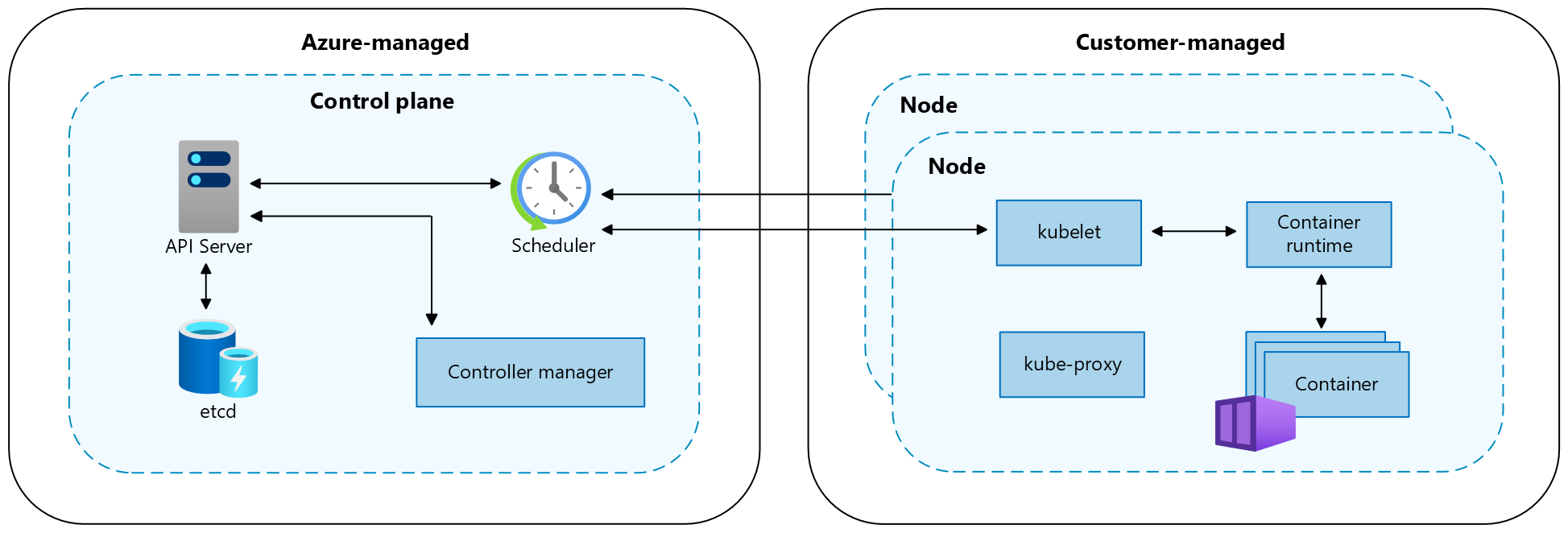
**What is AKS?**

AKS is a managed Kubernetes service that simplifies deploying, managing, and scaling containerized applications using Kubernetes. For more information, see [What is Azure Kubernetes Service (AKS)?](https://learn.microsoft.com/en-us/azure/aks/what-is-aks)



**Control plane**

The Azure managed control plane is composed of several components that help manage the cluster:

A screenshot of a computer

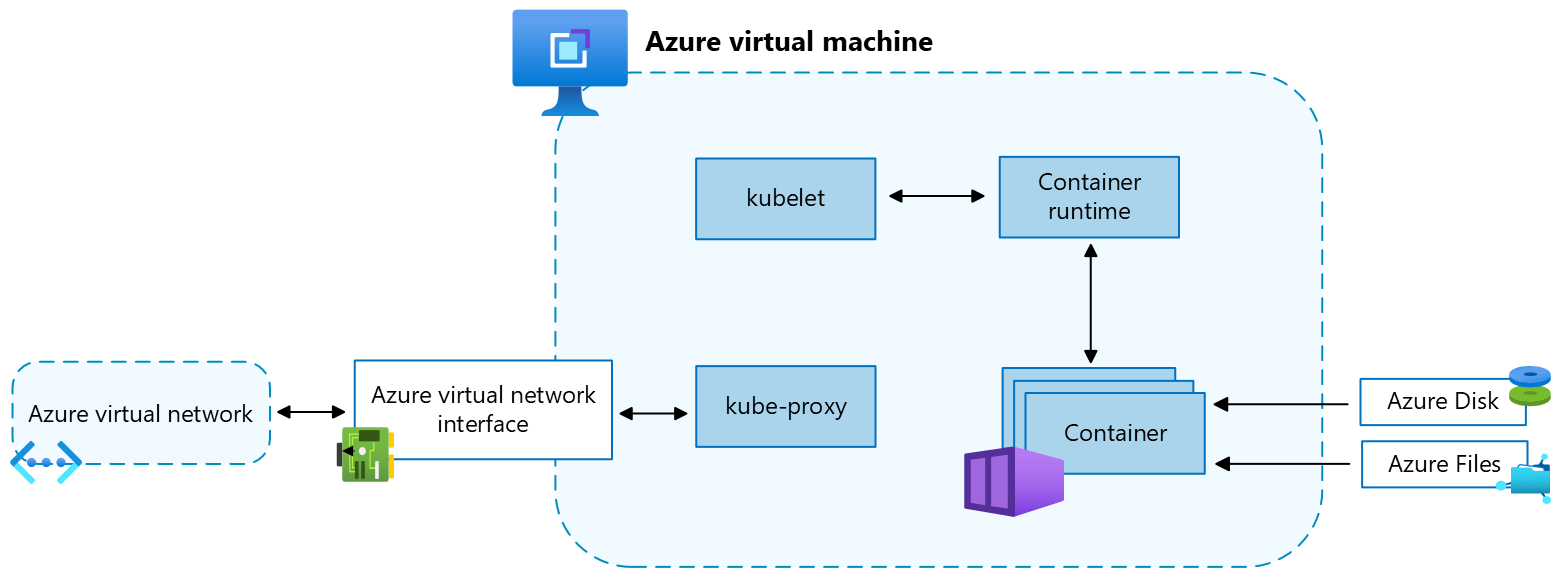
AI-generated content may be incorrect.

**Nodes**

Each AKS cluster has at least one node, which is an Azure virtual machine (VM) that runs Kubernetes node components. The following components run on each node:

A screenshot of a computer

AI-generated content may be incorrect.



**Pods**

A *pod* is a group of one or more containers that share the same network and storage resources and a specification for how to run the containers. Pods typically have a 1:1 mapping with a container, but you can run multiple containers in a pod.

**Namespaces**

Kubernetes resources, such as pods and deployments, are logically grouped into *namespaces* to divide an AKS cluster and create, view, or manage access to resources.

The following namespaces are created by default in an AKS cluster:

A screenshot of a computer

AI-generated content may be incorrect.

**Prepare an application for AKS**

1. git clone https://github.com/Azure-Samples/aks-store-demo.git
2. cd aks-store-demo
3. Create docker-compose file

services:

rabbitmq:

image: rabbitmq:3.13.2-management-alpine

container\_name: 'rabbitmq'

restart: always

environment:

- "RABBITMQ\_DEFAULT\_USER=username"

- "RABBITMQ\_DEFAULT\_PASS=password"

ports:

- 15672:15672

- 5672:5672

healthcheck:

test: ["CMD", "rabbitmqctl", "status"]

interval: 30s

timeout: 10s

retries: 5

volumes:

- ./rabbitmq\_enabled\_plugins:/etc/rabbitmq/enabled\_plugins

networks:

- backend\_services

order-service:

build: src/order-service

container\_name: 'order-service'

restart: always

ports:

- 3000:3000

healthcheck:

test: ["CMD", "wget", "-O", "/dev/null", "-q", "http://order-service:3000/health"]

interval: 30s

timeout: 10s

retries: 5

environment:

- ORDER\_QUEUE\_HOSTNAME=rabbitmq

- ORDER\_QUEUE\_PORT=5672

- ORDER\_QUEUE\_USERNAME=username

- ORDER\_QUEUE\_PASSWORD=password

- ORDER\_QUEUE\_NAME=orders

- ORDER\_QUEUE\_RECONNECT\_LIMIT=3

networks:

- backend\_services

depends\_on:

rabbitmq:

condition: service\_healthy

product-service:

build: src/product-service

container\_name: 'product-service'

restart: always

ports:

- 3002:3002

healthcheck:

test: ["CMD", "wget", "-O", "/dev/null", "-q", "http://product-service:3002/health"]

interval: 30s

timeout: 10s

retries: 5

environment:

- AI\_SERVICE\_URL=http://ai-service:5001/

networks:

- backend\_services

store-front:

build: src/store-front

container\_name: 'store-front'

restart: always

ports:

- 8080:8080

healthcheck:

test: ["CMD", "wget", "-O", "/dev/null", "-q", "http://store-front:80/health"]

interval: 30s

timeout: 10s

retries: 5

environment:

- VUE\_APP\_PRODUCT\_SERVICE\_URL=http://product-service:3002/

- VUE\_APP\_ORDER\_SERVICE\_URL=http://order-service:3000/

networks:

- backend\_services

depends\_on:

- product-service

- order-service

networks:

backend\_services:

driver: bridge

1. docker compose -f docker-compose-quickstart.yml up -d
2. docker images
3. docker ps
4. <http://localhost:8080>
5. docker compose down
6. Create ACR and save the acr name in ACRNAME variable

az group create --name AKS-rg --location eastus

az acr create --resource-group AKS-rg --name $ACRNAME --sku Basic

export ACRNAME= **aksdevopsvp**

1. Build images

az acr build --registry $ACRNAME --image aks-store-demo/product-service:latest ./src/product-service/

az acr build --registry $ACRNAME --image aks-store-demo/order-service:latest ./src/order-service/

az acr build --registry $ACRNAME --image aks-store-demo/store-front:latest ./src/store-front/

1. List images

az acr repository list --name $ACRNAME --output table

**Create AKS**

Install kubectl - az aks install-cli

Create Cluster

az aks create --resource-group AKS-rg --name myAKSCluster --node-count 2 --generate-ssh-keys --attach-acr $ACRNAME

az aks get-credentials --resource-group AKS-rg --name myAKSCluster

kubectl get nodes

Get your login server address using the [az acr list](https://learn.microsoft.com/en-us/cli/azure/acr) command and query for your login server

az acr list --resource-group AKS-rg --query "[].{acrLoginServer:loginServer}" --output table

Make sure you're in the cloned *aks-store-demo* directory, and then open the aks-store-quickstart.yaml manifest file with a text editor

<https://github.com/Azure-Samples/aks-store-demo/blob/main/aks-store-quickstart.yaml>

Update the image property for the containers by replacing *ghcr.io/azure-samples* with your ACR login server name.

containers:

...

- name: order-service

image: <acrName>.azurecr.io/aks-store-demo/order-service:latest

...

- name: product-service

image: <acrName>.azurecr.io/aks-store-demo/product-service:latest

...

- name: store-front

image: <acrName>.azurecr.io/aks-store-demo/store-front:latest

...

kubectl apply -f aks-store-quickstart.yaml

kubectl get pods

**Basic K8s labs**

https://labs.play-with-k8s.com/

ctrl + insert to copy and shift + insert to paste

**DEPLOYMENT**

apiVersion: apps/v1

kind: Deployment

metadata:

name: nginx-deployment

spec:

replicas: 3

selector:

matchLabels:

app: nginx

template:

metadata:

labels:

app: nginx

spec:

containers:

- name: nginx-container

image: nginx:latest

ports:

- containerPort: 80

**POD AND SERVICE**

apiVersion: v1

kind: Pod

metadata:

name: nginx

labels:

app.kubernetes.io/name: proxy

spec:

containers:

- name: nginx

image: nginx:stable

ports:

- containerPort: 80

name: http-web-svc

---

apiVersion: v1

kind: Service

metadata:

name: nginx-service

spec:

selector:

app.kubernetes.io/name: proxy

ports:

- name: name-of-service-port

protocol: TCP

port: 80

targetPort: http-web-svc