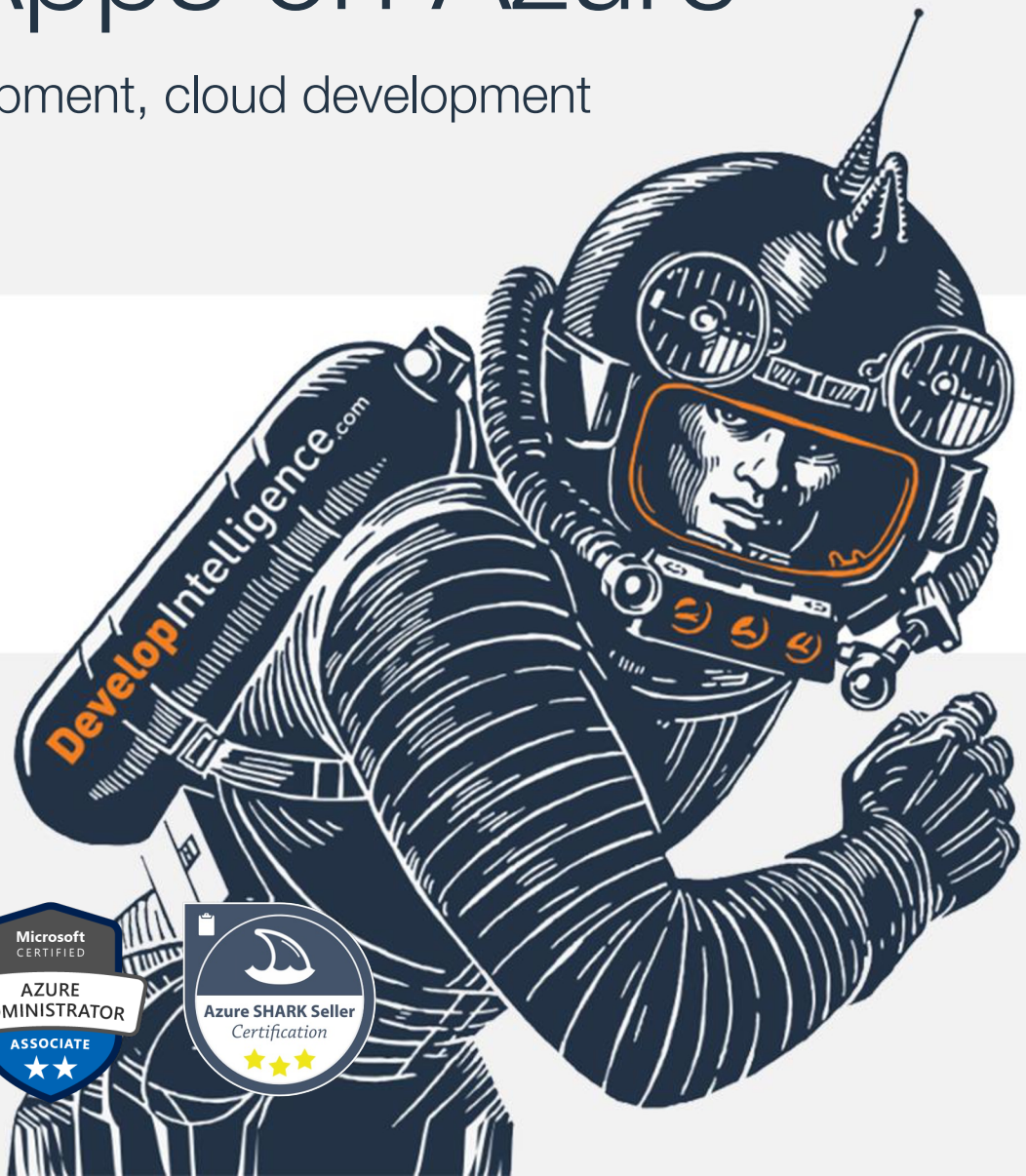
Developing Scalable Apps on Azure

Designed to demonstrate enterprise software development, cloud development and delivery.

Develop Intelligence



With Siddharth Dhawan
Microsoft Certified Trainer - Azure

Microsoft Certified Trainer MCT Trainer 2020 - 2021

Microsoft Certified Azure Solutions Architect Expert

Microsoft Certified DevOps Engineer Expert

Microsoft Certified Azure Security Engineer Associate

Microsoft Certified Azure Developer Associate

Microsoft Certified Azure Administrator Associate

Azure SHARK Seller Certification

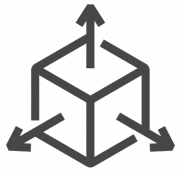
Why Containers?

Portability - Build once, deploy anywhere #kubernetes

Time to market – modernization accelerator

Low cost of ownership – run 10X to 100X more workloads than VMs

Security – Reduced attack surface, improved access control



Scalability – starts up in seconds, better resilience

Module Layout – Day 4

Develop Intelligence

What is container technology?

K8s Concepts

What is AKS?

Deploy a microservice to AKS using CLI

K8s Resource Management

Summary and Quiz

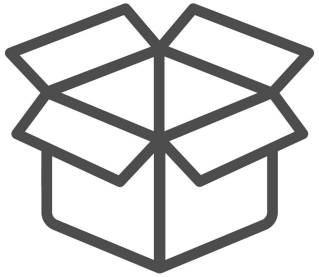
Module Layout – Day 1

Develop Intelligence

What is container technology?	K8s Concepts
What is AKS?	Deploy a microservice to AKS using CLI
K8s Resource Management	Summary and Quiz

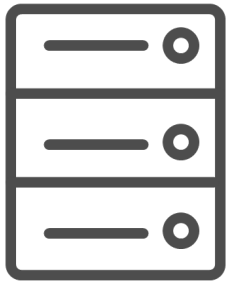
Traditional Software Development

Build software as a single codebase



Package code into a deployable entity

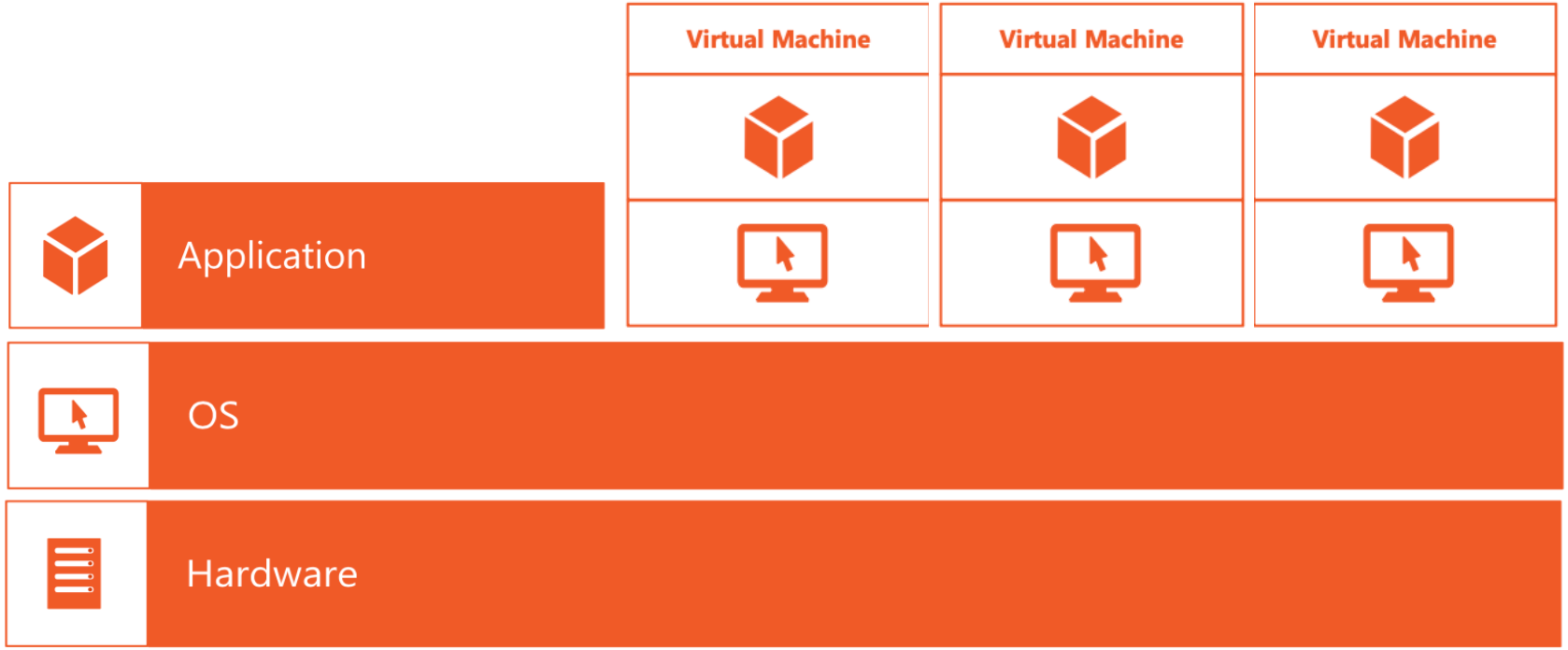
Deploy and monitor health



Migrate to healthy servers if hardware fails

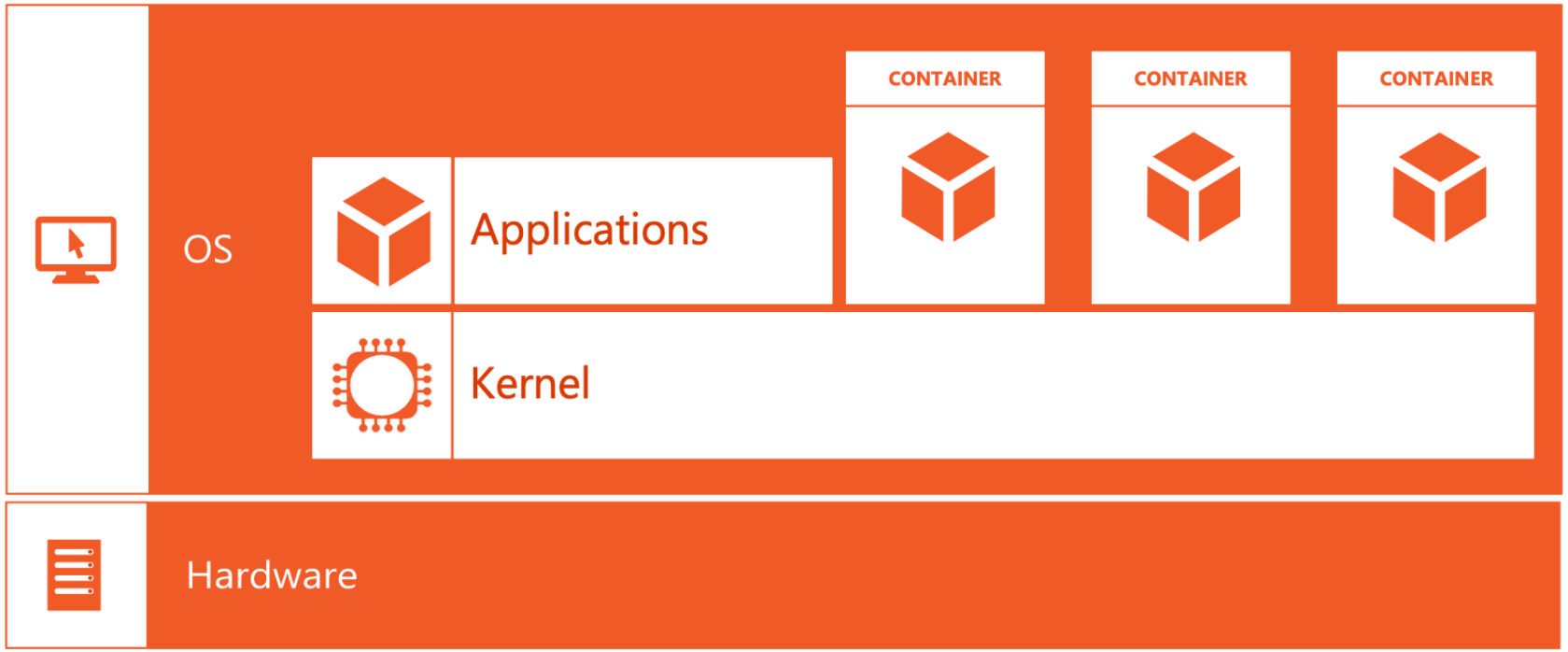
What is Container Technology?

Traditional virtual machines = hardware virtualization

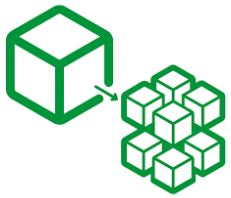


What is Container Technology?

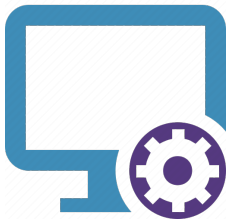
Containers = Operating system virtualization



Microservices

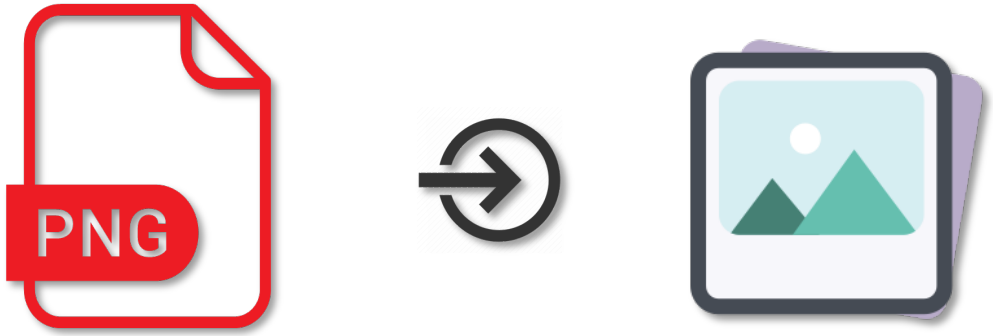


Microservices are broken down into smaller, independently running components called microservices



Microservices are de-coupled from each other and therefore they can be developed, deployed, updated and scaled individually

Each microservice exposes it’s functionality or services via an interface , typically a REST API


PNG

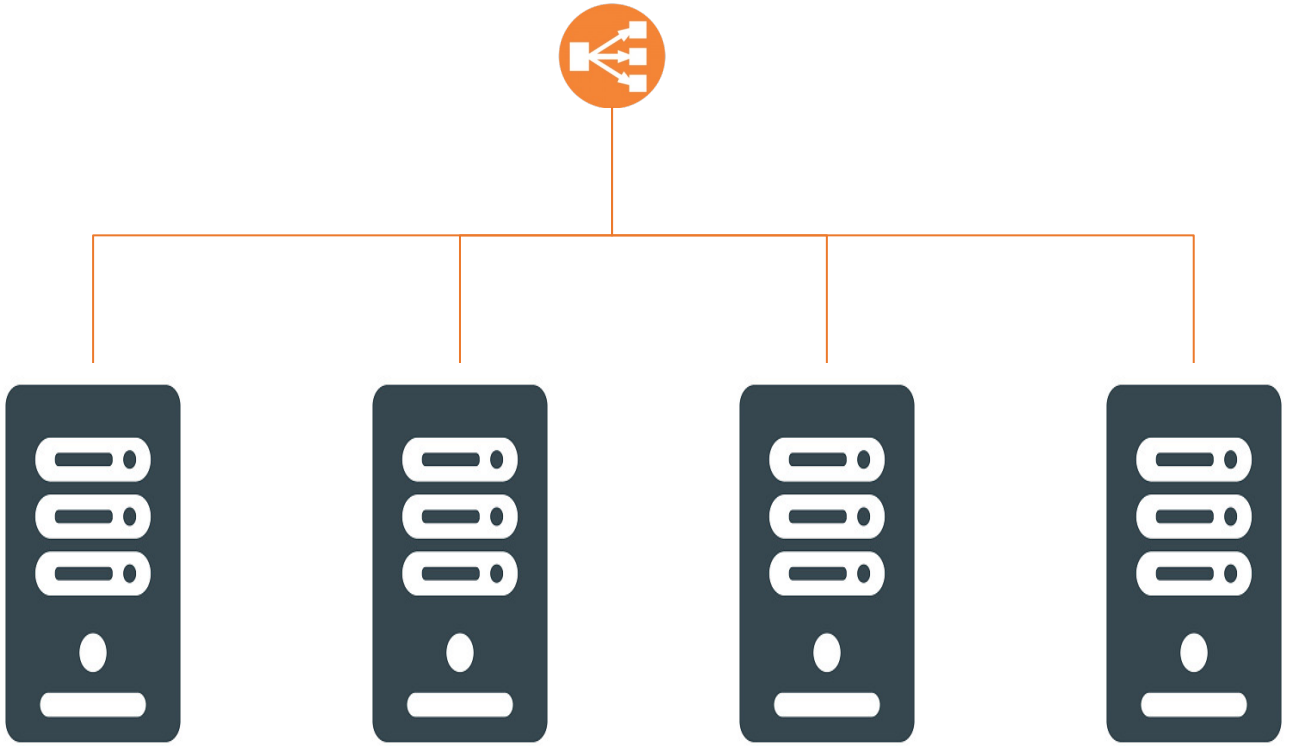

PNG


PNG


PNG

Vertical Scaling

Horizontal Scaling



GET /api/greyscale

GET /api/brightness

GET /api/contrast

Why Containers for Microservices ?

- Unit of deployment
- Simplified testing
- Unit of versioning
- Simplified scaling

Demo : Running your first Container

Running in Containers on a Single Machine

Containers – The Bad Part

Develop Intelligence

Many components, many moving parts

Difficult to manage inter-communication

Difficult to achieve high resource utilization

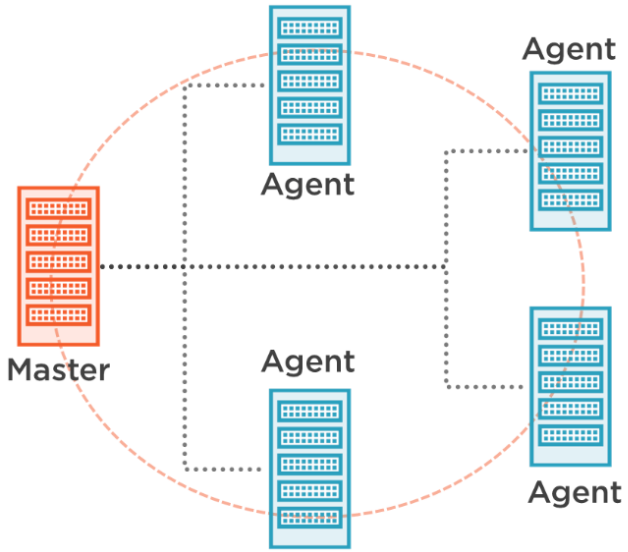
Manual management can be difficult

We need automation, which includes automatic scheduling, automatic configuration, supervision and failure handling

We need a ‘Container Orchestrator’

Kubernetes

Production Workloads Run on Clusters



Scalability

Fault Tolerance

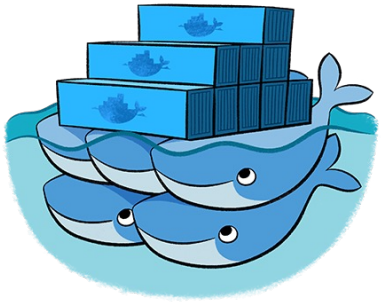
Automatic Recovery

Zero Downtime Deployments Resource Management Cross Machines

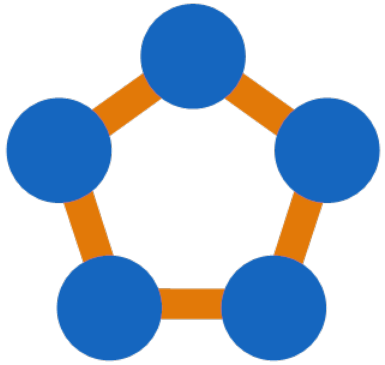
Container Composition

Options for Container Clusters

Apache Mesos



Docker Swarm



Azure Service Fabric

Kubernetes

Kubernetes ...

Enables developers to deploy their applications as often as they want

Enables the ops team by automatically monitoring and rescheduling those apps in the event of a hardware failure

The focus from supervising individual apps to mostly supervising and managing kubernetes and the rest of the infrastructure

Abstracts away the hardware infrastructure and exposes your whole datacenter as a single gigantic computational resource

Microsoft Azure

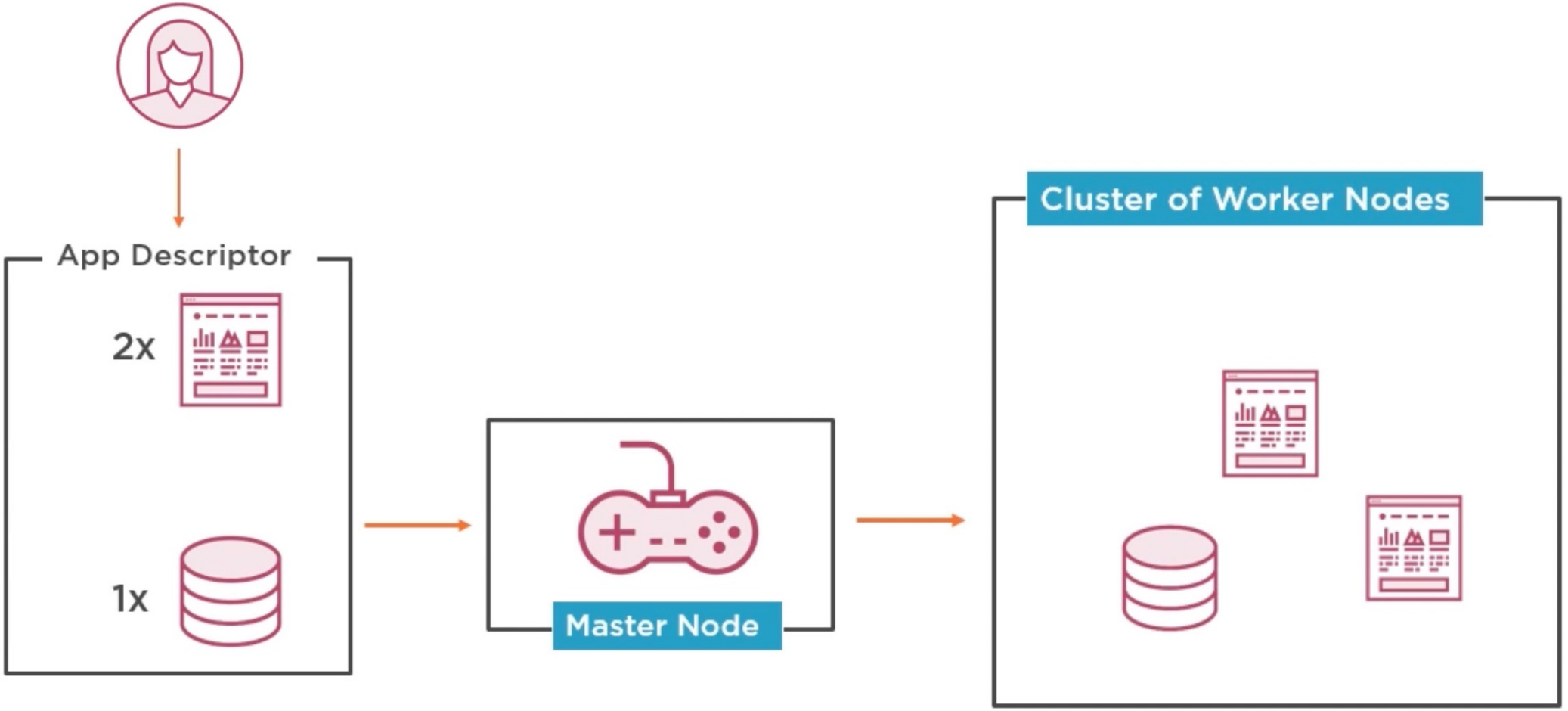
Kubernetes + Azure = Pure Awesomeness

Kubernetes Cluster Node Types

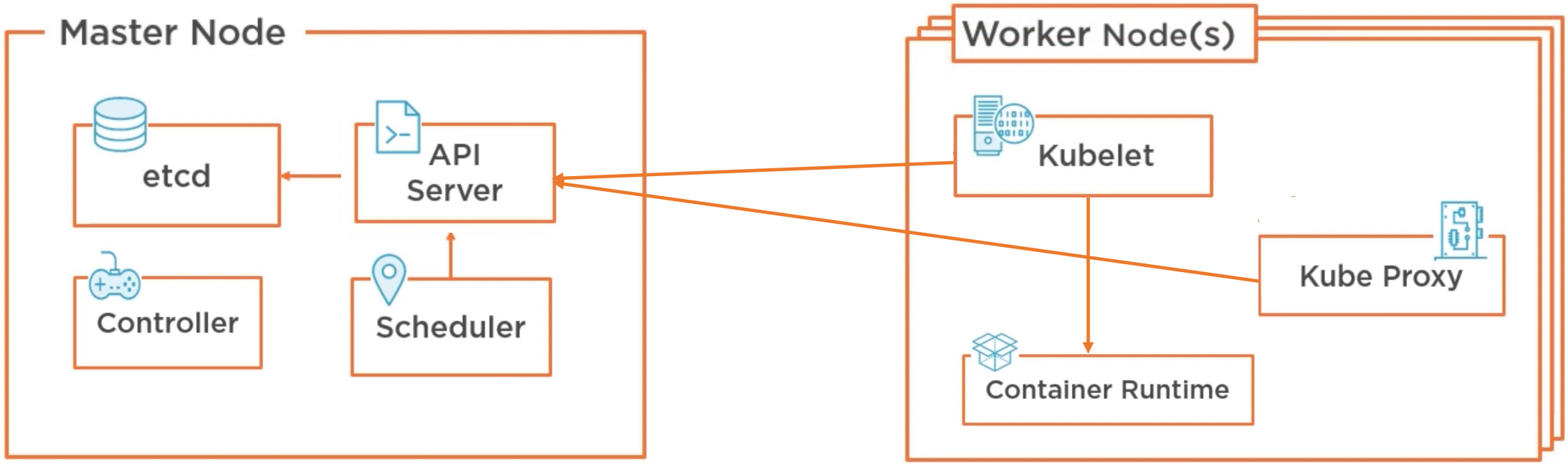
Master Node(s)

Worker Node(s)

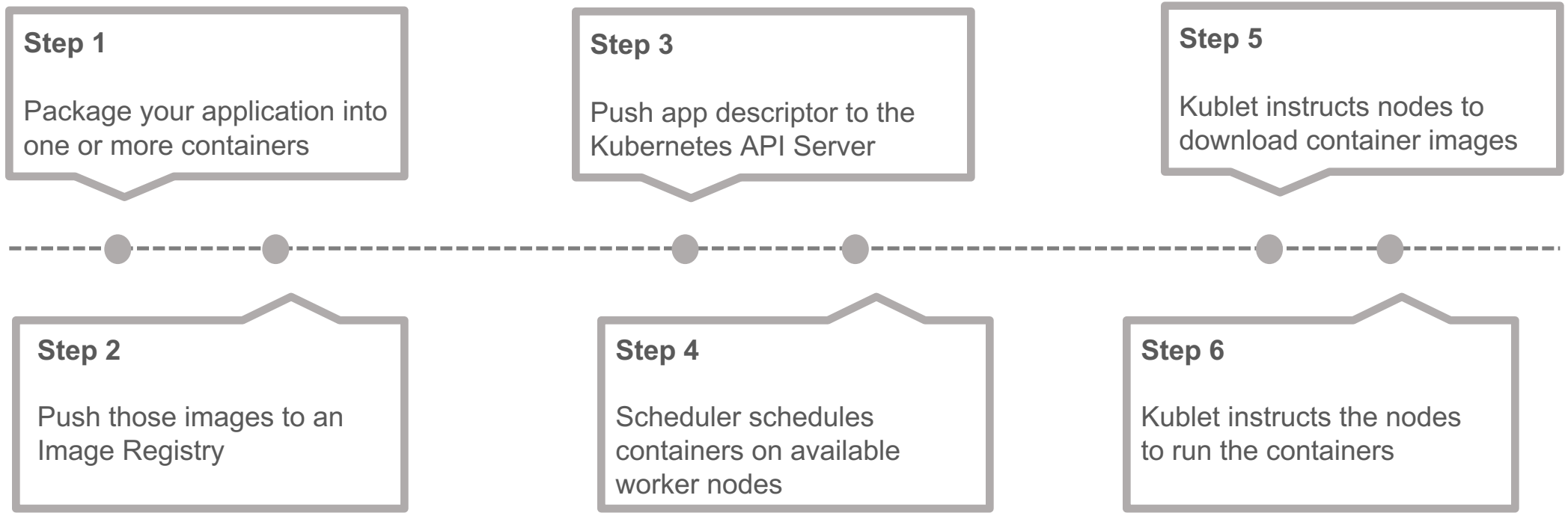
Kubernetes System



Kubernetes Architecture



Running an Application in Kubernetes



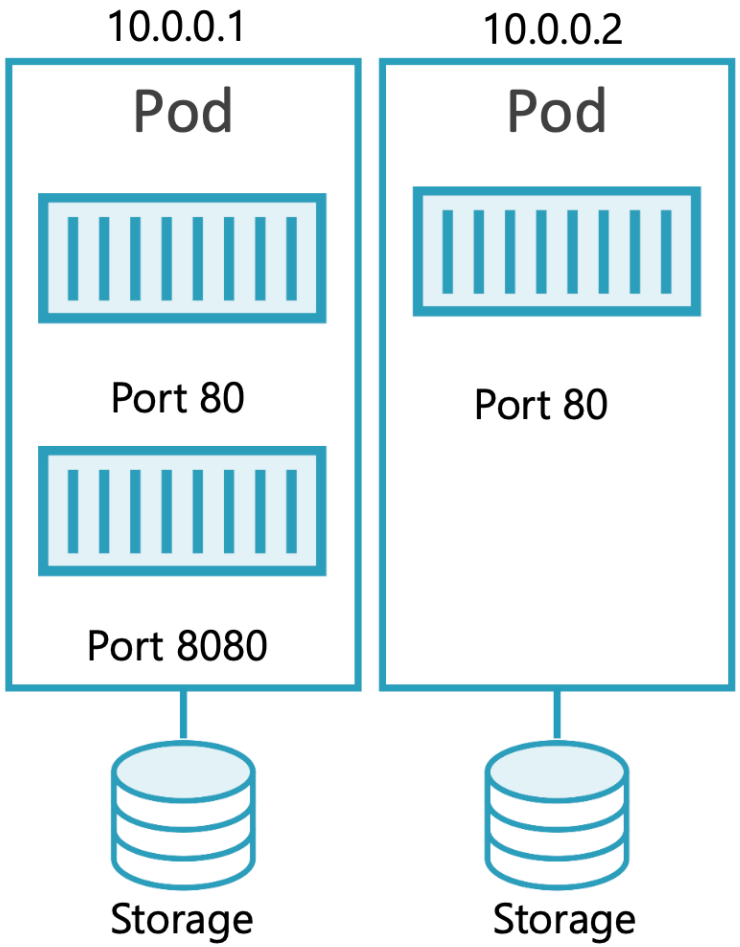
Excited !!!

Module Layout – Day 1

Develop Intelligence

What is container technology?	K8s Concepts
Desired State using YAML	Deploy a microservice to AKS using CLI
K8s Resource Management	Summary and Quiz

A POD



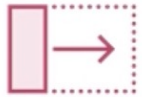
Group of 1 or more containers

Shared Storage

Shared Network
- Same IP-address
- Shared port-range

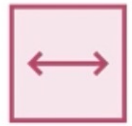
Well known patterns:
- Sidecar
- Proxy, bridge or adapter

Charasteristics of A POD



All the containers for a Pod will be run on the same Node

Any container running within a Pod will share the Node's network with any other containers in the same Pod



Containers within a Pod can share files through volumes, attached to the containers

A Pod has an explicit life cycle, and will always remain on the Node in which it was started

Namespaces

Pods are collected into namespaces, which are used to group pods

Namespaces can be used to provide quotas and limits around resource usage

If no namespace is specified while interacting with Kubernetes, the command assumes yiu are working with the ‘default’ namespace

Networks

All the containers in a Pod share the Node's network

All Nodes in a Kubernetes cluster are expected to be connected to each other and share a private cluster-wide network

Kubernetes runs containers within a Pod within this isolated network

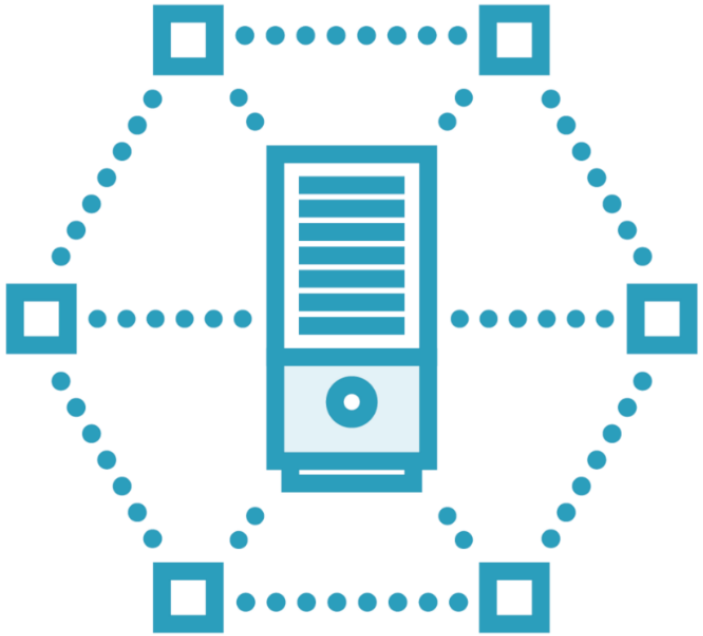
Controlleres

Kubernetes encourages Desired State deployment

You assert you want one or more resources to be in a certain state and with specific versions

Controllers are where the brains exist for tracking those resources and attempting to run your software as you described

A ReplicaSet and a Deployment



ReplicaSet
- A ReplicaSet's purpose is to maintain a stable set of replica Pods running at any given time
- Self-healing capability of K8s

Deployment
- Provides declarative management for Pods and ReplicaSet
- Deployments own and manage their ReplicaSet
- Primarily responsible for rolling out software updates

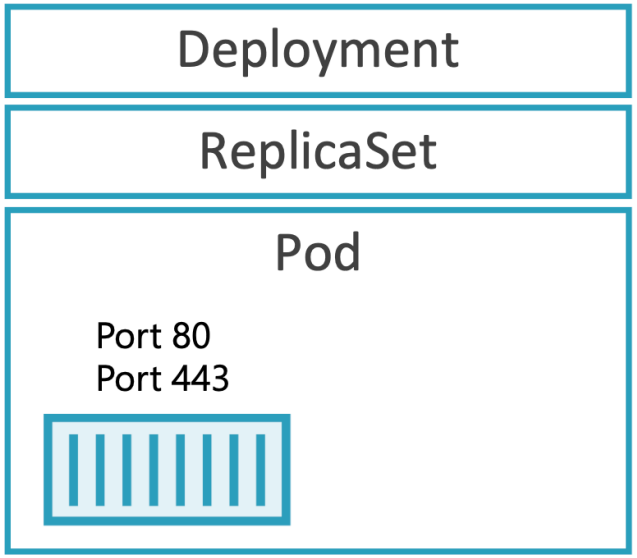
Services

Kubernetes resource used to provide an abstraction through to your Pods agnostic of the specific instances that are running

Can contain a Policy

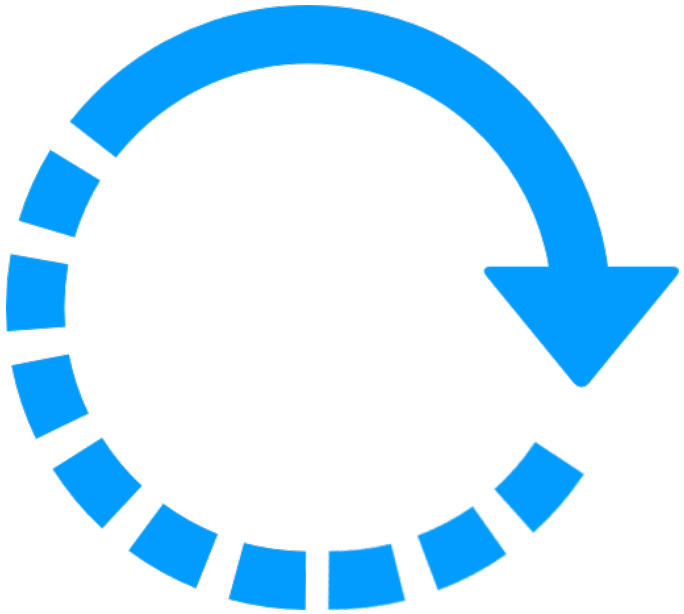
Emulates a software load balancer within Kubernetes

Desired State Defined in Yaml




```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: dep-globoticket-web
spec:
  replicas: 1
  selector:
    matchLabels:
      app: globoticket-web
  template:
    metadata:
      labels:
        app: globoticket-web
    spec:
      containers:
      - name: globoticket-web
        image: globoticket.azurecr.io/globoticket.web:433
        env:
        - name: ASPNETCORE_ENVIRONMENT
          value: Production
        - name: ASPNETCORE_URLS
          value: http://+:80
        - name: ApiConfigs__EventCatalog__Uri
          value: http://svc-globoticket-services-eventcatalog
        ports:
        - containerPort: 80
        - containerPort: 443
        resources:
          limits:
            cpu: "0.15"
      imagePullSecrets:
        - name: pullkey
```

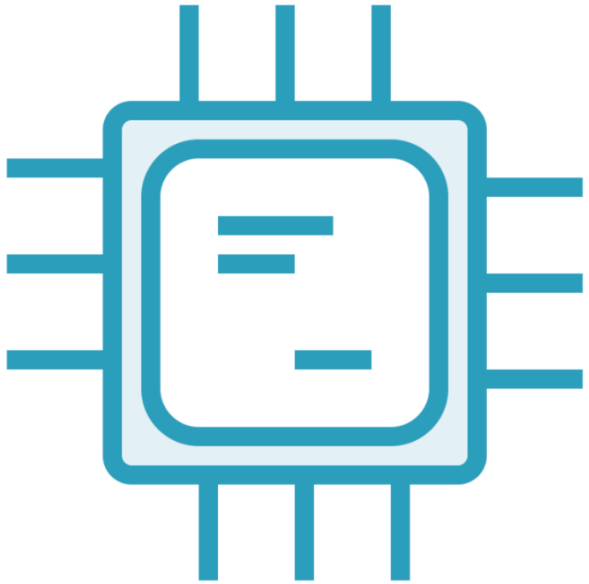
Rolling Updates



Zero Downtime Deployments
- New replica set is created - New pods are created
- Traffic is re-routed
- Old pods are deleted
- Old Replica set is deleted

Can be rolled back

Resource Management



Requests:
- Minimum required resources

Limits:
- Capped resource usage

How to Communicate between Pods?

Pods are mortal
- E.g., new deployment
- Need an abstract way to expose an application running on a set of pods

Service
- Single IP address and single DNS name
- Load-balance across pods

How to Communicate between Pods?

Important Note:

When building microservices, avoid microservice 2 microservice communication as much as possible.

Breaking up a monolith into multiple services that call each other is an anti pattern! It will result in a distributed monolith and will give you only
problems!

A microservice should operate on its own and should not need other services to do its work. If that is the case, you need to revise your design to create better isolated services!

Module Layout – Day 1

Develop Intelligence

What is container technology?	K8s Concepts
What is AKS?	Deploy a microservice to AKS using CLI
K8s Resource Management	Summary and Quiz

Azure Kubernetes Service (AKS)

Managed Kubernetes cluster running on Microsoft Azure with automated version upgrades and patching

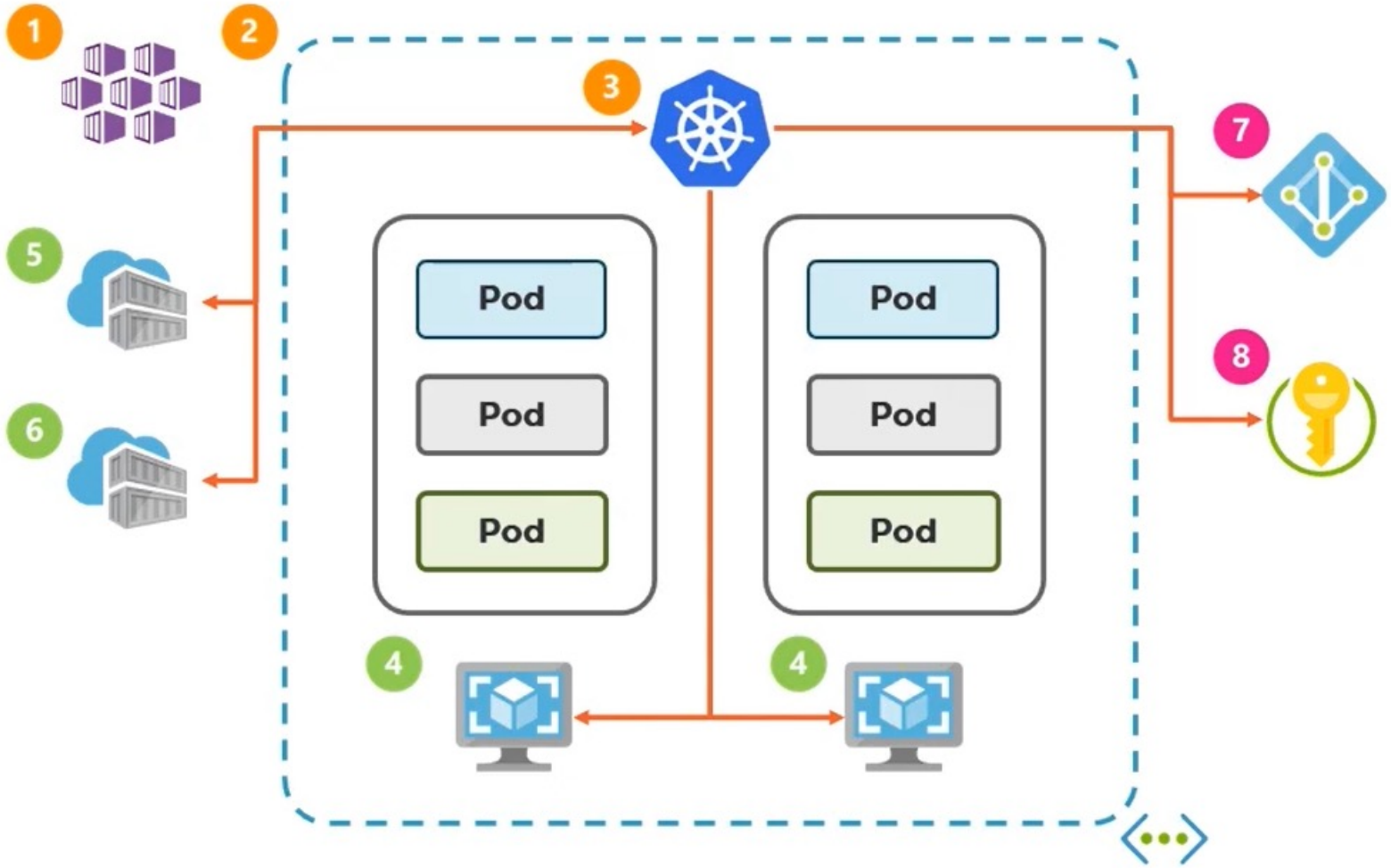
Kubernetes master service provided as a PaaS offering

Customer only pays for Kubernetes worker nodes: master is free

Azure Kubernetes Service Infrastructure

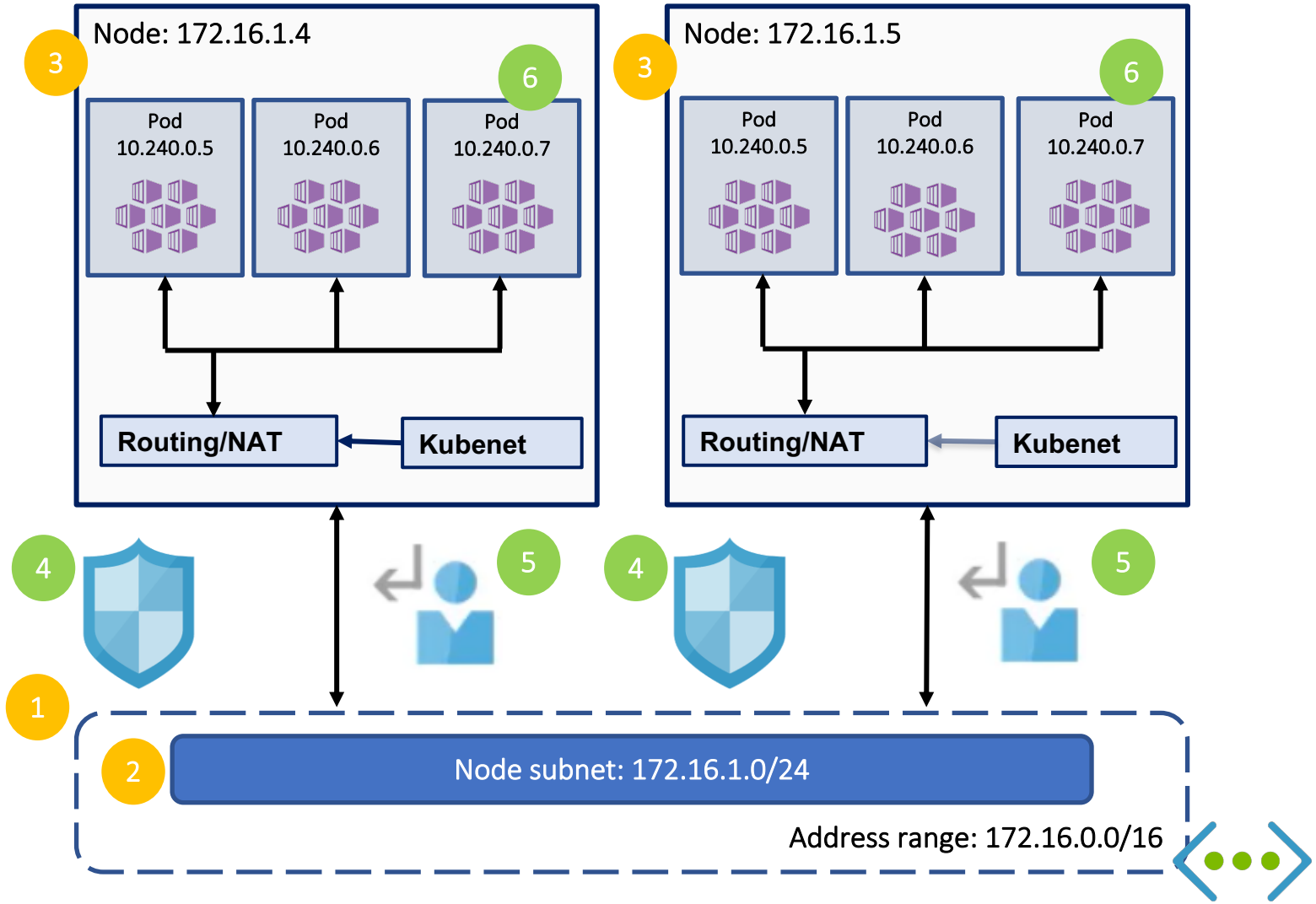
Develop Intelligence

1. Azure Kubernetes Services
2. Virtual Network
3. Kubernetes Cluster
4. Azure Virtual Machine
5. Azure Container Registry
6. Docker Hub
7. Azure Active Directory
8. Azure Key Vault



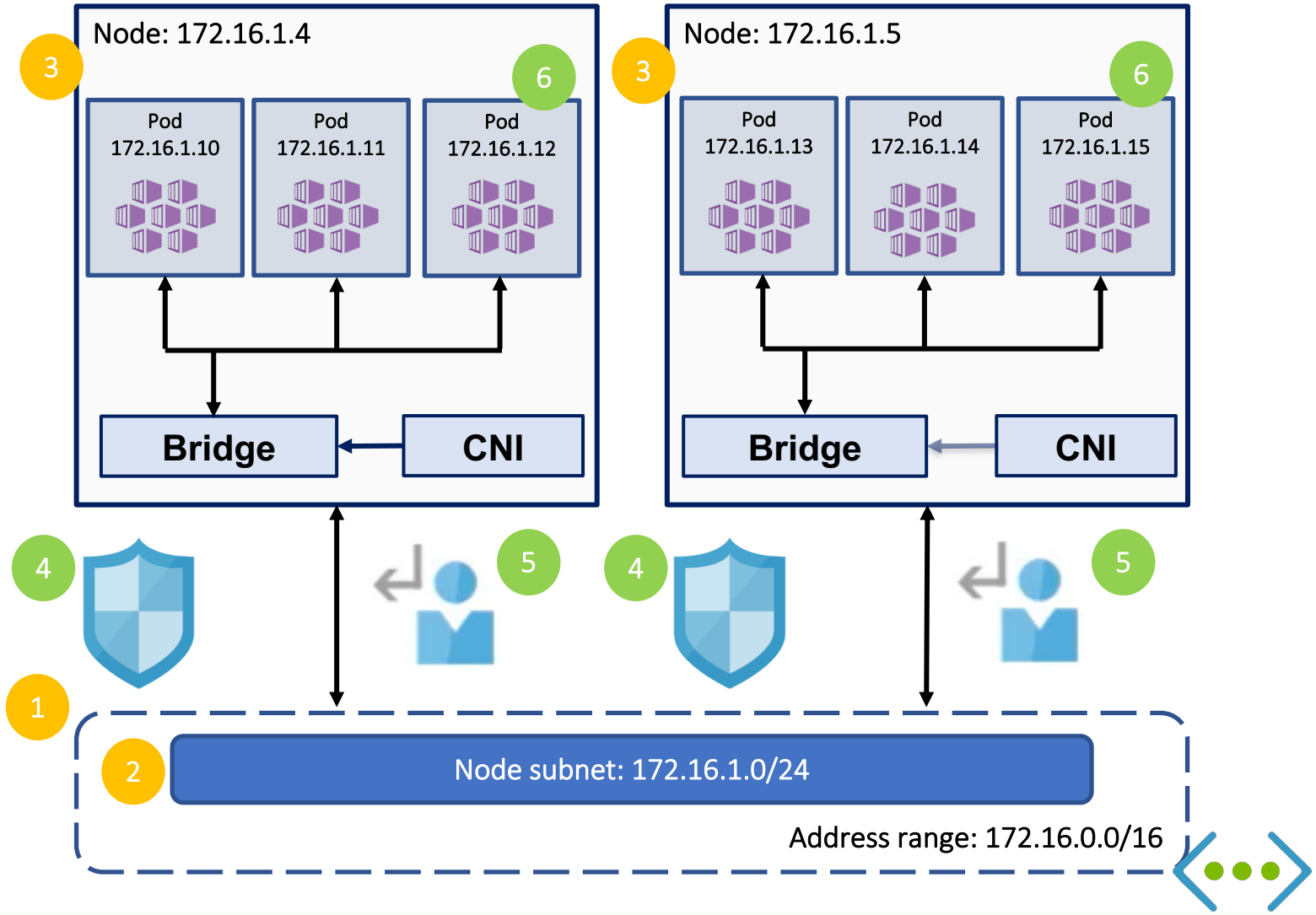
AKS Networking: Kubenet

1. Azure Virtual Network
2. Virtual Network Subnet
3. Cluster Nodes
4. Network Security Groups
5. User Defined Routes
6. Cluster Pods



AKS Networking: Container Networking Interface

1. Azure Virtual Network
2. Virtual Network Subnet
3. Cluster Nodes
4. Network Security Groups
5. User Defined Routes
6. Cluster Pods



Kubenet (Basic) vs Azure CNI (Advanced)

Develop Intelligence

Kubenet	Azure CNI
IP addresses are private to the cluster	IP addresses are taken from the subnet
AKS master manages network resources	Network resources managed independently
Pods are accessed via load balancers	Pods can be accessed directly
Pod-VM connectivity initiated by pod	Pod-VM connectivity initiated by pod or VM

Provision and Configure AKS Cluster using Azure Portal

- Create a new resource group for AKS Cluster
- Select location which supports Azure Kubernetes Service
- Create a Service Principal or generated automatically
- Access AKS Cluster from local system
- Check resources in portal and Command line
- Browse Kubernetes Dashboard

Azure CNI: Planning Considerations

Develop Intelligence

Network must allow outbound connectivity

Only one AKS cluster per subnet

IP addresses are reserved for each cluster node

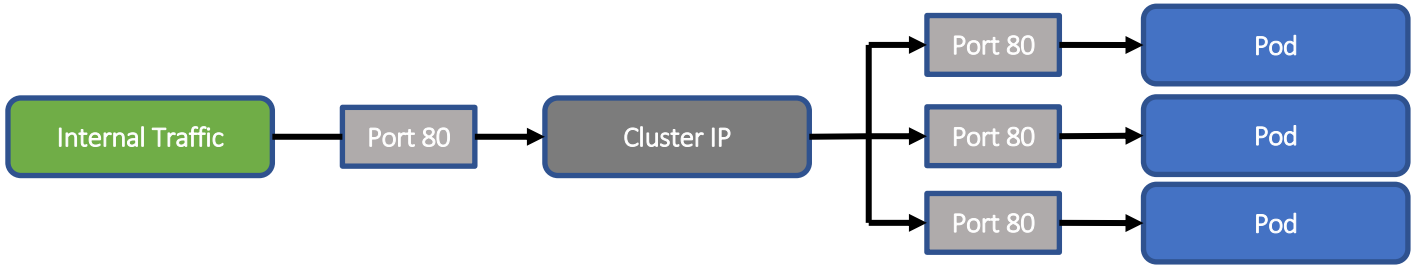
Up to 250 pods per node (default is 30 pods)

Plan for additional network resources

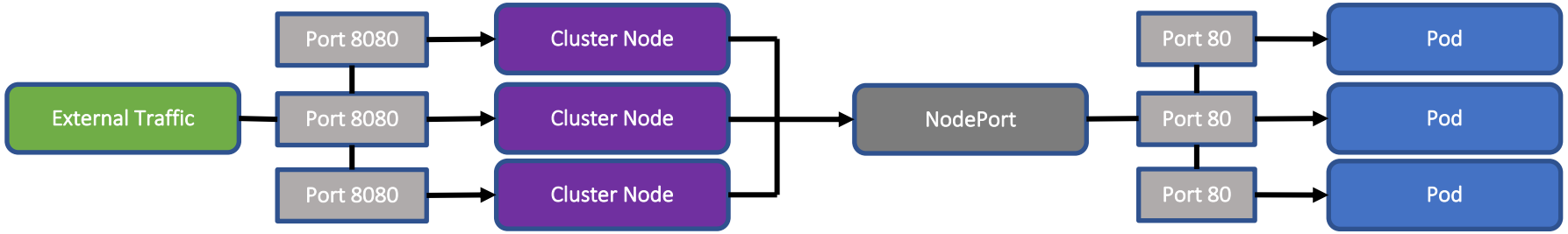
Service principal requires Network Contributor rights

Publishing Kubernetes Services

- ClusterIP: An internal IP within the cluster.

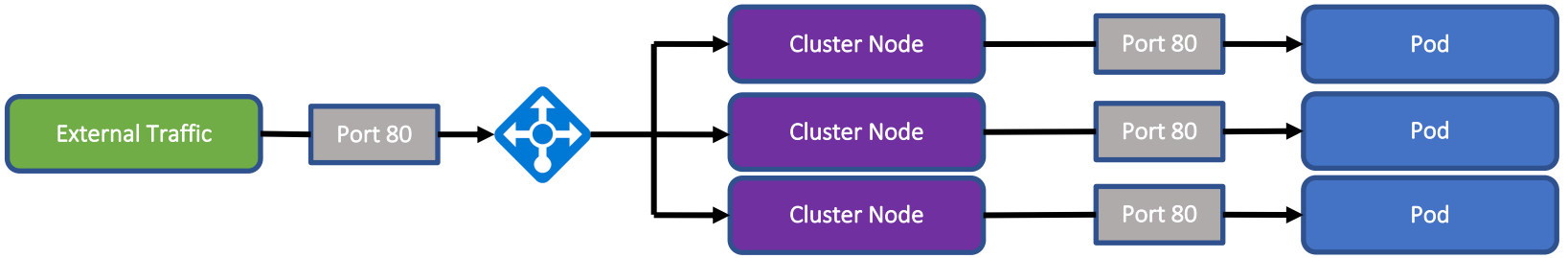


- NodePort: Port mapping from node to application

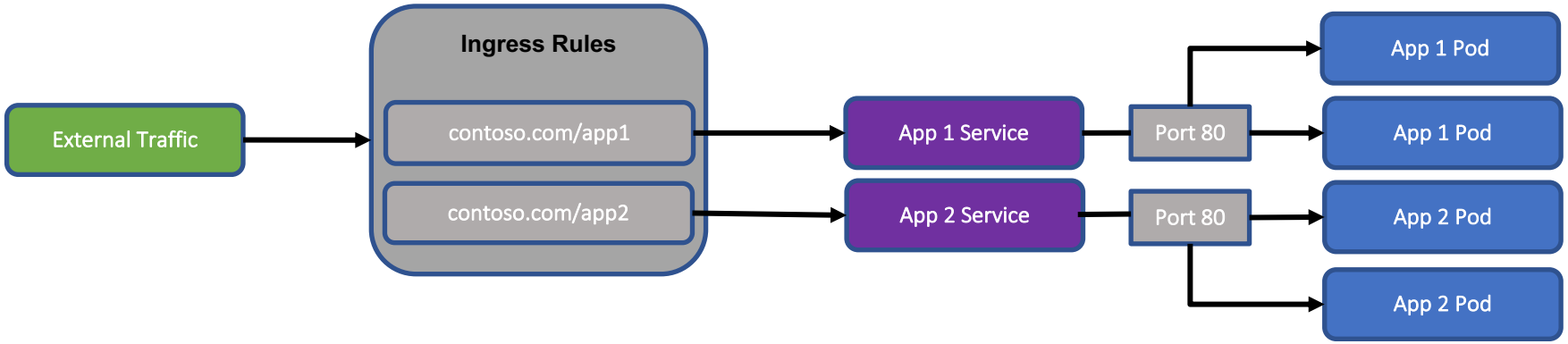


Publishing Kubernetes Services

- LoadBalancer: **Exposes** NodePort **and** ClusterIP **services to external traffic**



- Ingress: **Layer 7 controller to distribute application traffic based**



Imperative Deployment to AKS

- Retag and Push sample image to Docker Hub
- Deploy to AKS
- Expose the Service
- Scale the deployment manually
- Update the version of the deployment
- Rollback the deployment

AKS Scaling

Manual: Use az aks scale **to increase or decrease the node count**

Virtual Node: Burst out using Azure Container Instances

Cluster Autoscaler: Use VMSS to enable automatic scaling

Cluster Autoscaler Functionality

Cluster Autoscaler: scales node count to support new pods

Horizontal pod Autoscaler: Scaled pods to support service requirements

Supports multiple node pools feature

Container Registries

Docker images can be stored locally

Use container registries to share images

Docker Hub hosts public images

- https://hub.docker.com/
- Also supports private image hosting

Azure Container Registry (ACR)

- Store containers in Azure
- Can build images automatically

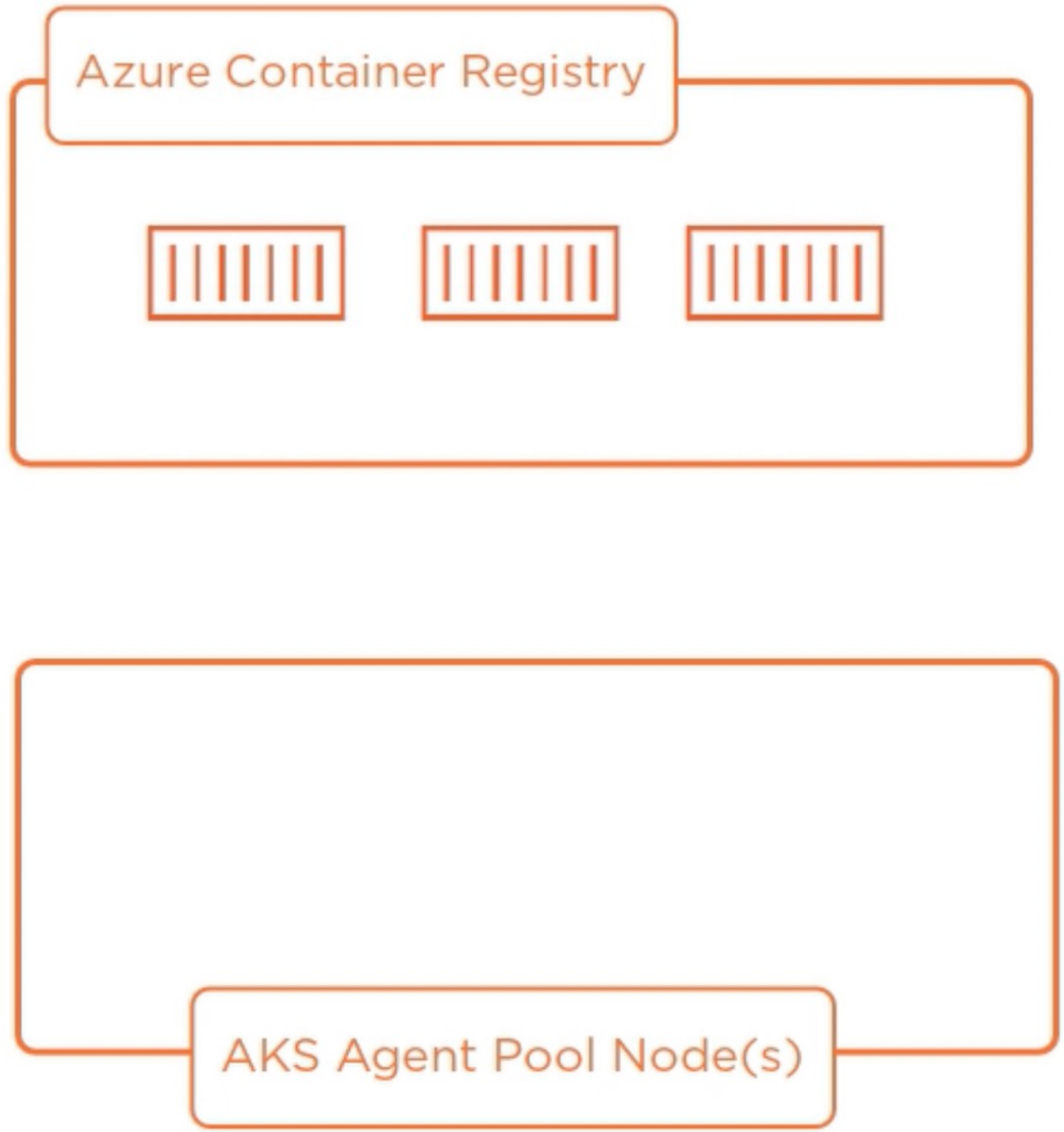
Container Registry Security

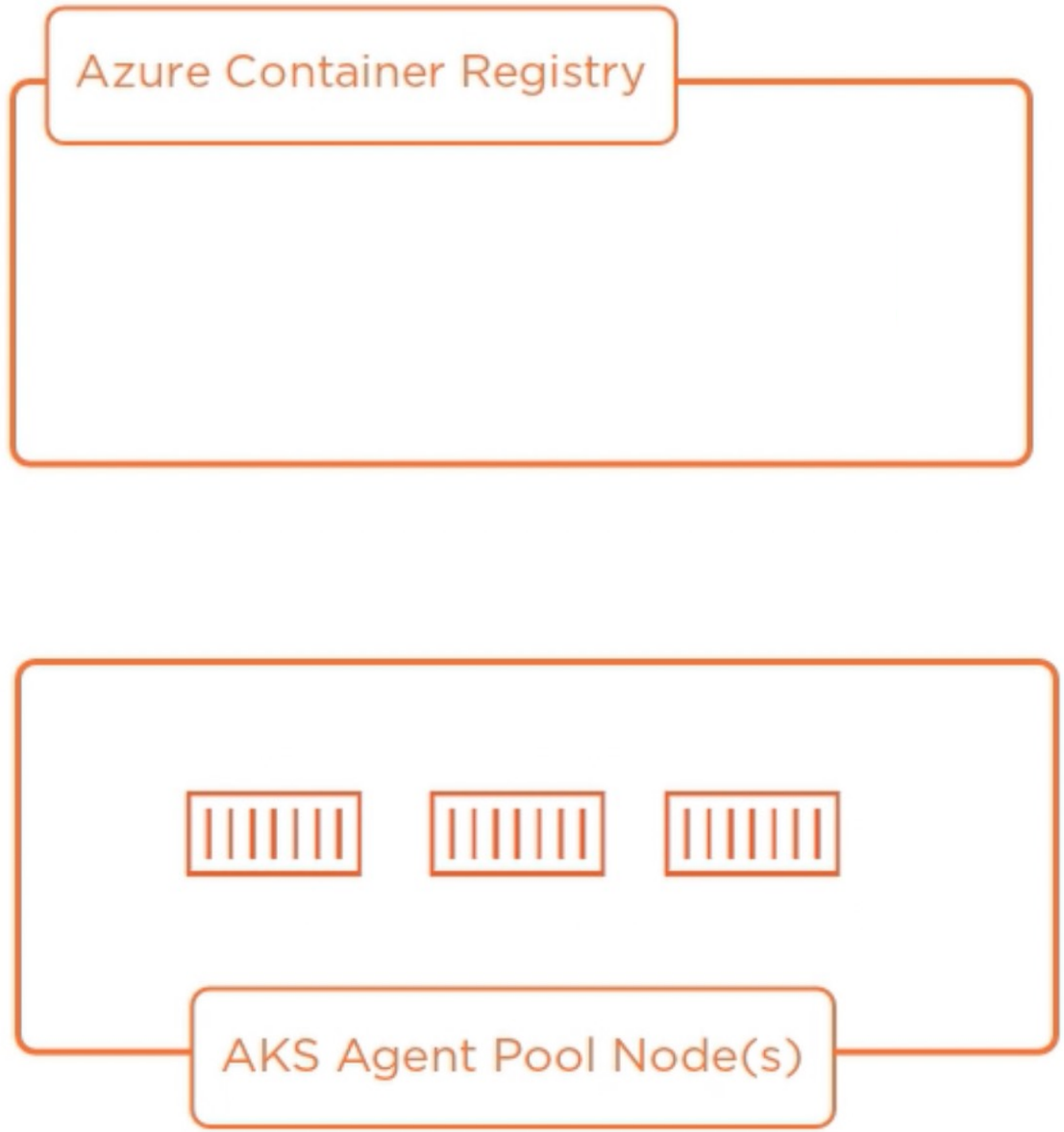
Images not accessible to the public

Can restrict access rights

Only allow trusted images

Automatically update to use latest security updates


Azure Container Registry
AKS Agent Pool Node(s)


Azure Container Registry
AKS Agent Pool Node(s)

Provision and Configure Azure Container Registry

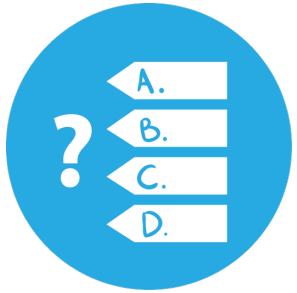
- Create an Azure Container Registry in Azure Portal
- Create a Docker image using a Dockerfile
- Log in to ACR and push Docker image in it
- Run container using Docker Image in ACR
- Verify container service and image in ACR

Declarative Deployment to AKS

Pre-requisites:

1. Install .NET Core
2. Install Docker
3. Install VS Code with the below extensions: Docker, Git, Kubernetes, C#
4. Install Helm 3
5. Azure CLI

Quiz



1. Docker Desktop is an app for building and sharing containerized apps and microservices available on which of the following operating systems?

a. macOS only
b. Linux only
c. Windows, macOS, and Windows Subsystem for Linux (WSL)

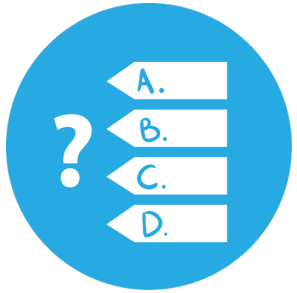
2. Which is correct Docker command to rebuild a container image?

a. docker rebuild
b. docker compile
c. docker build

3. Which of the following sentences describe a container image the best?

a. A container image is a read-only portable package that contains software and may include an operating system.
b. A container image is a set of commands that builds a container.
c. A container image is a read-only portable package that contains software.

Quiz



4. A container is launched using the --publish 8080:80 flag. Which of the following options is the most likely network configuration used for the container?

a. none
b. Bridge
c. Host

5. Suppose you work for a company that builds a Massively Multiplayer Online (MMO) game. You decide to move all your services to Azure Kubernetes service. Which of the following components will contribute to your monthly Azure charge?

a. The Master node
b. Per deployed Pod
c. Per node VM

6. Which of the following Kubernetes objects would you use to expose deployments

a. pods
b. service
c. Replica set

THANK YOU

Develop
Intelligence