Azure Kubernetes Services (AKS) Workshop

Workshop Overview

This workshop provides hands-on experience with Azure Kubernetes Service (AKS), covering cluster creation, application deployment, scaling, and management.

Duration: 4-6 hours
Level: Intermediate
Prerequisites: Basic knowledge of containers, Docker, and Azure

Learning Objectives

By the end of this workshop, you will be able to:

- Create and configure an AKS cluster
- Deploy applications to AKS
- Manage pods, services, and deployments
- Implement scaling and updates
- Monitor and troubleshoot AKS clusters
- Secure AKS workloads

Prerequisites Setup

Required Tools

```bash
# Install Azure CLI
curl -sL https://aka.ms/InstallAzureCLIDeb | sudo bash

# Install kubectl
curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl

# Install Helm
curl https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3 | bash
```

Azure Login

```bash
az login
az account set --subscription "your-subscription-id"
```

Lab 1: Creating an AKS Cluster

1.1 Create Resource Group

bash

```bash
# Set variables
RESOURCE_GROUP="aks-workshop-rg"
LOCATION="eastus"
CLUSTER_NAME="aks-workshop-cluster"

# Create resource group
az group create --name $RESOURCE_GROUP --location $LOCATION
```

1.2 Create AKS Cluster

bash

```bash
# Create AKS cluster with basic configuration
az aks create \
  --resource-group $RESOURCE_GROUP \
  --name $CLUSTER_NAME \
  --node-count 2 \
  --node-vm-size Standard_B2s \
  --generate-ssh-keys \
  --enable-managed-identity \
  --enable-addons monitoring
```

1.3 Connect to Cluster

bash

```bash
# Get cluster credentials
az aks get-credentials --resource-group $RESOURCE_GROUP --name $CLUSTER_NAME

# Verify connection
kubectl get nodes
kubectl cluster-info
```

Lab 2: Deploying Your First Application

2.1 Deploy a Sample Application

Create a deployment file:

yaml

```yaml
# nginx-deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.21
        ports:
        - containerPort: 80
---
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  selector:
    app: nginx
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
  type: LoadBalancer
```

Deploy the application:

```bash
kubectl apply -f nginx-deployment.yaml
kubectl get deployments
kubectl get services
kubectl get pods
```

2.2 Access the Application

```bash
# Get external IP (may take a few minutes)
kubectl get service nginx-service --watch

# Once external IP is assigned, test the application
curl http://EXTERNAL-IP
```

Lab 3: Working with Kubernetes Resources

3.1 Scaling Applications

```bash
# Scale deployment
kubectl scale deployment nginx-deployment --replicas=5
kubectl get pods

# Auto-scaling with HPA
kubectl autoscale deployment nginx-deployment --cpu-percent=50 --min=3 --max=10
kubectl get hpa
```

3.2 Rolling Updates

```bash
# Update nginx image
kubectl set image deployment/nginx-deployment nginx=nginx:1.22
kubectl rollout status deployment/nginx-deployment

# Check rollout history
kubectl rollout history deployment/nginx-deployment

# Rollback if needed
kubectl rollout undo deployment/nginx-deployment
```

3.3 ConfigMaps and Secrets

```bash
```

```
# Create ConfigMap
kubectl create configmap app-config --from-literal=DATABASE_URL=mysql://localhost:3306/myapp

# Create Secret
kubectl create secret generic app-secrets --from-literal=DB_PASSWORD=supersecret

# View resources
kubectl get configmaps
kubectl get secrets
```

Lab 4: Advanced AKS Features

4.1 Enable Azure Container Insights

```bash
# Enable Container Insights
az aks enable-addons \
  --resource-group $RESOURCE_GROUP \
  --name $CLUSTER_NAME \
  --addons monitoring
```

4.2 Configure Ingress Controller

```bash
# Install NGINX Ingress Controller using Helm
helm repo add ingress-nginx https://kubernetes.github.io/ingress-nginx
helm repo update

helm install ingress-nginx ingress-nginx/ingress-nginx \
  --create-namespace \
  --namespace ingress-basic \
  --set controller.service.annotations."service\.beta\.kubernetes\.io/azure-load-balancer-health-probe-request-path"=/h
```

4.3 Deploy Application with Ingress

```yaml
```

```yaml
# app-with-ingress.yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: nginx-ingress
  annotations:
    kubernetes.io/ingress.class: nginx
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  rules:
  - host: myapp.example.com
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: nginx-service
            port:
              number: 80
```

Lab 5: Security and Networking

5.1 Network Policies

yaml

```yaml
# network-policy.yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: deny-all
spec:
  podSelector: {}
  policyTypes:
  - Ingress
  - Egress
```

5.2 Pod Security Standards

yaml

```yaml
# pod-security-policy.yaml
apiVersion: v1
kind: Pod
metadata:
  name: secure-pod
spec:
  securityContext:
    runAsNonRoot: true
    runAsUser: 1000
    fsGroup: 2000
  containers:
  - name: app
    image: nginx:1.21
    securityContext:
      allowPrivilegeEscalation: false
      readOnlyRootFilesystem: true
      capabilities:
        drop:
        - ALL
```

Lab 6: Monitoring and Logging

6.1 View Cluster Metrics

```bash
# Check node resource usage
kubectl top nodes

# Check pod resource usage
kubectl top pods

# View logs
kubectl logs -l app=nginx
kubectl logs deployment/nginx-deployment
```

6.2 Set Up Prometheus and Grafana

```bash
```

```
# Add Prometheus Helm repository
helm repo add prometheus-community https://prometheus-community.github.io/helm-charts
helm repo update

# Install Prometheus and Grafana
helm install monitoring prometheus-community/kube-prometheus-stack \
  --namespace monitoring \
  --create-namespace
```

Lab 7: Cluster Management

7.1 Node Pool Management

```
bash

# Add new node pool
az aks nodepool add \
  --resource-group $RESOURCE_GROUP \
  --cluster-name $CLUSTER_NAME \
  --name nodepool2 \
  --node-count 2 \
  --node-vm-size Standard_B4ms

# List node pools
az aks nodepool list --resource-group $RESOURCE_GROUP --cluster-name $CLUSTER_NAME

# Scale node pool
az aks nodepool scale \
  --resource-group $RESOURCE_GROUP \
  --cluster-name $CLUSTER_NAME \
  --name nodepool2 \
  --node-count 3
```

7.2 Cluster Upgrades

```
bash

# Check available upgrades
az aks get-upgrades --resource-group $RESOURCE_GROUP --name $CLUSTER_NAME

# Upgrade cluster (example)
az aks upgrade \
  --resource-group $RESOURCE_GROUP \
  --name $CLUSTER_NAME \
  --kubernetes-version 1.28.5
```

Lab 8: Storage and Persistent Volumes

8.1 Azure Disk Storage

yaml

```yaml
# azure-disk-pvc.yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: azure-disk-pvc
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
  storageClassName: managed-premium
```

8.2 Deploy Application with Persistent Storage

yaml

```yaml
# app-with-storage.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mysql-deployment
spec:
  selector:
    matchLabels:
      app: mysql
  template:
    metadata:
      labels:
        app: mysql
    spec:
      containers:
      - name: mysql
        image: mysql:8.0
        env:
        - name: MYSQL_ROOT_PASSWORD
          valueFrom:
            secretKeyRef:
              name: mysql-secret
              key: password
        volumeMounts:
        - name: mysql-storage
          mountPath: /var/lib/mysql
      volumes:
      - name: mysql-storage
        persistentVolumeClaim:
          claimName: azure-disk-pvc
```

Lab 9: CI/CD Integration

9.1 Azure DevOps Pipeline Example

yaml

```yaml
# azure-pipelines.yml
trigger:
- main

variables:
  azureServiceConnection: 'your-service-connection'
  resourceGroup: 'aks-workshop-rg'
  kubernetesCluster: 'aks-workshop-cluster'
  imageRepository: 'myapp'
  containerRegistry: 'myregistry.azurecr.io'

stages:
- stage: Build
  jobs:
  - job: Build
    pool:
      vmImage: 'ubuntu-latest'
    steps:
    - task: Docker@2
      inputs:
        containerRegistry: '$(containerRegistry)'
        repository: '$(imageRepository)'
        command: 'buildAndPush'
        Dockerfile: '**/Dockerfile'
        tags: '$(Build.BuildId)'

- stage: Deploy
  jobs:
  - deployment: Deploy
    environment: 'production'
    pool:
      vmImage: 'ubuntu-latest'
    strategy:
      runOnce:
        deploy:
          steps:
          - task: KubernetesManifest@0
            inputs:
              action: 'deploy'
              kubernetesServiceConnection: '$(azureServiceConnection)'
              namespace: 'default'
              manifests: 'k8s/deployment.yaml'
```

Lab 10: Troubleshooting

10.1 Common Debugging Commands

```bash
# Check cluster status
kubectl get all --all-namespaces

# Describe resources for details
kubectl describe pod <pod-name>
kubectl describe service <service-name>
kubectl describe node <node-name>

# Check events
kubectl get events --sort-by=.metadata.creationTimestamp

# Check logs
kubectl logs <pod-name> -c <container-name>
kubectl logs -l app=myapp --previous

# Port forwarding for debugging
kubectl port-forward pod/<pod-name> 8080:80
```

10.2 Resource Issues

```bash
# Check resource usage
kubectl top nodes
kubectl top pods

# Check resource quotas
kubectl get resourcequota
kubectl describe resourcequota

# Check node conditions
kubectl describe nodes | grep -A 5 Conditions
```

Workshop Exercises

Exercise 1: Deploy a Multi-Tier Application

Deploy a complete web application with:

- Frontend (React/Angular)
- Backend API (Node.js/Python)

- Database (PostgreSQL/MongoDB)
- Implement proper service communication

Exercise 2: Implement Blue-Green Deployment

- Set up two identical environments
- Deploy new version to green environment
- Switch traffic using ingress controller
- Implement rollback strategy

Exercise 3: Set Up Monitoring Dashboard

- Configure Prometheus to scrape application metrics
- Create custom Grafana dashboards
- Set up alerting rules
- Test alert notifications

Cleanup

Remove Workshop Resources

```bash
# Delete the resource group (removes everything)
az group delete --name $RESOURCE_GROUP --yes --no-wait

# Or delete individual resources
kubectl delete all --all
az aks delete --resource-group $RESOURCE_GROUP --name $CLUSTER_NAME --yes --no-wait
```

Additional Resources

Documentation

- Azure Kubernetes Service Documentation
- Kubernetes Documentation
- kubectl Cheat Sheet

Best Practices

- Use namespaces for resource organization
- Implement resource quotas and limits
- Regular security scanning and updates

- Backup and disaster recovery planning
- Cost optimization strategies

Next Steps

- Explore GitOps with ArgoCD or Flux
- Implement service mesh (Istio/Linkerd)
- Advanced security with Azure Policy
- Multi-cluster management
- Serverless containers with Azure Container Instances

Workshop Feedback

Please provide feedback on:

- Content difficulty and pacing
- Lab exercise clarity
- Additional topics of interest
- Technical issues encountered

Note: This workshop requires an active Azure subscription. Some services may incur costs. Always clean up resources after the workshop to avoid unnecessary charges.