Running Kubernetes in Azure

VAIBHAV GUJRAL
CLOUD ARCHITECT | MICROSOFT AZURE MVP

About me

Director, Global Microsoft Cloud CoE at Capgemini

Born and brought up in India and based out of Omaha, NE since 2016

Microsoft Azure MVP since 2020

Leader, Omaha Azure User Group(https://omahaazure.org)

15+ cloud certifications and counting…

What is Kubernetes?

Kubernetes is a portable, extensible, open-source platform for automating the deployment, scaling, and management of containerized workloads.

Kubernetes (k(j)uːbərˈnɛtɪs)

Greek for “helmsman of a ship”

Kubernetes History

First announced by Google in 2014

Heavily influenced by Google's ***Borg*** system.

Original codename for Kubernetes project was Project 7 (a reference to the Star Trek ex-Borg character Seven of Nine)

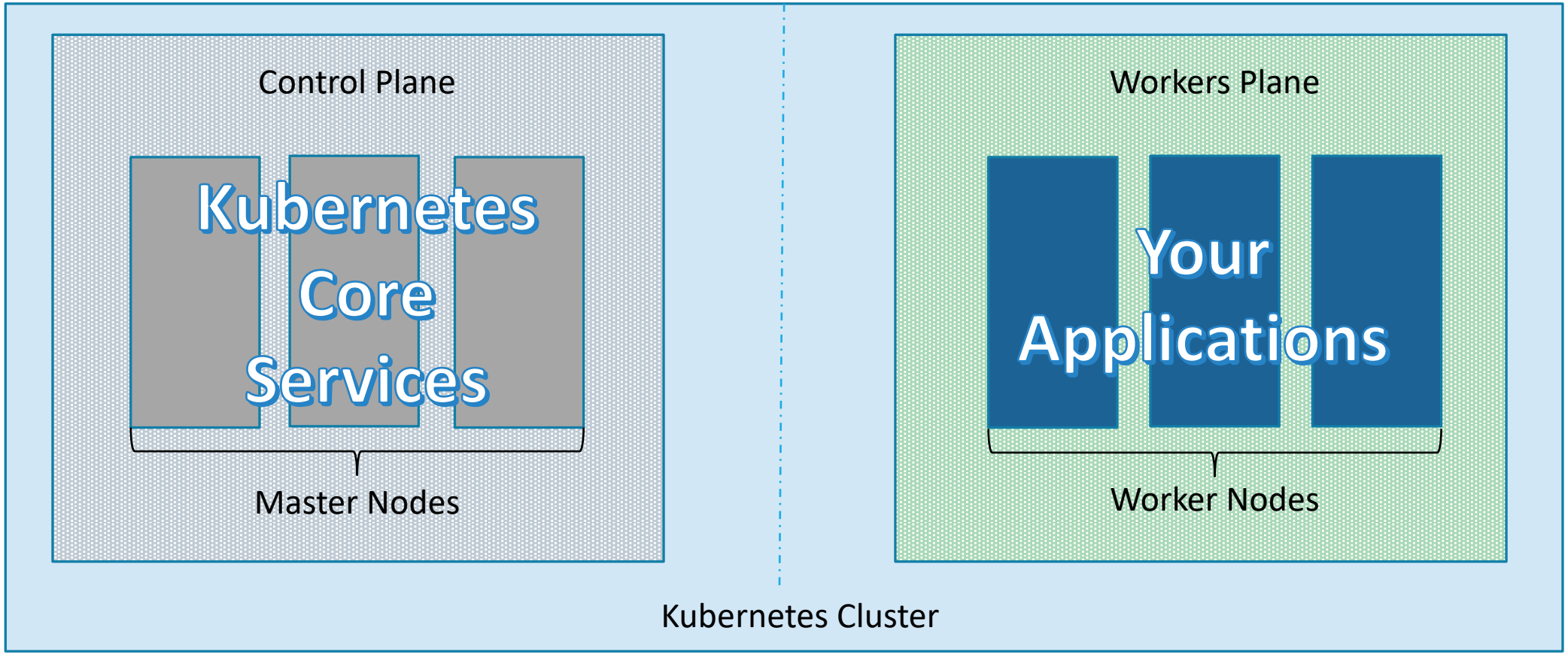
V1.0 was released on July 21, 2015

Current version is *1.26.1 (released on 01/18/2023)* (Release History)

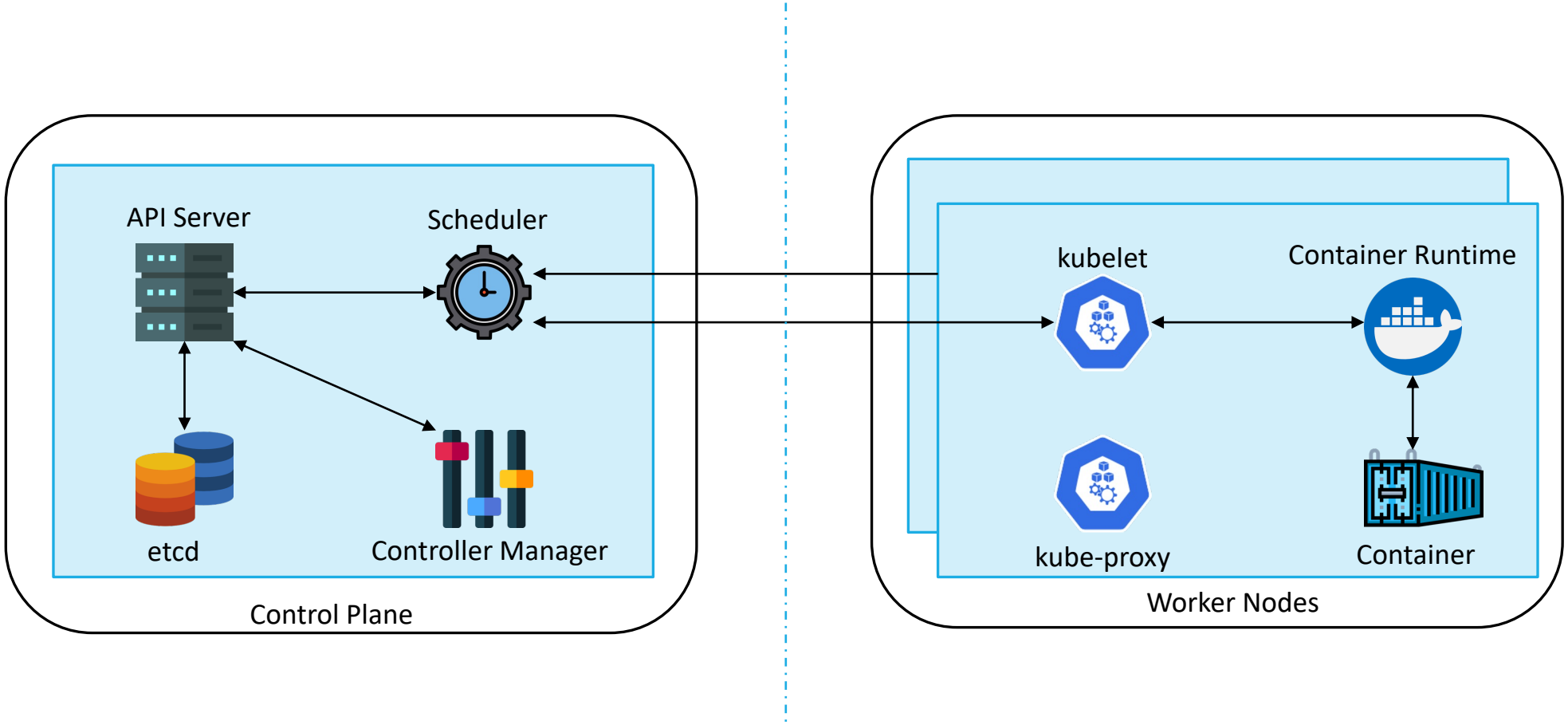
Version *1.27* will be available 04/11/2023 (Schedule)

Originally written in C++, the current system is written in Go language.

Kubernetes Architecture – Big Picture



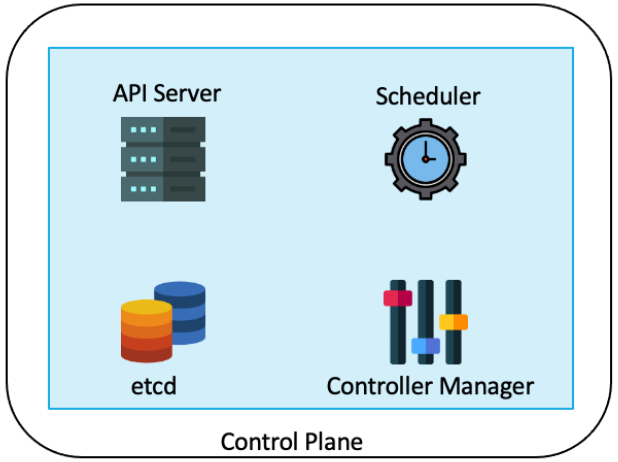
Kubernetes Architecture

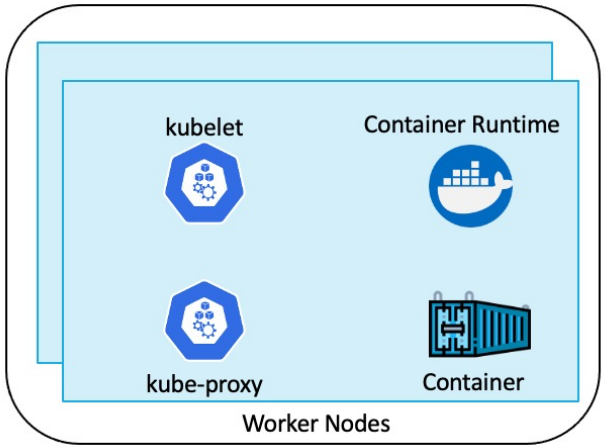


Kubernetes Components

1. **Control Plane**: manages the agent nodes and the pods in the cluster
   - **api-server**: front end of the Kubernetes control plane; exposes Kubernetes API
   - **controller-manager**: runs the controller processes
   - **scheduler**: tracks newly created pods and selects node to run them on
   - **etcd**: stores the state of the cluster (config, running workloads status, etc.)

2. **Worker nodes**: run your application workloads
   - **Pods**: a collection of containers co-located on a single machine
   - **kube-proxy**: a network proxy that runs on each node in a cluster
   - **kubelet**: agent that runs on each node in a cluster; ensures containers are running in a pod
   - **Containers**: software responsible for running containers





Kubernetes Core Concepts

Pods are smallest unit in Kubernetes providing an abstraction over containers. Pods are ephemeral and get their own IP Address.

Services provide a persistent IP Address for a set of pods running an application and acts like a load balancer. The lifecycle of a service is not linked to the lifecycle of a pod.

Ingress exposes HTTP and HTTPS routes from outside the cluster to services within the cluster.

ConfigMaps are text-based key-value stores to store the external configuration for your application.

Secrets are base-64 encoded store for confidential data like passwords and secrets.

Volumes offer data storage for persistent data that needs to exist beyond the lifecycle of a pod.

Kubernetes Core Concepts

A ***ReplicaSet***'s purpose is to maintain a stable set of replica Pods running at any given time.

A ***Deployment*** provides declarative updates for Pods and ReplicaSets.

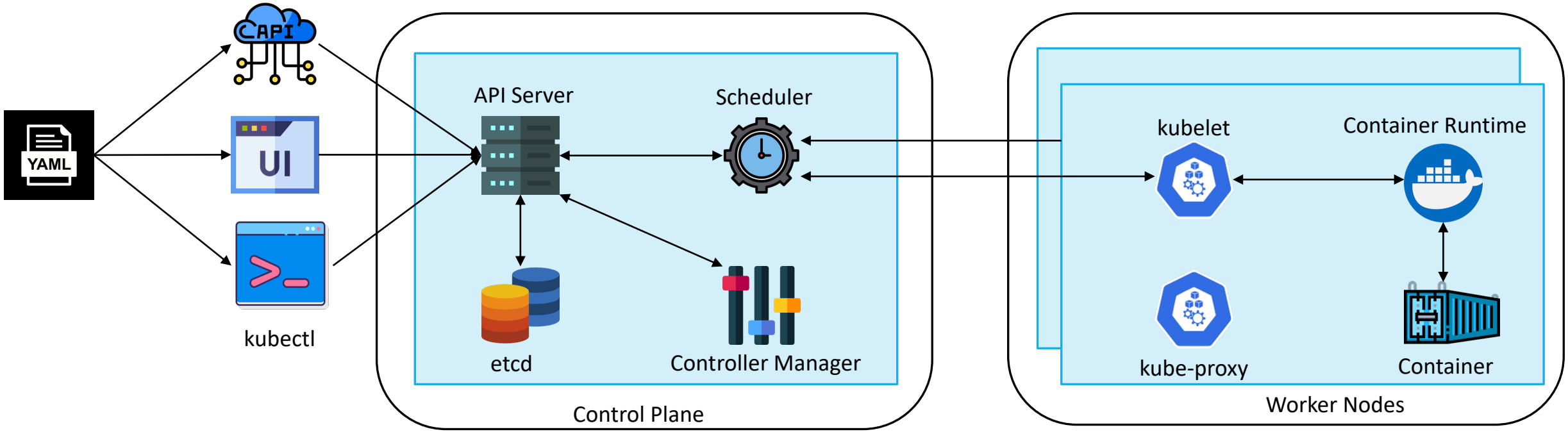
A ***StatefulSet*** is similar to deployment, but it maintains a sticky identity for each of their Pods.

A ***DaemonSet*** is used for deploying ongoing background tasks that you need to run on all or certain nodes, and which do not require user intervention.

A ***Job*** creates one or more Pods and continues to retry execution of the Pods until a specified number of them successfully terminate.

A ***CronJob*** creates Jobs on a repeating schedule.

Managing Kubernetes Cluster



[GitHub - kelseyhightower/kubernetes-the-hard-way: Bootstrap Kubernetes the hard way on Google Cloud Platform. No scripts.]

Azure

+

Kubernetes

=

Azure Kubernetes Service

Azure Kubernetes service

Hosted Kubernetes service in Azure

Reduces the complexity and operational overhead of managing Kubernetes

The control plane is provided as a managed Azure resource abstracted from the user

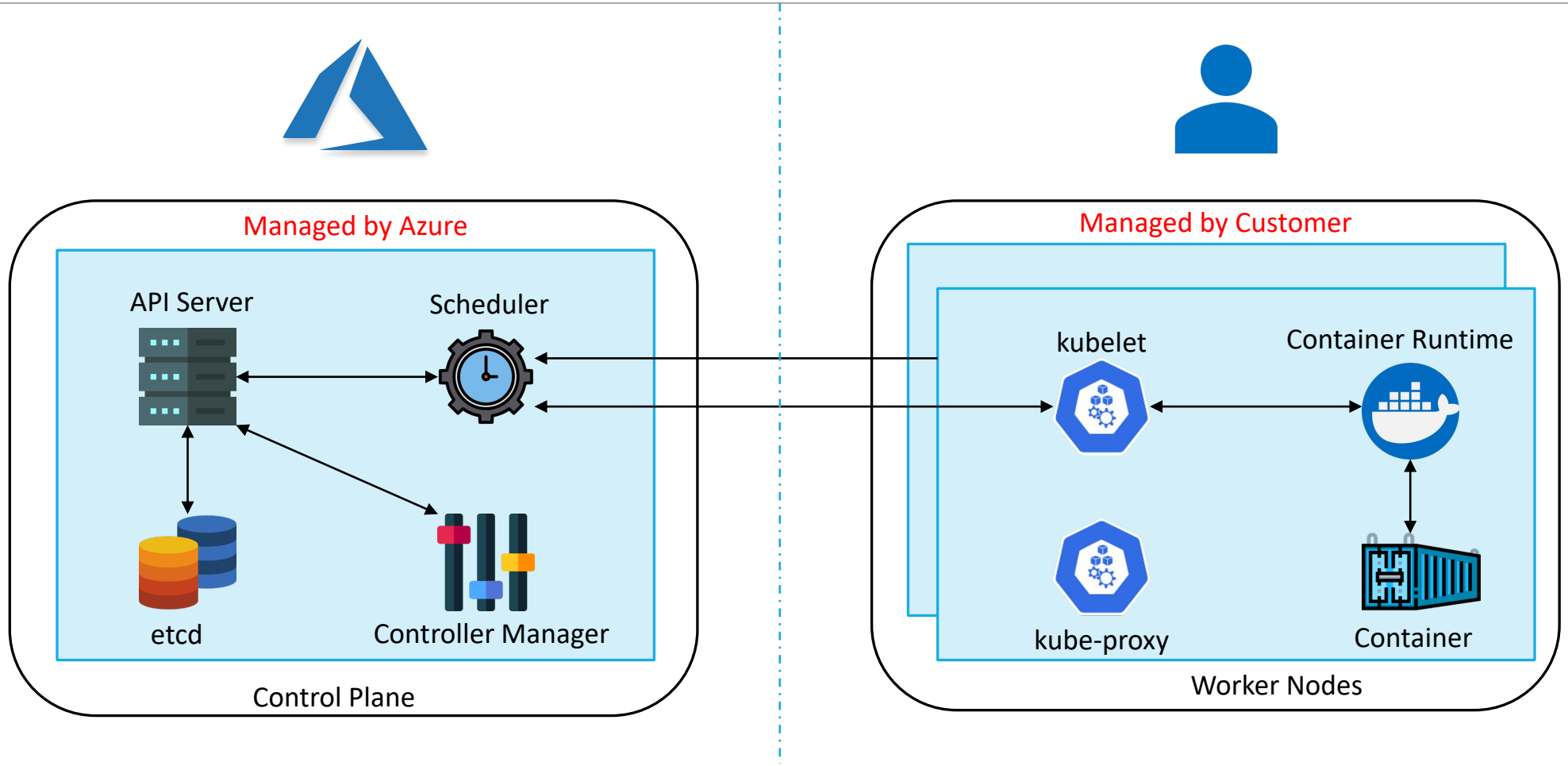
The control plane includes the core Kubernetes components like kube-apiserver, etcd, kube-scheduler and kube-controller-manager

An AKS cluster has one or more worker nodes, which is an Azure virtual machine (VM) that runs the Kubernetes node components and container runtime

You only pay for worker nodes and control plane comes at no charge to you

https://docs.microsoft.com/en-us/azure/aks/

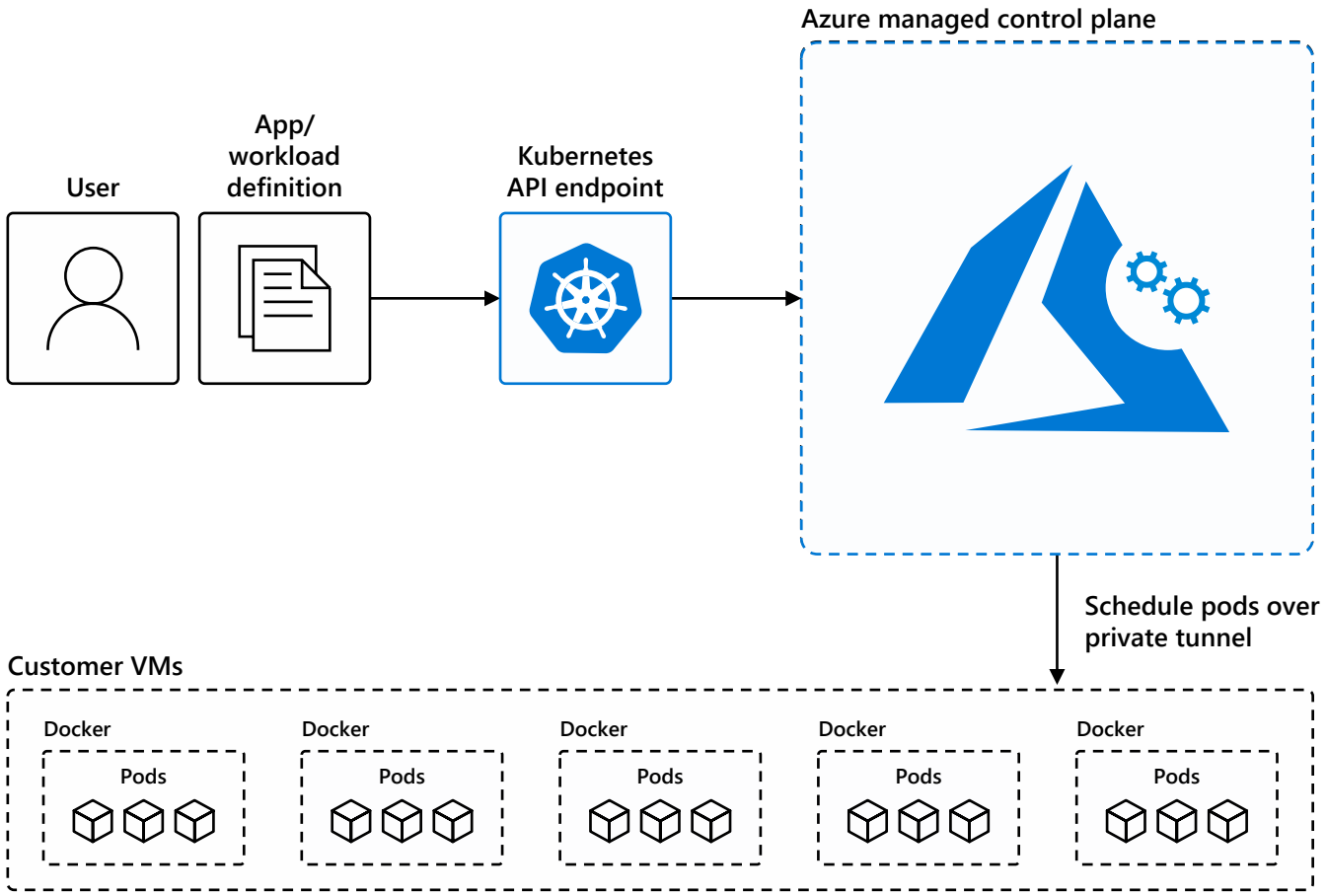
Azure Kubernetes Service



Shared Responsibility

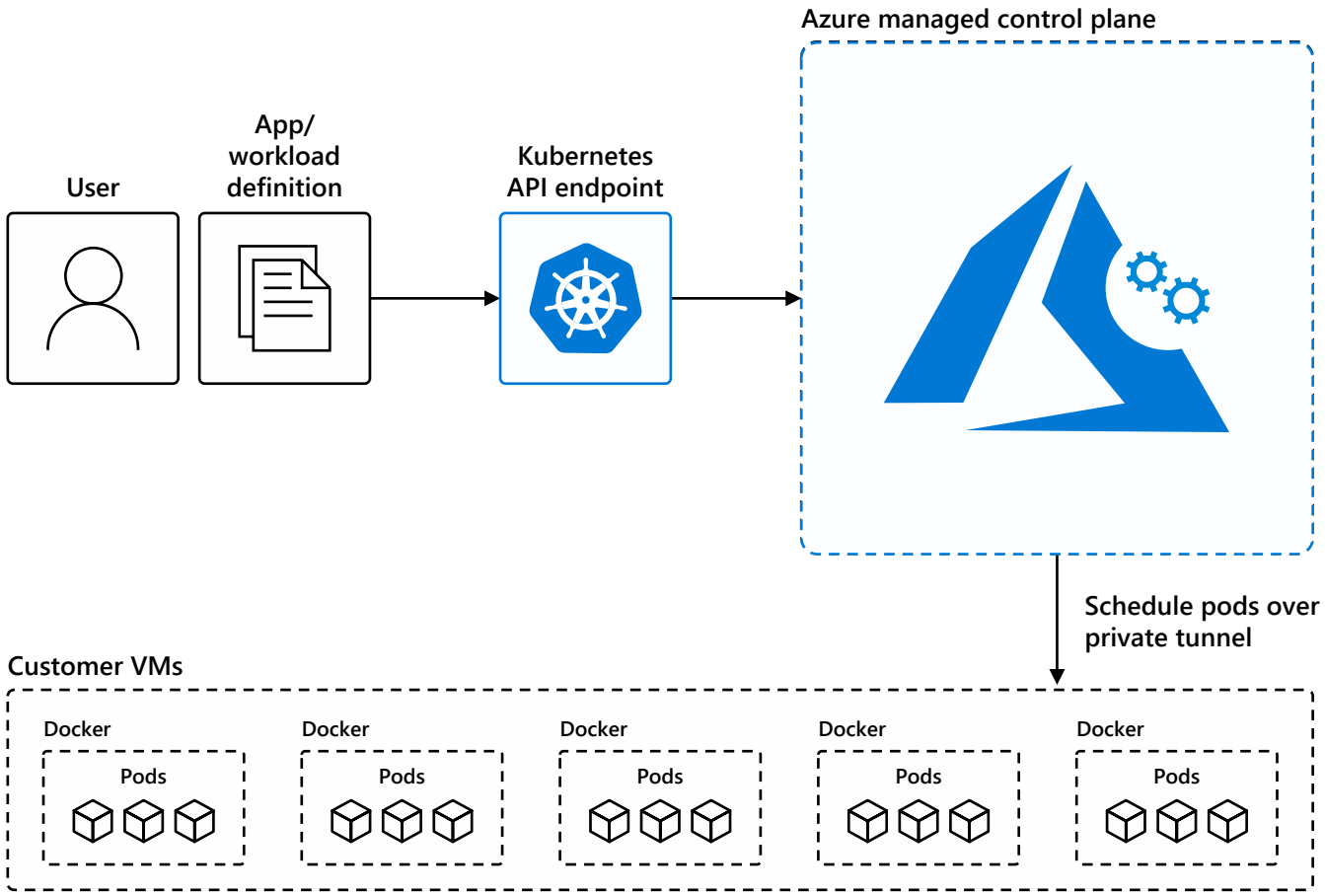
Responsibilities	DIY with Kubernetes	Managed Kubernetes on Azure
Containerization	Customer	Microsoft / Customer
Application iteration, debugging	Customer	Microsoft / Customer
CI/CD	Customer	Microsoft / Customer
Provisioning, upgrades, patches	Customer	Microsoft
Reliability availability	Customer	Microsoft
Scaling	Customer	Microsoft
Monitoring and logging	Customer	Microsoft

Customer Microsoft



Benefits of AKS

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



Create/Configure AKS Cluster

1. Using Azure Command Line Interface (CLI)

```
az aks create --resource-group myResourceGroup --name myAKSCluster --node-count 1 --generate-ssh-keys
```

2. Using Azure PowerShell

```
New-AzAksCluster -ResourceGroupName myResourceGroup -Name myAKSCluster -NodeCount 1
```

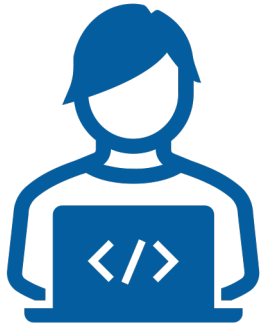
3. Using Azure Portal
4. Using ARM Templates and Bicep
5. Using Azure REST API

```
PUT
https://management.azure.com/subscriptions/{subscriptionId}/resourceGroups/{resourceGroupName}/
providers/Microsoft.ContainerService/managedClusters/{resourceName}?api-version=2021-05-01
```

Manage Azure Kubernetes Service

Task	The Old Way	With Azure
Create a cluster	Provision network and VMs Install dozens of system components including etcd Create and install certificates Register agent nodes with control plane	`az aks create`
Upgrade a cluster	Upgrade your master nodes Cordon/drain and upgrade Agent nodes individually	`az aks upgrade`
Scale a cluster	Provision new VMs Install system components Register nodes with API server	`az aks scale`

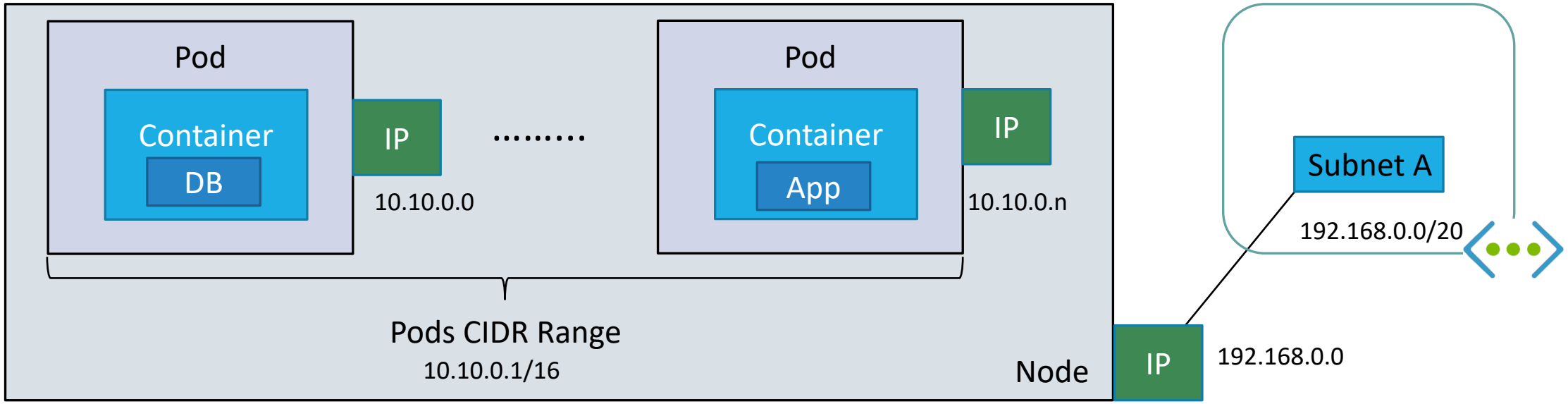

DEMO

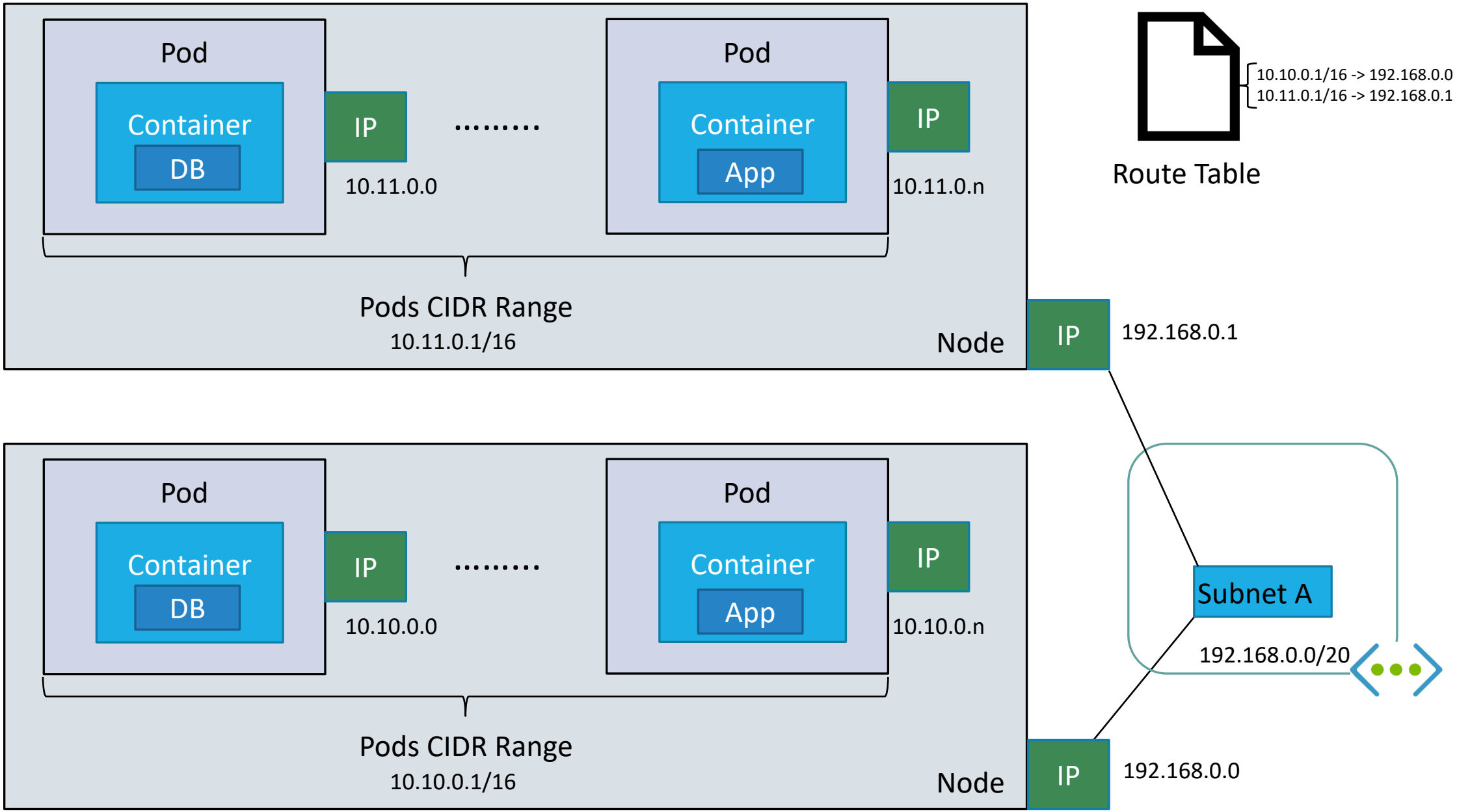


AKS Networking

Two different options -

1. ***Kubenet* networking** - The network resources are typically created and configured as the AKS cluster is deployed.

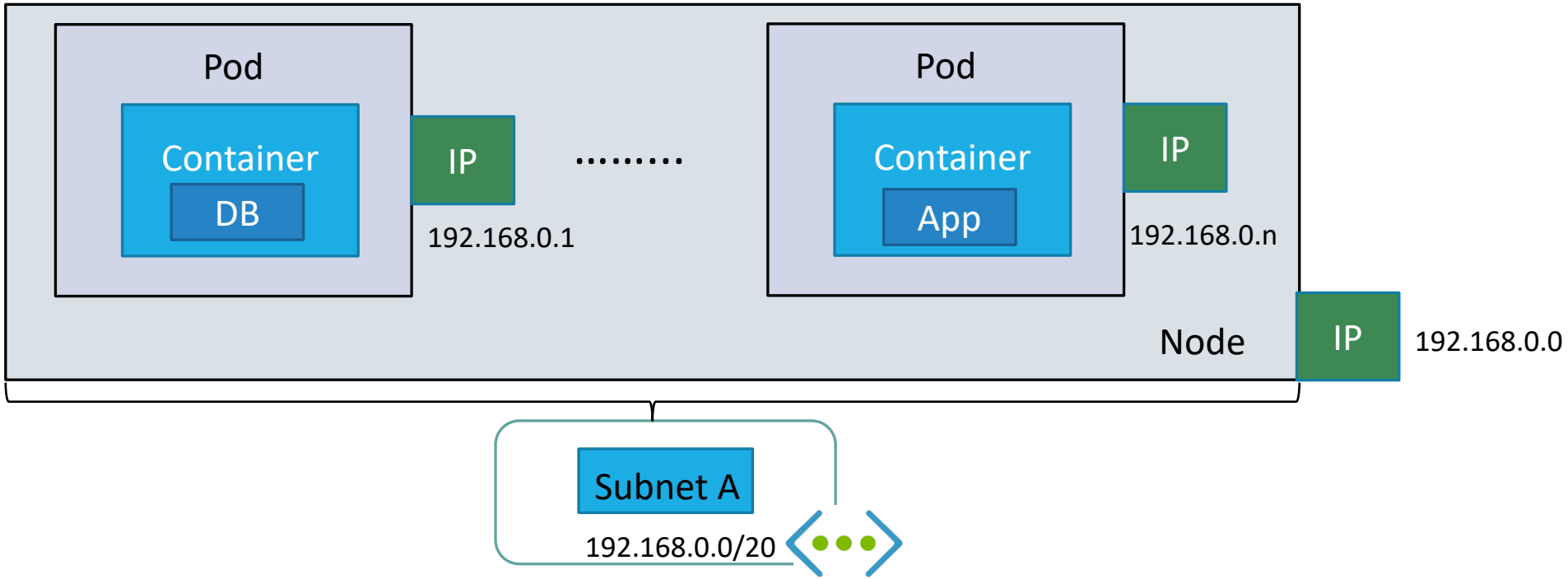



Pod
Container
DB
IP
10.11.0.0
.....
Pod
Container
App
IP
10.11.0.n
Pods CIDR Range
10.11.0.1/16
Node
IP
192.168.0.1
10.10.0.1/16 -> 192.168.0.0
10.11.0.1/16 -> 192.168.0.1
Route Table
Pod
Container
DB
IP
10.10.0.0
.....
Pod
Container
App
IP
10.10.0.n
Pods CIDR Range
10.10.0.1/16
Node
IP
192.168.0.0
Subnet A
192.168.0.0/20

AKS Networking

Two different options -

2. ***Azure Container Networking Interface (CNI)* networking** - The AKS cluster is connected to existing virtual network resources and configurations.



AKS Security

Enforce compliance rules with Azure Policy

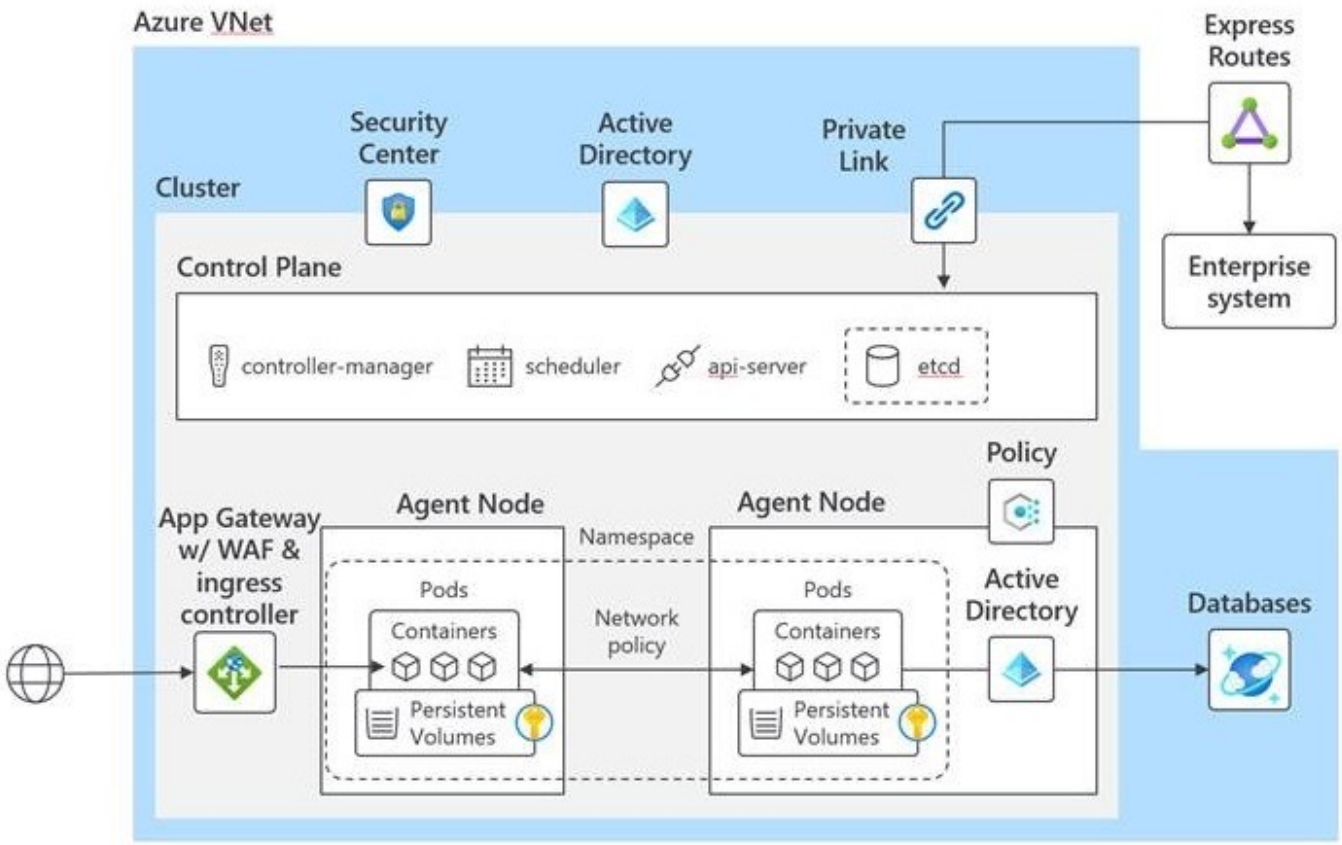
Identity and access control using Azure Active Directory

Encrypt using your own keys, stored in Azure Key Vault

Gain unmatched security management with Azure Security Center integration

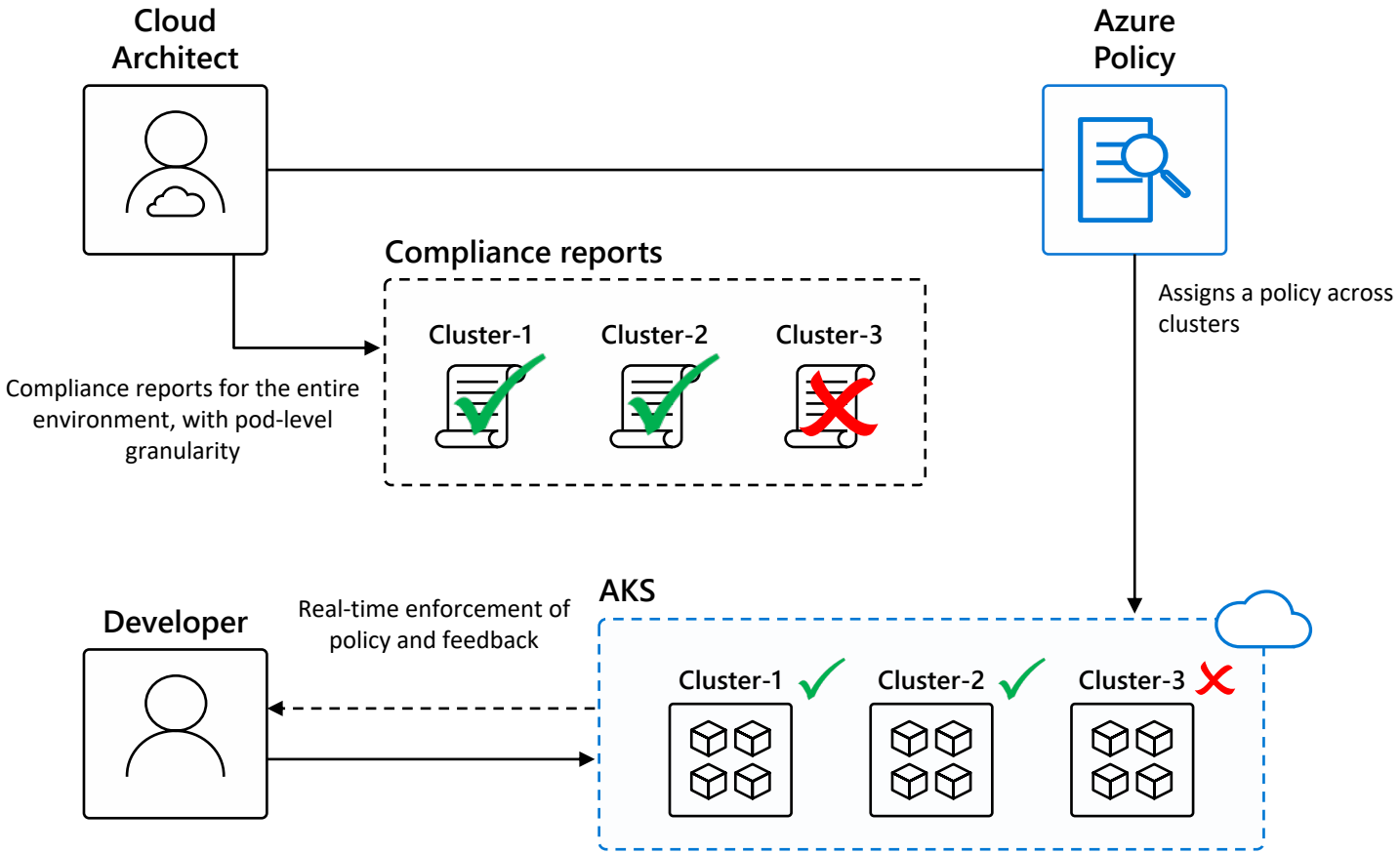
Interact securely with Kubernetes API server using Azure Private Link

Use application gateway (and WAF) with Ingress Controller

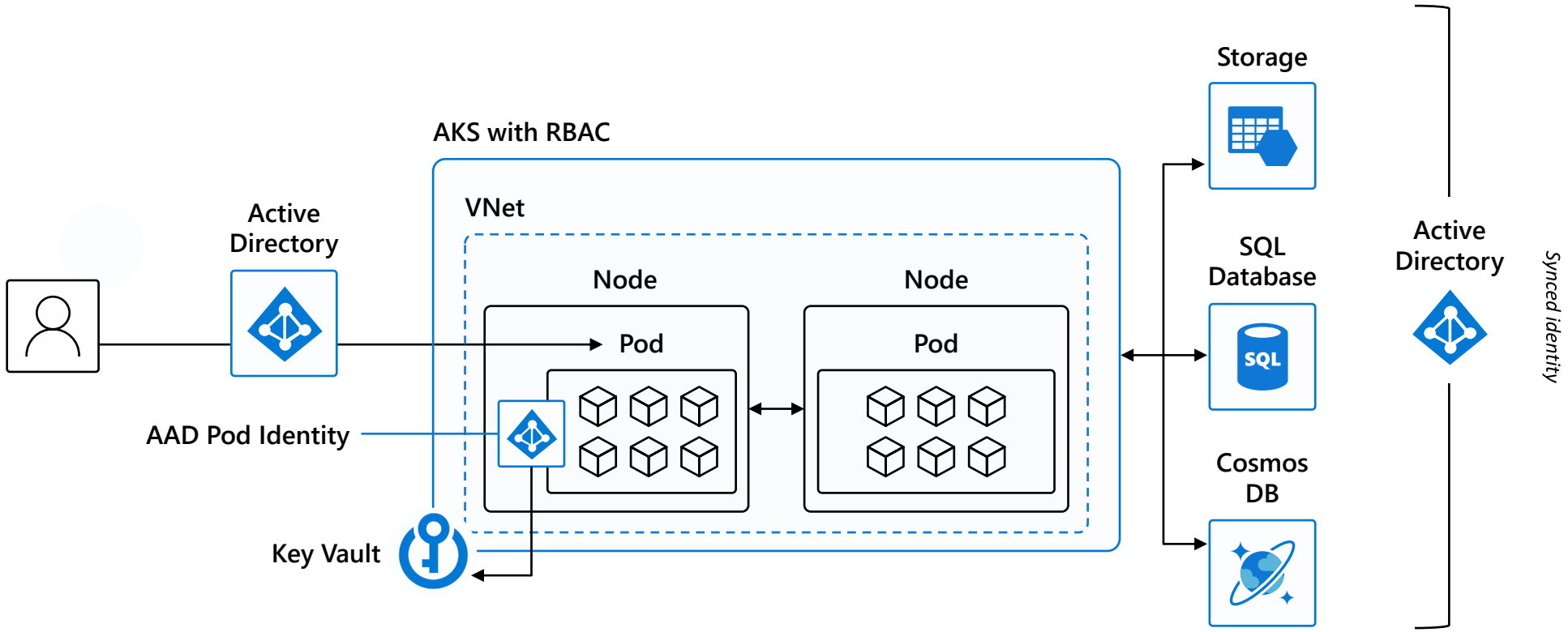


AKS Governance with Azure Policies

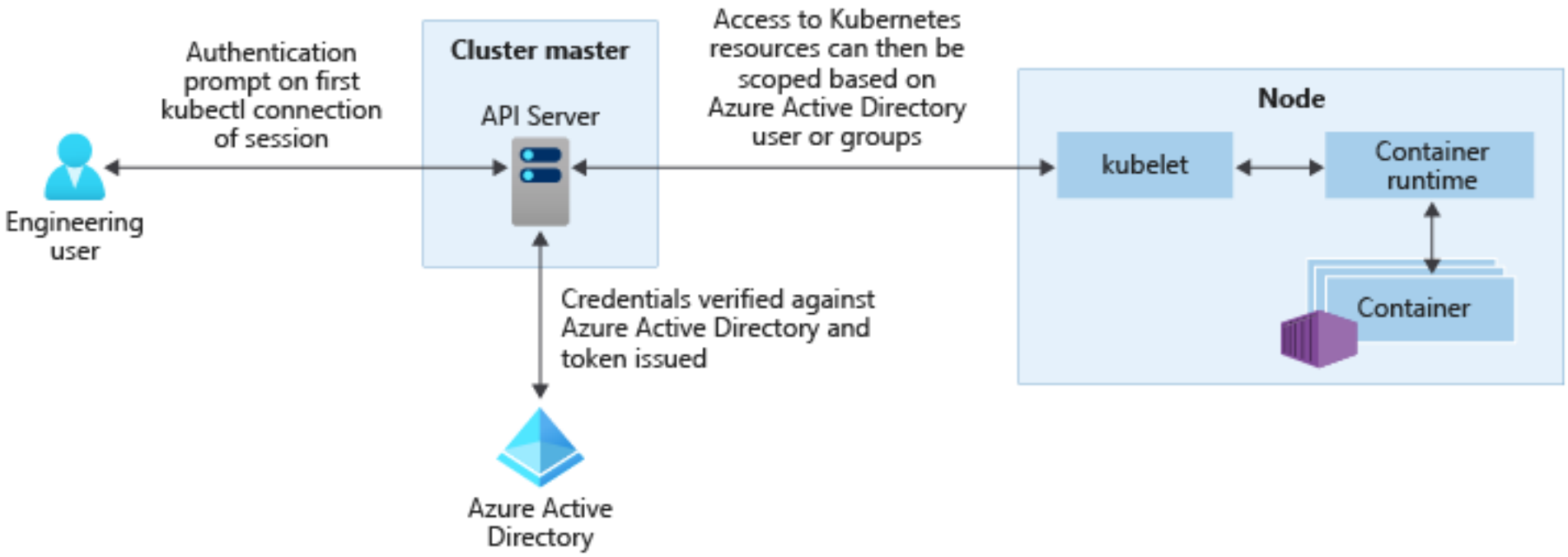
1. Cloud architect assigns a deployment policy across cluster(s)
2. Developer uses standard Kubernetes API to deploy to the cluster
3. Real-time deployment enforcement (acceptance/denial) provided to developer based on policy
4. Cloud architect obtains compliance report for the entire environment and can drill down to individual pod level



AKS Identity and Management



AKS Identity and Management

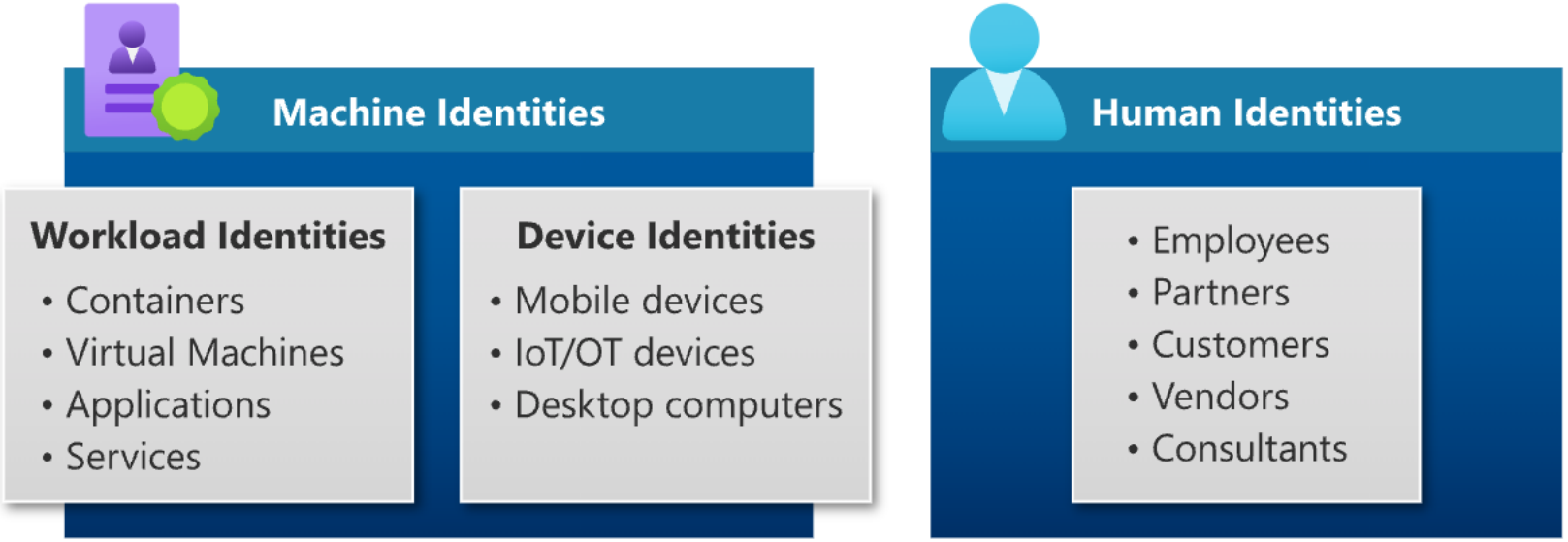


https://docs.microsoft.com/en-us/azure/aks/concepts-identity#azure-ad-integration

Azure AD Workload Identities

Used by a software workload (such as an application, service, script, or container) to authenticate and access other services and resources.

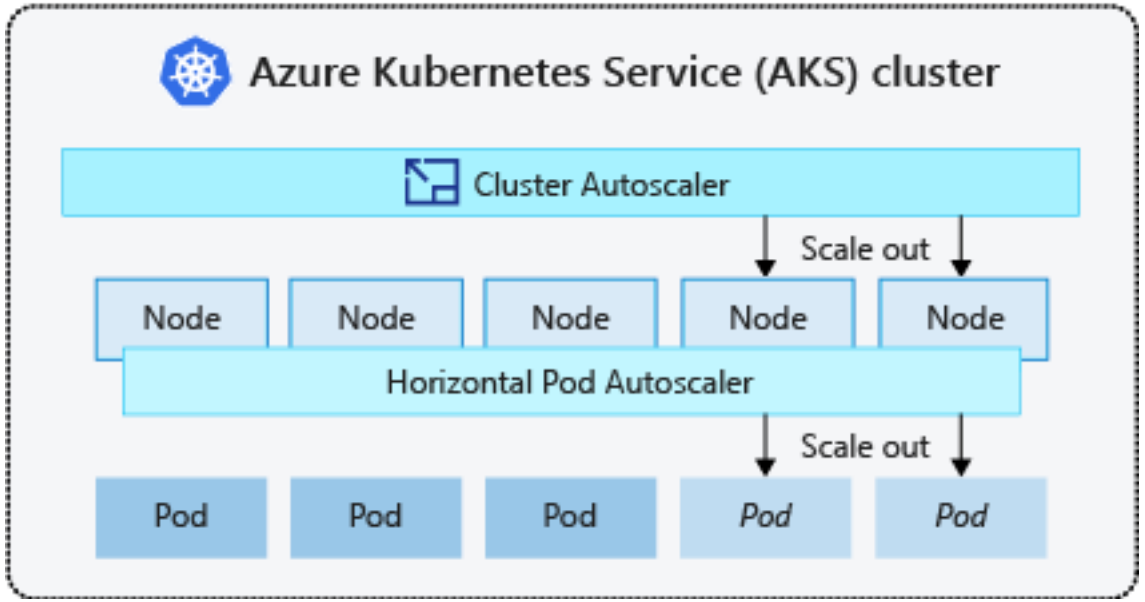
In Azure Active Directory (Azure AD), workload identities are applications, service principals, and managed identities.



AKS Auto-scaling

The **cluster autoscaler** watches for pods that can't be scheduled on nodes because of resource constraints. The cluster then automatically increases the number of nodes.

The **horizontal pod autoscaler** uses the Metrics Server in a Kubernetes cluster to monitor the resource demand of pods. If an application needs more resources, the number of pods is automatically increased to meet the demand.



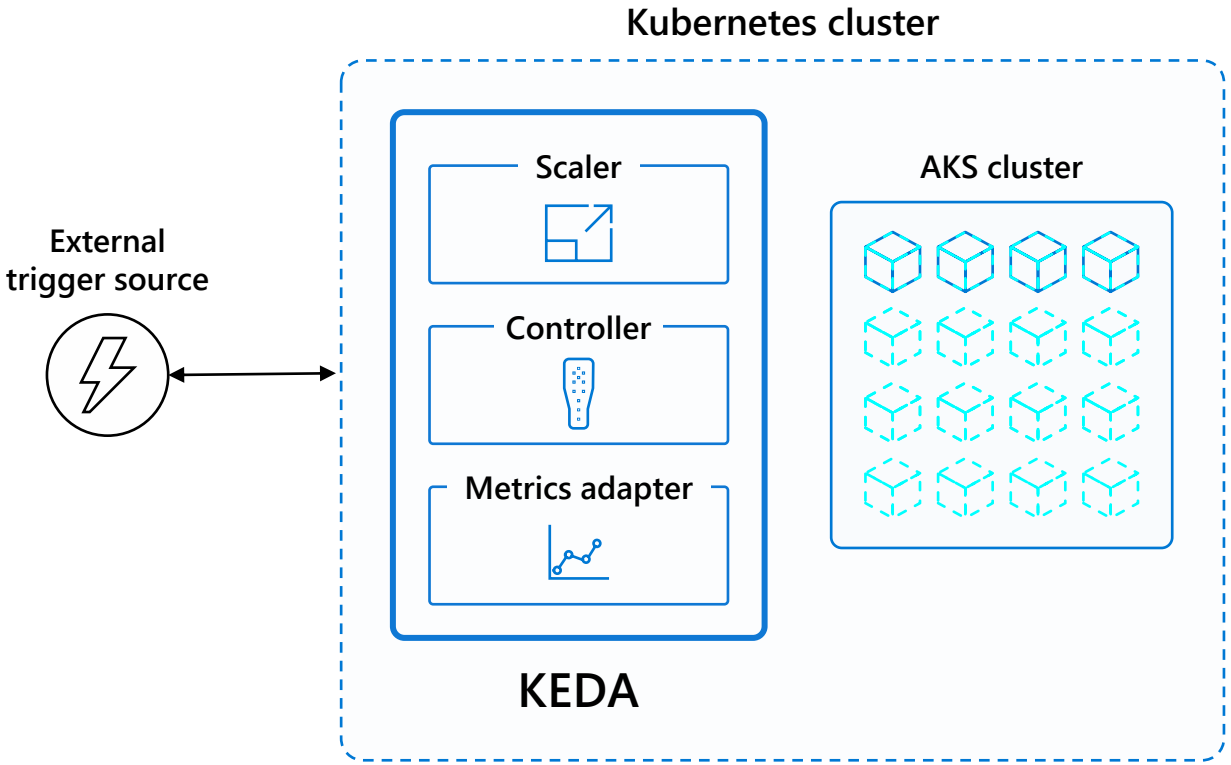
Kubernetes-based event-driven auto-scaling (KEDA)

Open-source component jointly built by Microsoft and RedHat

Event-driven container creation & scaling
Allows containers to “scale to zero” until an event comes in, which will then create the container and process the event, resulting in more efficient utilization and reduced costs

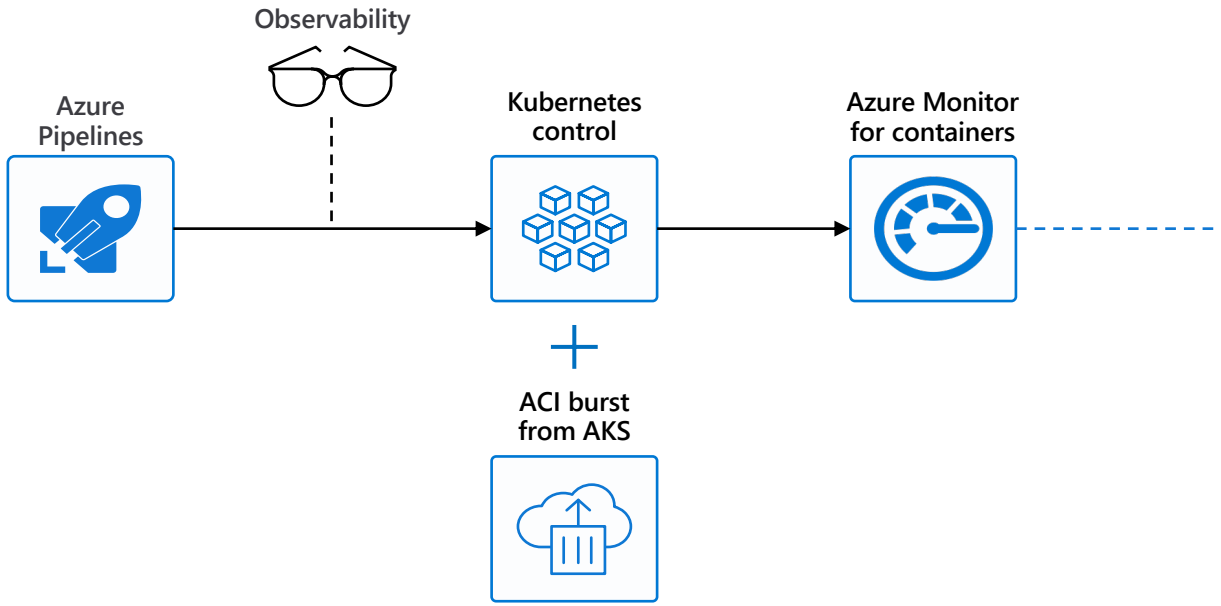
Native triggers support
Containers can consume events directly from the event source, instead of routing events through HTTP

Can be used in any Kubernetes service
This includes in the cloud (e.g., AKS, EKS, GKE, etc.) or on-premises with OpenShift—any Kubernetes workload that requires scaling by events instead of traditional CPU or memory scaling can leverage this component.



https://docs.microsoft.com/en-us/azure/azure-functions/functions-kubernetes-keda

Azure Monitor for Containers



Visualization
Visualize overall health and performance from clusters to containers with drill downs and filters

Insights
Provide insights with multi-cluster health roll up view

Monitor & Analyze
Monitor and analyze Kubernetes and container deployment performance, events, health, and logs

Response
Native alerting with integration to issue managements and ITSM tools

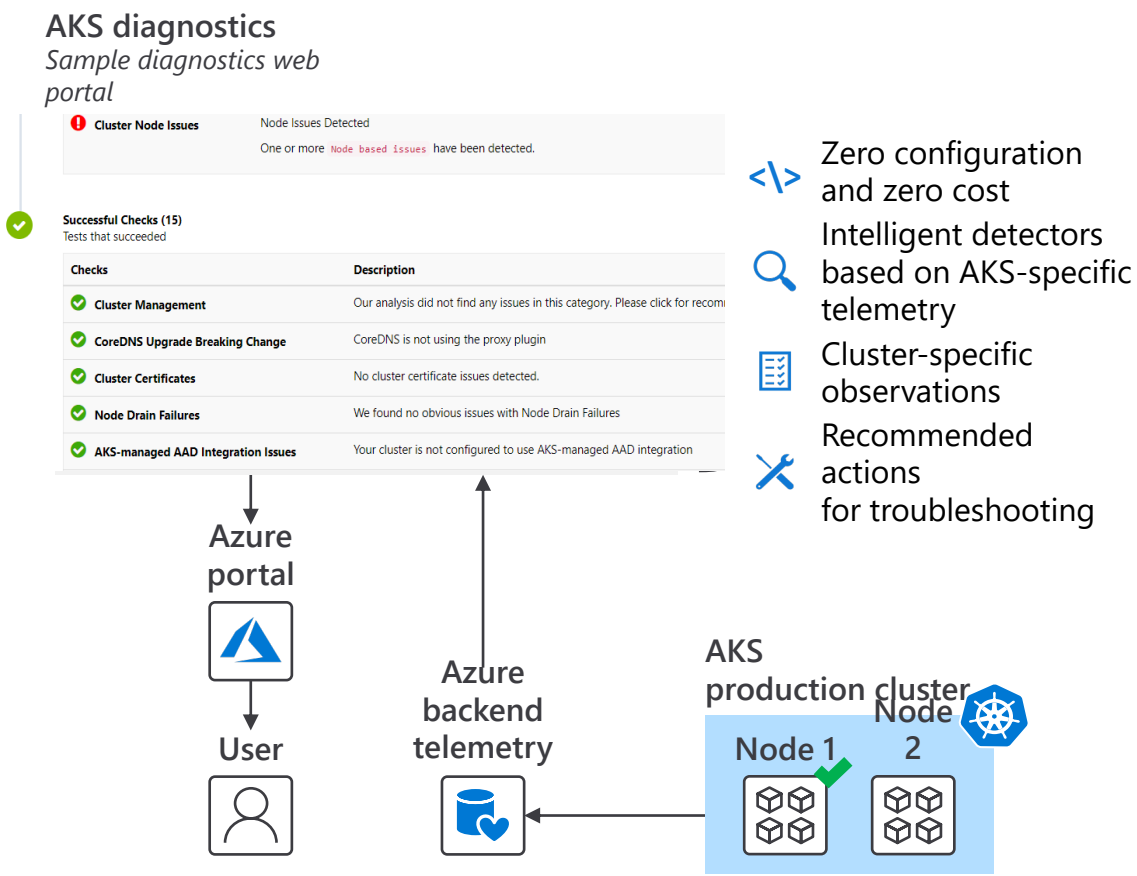
Observability
Observe live container logs on container deployment status

AKS Diagnostics

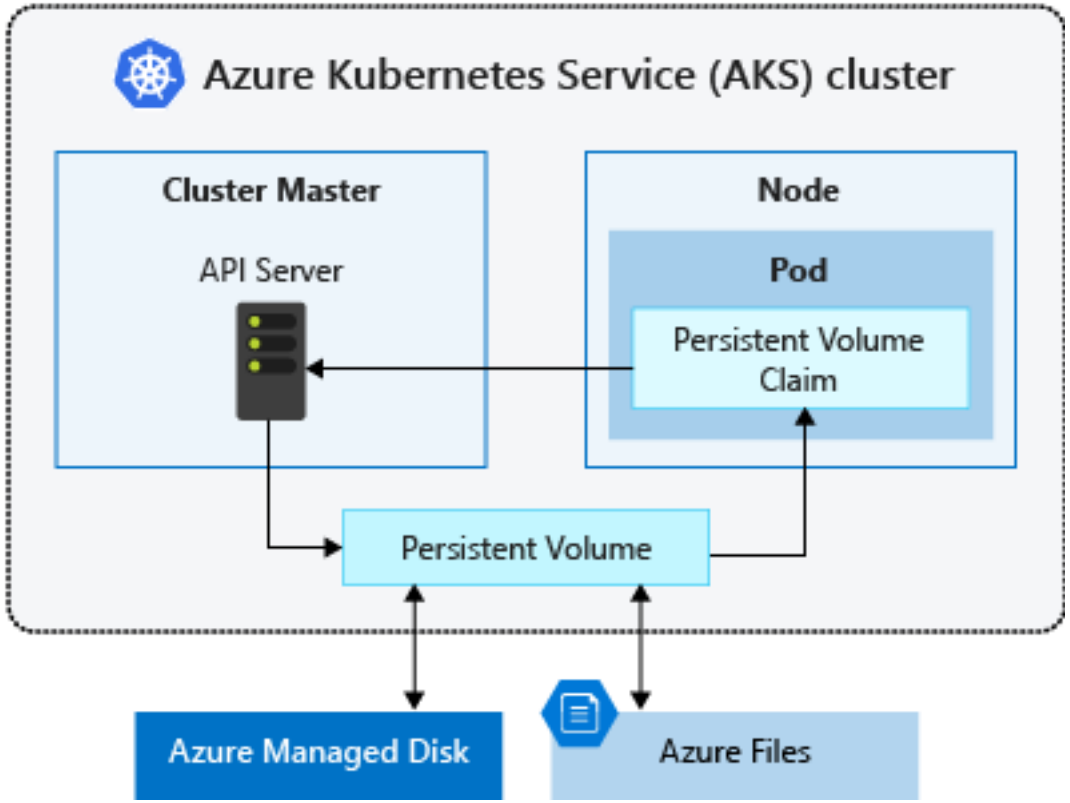
Faster resolution of common issues with an intelligent, self-diagnostic experience right in the portal

Cluster-specific observations

Recommended actions for troubleshooting



AKS Storage Options



Azure Container registry

First-class Azure resource

Managed, private Docker registry service based on the open-source Docker Registry 2.0.

Can be used with existing container development and deployment pipelines

Use Azure Container Registry Tasks to build container images in Azure

Three pricing tiers:

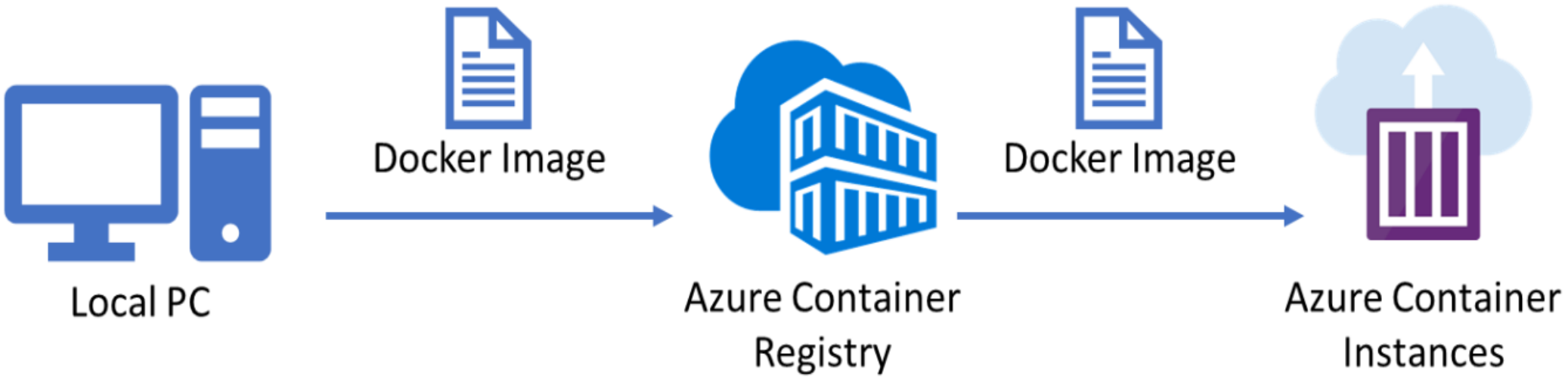
1. Basic
2. Standard
3. Premium

https://azure.microsoft.com/en-us/pricing/details/container-registry/
https://docs.microsoft.com/en-us/azure/container-registry/container-registry-skus

Azure container instances

Fastest and simplest way to run a container in Azure

https://docs.microsoft.com/en-us/azure/container-instances/

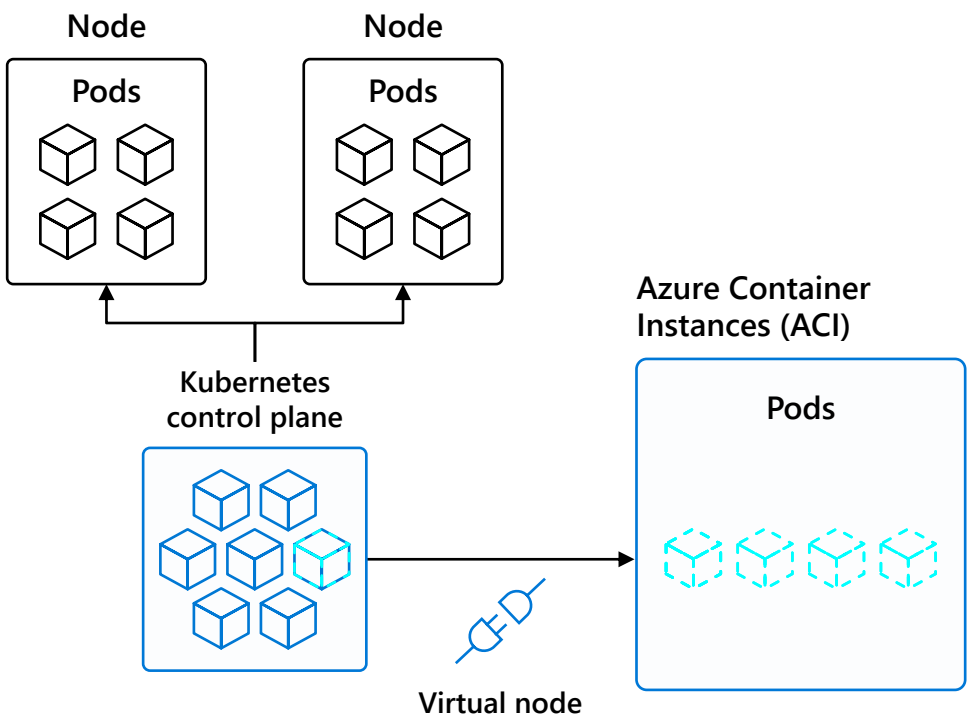


Serverless Kubernetes with Virtual Nodes

Elastically provision compute capacity in seconds

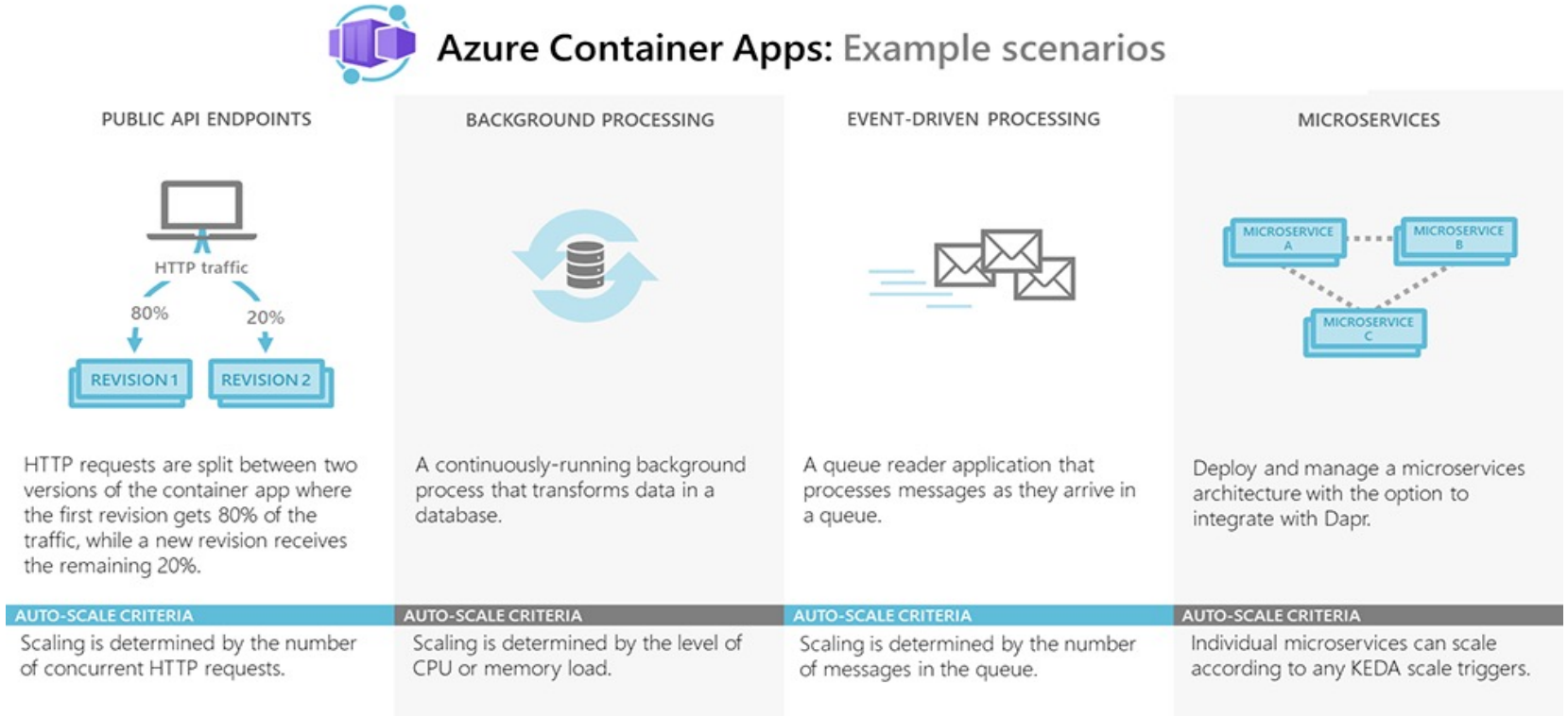
No infrastructure to manage

Built on open sourced Virtual Kubelet technology, donated to the Cloud Native Computing Foundation (CNCF)

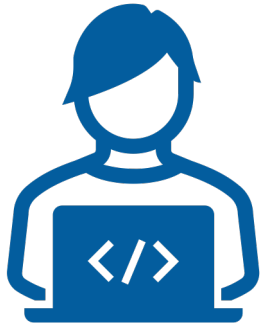


Azure Container Apps

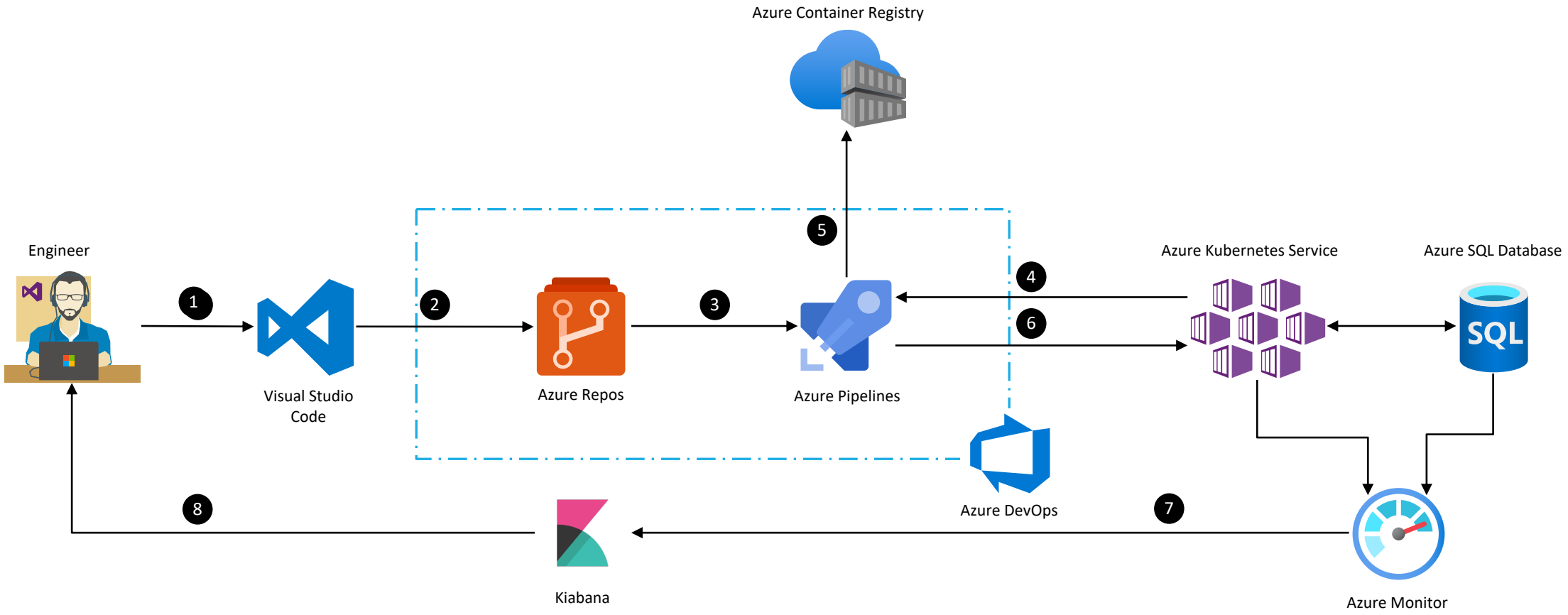
Azure Container Apps enables you to run microservices and containerized applications on a serverless platform.



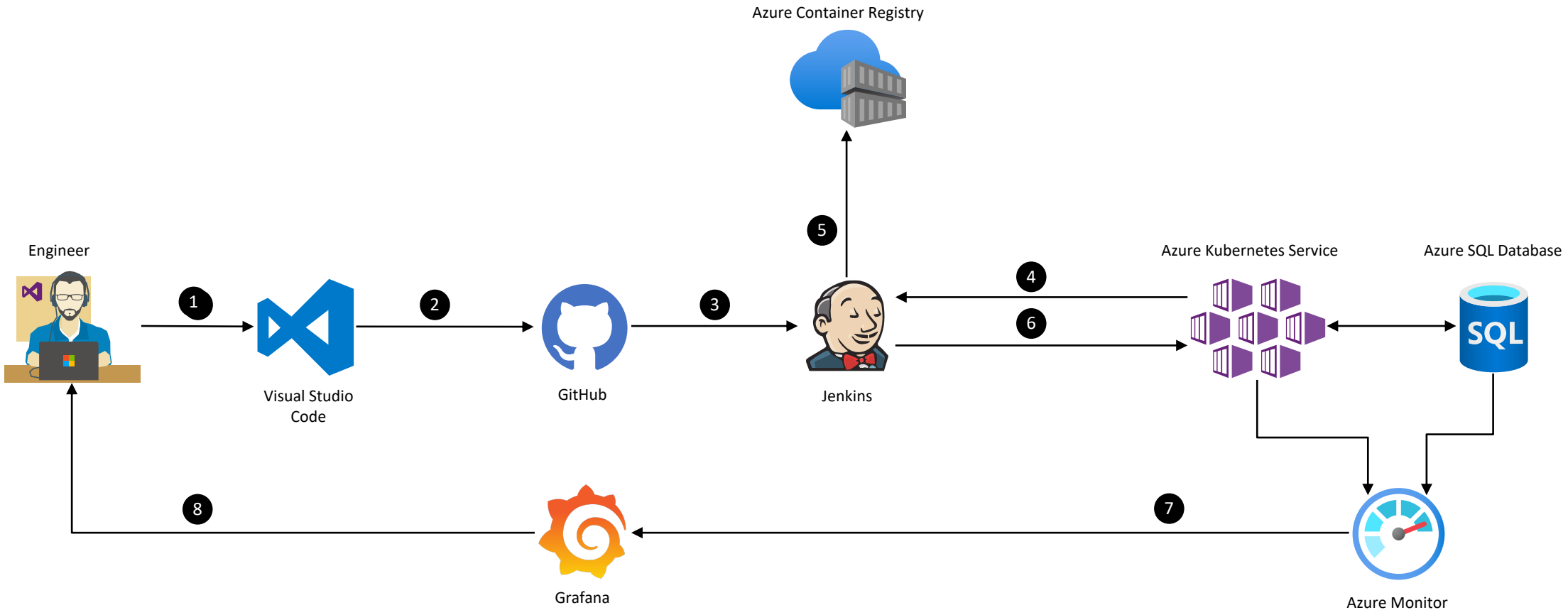

DEMO



CI/CD using Azure DevOps

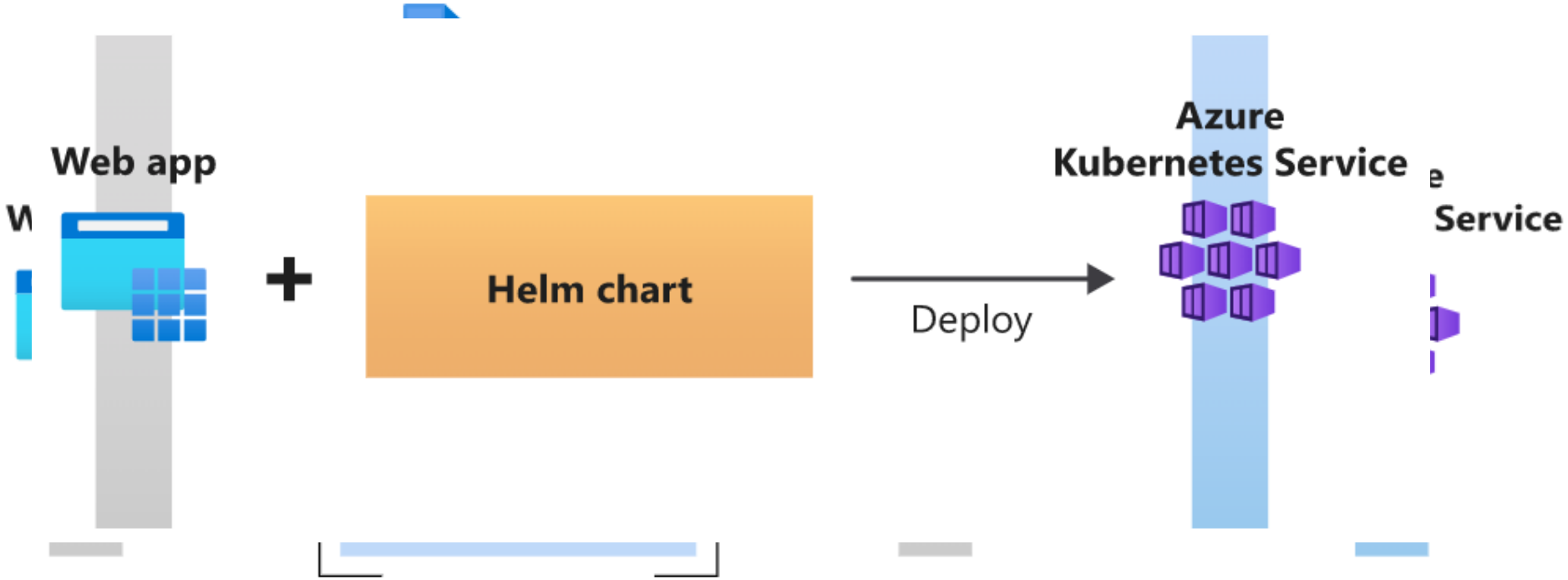


CI/CD using GitHub and Jenkins



Helm Package Manager

Helm is a package manager for Kubernetes that combines all your application's resources and deployment information into a single deployment package.



AKS – Best Practices

Networking configuration

- Network topology
- Plan the IP addresses
- Deploy Ingress resources

Cluster compute

- Compute for the base cluster
- Container image reference
- Policy management

Identity management

- Integrate Azure AD for the cluster
- Integrate Azure AD for the workload

Secure data flow

- Secure the network flow
- Add secret management

Business continuity

- Scalability
- Cluster and node availability
- Availability and multi-region support

Operations

- Cluster and workload CI/CD pipelines
- Cluster health and metrics
- Cost management and reporting

https://docs.microsoft.com/en-us/azure/aks/best-practices

Further Reading

Kubernetes on Azure - https://aka.ms/K8sonAzure

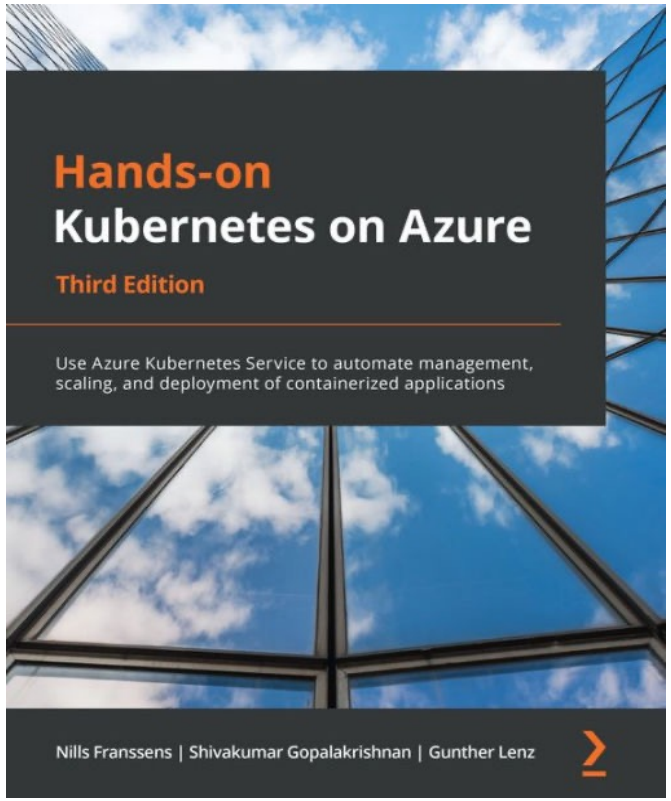
Microsoft Learn - https://aka.ms/LearnKubernetes

What is Kubernetes - https://aka.ms/k8sLearning

AKS Case Studies - https://aka.ms/AKS/casestudy

AKS Roadmap - https://aka.ms/k8sroadmap

Getting started for free - https://aka.ms/AKS/trial




Q&A

Contact Information

https://vaibhavgujral.com

@vaibhavgujral_

https://www.linkedin.com/in/vaibhavgujral/

https://www.youtube.com/c/VaibhavGujral

vaibhav@vaibhavgujral.com

LinkedIn

Twitter

Email

Slides


Thank
You!