Microsoft

AZ-203.1
Module 03: Create containerized solutions

Kishore Chowdary

Microsoft

Topics

- Azure Kubernetes Service (AKS)
- Deploy an AKS cluster
- Publish a container image to Azure Container Registry
- Run container images in Azure Container Instances

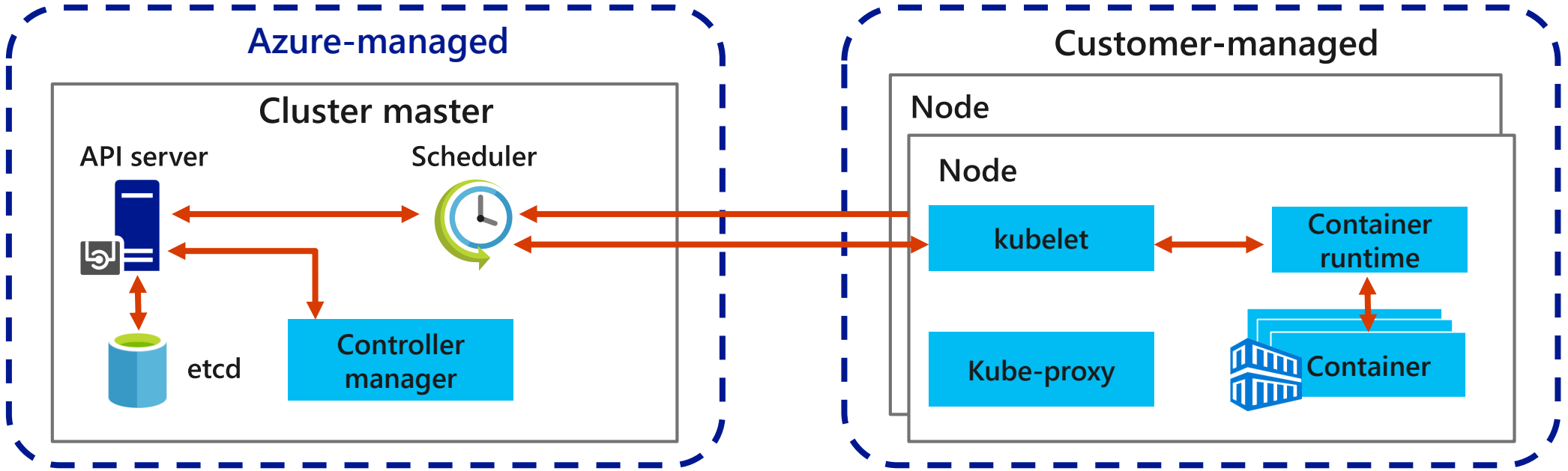
Lesson 01: Azure Kubernetes Service

Kubernetes

- Manages container-based applications
  - Along with networking and storage requirements
  - Focused on application workloads instead of infrastructure components
- Makes it easier to orchestrate large solutions using a variety of containers
  - Application containers
  - Storage containers
  - Middleware containers
  - Even more…
- Applications are described declaratively
  - Use YAML files to describe application
  - Kubernetes handles management and deployment

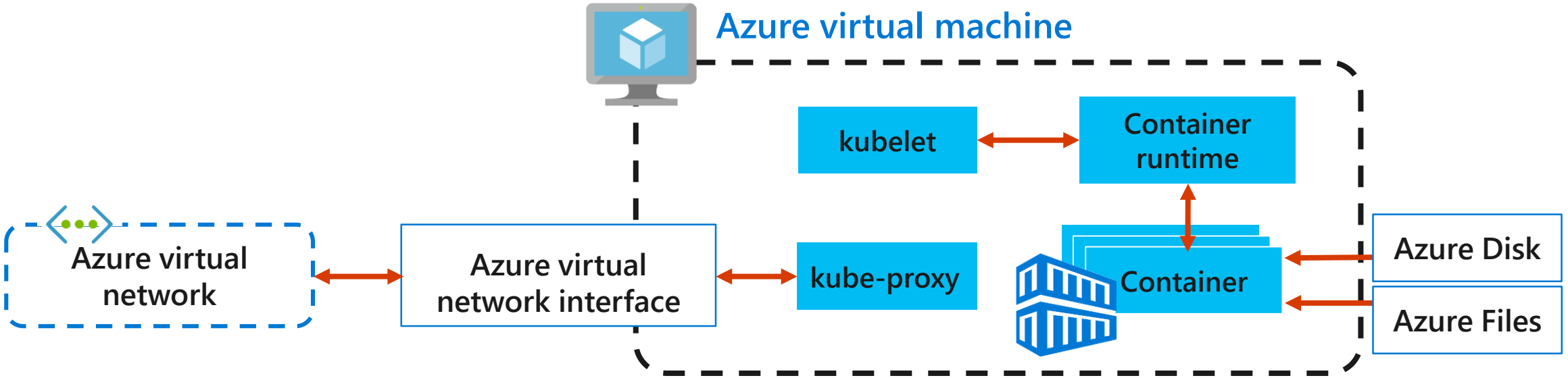
Kubernetes cluster architecture

- Cluster master
  - Offers dedicated nodes that provide core Kubernetes services and orchestration
- Nodes
  - Run application workloads



Kubernetes nodes

- Each individual node is an Azure virtual machine (VM)
  - Contains Kubernetes node components needed to communicate with the cluster master and the internet
  - Contains the container runtime for your applications
    - In Azure Kubernetes Service (AKS), Docker is the container runtime



Kubernetes terminology

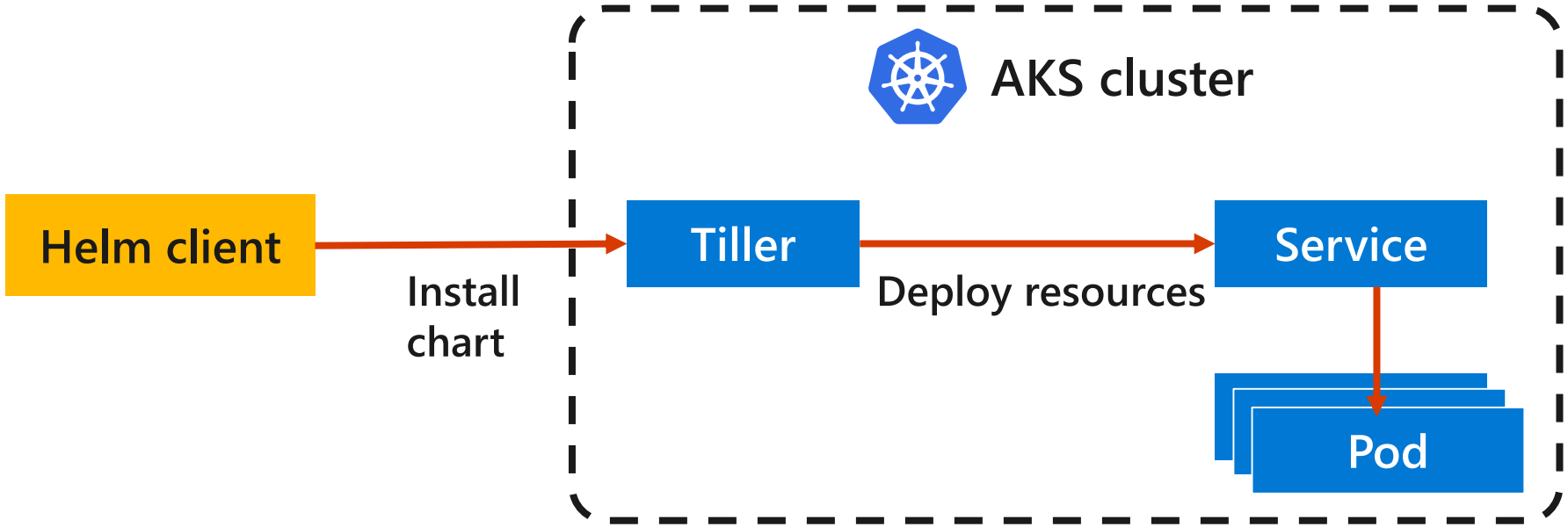
- Nodes
  - Individual VM running containerized applications
- Pools
  - Groups of nodes with identical configurations
- Pods
  - Single instance of an application
  - It’s possible for a pod to contain multiple containers within the same node
- Deployments
  - One or more identical pods managed by Kubernetes
- Manifests
  - YAML file describing a deployment

Azure Kubernetes Service (AKS)

- Simple management of cluster of VMs by using Kubernetes
  - Removes infrastructure complication and planning
  - No cluster charges, just used resources
  - Secure, reliable, highly scalable
  - Supports node autoscaling to meet resource utilization
  - Supports in-place upgrade of clusters to the latest version of Kubernetes
  - Direct integration with Container Registry for Docker images
  - Azure Virtual Network integration for isolated network traffic

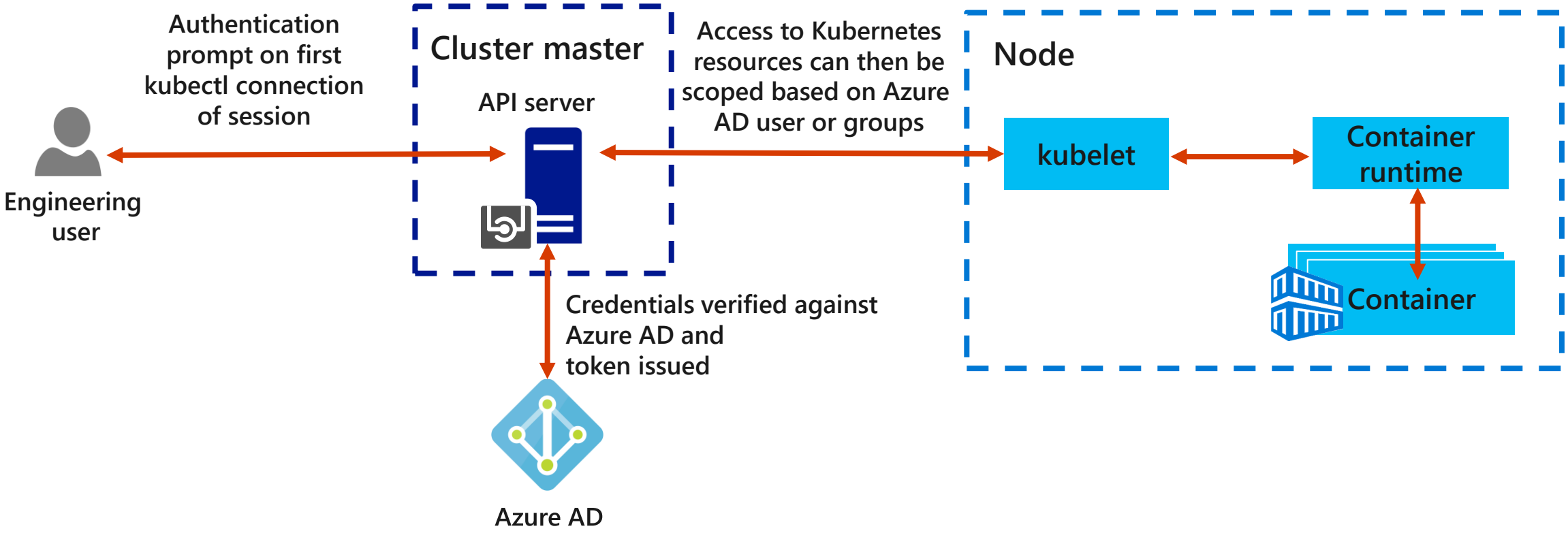
Package management with Helm

- Helm charts are a package that makes it easier to manage applications
- Helm packages contain:
  - Kubernetes YAML manifest(s)
  - Packaged version of application code
- Can be stored locally or in a remote repository



Azure AD integration

You can use Azure AD as an integrated identity solution



Role-based access control (RBAC)

- Assign users or groups permissions to manage Kubernetes
- Permissions include:
  - Create or modify resources
  - View logs
  - Deploy application workloads
- Permissions can be scoped to a single namespace or the entire AKS cluster
- Permissions can be encapsulated in a RBAC role definition

Security

- Master security
  - Fully managed by Microsoft without any need for user input
  - Has a public IP and fully qualified domain name (FQDN) by default, and this can be managed by using RBAC or Azure AD
- Node security
  - Automatically deployed with the latest OS security updates and configuration
  - Azure platform automatically applies OS security updates
    - If updates require a reboot, you must do this manually
  - Nodes are deployed into a virtual network private subnet
- Cluster upgrades
  - You can trigger an AKS platform upgrade by using the existing orchestration tools
  - AKS safely drains each node and performs upgrades

Networking security

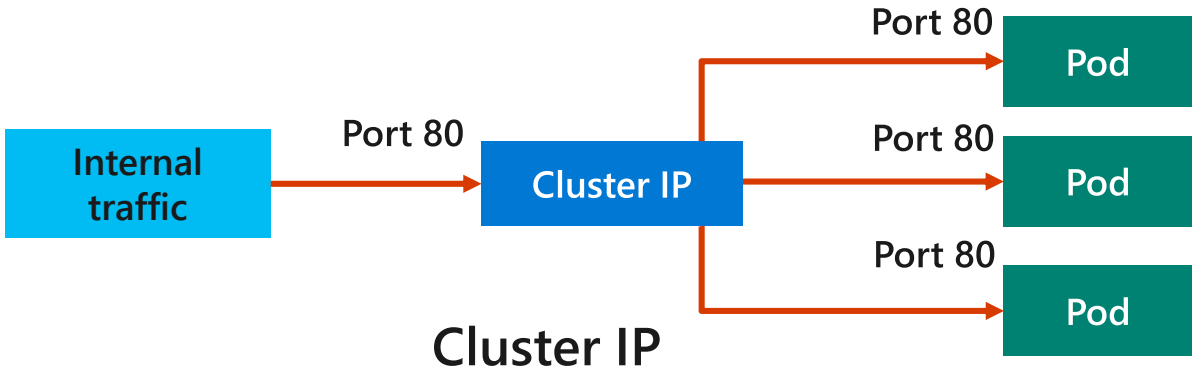
- Network Security Groups
  - Built-in Azure service to protect resources within virtual networks
  - Can define rules to manage:
    - Destination or source IP ranges
    - Portals
    - Protocols
  - Default rules are created to allow:
    - TLS traffic to Kubernetes API server
    - SSH access to individual nodes
  - Network security group (NSG) is automatically modified by AKS as you create services with:
    - Ingress routes
    - Port mappings
    - Load balancers

Networking connectivity

Services group pods together to provide network connectivity

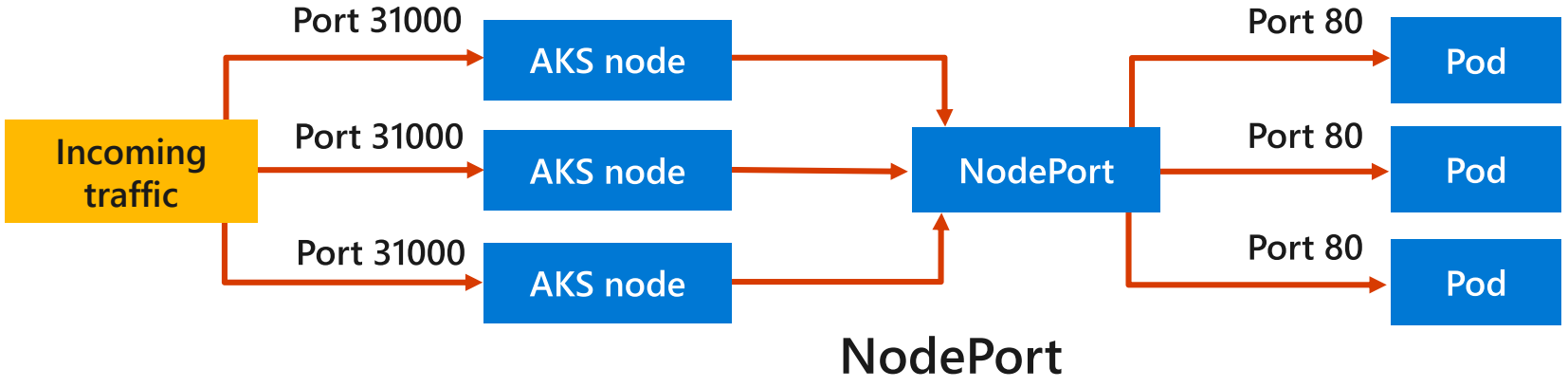
Cluster IP

Creates internal-only IP address for use within the cluster



NodePort

Creates a port mapping on a specific node for direct access

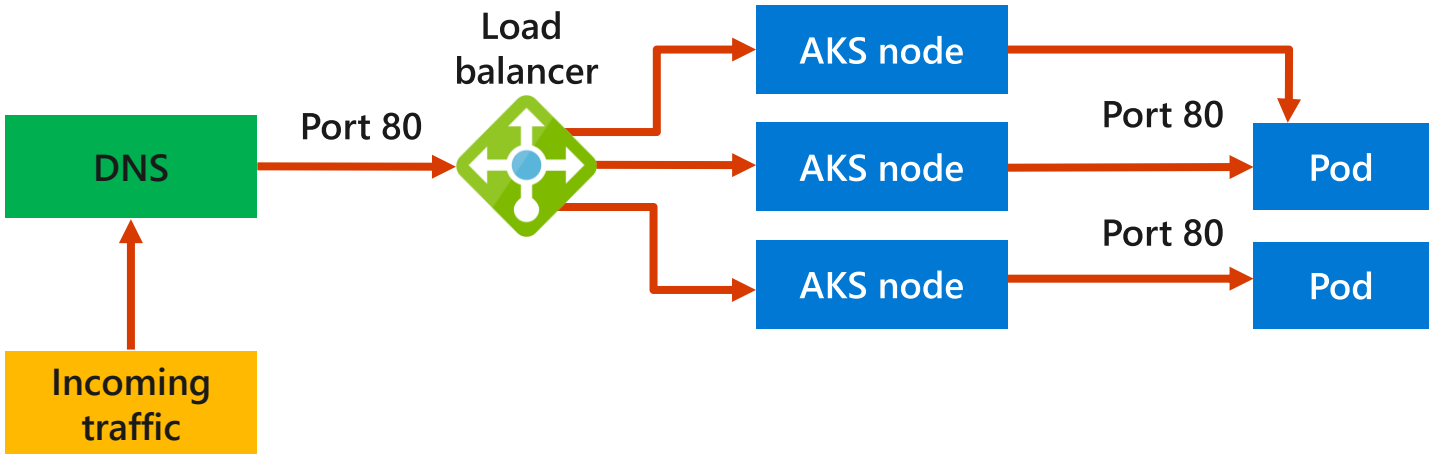


Networking connectivity (continued)

Services group pods together to provide network connectivity

LoadBalancer

Creates an Azure load balancer resource with an external IP address

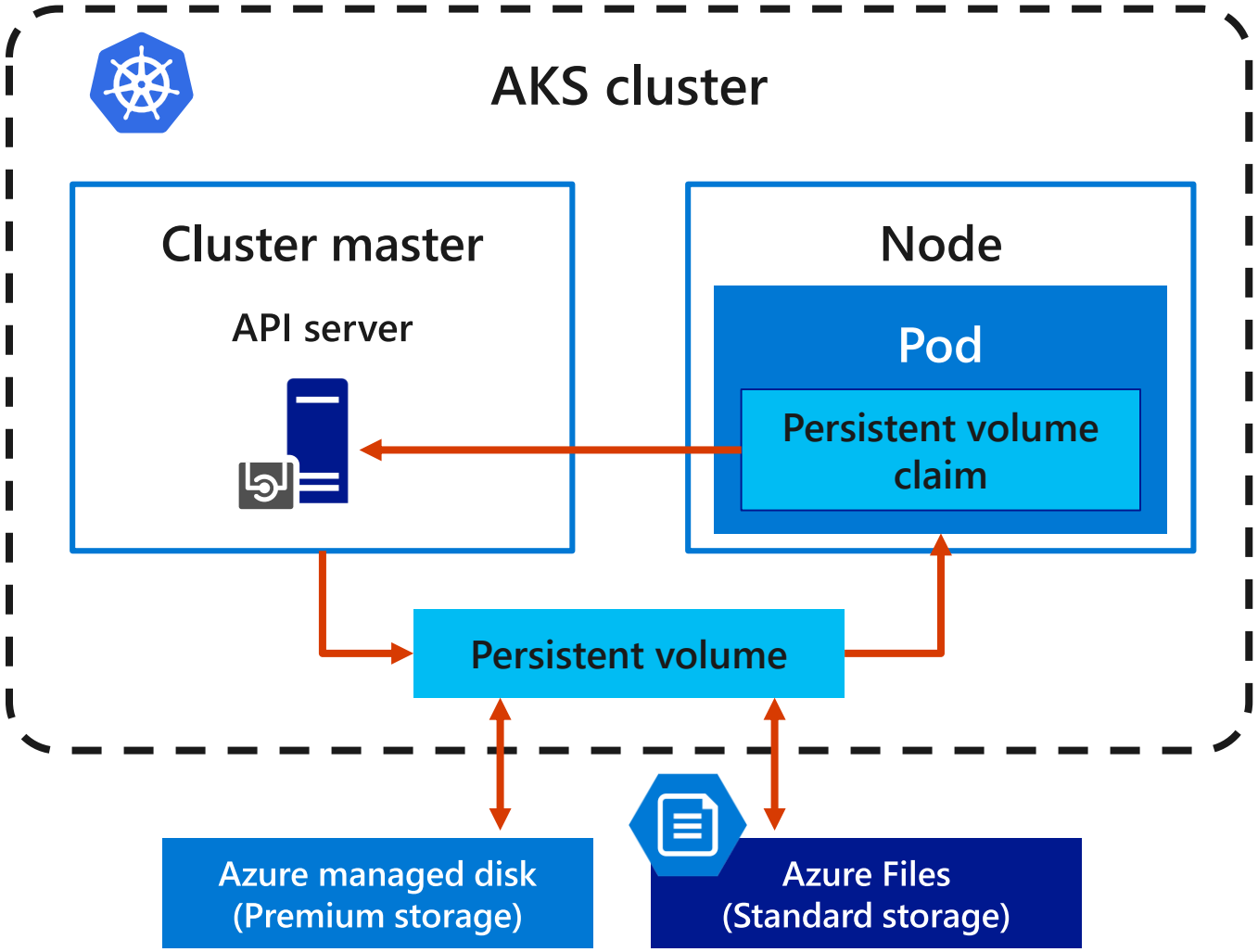


ExternalName

Creates a direct DNS entry

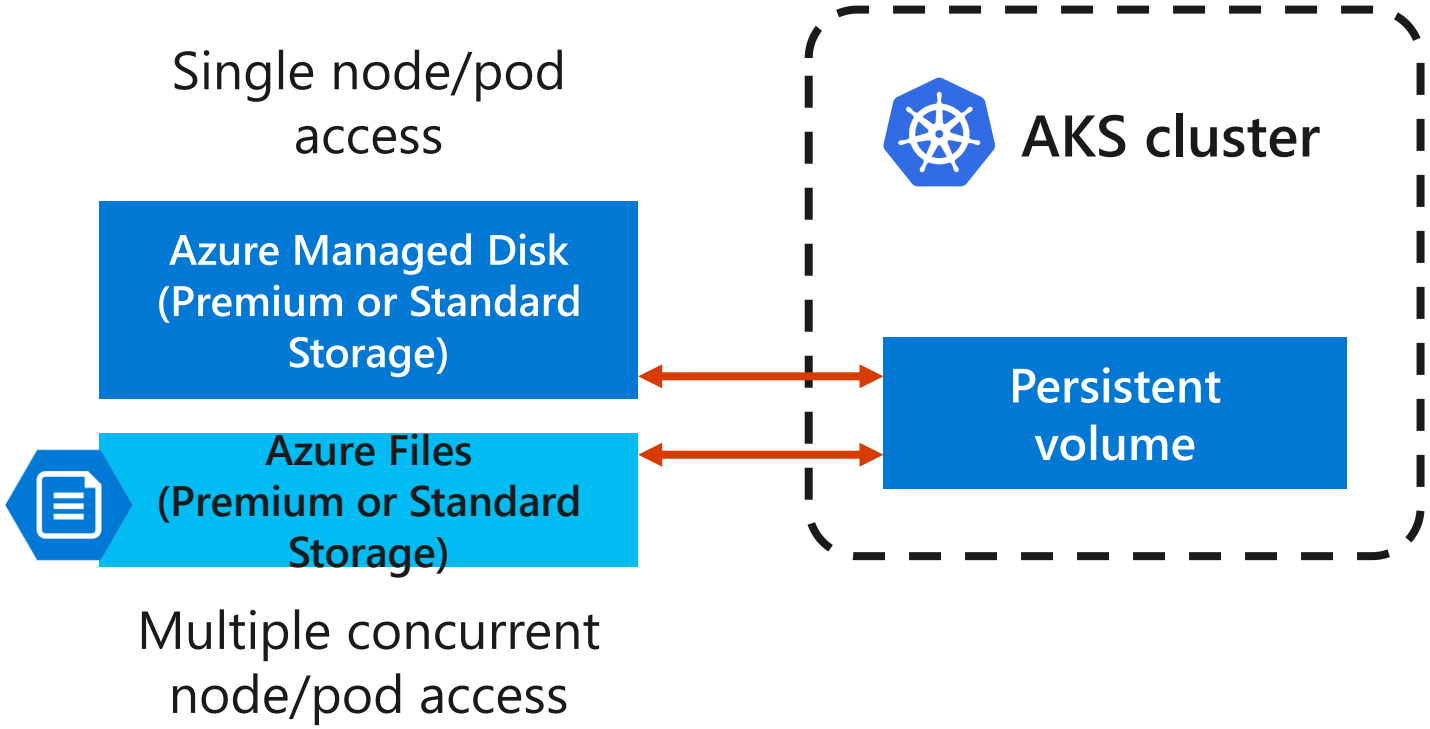
Storage

- Local storage on the node is fast and simple to use
  - Local storage might not be available after the pod is deleted
- Multiple pods may share data volumes
- Storage could potentially be reattached to another pod



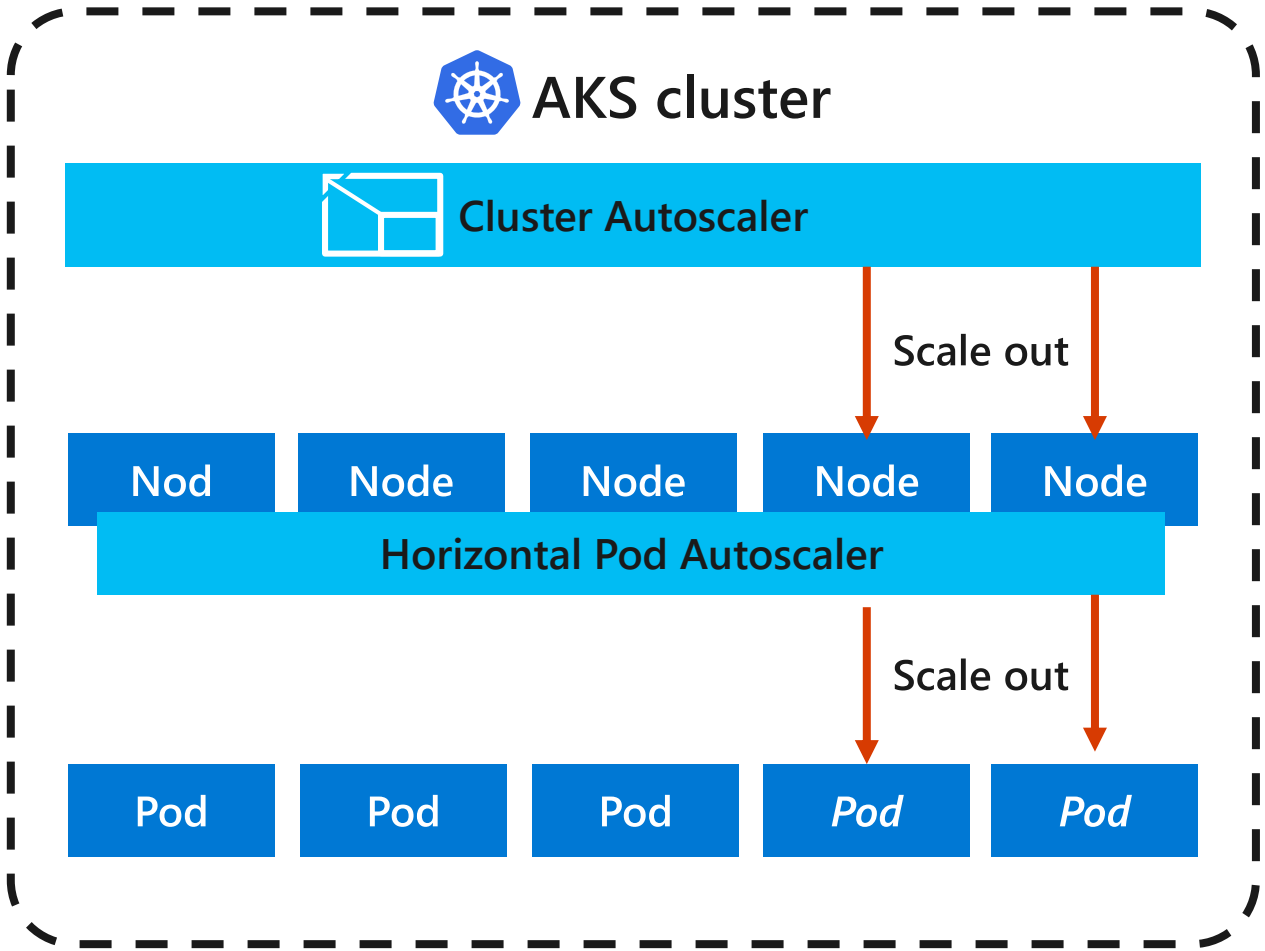
Persistent storage volumes

- Pods can use storage that is persistent
  - Storage exists beyond the lifetime of the pod
  - Storage can be a service, such as Azure Files or an Azure Managed Disk



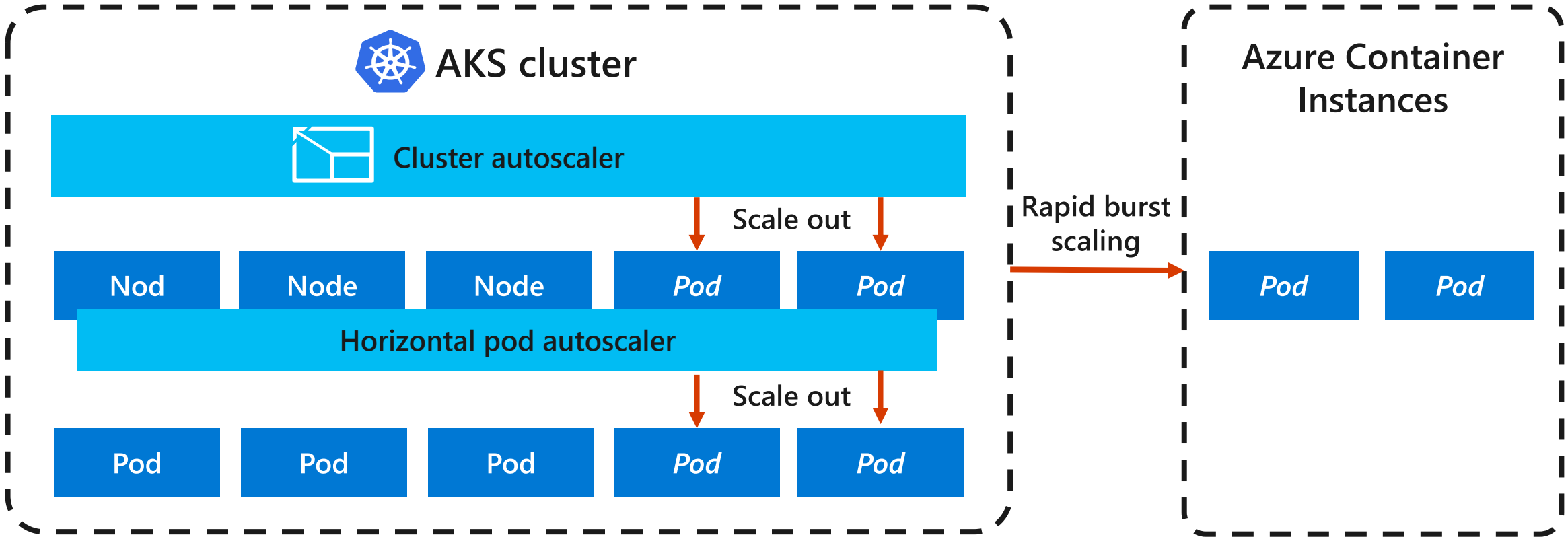
Scaling

- Applications might grow beyond the capacity of a single pod
- Kubernetes has built-in autoscalers to automatically create instances when they are needed
  - Horizontal pod autoscaler
    - Automatically scales replicas based on metrics
  - Cluster autoscaler
    - Adjusts the number of nodes based on the requested compute resources



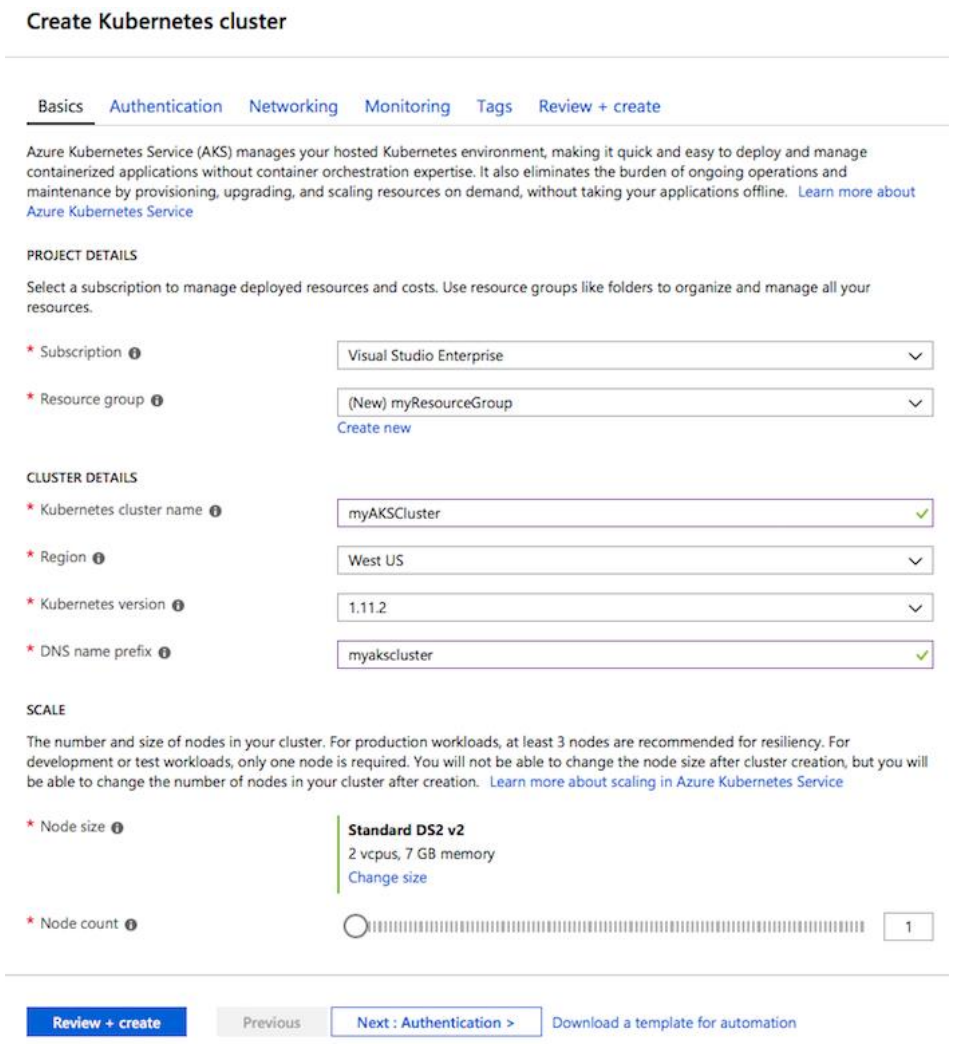
Scaling to Azure Container Instances

If you need to **rapidly** grow your AKS cluster, you can create new pods in Azure Container Instances



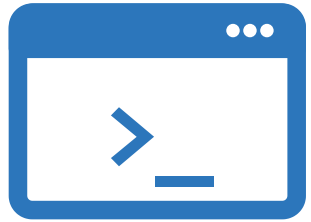
Lesson 02: Deploy an AKS cluster

Deploying an AKS cluster by using the Azure portal



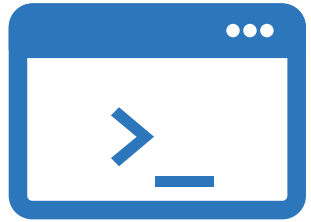
Deploying to AKS by using Azure CLI

```
# Create resource group
az group create --name mygroup --location eastus

# Create AKS cluster
az aks create --resource-group mygroup --name mycluster --node-count 1 --enable-addons
monitoring --generate-ssh-keys
```



Connecting a kubectl client to AKS

```
# Install kubectl command line client locally
az aks install-cli

# Get credentials
az aks get-credentials --resource-group mygroup --name mycluster

# Get a list of cluster nodes
kubectl get nodes
```

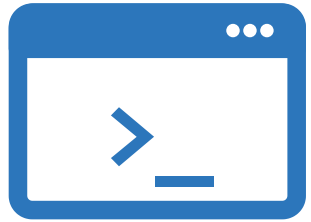


Kubernetes manifest

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  selector:
    matchLabels:
      app: nginx
    replicas: 2 # tells deployment to run 2 pods matching the template
    template:
      metadata:
        labels:
          app: nginx
      spec:
        containers:
        - name: nginx
          image: nginx:1.7.9
          ports:
          - containerPort: 80
```

Deploying application to AKS

```
# Get a list of cluster nodes
kubectl get nodes

# Run the application
kubectl apply -f example.yaml

# Monitor progress of the deployment
kubectl get service nginx-deployment -watch

# View pods related to this deployment
kubectl get pods -l app=nginx
```



Demo: Deploying to AKS by using Azure CLI and Portal

Lesson 03: Publish a container image to Container Registry

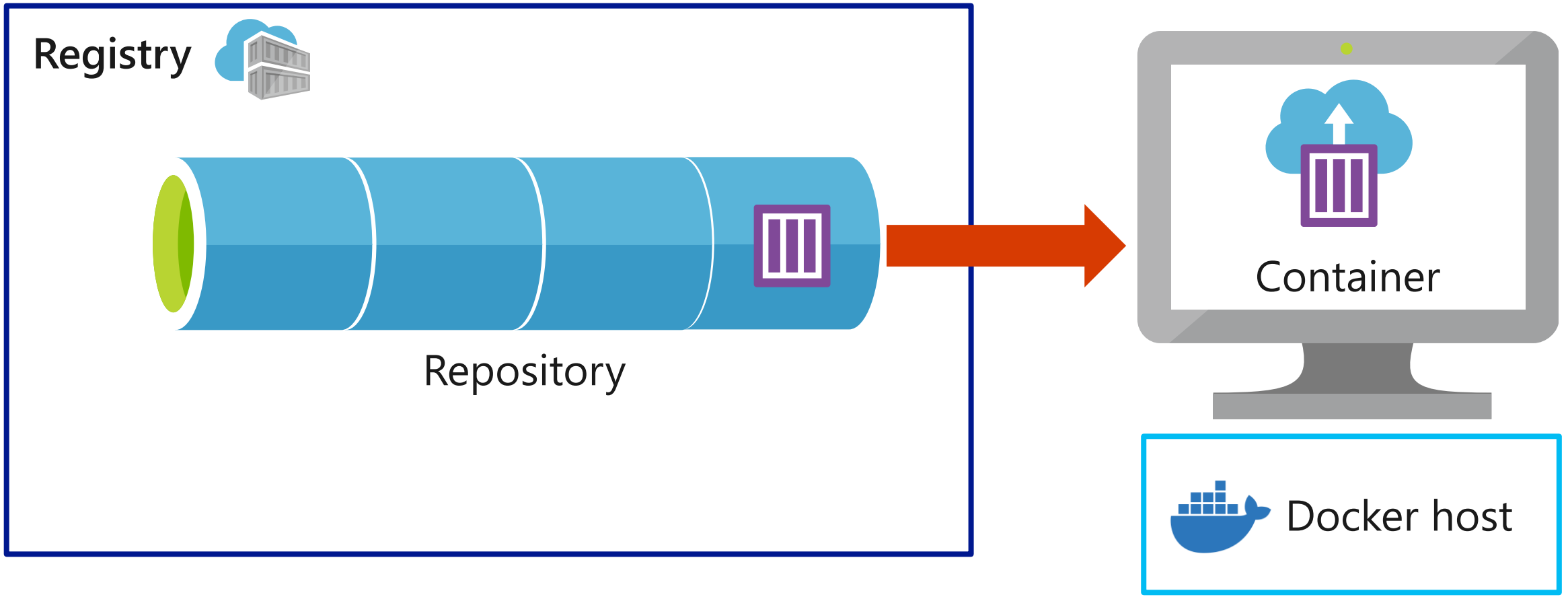
Azure Container Registry (ACR)

- Managed Docker registry service
  - Based on the open-source Docker Registry 2.0
- Stores and manages private Docker container images
- Tight integration with multiple Azure services that support these Docker containers:
  - Azure App Service
  - Azure Batch
  - Azure Service Fabric
  - Azure Kubernetes Service

Key terminology

- **Registry**
  - A service that stores container images
- **Repository**
  - A group of related container images
- **Image**
  - A point-in-time snapshot of a Docker-compatible container
- **Container**
  - A software application and its dependencies running in an isolated environment

Docker containers and registries

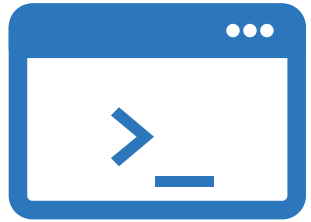


Container Registry SKUs

SKU	Description
Basic	• Ideal for developers learning about Container Registry • Same programmatic capabilities as Standard and Premium, however, there are size and usage constraints
Standard	• Same capabilities as Basic, but with increased storage limits and image throughput. • Should satisfy the needs of most production scenarios.
Premium	• Higher limits on constraints, such as storage and concurrent operations, including enhanced storage capabilities to support high-volume scenarios. • Adds features like geo-replication for managing a single registry across multiple regions

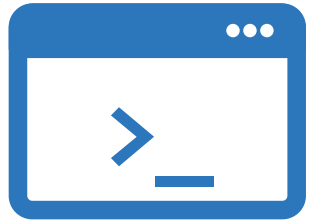
Create a container registry by using Azure CLI

```
# Create a Container Registry instance
az acr create --resource-group <group> --name <acr-name> --sku Basic

# Login to Container Registry
az acr login --name <acrName>
```



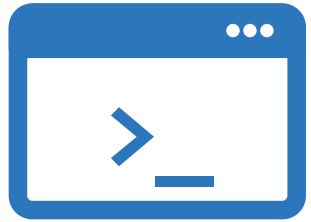
Build a Docker image for Container Registry

```
# Pull existing Docker image
docker pull microsoft/aci-helloworld

# Obtain the full login server name of the Container Registry instance
az acr list --resource-group <group> --query "[].{acrLoginServer:loginServer}" --output
table

# Tag image with full login server name prefix
docker tag microsoft/aci-helloworld <acrLoginServer>/aci-helloworld:v1

# Push image to Container Registry
docker push <acrLoginServer>/aci-helloworld:v1
```



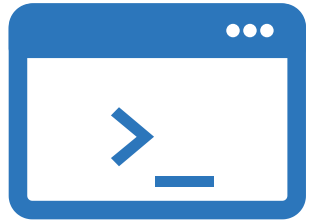
View a deployed image in Container Registry by using Azure CLI

```
# List container images
az acr repository list --name <acrName> --output table

# List the tags on the aci-helloworld repository
az acr repository show-tags --name <acrName> --repository aci-helloworld --output table
```



Deploy an image to Container Registry by using Azure CLI

```
# Enable admin user
az acr update --name <acrName> --admin-enabled true

# Query for the password
az acr credential show --name <acrName> --query "passwords[0].value"

# Deploy container image
az container create --resource-group <group> --name acr-quickstart --image
<acrLoginServer>/aci-helloworld:v1 --cpu 1 --memory 1 --registry-username <acrName> --
registry-password <acrPassword> --dns-name-label <fqdn> --ports 80

# View container FQDN
az container show --resource-group myResourceGroup --name acr-quickstart `
    --query instanceView.state
```



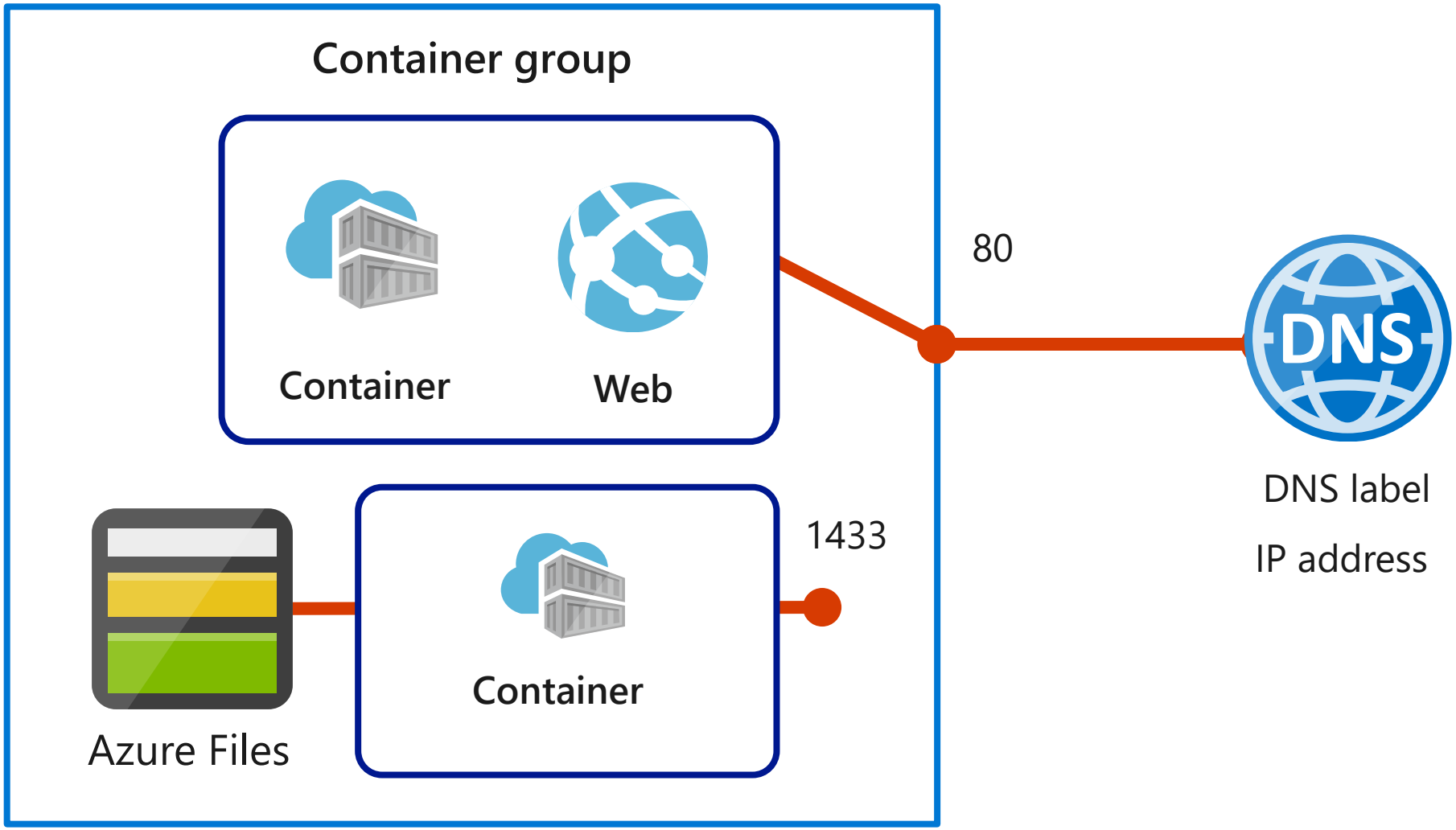
Demo: Deploy an image to Container Registry

Lesson 04: Run container images in Container Instances

Azure Container Instances (ACI)

- Simplest way to run a container in Azure:
  - Doesn’t require IaaS provisioning
  - Doesn’t require the adoption of a higher-level service
- Ideal for one-off, isolated container instances:
  - Simple applications
  - Task automation
  - Build jobs
- Supports Linux and Windows containers
- Supports direct mounting of Azure Files shares
- Container can be provisioned with public IP address and DNS name

Container groups



Container Instances features

Feature	Description
Fast startup times	Containers can start in seconds without the need to provision and manage VMs
Public IP connectivity and DNS name	Containers can be directly exposed to the internet with an IP address and a fully qualified domain name (FQDN)
Hypervisor-level security	Container applications are as isolated in a container as they would be in a VM
Custom sizes	Container nodes can be scaled dynamically to match actual resource demands for an application
Persistent storage	Containers support direct mounting of Azure Files shares
Linux and Windows containers	The same API is used to schedule both Linux and Windows containers
Co-scheduled groups	Container Instances supports scheduling of multicontainer groups that share host machine resources
Virtual network deployment	Container Instances can be deployed into an Azure virtual network

Docker terminology

- **Container**
  - A standardized "unit of software" that contains everything required for an application to run
- **Container Image**
  - A template that can be used to create one or more containers
- **Build**
  - The process of creating a container image using a set of instructions
- **Pull**
  - The process of downloading a container image from a container registry
- **Push**
  - The process of uploading a container image to a container registry
- **Dockerfile**
  - A text file that contains instructions required to build a Docker image.

Demo: Deploy a container to Container Instances

Microsoft

Review

- Azure Kubernetes Service
- Deploy an AKS cluster
- Publish a container image to Azure Container Registry
- Run container images in Azure Container Instances