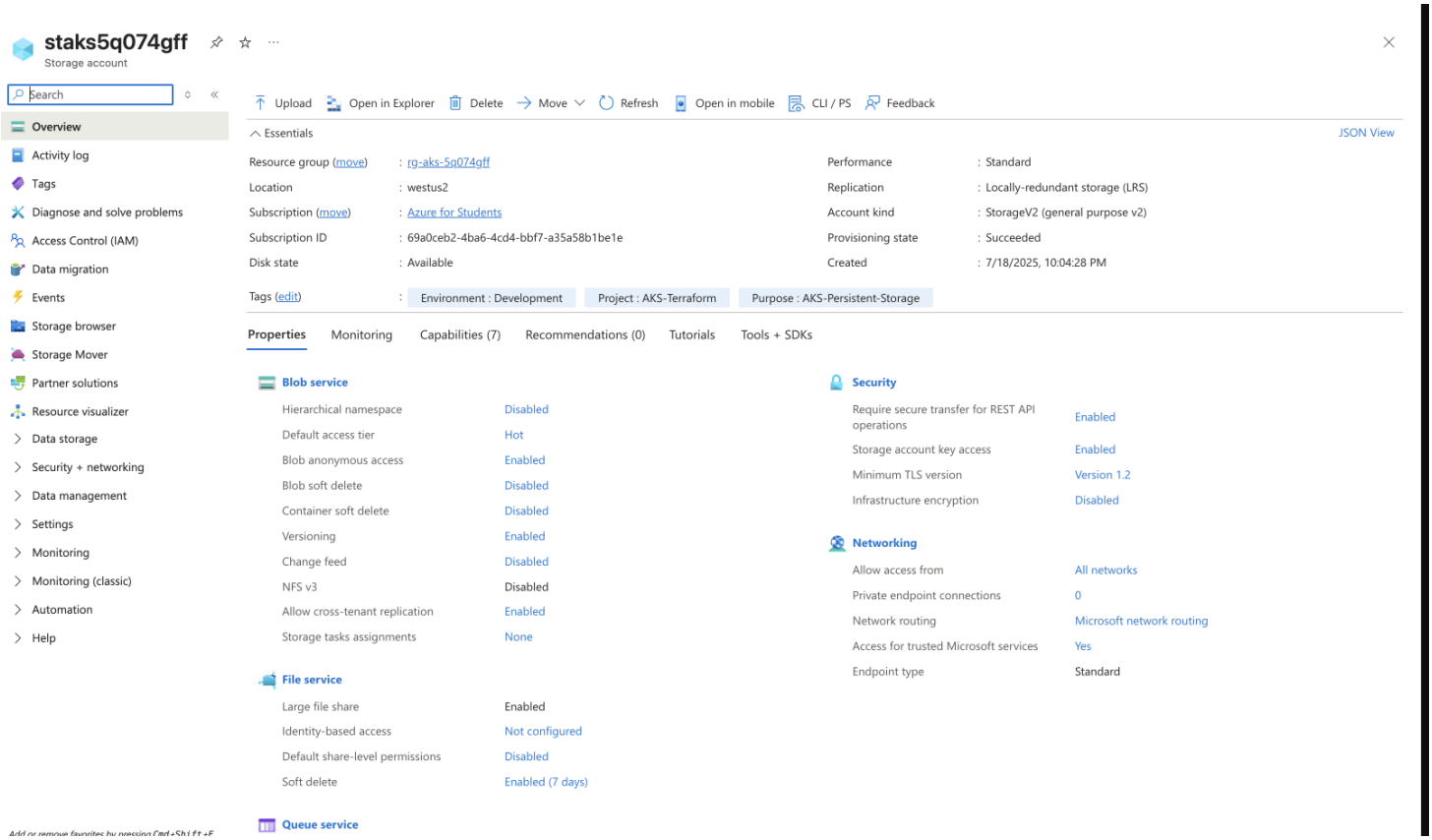
Q3: Azure Blob Storage + AKS Persistent Volume Integration + CI/CD Deployment

Screenshots

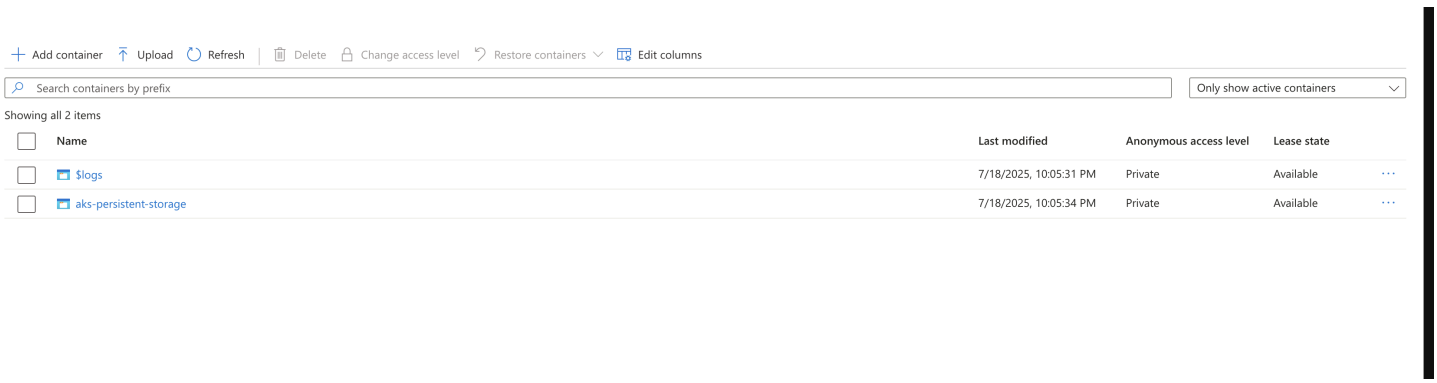
1. Azure Storage Accounts



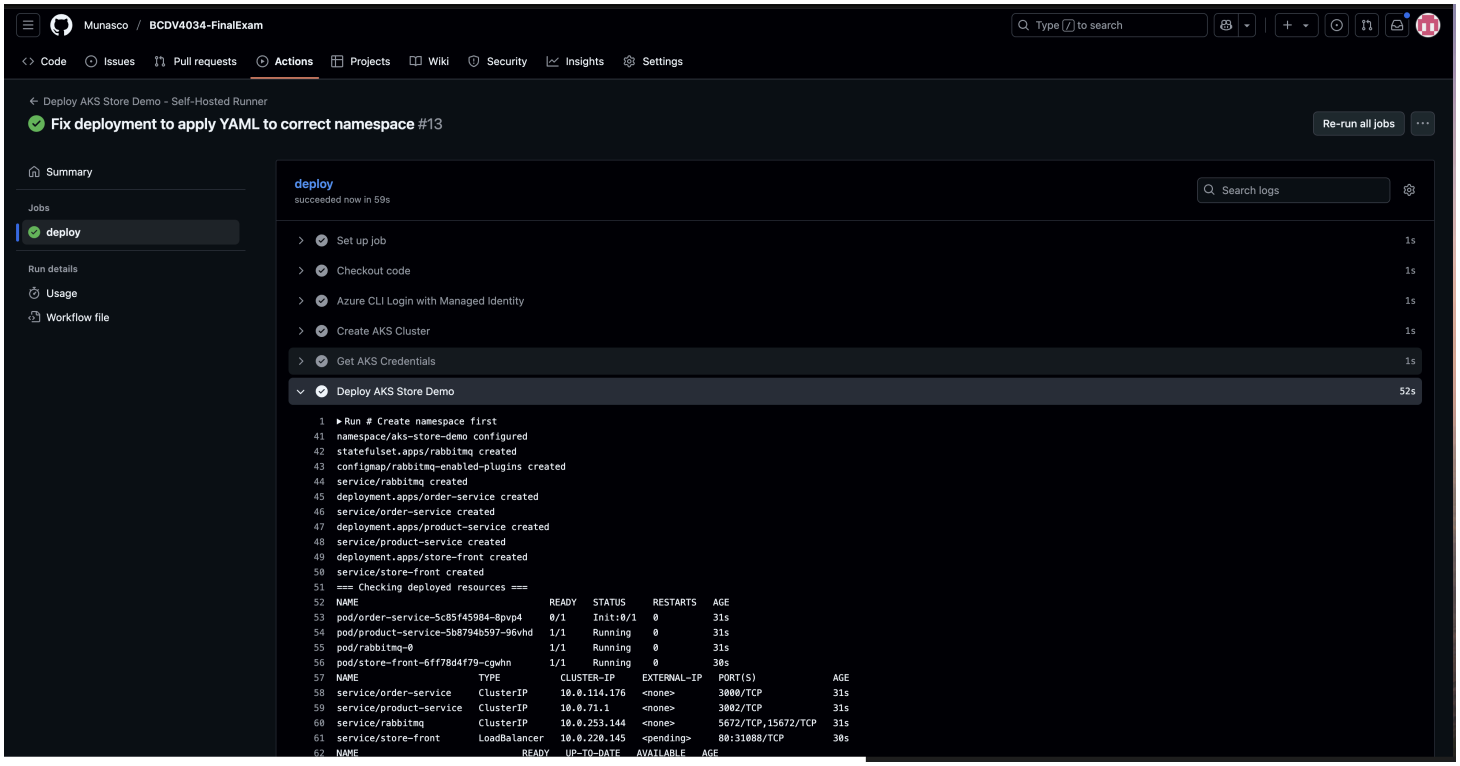
2. AKS Storage Status

```
Finalizers:      [kubernetes.io/pv-protection]
StorageClass:    azure-blob-storage
Status:          Bound
Claim:           default/aks-blob-pvc
Reclaim Policy:  Retain
Access Modes:    RWX
VolumeMode:      Filesystem
Capacity:        10Gi
Node Affinity:   <none>
Message:
Source:
    Type:              CSI (a Container Storage Interface (CSI) volume so
    Driver:            blob.csi.azure.com
    FSType:
    VolumeHandle:      staks5q074gff-aks-persistent-storage
    ReadOnly:          false
    VolumeAttributes:      containerName=aks-persistent-storage
                           protocol=fuse
Events:                <none>
munachiernest-eze@Munachis-MacBook-Pro BCDV4034-FinalExam %
```

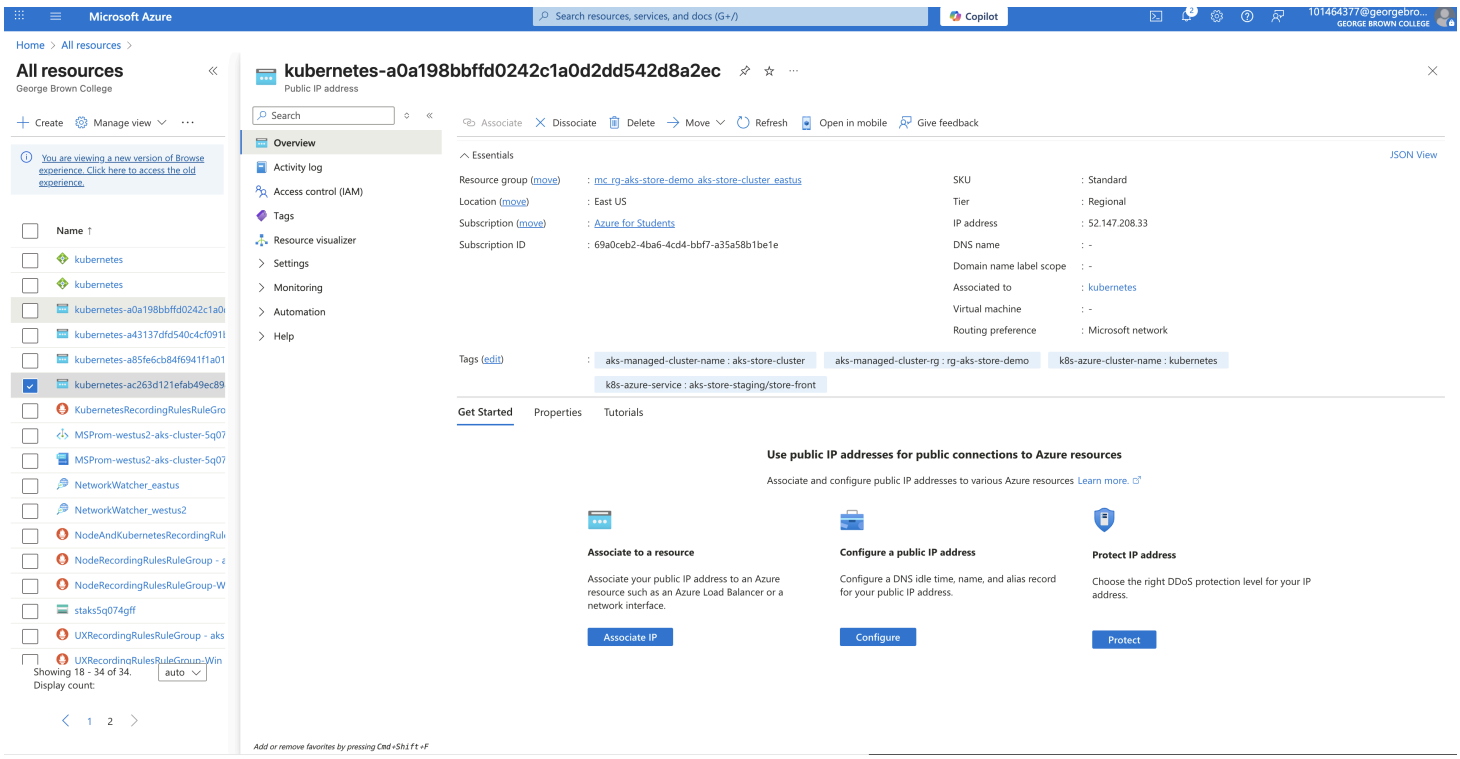
3. Persistent Container



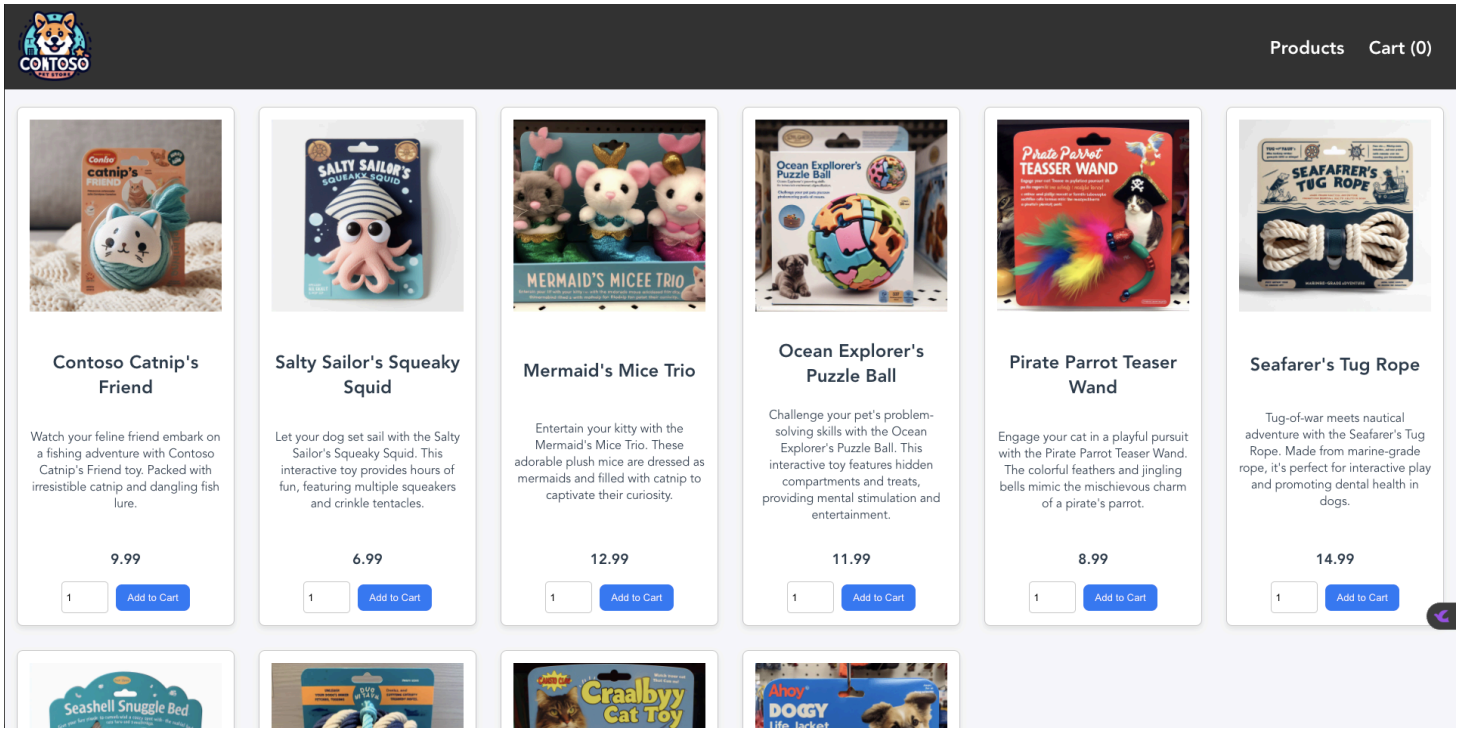
4. CI/CD Pipeline Success



5. AKS Store Demo Deployment



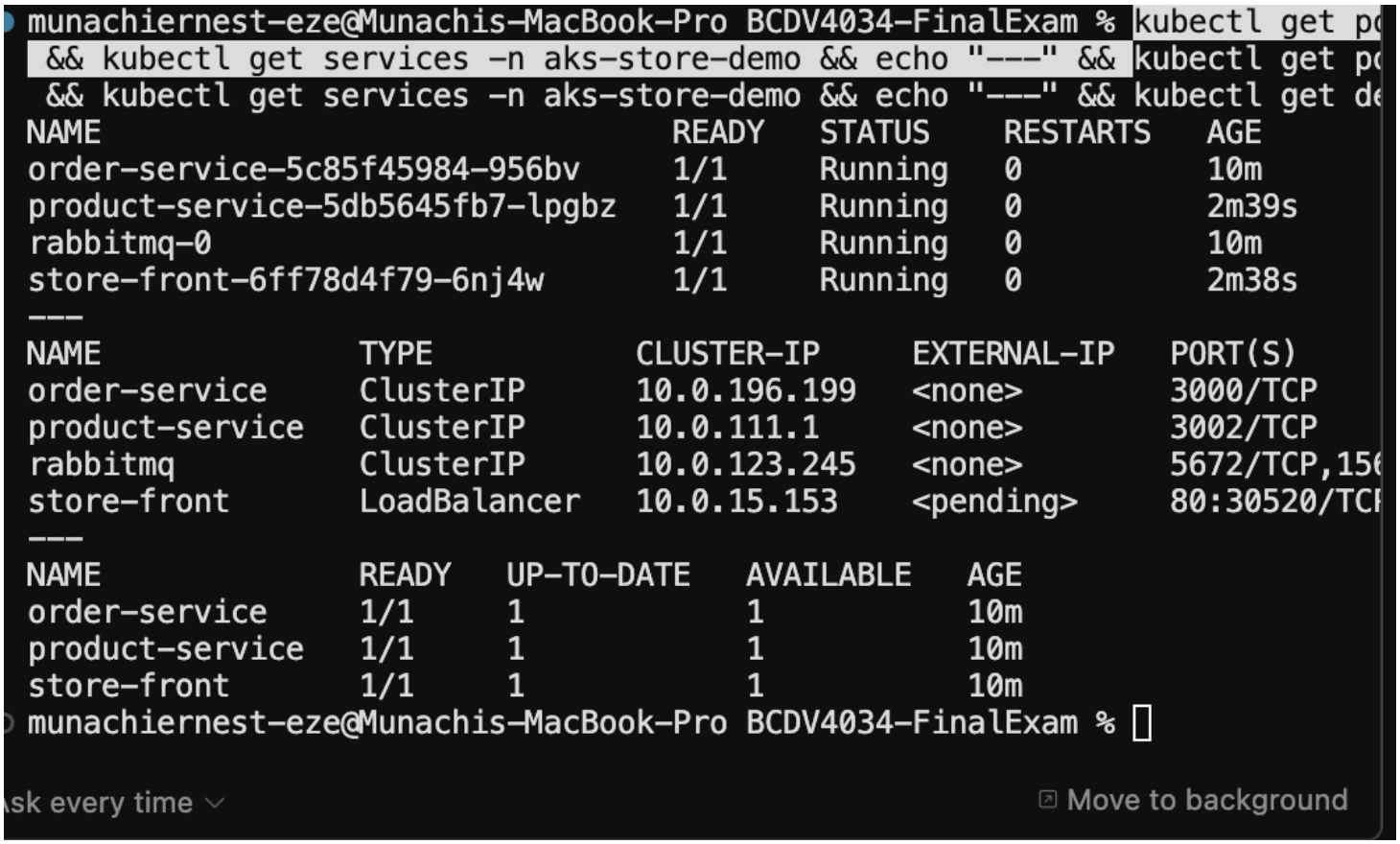
6. AKS Store Application Running



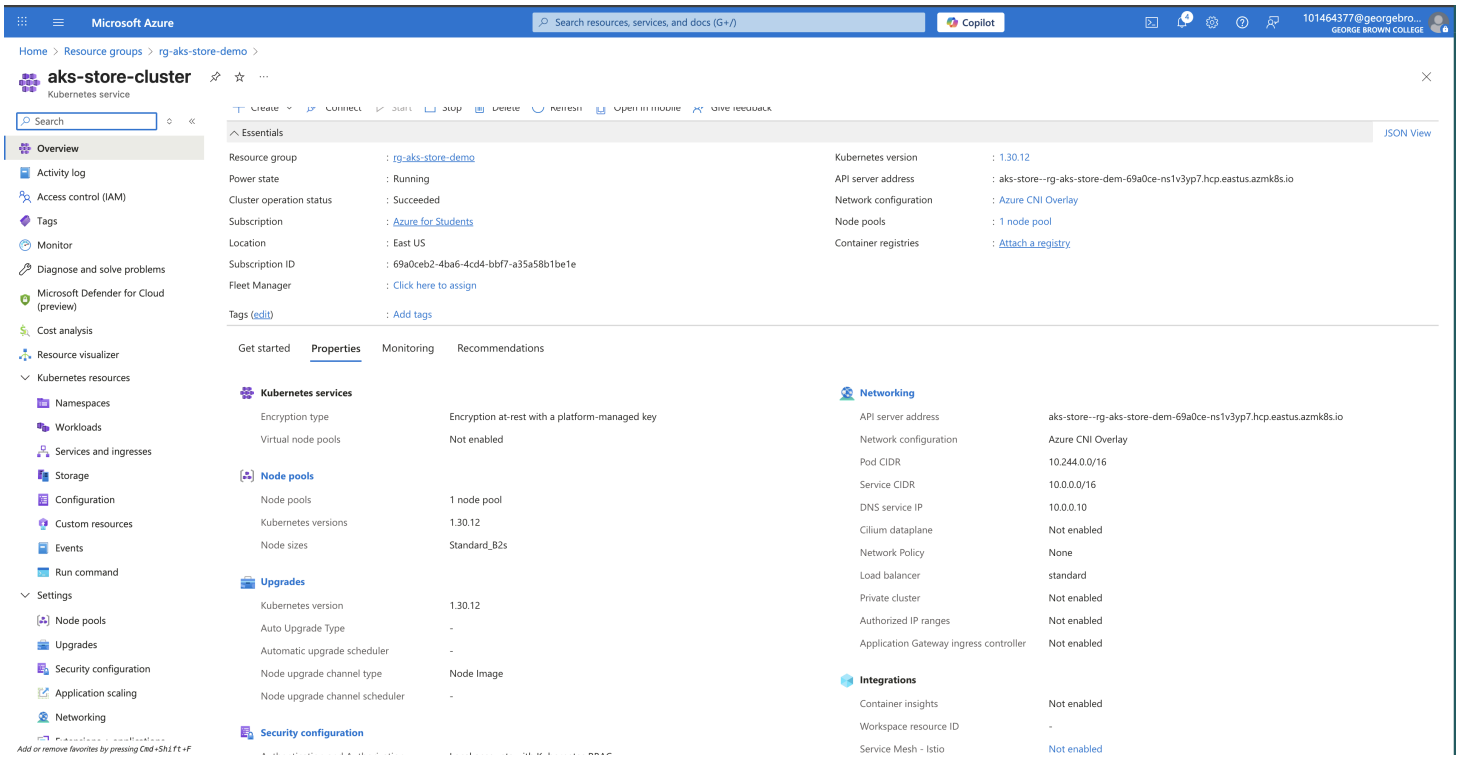
7. AKS Store with Security Context

```
munachiernest-eze@Munachis-MacBook-Pro BCDV4034-FinalExam % kubectl get pod product-service-5db5645fb7-lpgbz -n aks-store-demo -o yaml | grep -A 20 securityContext
    kubectl get pod product-service-5db5645fb7-lpgbz -n aks-store-demo -o yaml | grep -A 20 securityContext
    securityContext:
      allowPrivilegeEscalation: false
      capabilities:
        drop:
        - ALL
      readOnlyRootFilesystem: false
      runAsNonRoot: true
      runAsUser: 1000
    terminationMessagePath: /dev/termination-log
    terminationMessagePolicy: File
    volumeMounts:
    - mountPath: /var/run/secrets/kubernetes.io/serviceaccount
      name: kube-api-access-zvdzp
      readOnly: true
  dnsPolicy: ClusterFirst
  enableServiceLinks: true
  nodeName: aks-default-39692054-vmss000002
  nodeSelector:
    kubernetes.io/os: linux
  preemptionPolicy: PreemptLowerPriority
  priority: 0
--
  securityContext:
    fsGroup: 1000
    runAsGroup: 1000
    runAsNonRoot: true
    runAsUser: 1000
  serviceAccount: default
  serviceAccountName: default
  terminationGracePeriodSeconds: 30
  tolerations:
  - effect: NoExecute
    key: node.kubernetes.io/not-ready
    operator: Exists
    tolerationSeconds: 300
  - effect: NoExecute
    key: node.kubernetes.io/unreachable
    operator: Exists
    tolerationSeconds: 300
  - effect: NoSchedule
    key: node.kubernetes.io/memory-pressure
    operator: Exists
  volumes:
```

8. AKS Store with Security



9. AKS Store Cluster with Dependencies



Overview

This document demonstrates the integration of Azure Blob Storage with Azure Kubernetes Service (AKS) through persistent volumes, along with a complete CI/CD pipeline for automated deployment of the AKS store demo application.

Files Submitted

1. Terraform Configuration

`storage-account.tf` - Creates Azure Storage Account and Blob Container

2. Kubernetes Storage Resources

`k8s-storage-class.yaml` - StorageClass for Azure Blob CSI driver
`k8s-persistent-volume.yaml` - PersistentVolume for blob storage
`k8s-persistent-volume-claim.yaml` - PVC for requesting storage
`k8s-storage-secret.yaml` - Secret for storage credentials

3. Application Deployment

`store-front-with-storage.yaml` - Deployment with volume mount

4. CI/CD Pipeline

`.github/workflows/deploy-aks-store-runner.yaml` - GitHub Actions workflow for automated deployment

CI/CD Pipeline Details

Architecture

- **Self-hosted GitHub Runner**: Running on Azure VM for secure access to Azure resources
- **Managed Identity Authentication**: Using user-assigned managed identity for secure Azure authentication
- **Automated Deployment**: Triggers on push to main branch or manual dispatch

Pipeline Steps

1. **Checkout Code**: Retrieves the latest code from the repository
2. **Azure CLI Login**: Authenticates using managed identity
3. **AKS Cluster Management**: Creates AKS cluster if it doesn't exist
4. **Get Credentials**: Retrieves kubectl credentials for the cluster
5. **Deploy Application**: Applies the AKS store demo to the cluster
6. **Verify Deployment**: Checks deployment status and service URLs

Security Features

- ✅ **Managed Identity**: No secrets stored in workflow
- ✅ **Self-hosted Runner**: Secure execution environment
- ✅ **Namespace Isolation**: Deploys to dedicated namespace
- ✅ **Error Handling**: Robust error handling and retry logic

Current Status

✅ **Azure Storage Account**: `staks5q074gff` - Created
✅ **Blob Container**: `aks-persistent-storage` - Created
✅ **Persistent Volume**: `aks-blob-pv` - Bound
✅ **Storage Class**: `azure-blob-storage` - Active
✅ **Deployment**: `store-front-with-storage` - Deployed
✅ **CI/CD Pipeline**: Automated deployment workflow - Active
✅ **AKS Store Demo**: Successfully deployed via CI/CD
✅ **LoadBalancer Service**: `store-front` service with external access
✅ **All Services**: `product-service`, `order-service`, `rabbitmq` - Running

Deployment Results

Services Deployed

- **store-front**: LoadBalancer service (external access)
- **product-service**: ClusterIP service
- **order-service**: ClusterIP service
- **rabbitmq**: StatefulSet with ClusterIP service

Pod Status

- All pods running successfully
- Services properly configured
- LoadBalancer pending external IP assignment

CI/CD Success Metrics

- ✅ Workflow completes successfully
- ✅ All deployment steps pass
- ✅ Services accessible within cluster
- ✅ External access via LoadBalancer