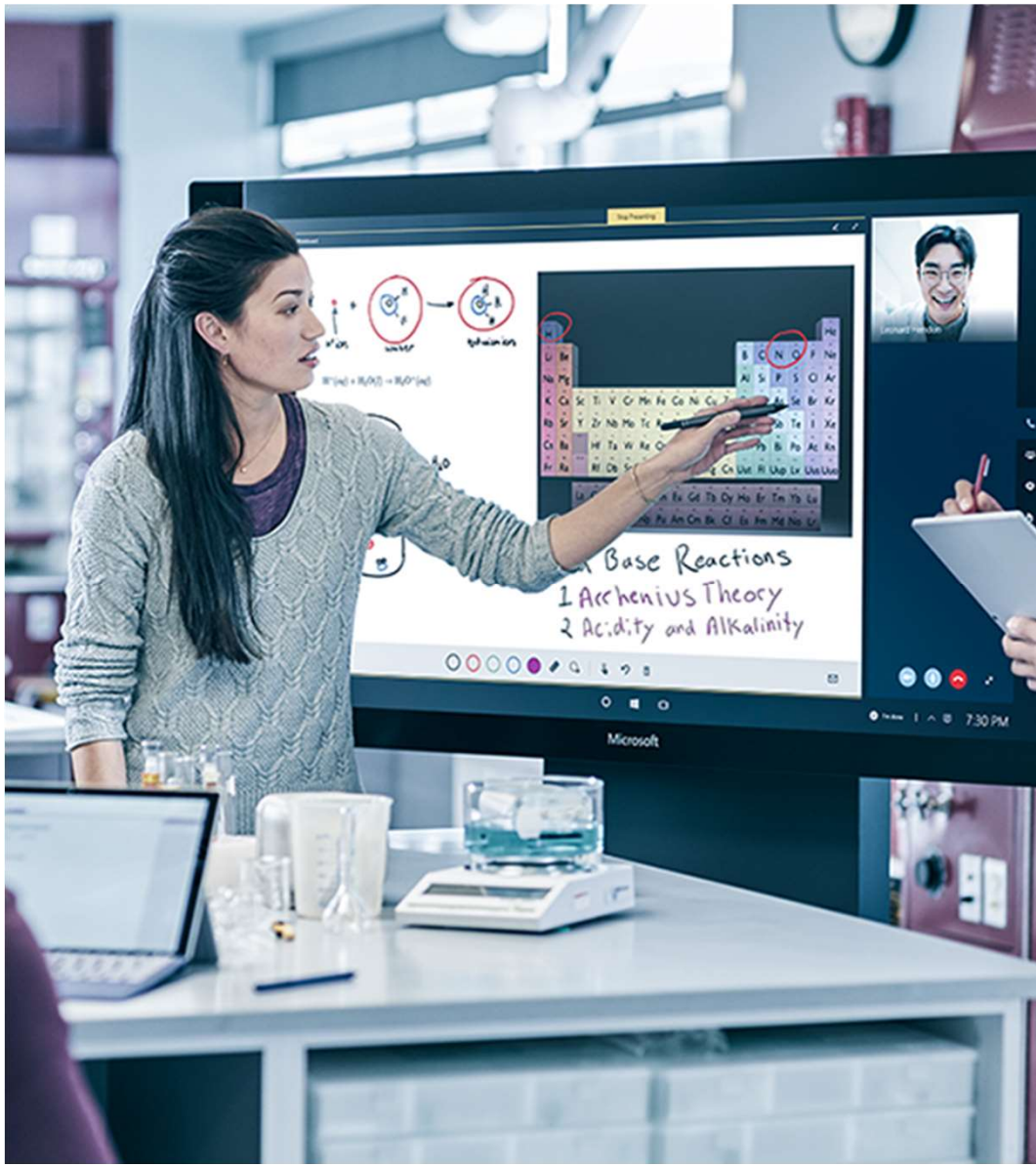
- What is the atomic unit of a cluster?
  - Pods
- What object allows me to define the number of instances of a pod?
  - Deployment
- What are two ways to define metadata for an object?
  - Labels and annotations

Module Summary

- Containers are a component of a Pod
- Pods are the atomic unit of a Kubernetes Cluster
- Objects like Deployments control the configuration and replication of pods
- Objects like services, secrets, configmaps, and volumes expose or connect the pod to other resources

Lab: Module 4

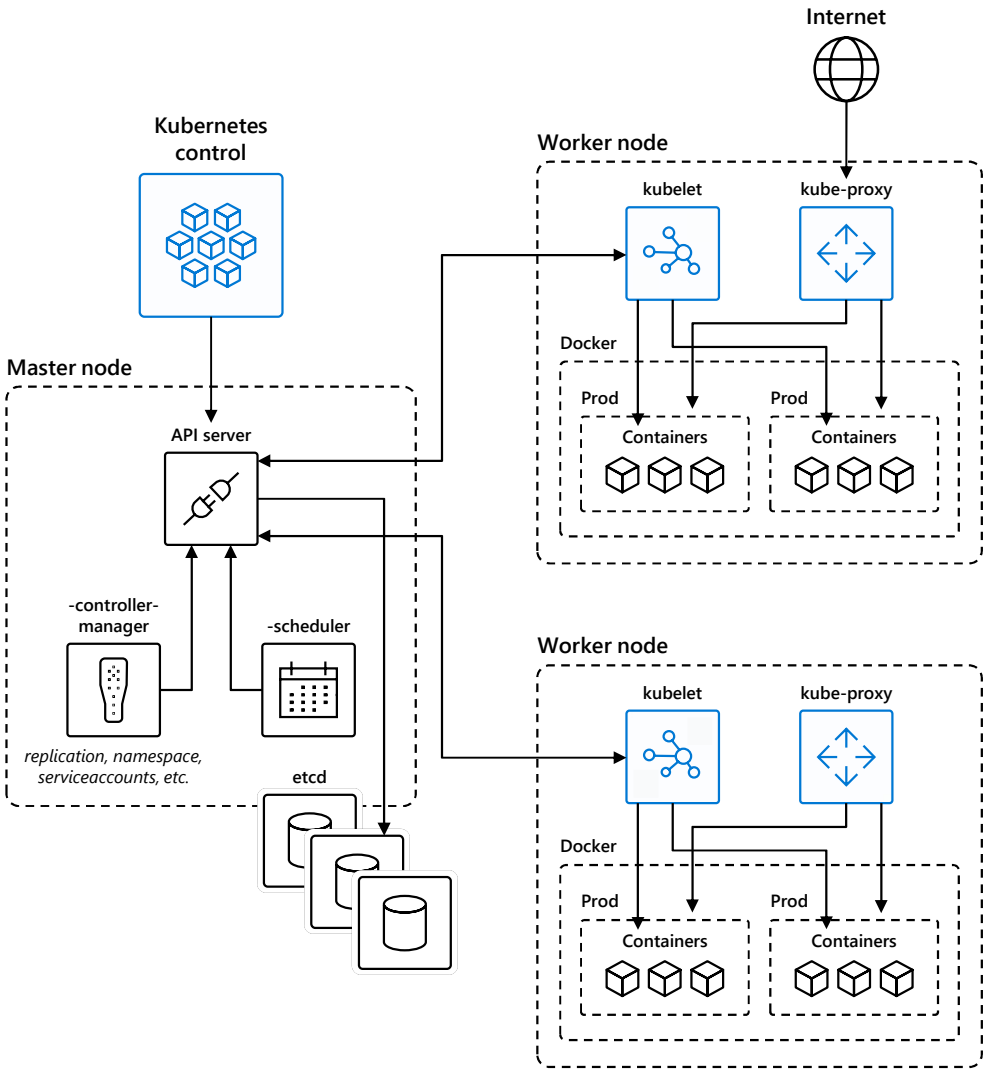
Kubernetes Core Concepts



Extra Slides

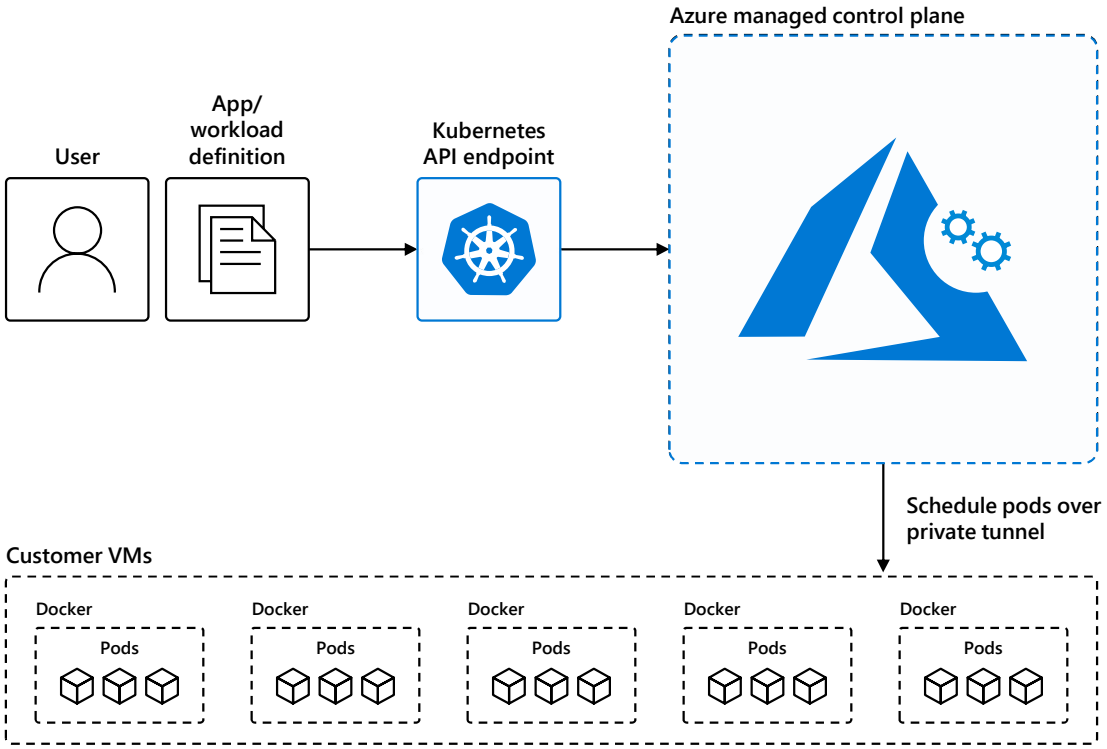
Kubernetes 101

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



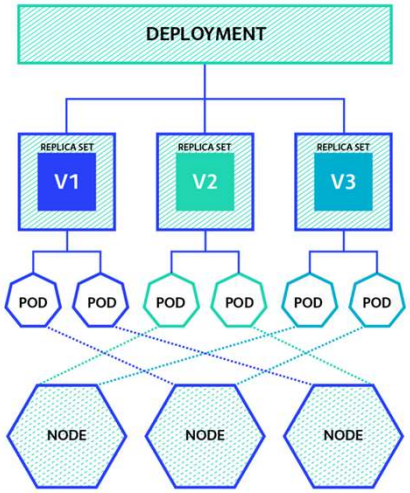
Kubernetes Overview

What is it?

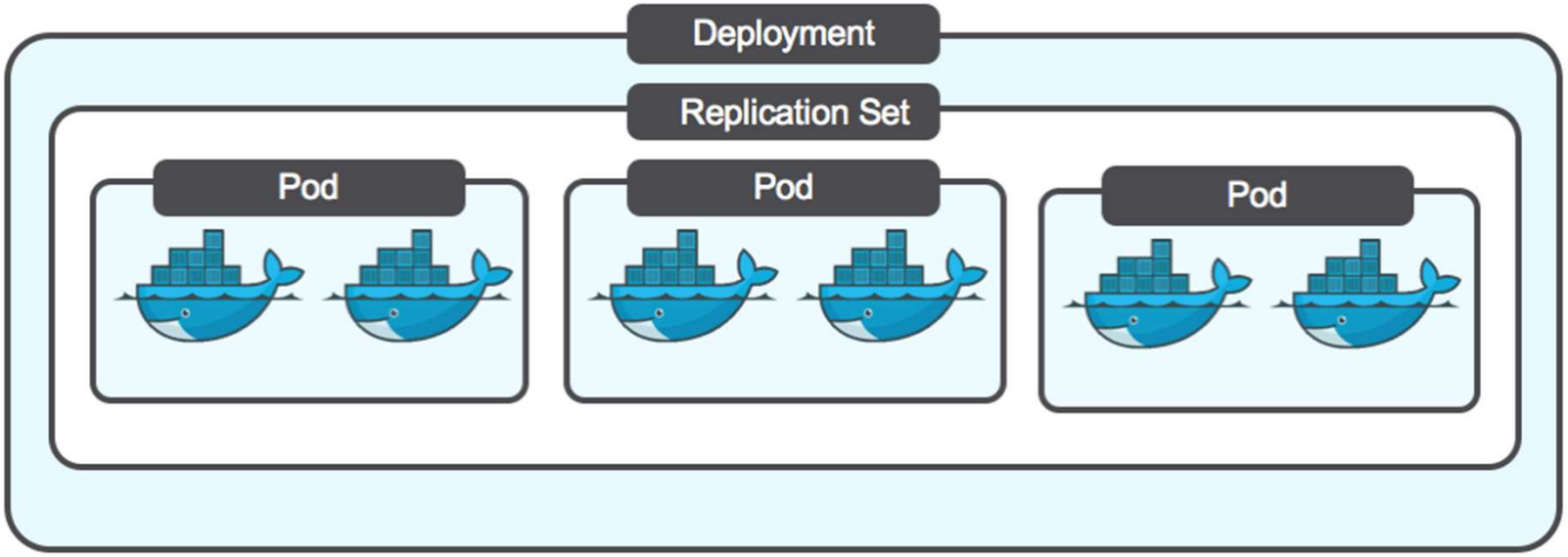
- It is a Container Orchestrator - Coordinates Docker Containers
- Made up of Master Nodes and Worker Nodes
- Uses the **kubectl** CLI to issue commands

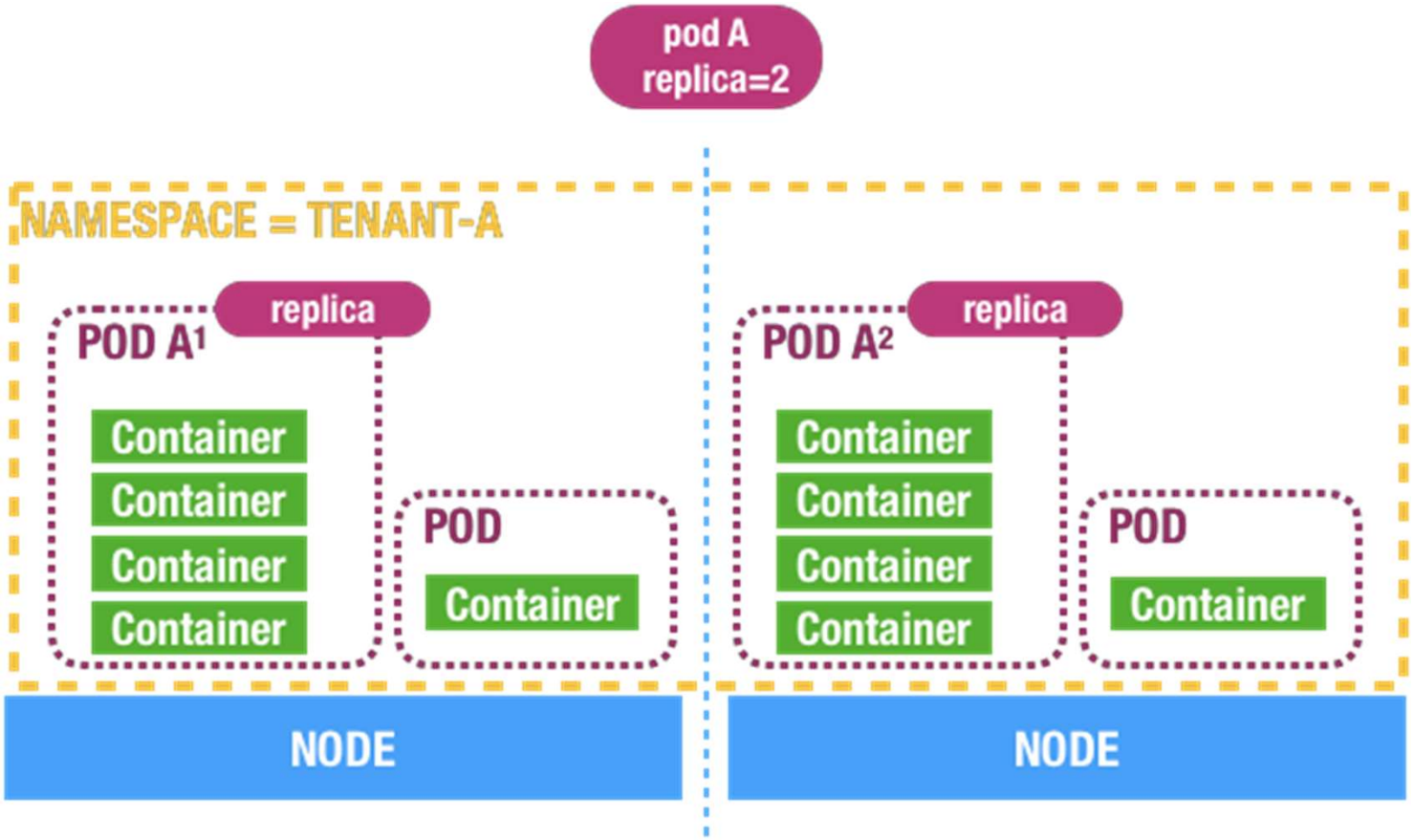
What goes on it?

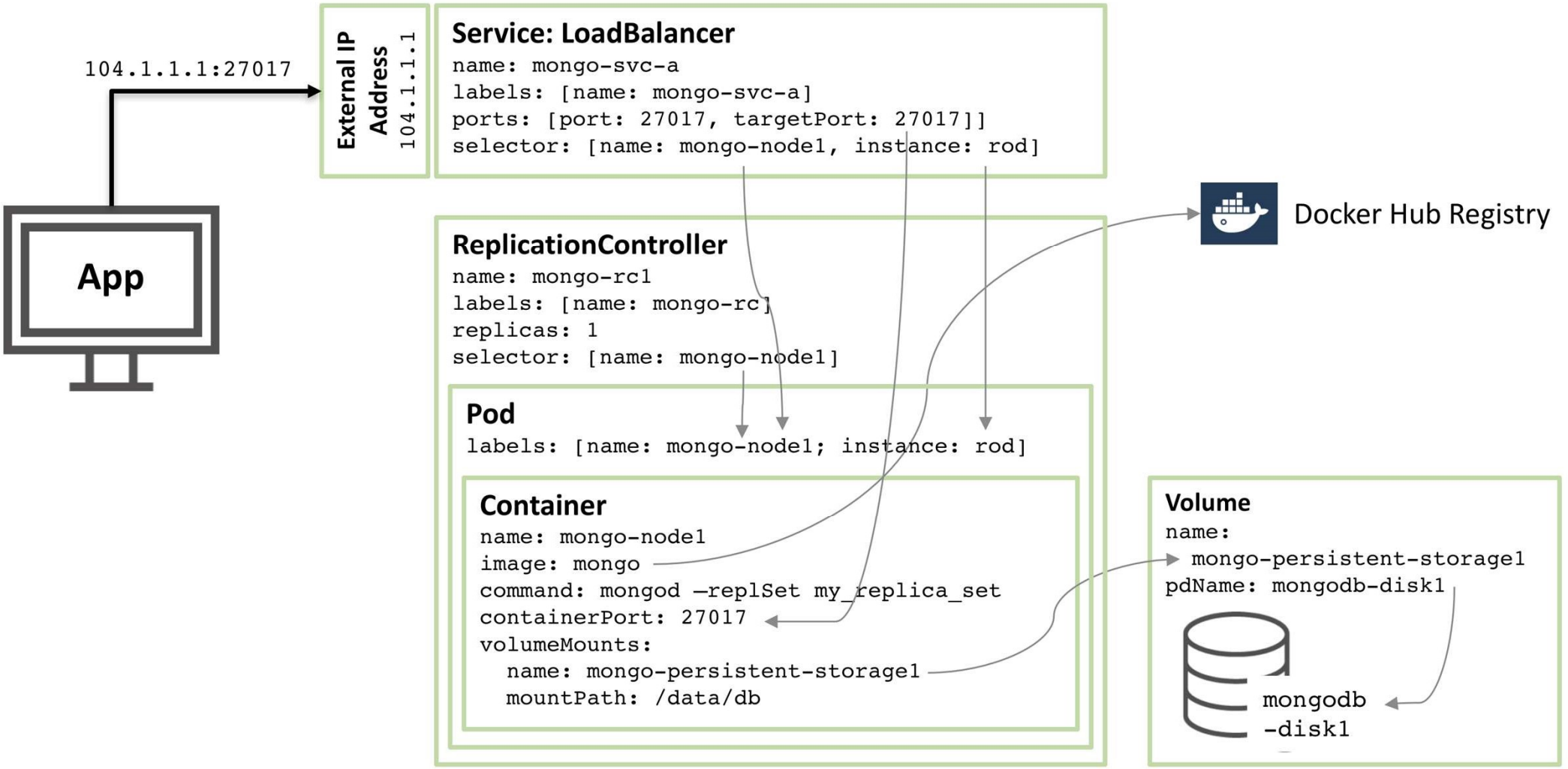
- Docker containers wrapped in a *Pod* (atomic unit of a cluster)
- A *Pod* is 'controlled' via various *Controllers*
- Objects are defined in *YAML* files


DEPLOYMENT
REPLICA SET
V1
REPLICA SET
V2
REPLICA SET
V3
POD
POD
POD
POD
POD
POD
NODE
NODE
NODE


Kubernetes Components
Pods: container or containers that form a single logical group
Replica Set: Supervises the pods
Deployment: Abstracts replica sets and pods
Service: Creates an internal and/or external URL and load balances across pods
Ingress: set of externally available routes that map to services

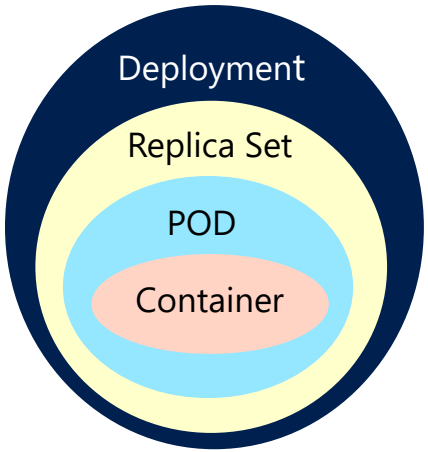

Deployment
Replication Set
Pod
Pod
Pod


pod A
replica=2
NAMESPACE = TENANT-A
replica
POD A1
Container
Container
Container
Container
POD
Container
replica
POD A2
Container
Container
Container
Container
POD
Container
NODE
NODE


104.1.1.1:27017
App
External IP Address 104.1.1.1
Service: LoadBalancer
name: mongo-svc-a
labels: [name: mongo-svc-a]
ports: [port: 27017, targetPort: 27017]]
selector: [name: mongo-node1, instance: rod]
Docker Hub Registry
ReplicationController
name: mongo-rc1
labels: [name: mongo-rc]
replicas: 1
selector: [name: mongo-node1]
Pod
labels: [name: mongo-node1; instance: rod]
Container
name: mongo-node1
image: mongo
command: mongod –replSet my_replica_set
containerPort: 27017
volumeMounts:
name: mongo-persistent-storage1
mountPath: /data/db
Volume
name:
mongo-persistent-storage1
pdName: mongodb-disk1
mongodb
-disk1

Kubernetes Concepts

- Kubernetes is actually an object store, along with code that interacts with those objects
- Each Kubernetes object has three parts
  - Metadata – You describe the object
  - A specification – You describe the desired state of the object
  - The current observed status – Reported by Kubernetes


Deployment
Replica Set
POD
Container

Deployments: Updates as a Service

Reliable mechanism for creating, updating and managing Pods

Deployment manages replica changes, including rolling updates and scaling

Edit Deployment configurations in place with kubectl edit or kubectl apply

Managed rollouts and rollbacks

Status: BETA in Kubernetes v1.2

Deployment

Rollouts as a service

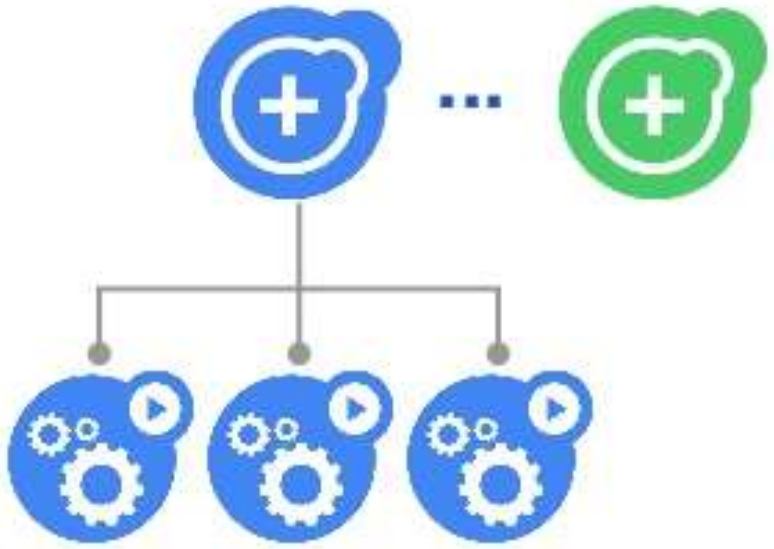
- updates to pod template will be rolled out by controller
- can choose between rolling update and recreate

Enables declarative updates

- manipulates replication controllers and pods so clients don't have to

Status: **ALPHA** in Kubernetes v1.1

```
Deployment
  - strategy: {type: RollingUpdate}
  - replicas: 3
  - selector:
      - app: my-app
```



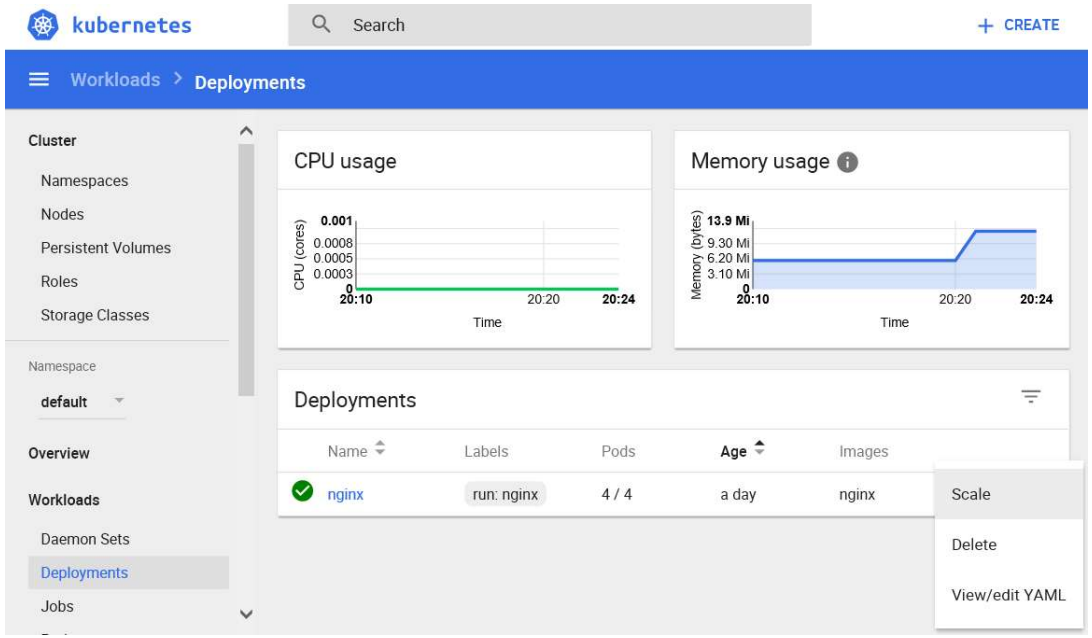
Kubernetes – Scaling Pods

- Using the kubectl CLI

```
PS C:\> kubectl scale deployments/nginx --replicas=4
deployment "nginx" scaled
```

- From the dashboard

⇒Scaling pods is really fast depending on container time to start and resource availability



Kubernetes – Autoscaling Pods

- Works on built-in metrics as well as custom or application-provided metrics
- Can be setup imperatively with **kubectl autoscale** command or declaratively with the **HorizontalPodAutoscaler** K8s object
- Convenient to scale **deployments**

```
kubectl autoscale deployment azure-vote-front --cpu-percent=50 --min=3 --max=10
```

```
apiVersion: autoscaling/v2beta2
kind: HorizontalPodAutoscaler
metadata:
  name: php-apache
  namespace: default
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: php-apache
  minReplicas: 1
  maxReplicas: 10
  metrics:
  - type: Resource
    resource:
      name: cpu
      target:
        type: Utilization
        averageUtilization: 50
```