**Azure DevOps & Cloud Capstone Project Report**

**Project Overview**

This capstone project demonstrates the complete DevOps lifecycle of a cloud-native e-commerce application using Azure services. The project integrates Infrastructure as Code (IaC), CI/CD pipelines, monitoring, security scanning, and high availability/disaster recovery using two AKS clusters deployed in different regions.

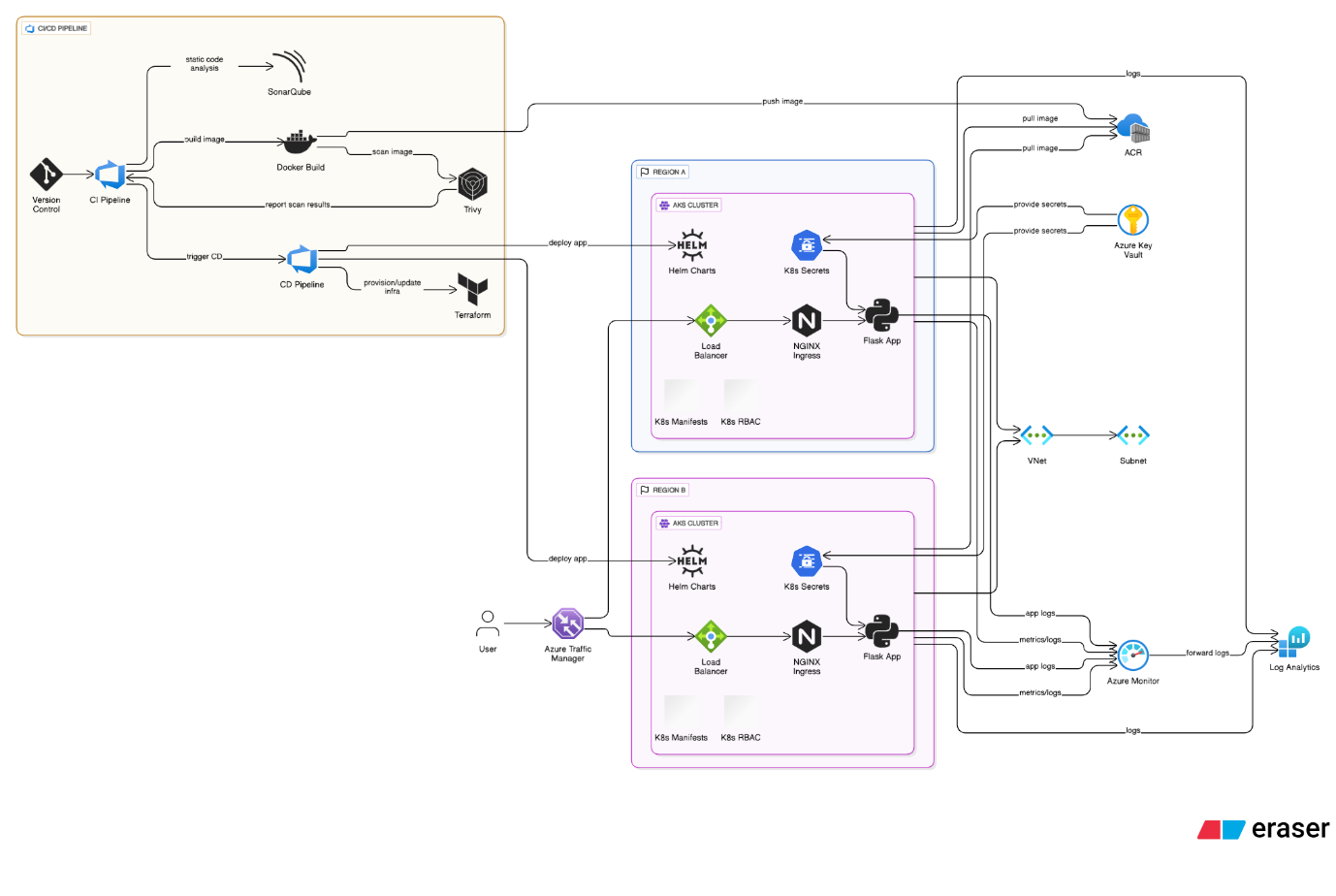
**Author: Sethumadav Somasundaram  
Date: 25-06-2025**

**Github links:**

[**ssethumadav/capstone**](https://github.com/ssethumadav/capstone)

[**ssethumadav/Capstone-infra1**](https://github.com/ssethumadav/Capstone-infra1)

**Project Architecture:**

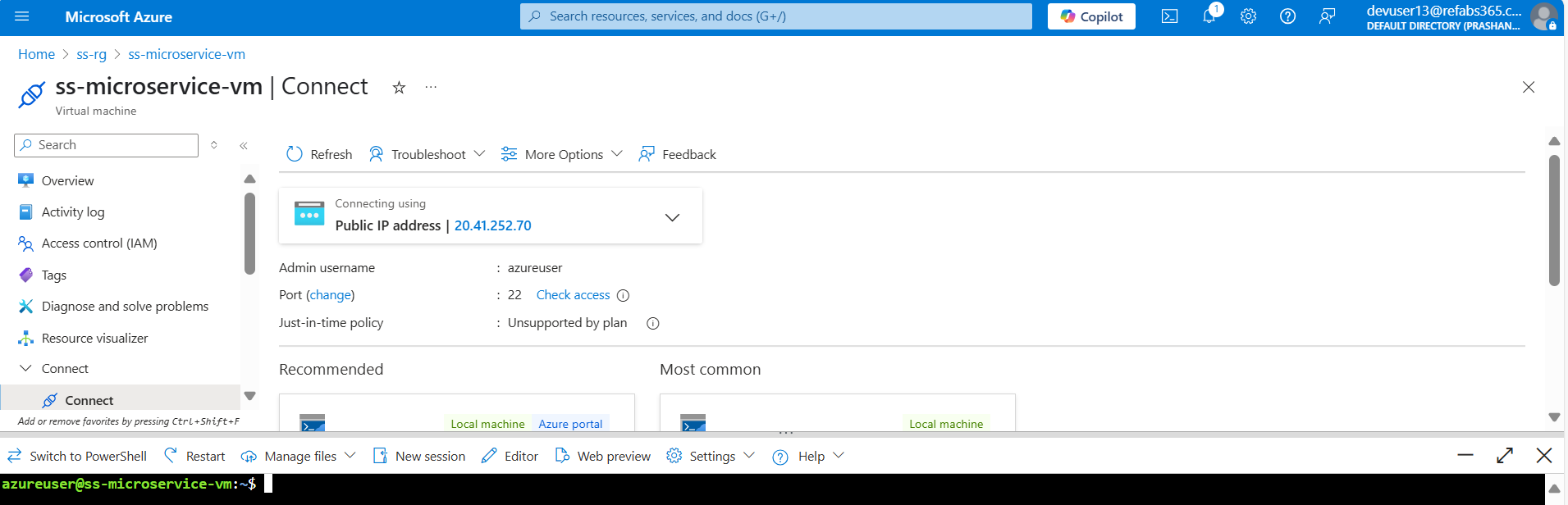


**1.Application Hosting on Azure VM (Initial Phase)**

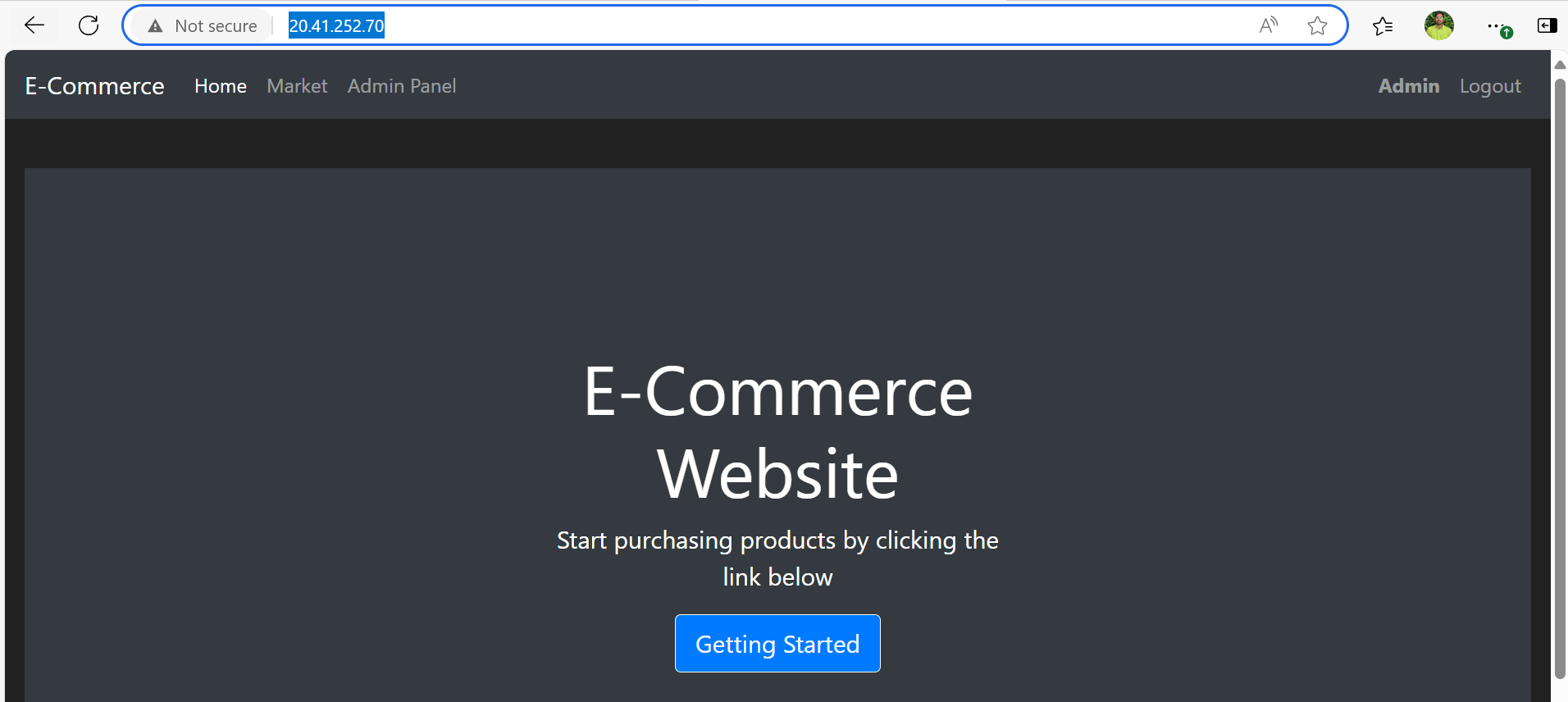
* A Python Flask-based e-commerce application was hosted on an Azure Virtual Machine.
* Connected the app to an Azure SQL Database to manage product and user data.

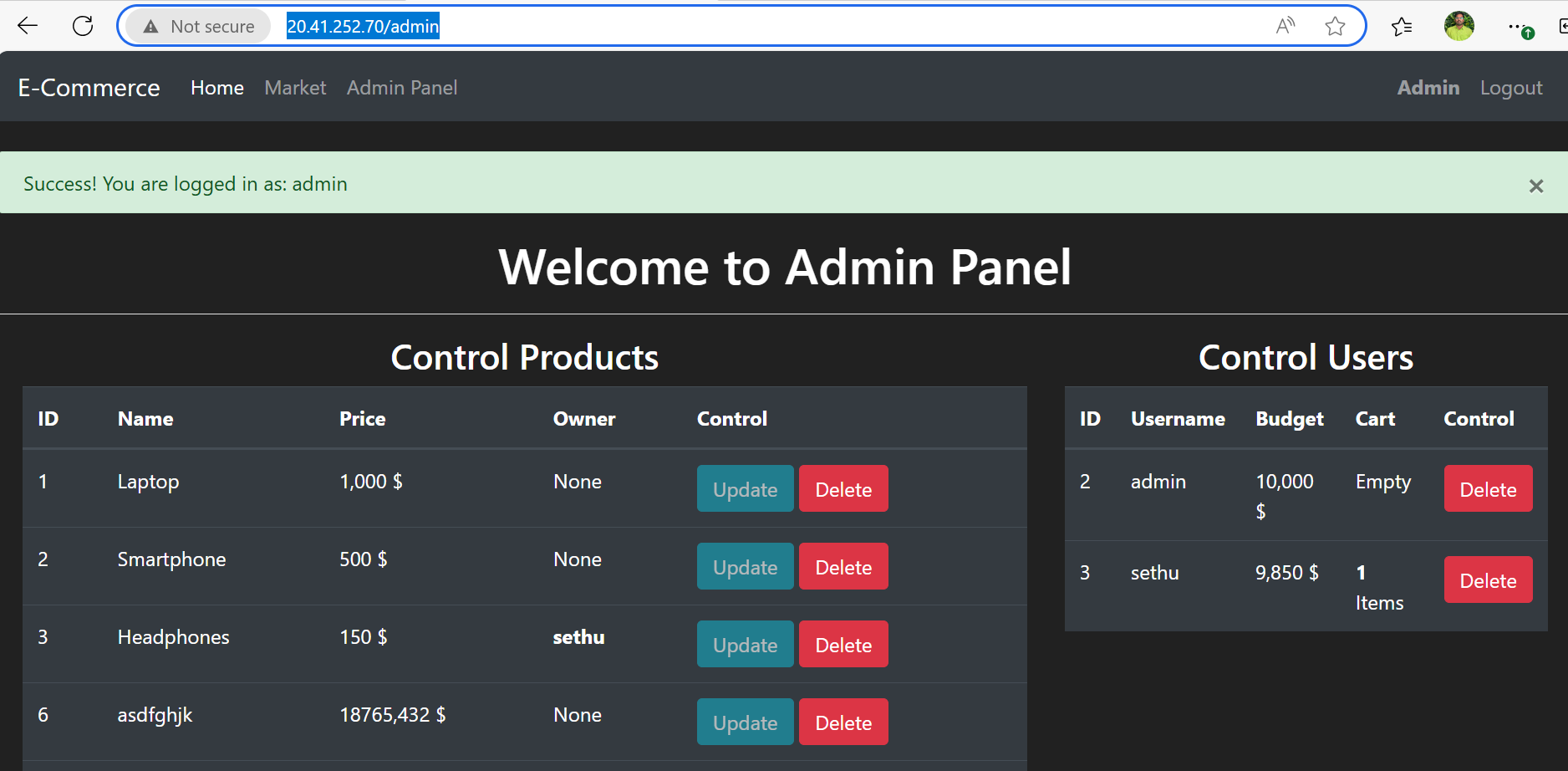
**Screenshots :**

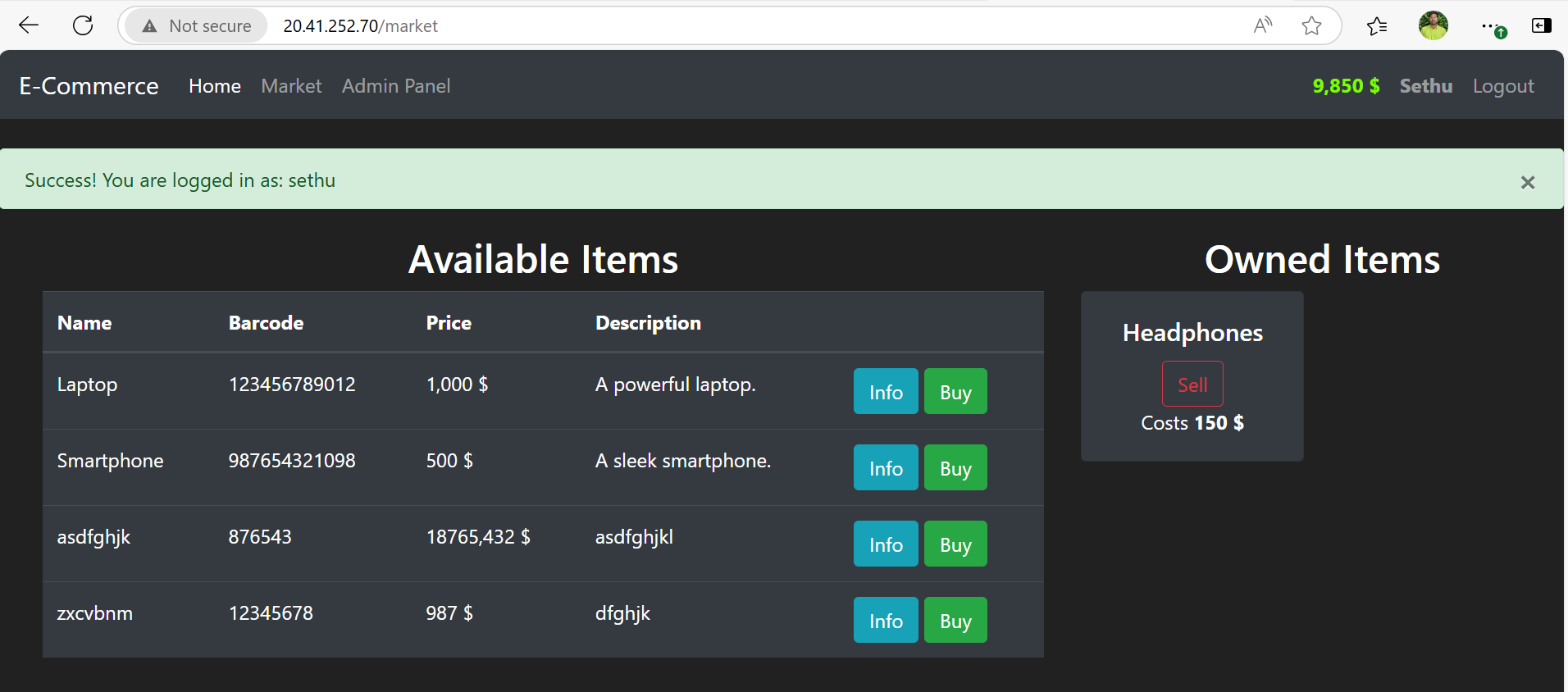
* Azure VM running status in portal.



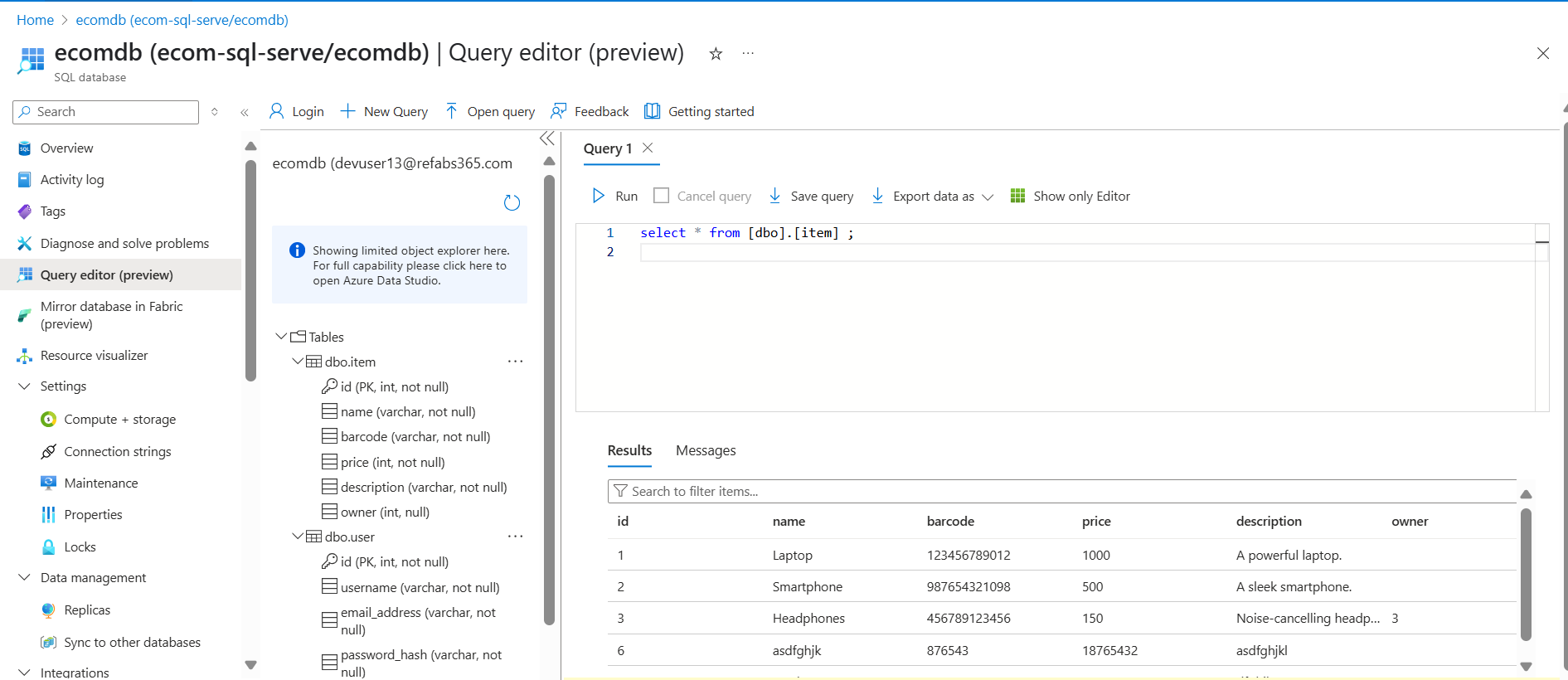
* Application homepage on the browser using vm public ip ([E-Commerce](http://20.41.252.70/)).







. Azure SQL DB query editor showing tables/data.

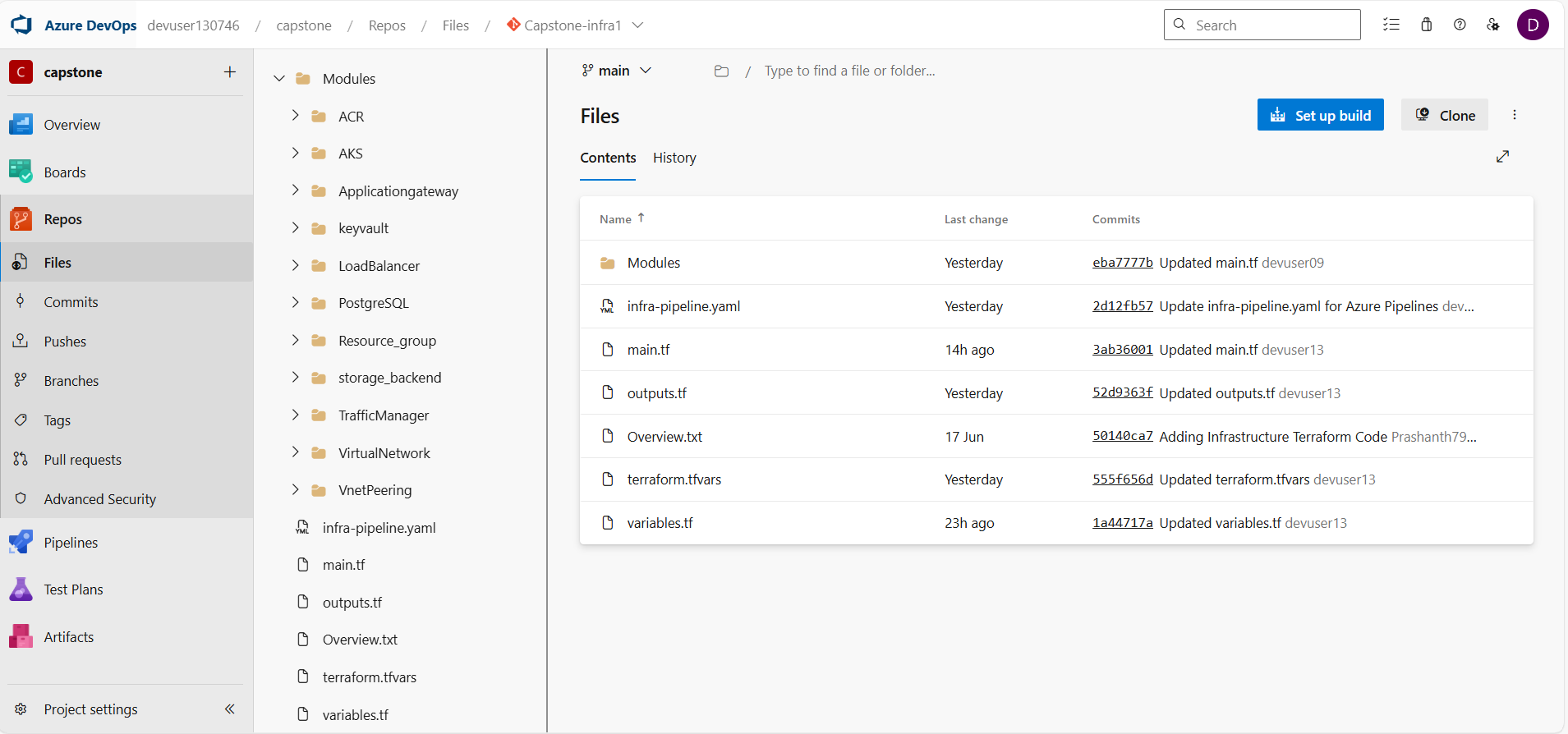


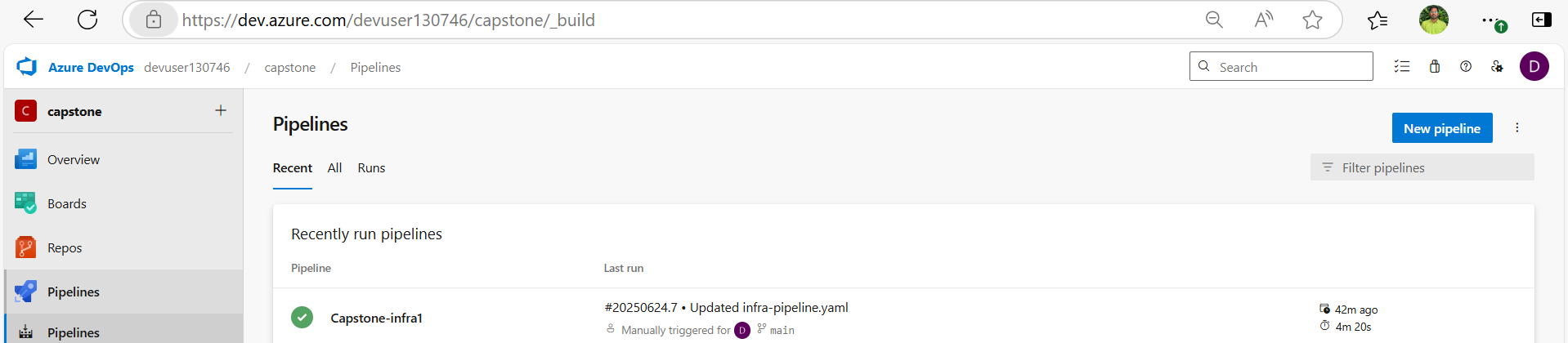
**2. Infrastructure as Code with Terraform**

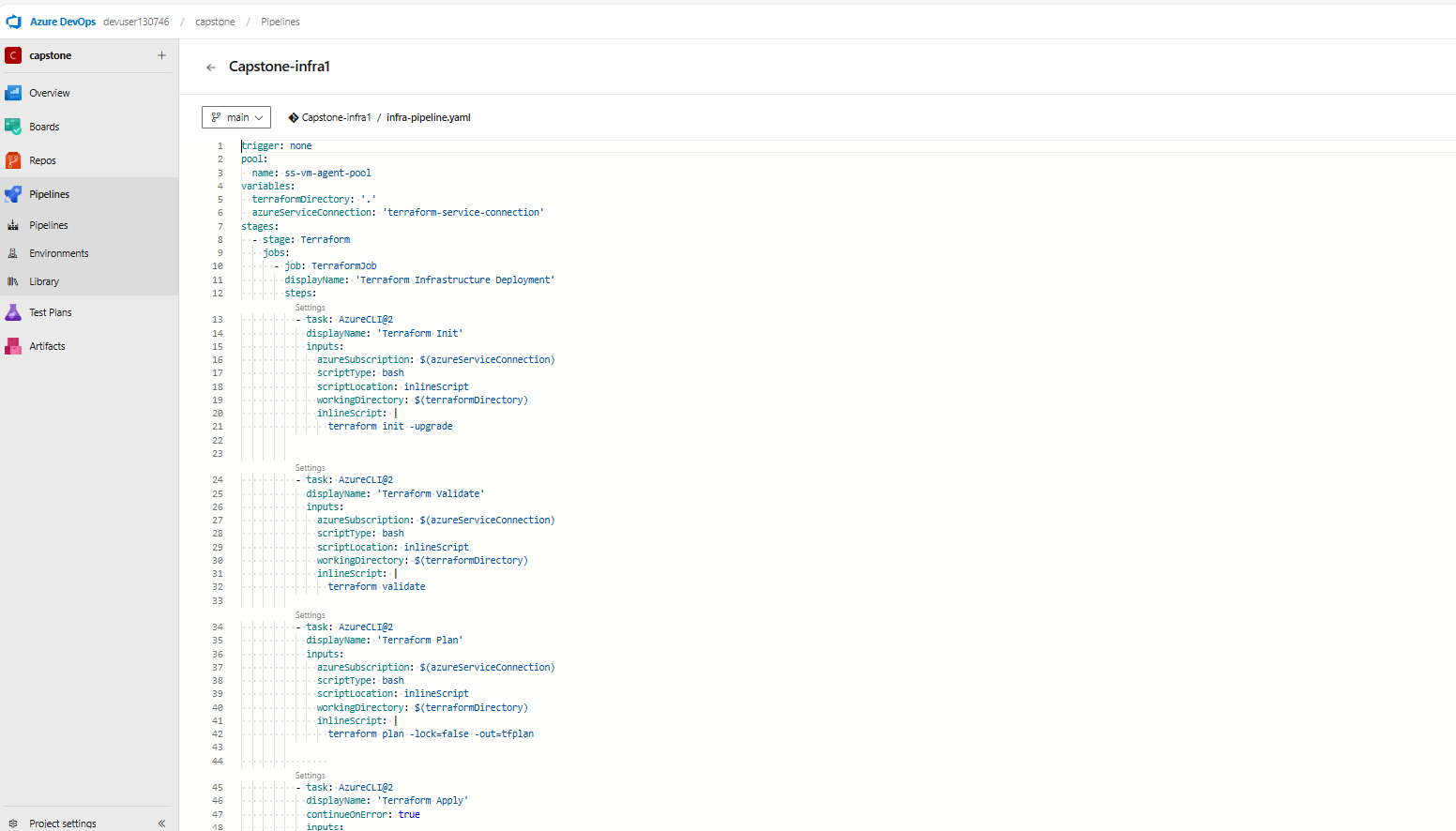
* Created Terraform modules for:
  + Azure Container Registry (ACR)
  + Azure Kubernetes Service (AKS)
  + Networking setup (VNet, subnets)
* Provisioned two AKS clusters in separate Azure regions for high availability.

**Screenshots :**

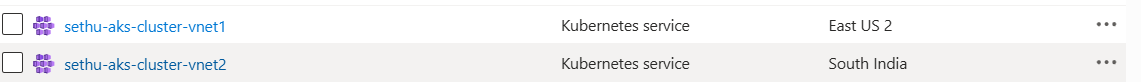
* Terraform directory structure.



* Successful Terraform apply logs or Azure DevOps Terraform pipeline output. 



* Azure portal showing two AKS clusters in different regions.

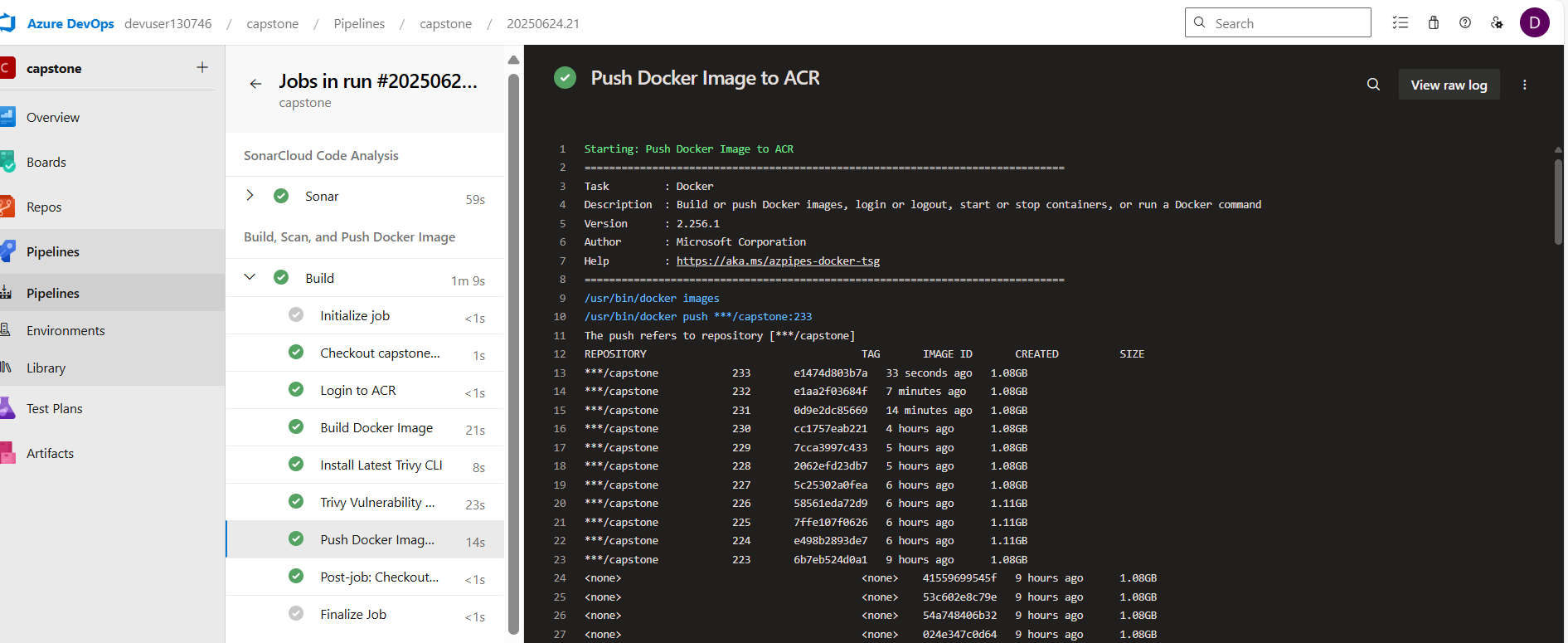


**3. CI Pipeline: Docker Image Build & Push**

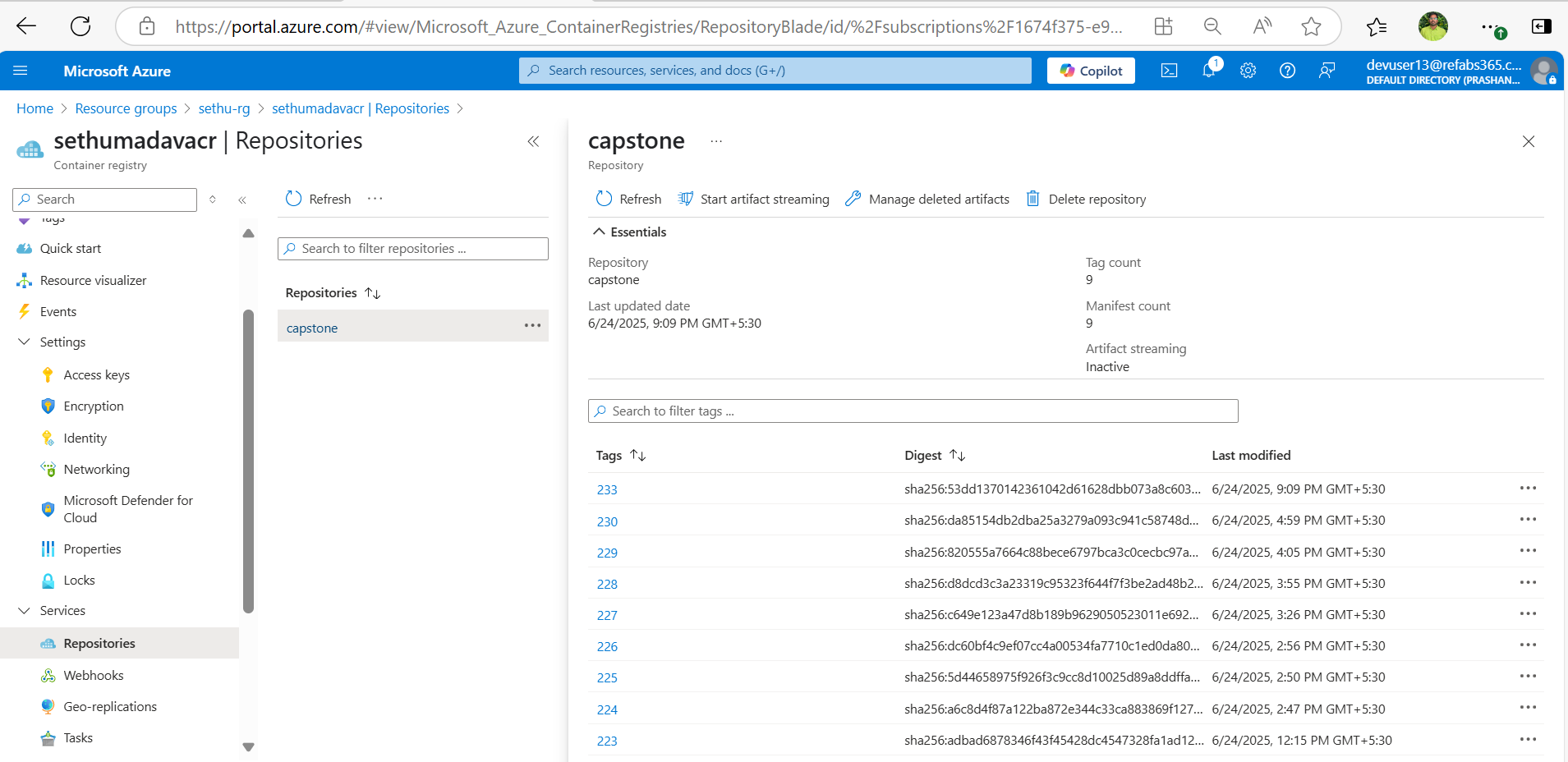
* Set up an Azure DevOps CI pipeline that:
  + Logs in to ACR
  + Builds a Docker image from the app source code
  + Pushes the image to the ACR

**Screenshots :**

* Azure DevOps CI pipeline run with stages shown (build, push).



* ACR showing the uploaded image.

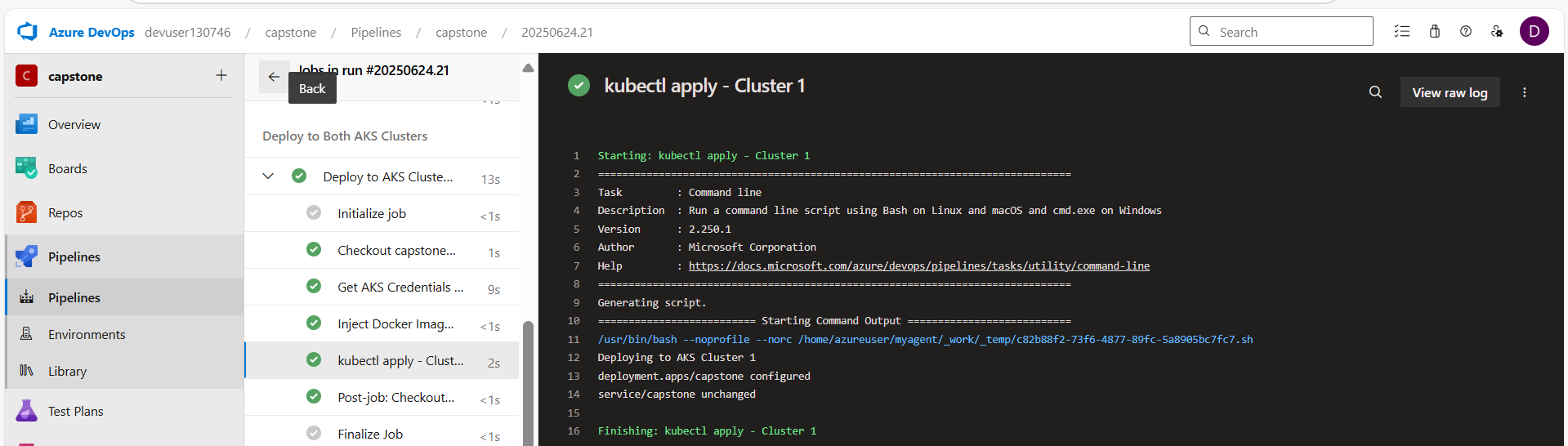


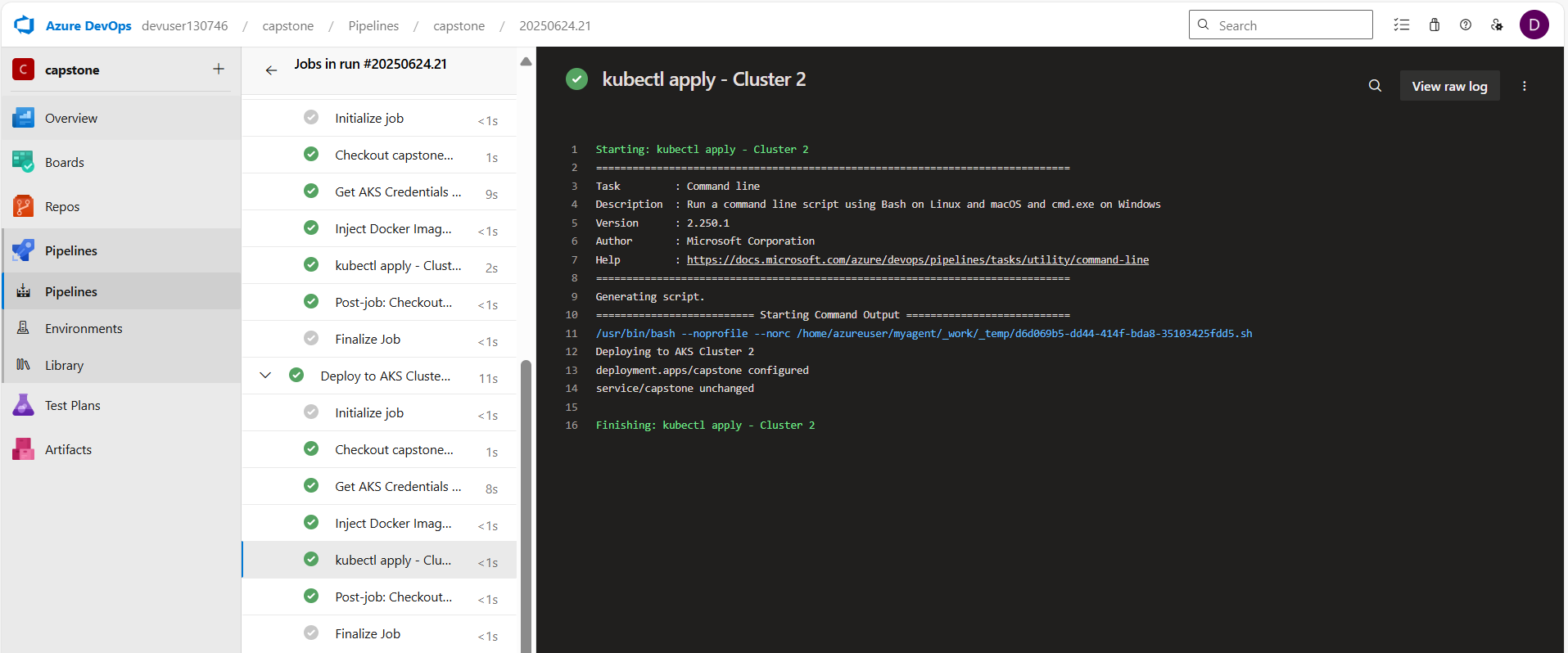
**4. CD Pipeline: Deploy to Two AKS Clusters**

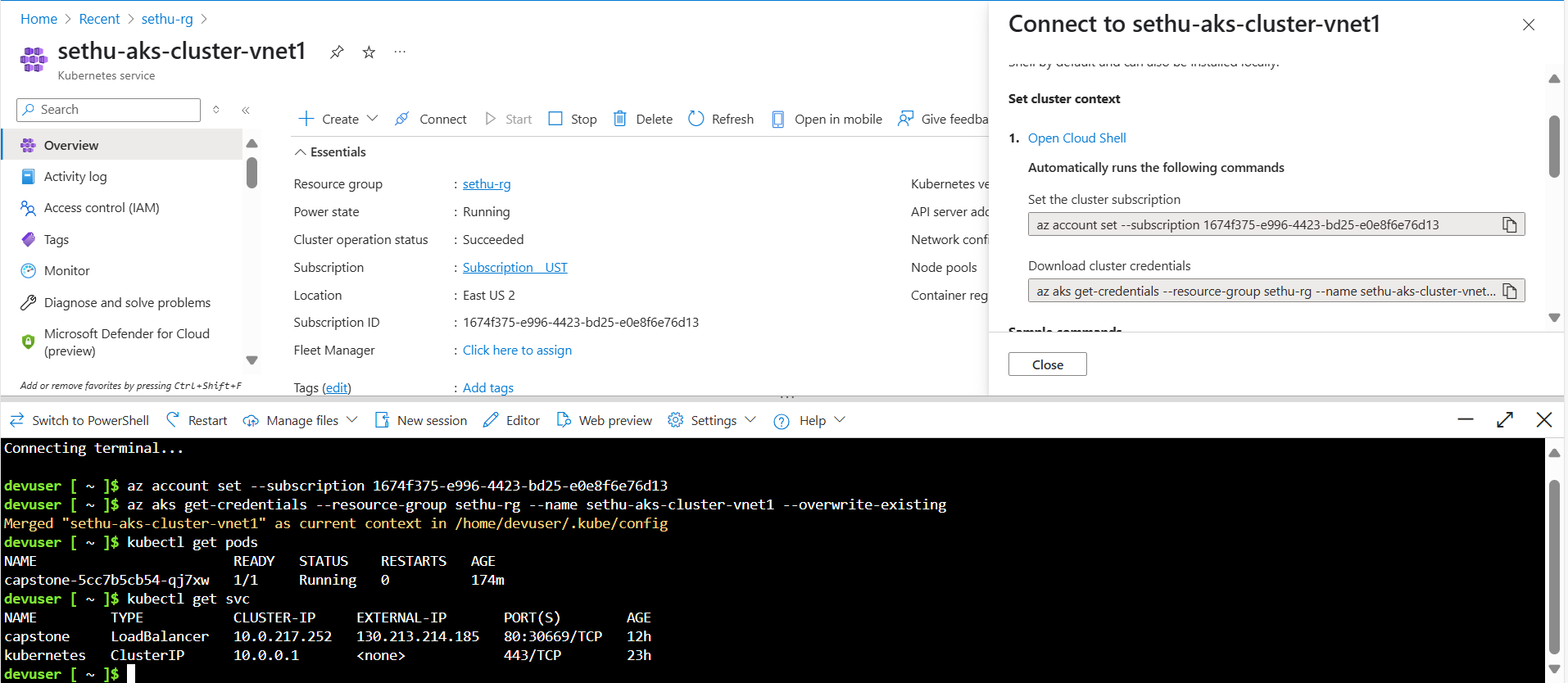
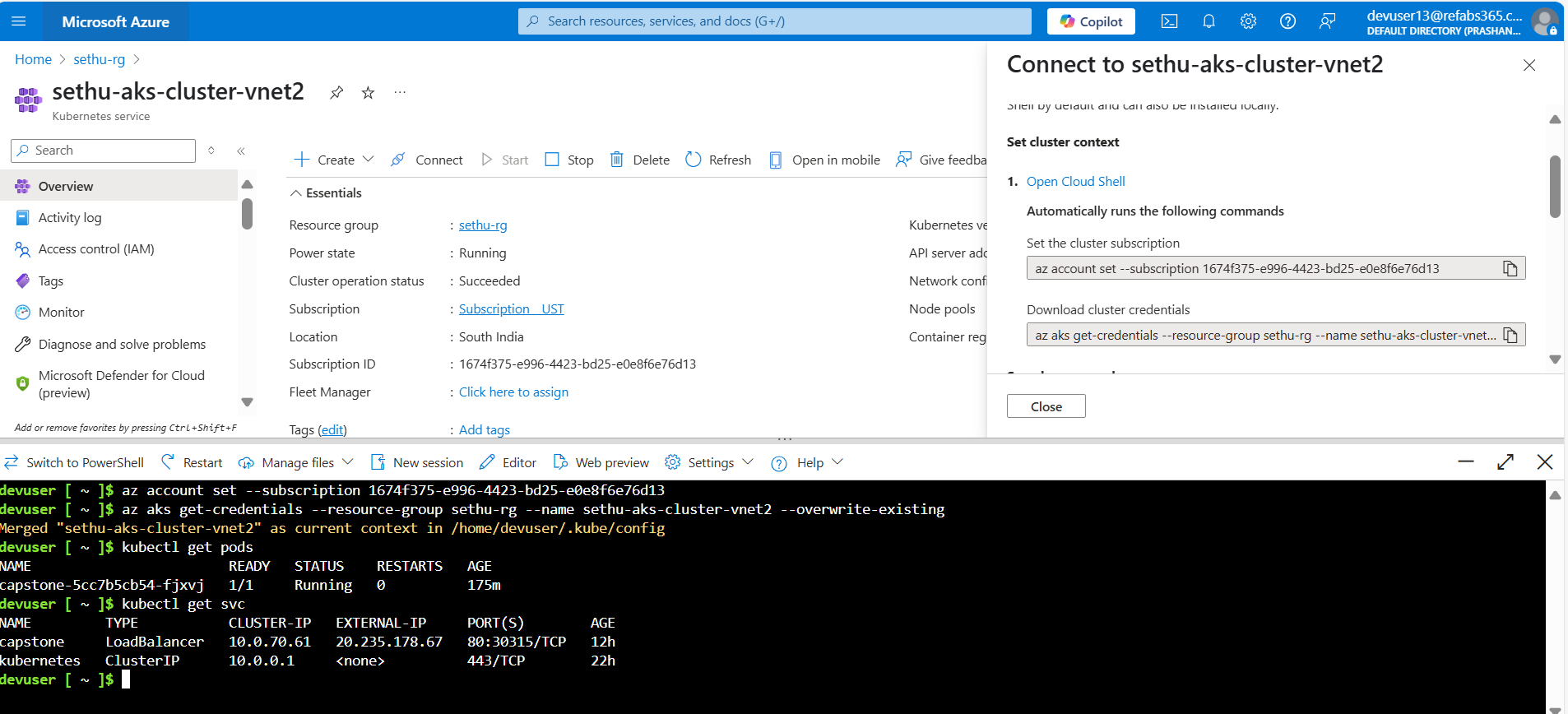
* Configured a CD pipeline to:
  + Pull the latest Docker image from ACR
  + Deploy the app to both AKS clusters using Kubernetes manifests

**Screenshots :**

* Azure DevOps CD pipeline run output.





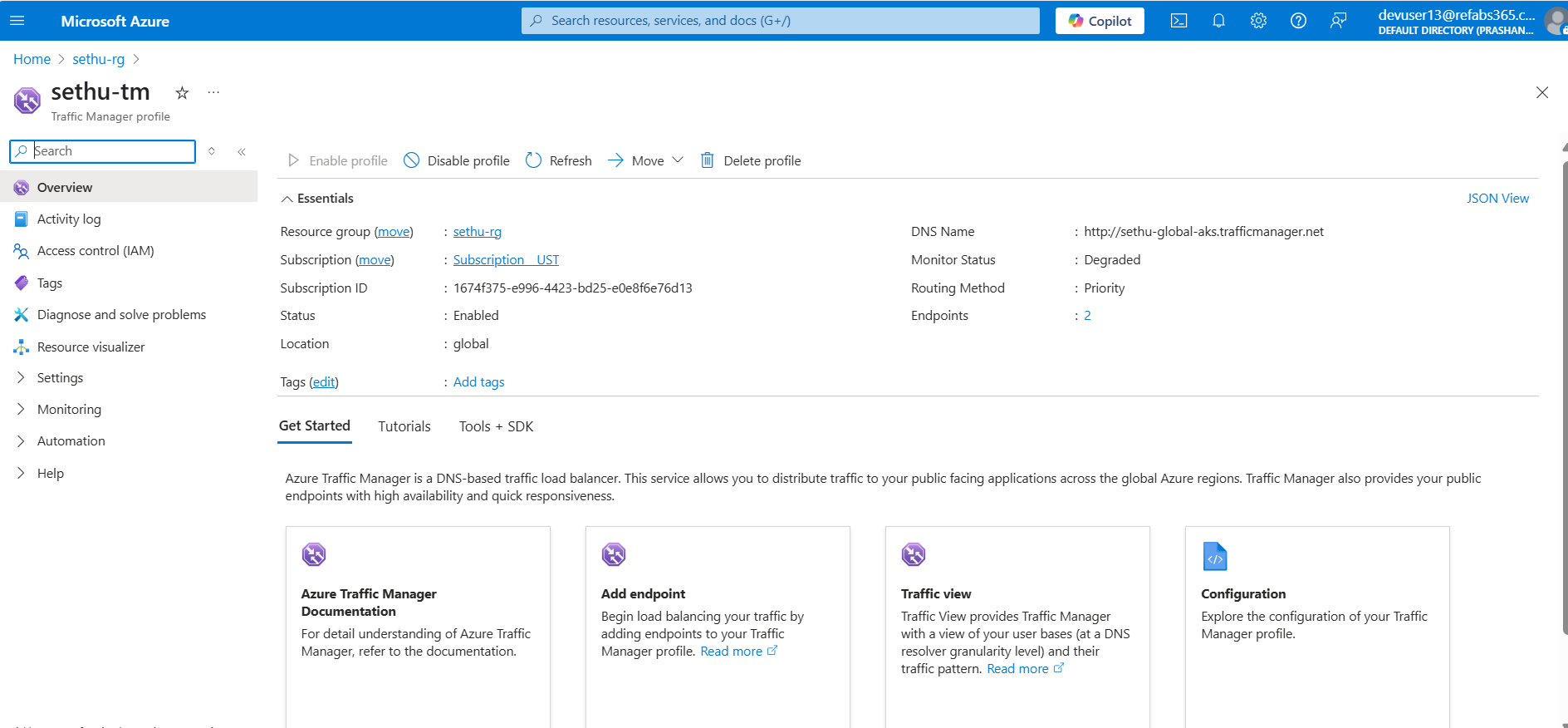
* kubectl get pods and kubectl get svc from both clusters.
* 
* 

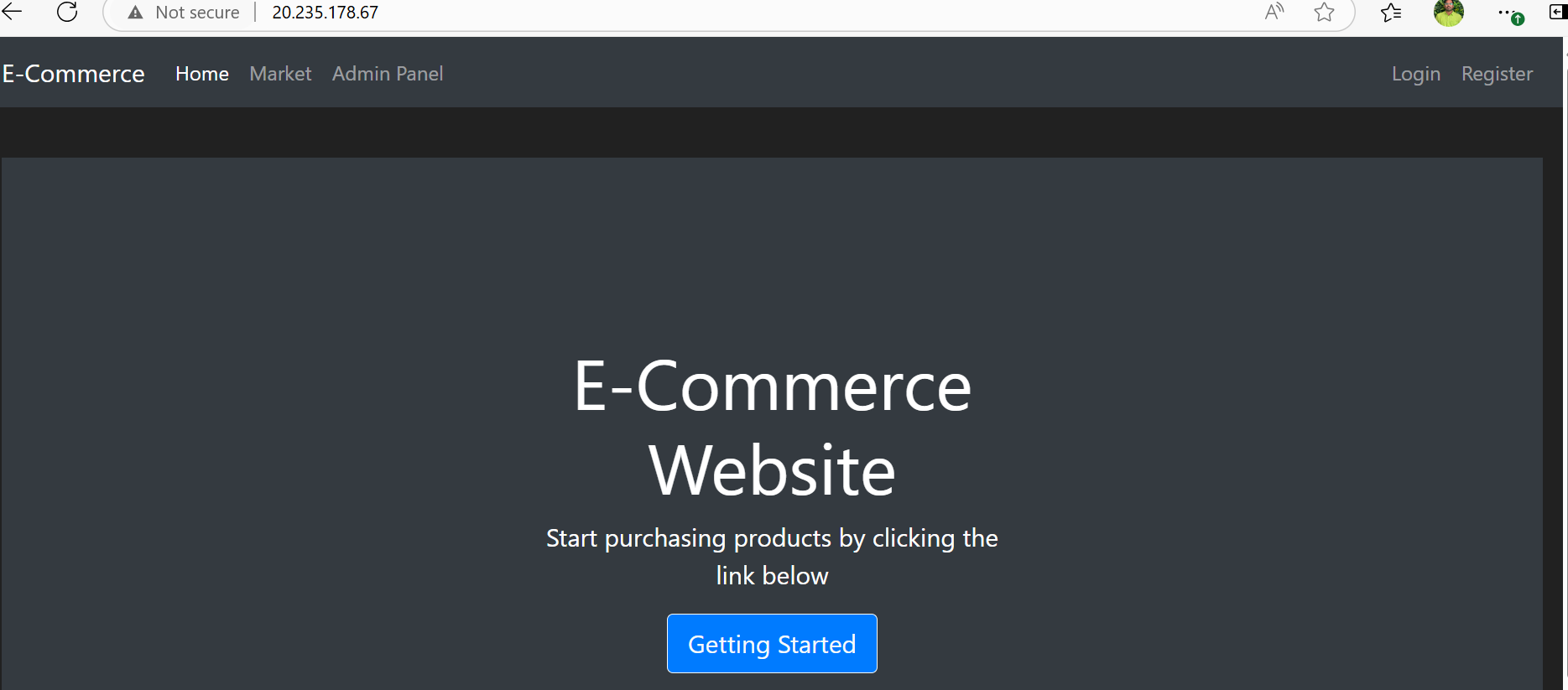
**5. Traffic Manager Configuration for HA/DR**

* Set up Azure Traffic Manager to:
  + Route traffic to the AKS cluster that is healthy and active
  + Ensure application availability in case of regional failure

**Screenshots :**

* Traffic Manager profile in Azure portal.



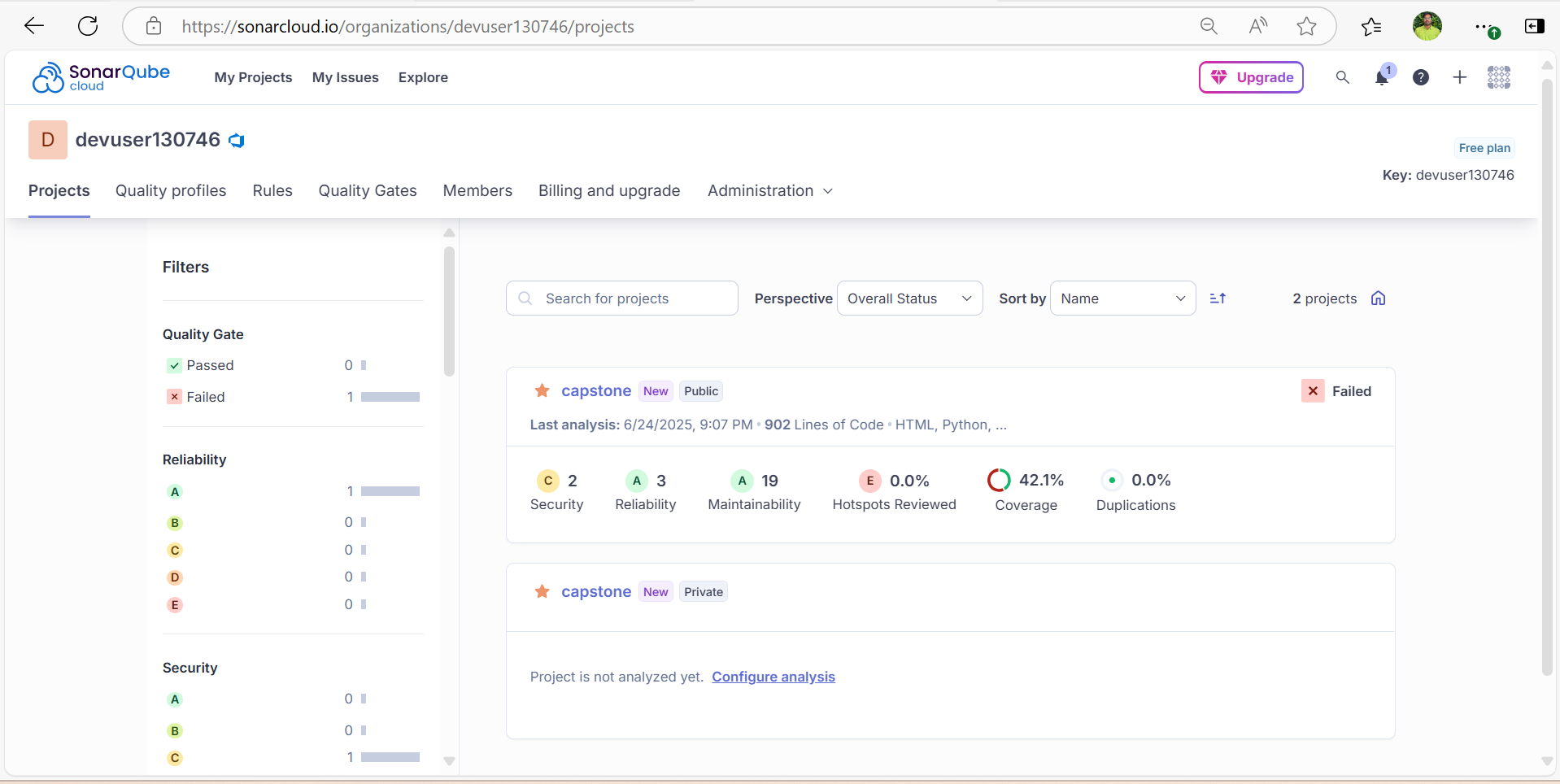
* Testing failover using browser or curl after simulating failure in one region.so using cluster 2 in another region:
* 

**6. SonarQube Integration for Code Quality**

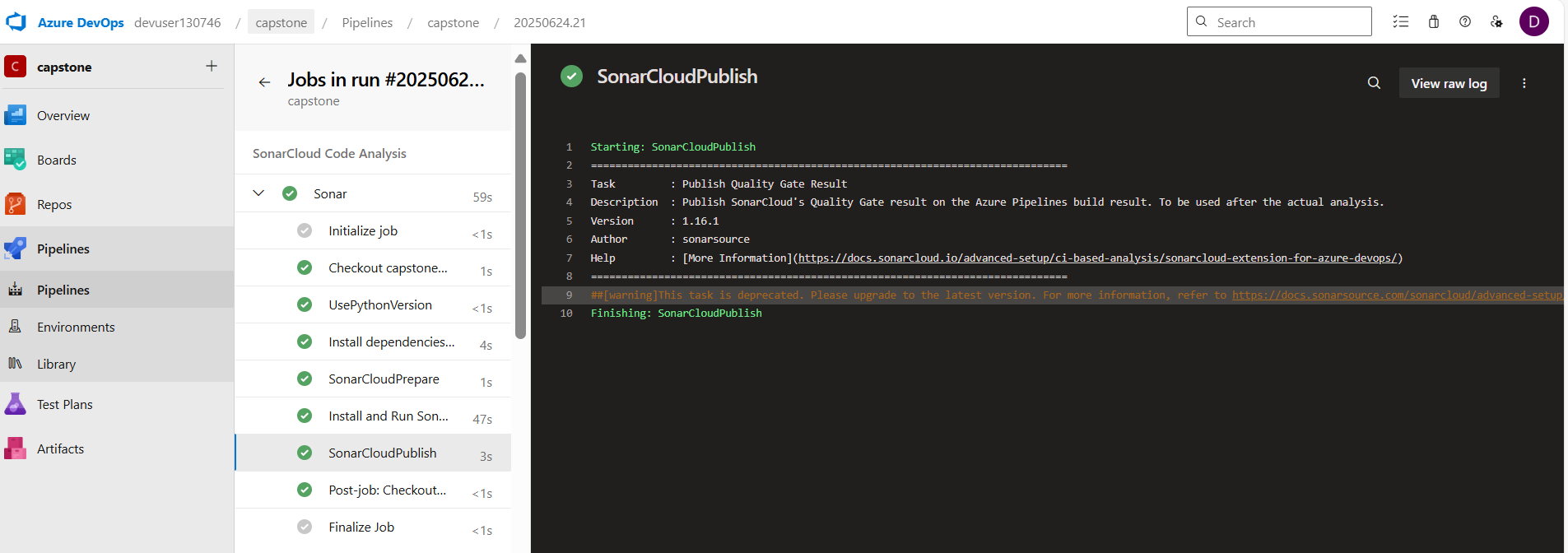
* Integrated SonarQube with Azure DevOps to scan the Python codebase for:
  + Code smells
  + Bugs
  + Vulnerabilities
  + Technical debt

**Screenshots :**

* SonarQube dashboard showing scan results.



* Pipeline stage showing successful SonarQube scan.

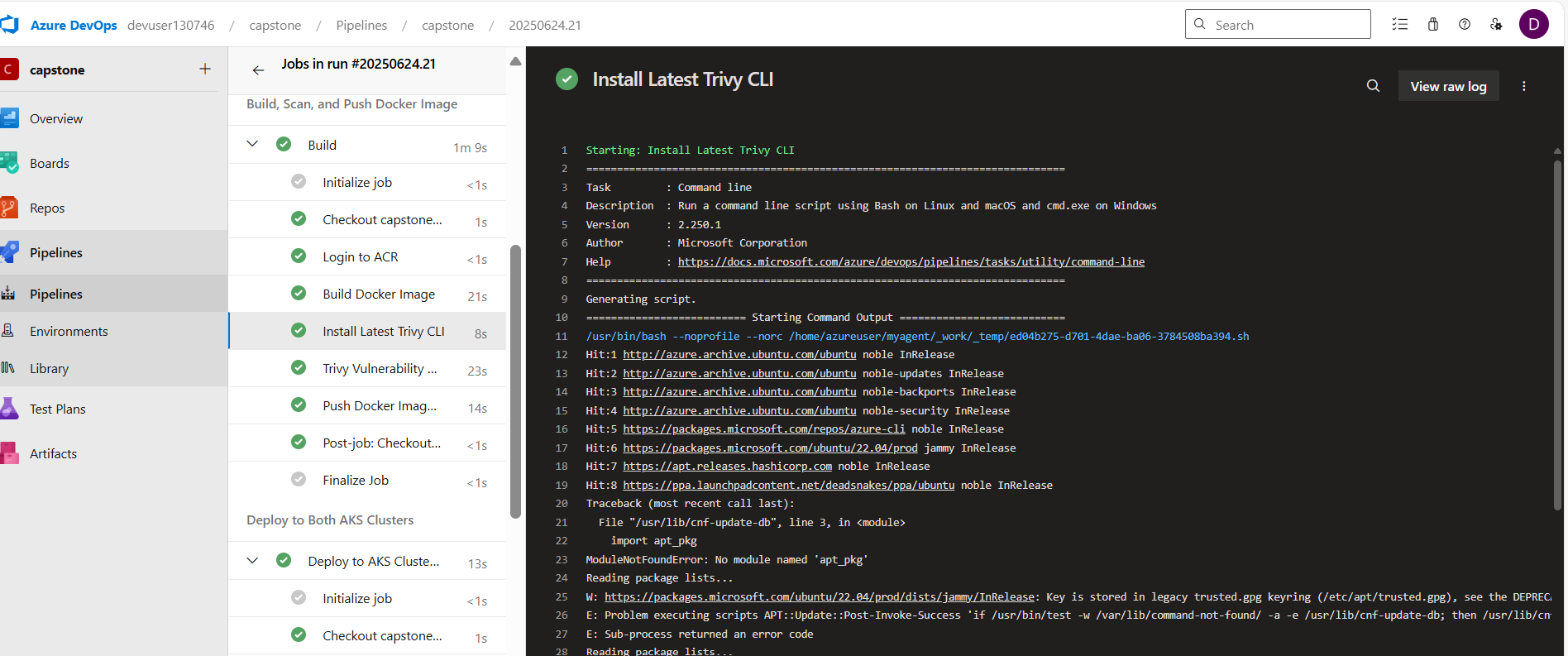


**7. Container Image Scanning with Trivy**

* Integrated Trivy into the CI pipeline to scan Docker images for:
  + OS-level vulnerabilities
  + Python package vulnerabilities
* Ensured the scan does not fail the build but highlights high/critical issues.

**Screenshots :**

* Trivy scan output in pipeline logs.



* vulnerability report (highlighted rows).



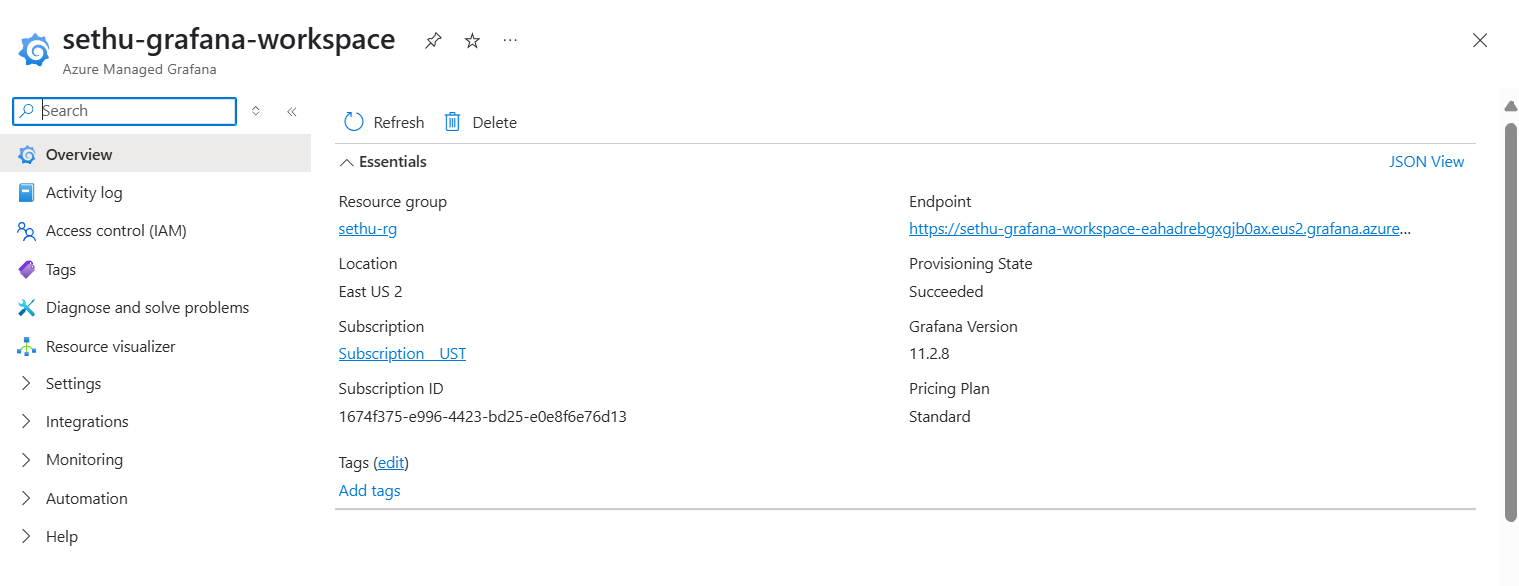
**8. Monitoring and Observability**

Configured **Grafana** for monitoring and observability:

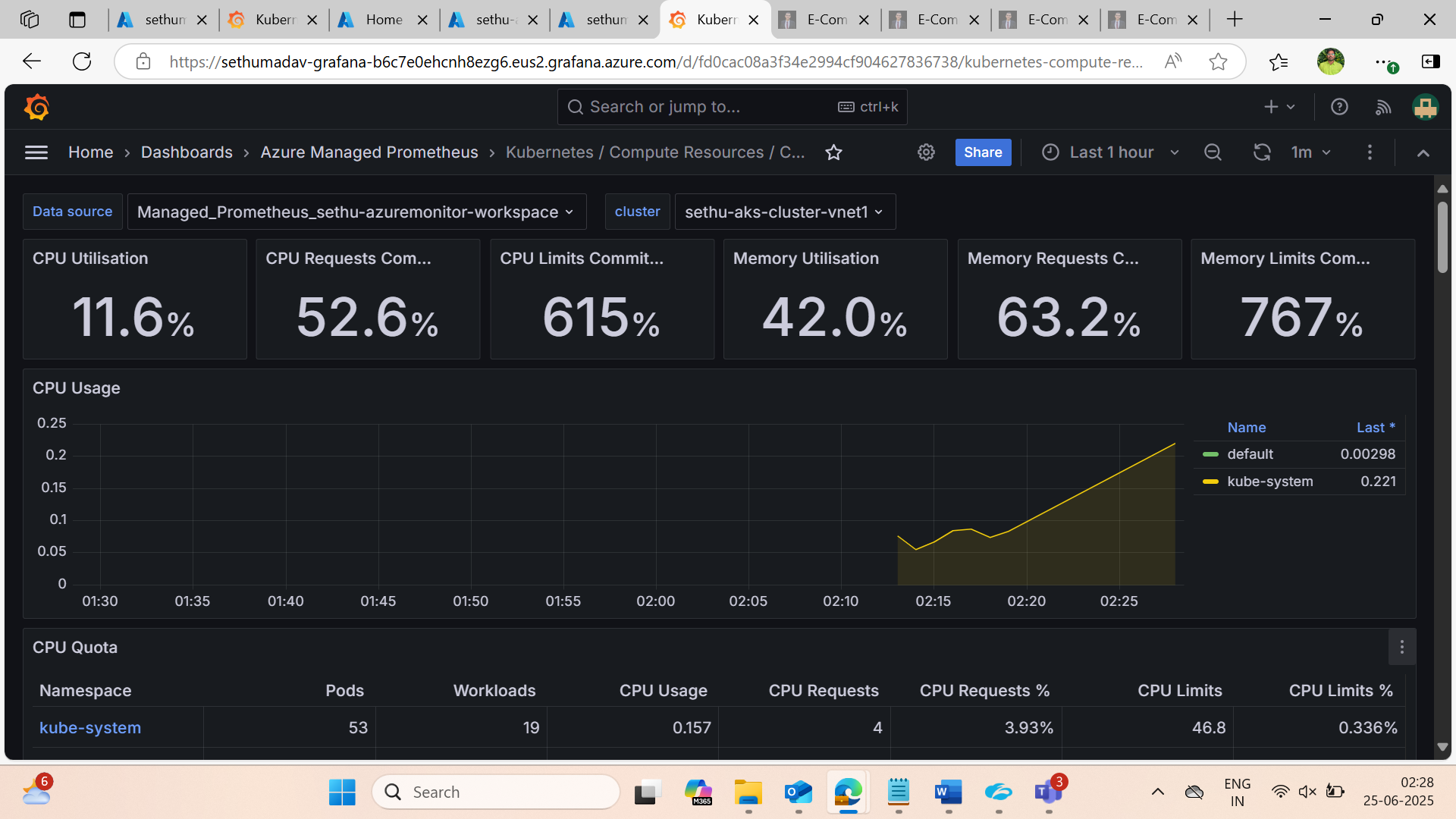
* Integrated Grafana with Azure Kubernetes Service (AKS) to collect metrics and logs.
* Connected Grafana to Prometheus (deployed in the AKS cluster) for real-time monitoring.
* Enabled dashboards to visualize application health, pod metrics, CPU/memory usage, and request rates.

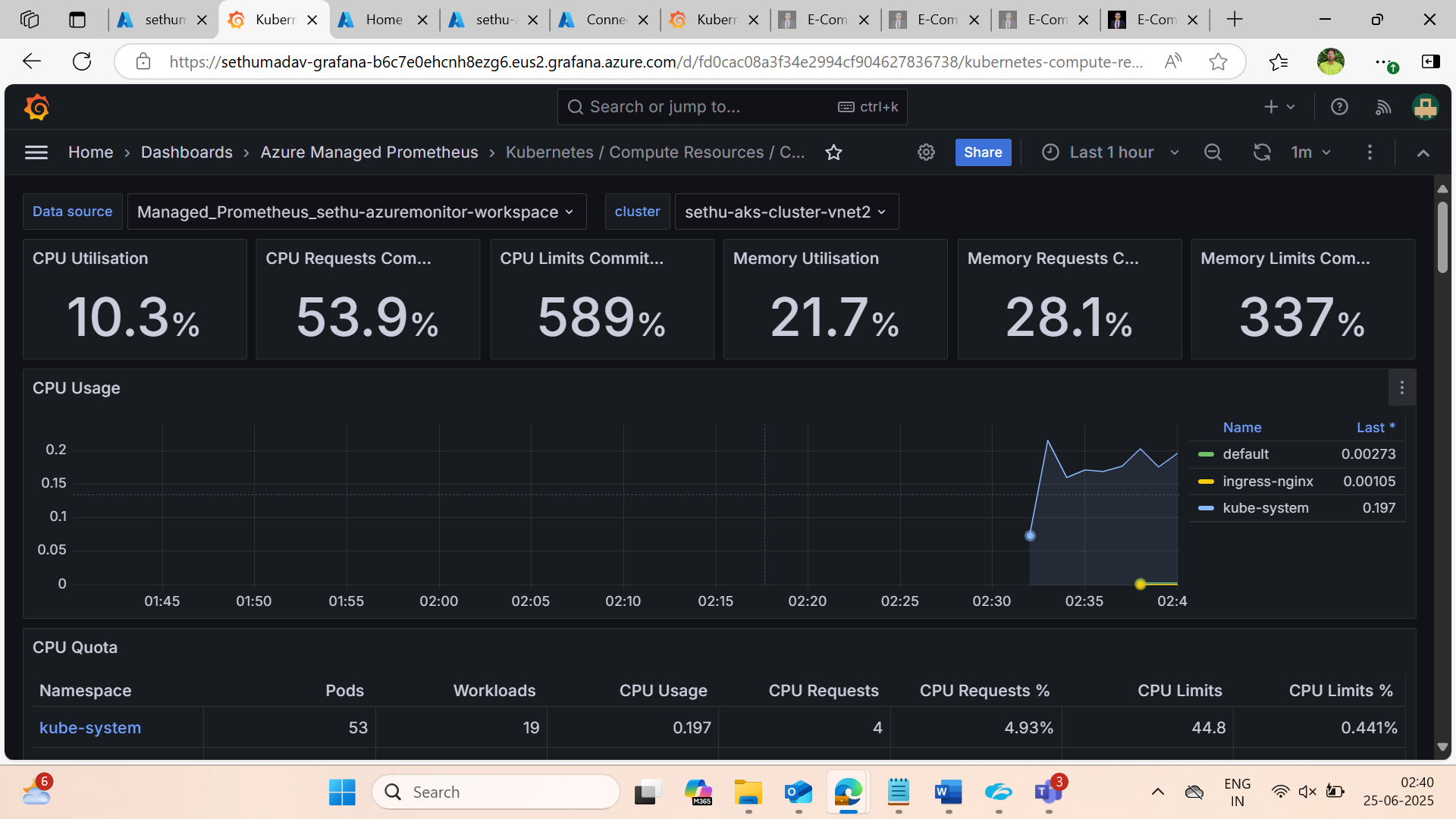
**Screenshots :**

* Log Analytics workspace with log queries.



* Azure Monitor Insights for AKS.





**Conclusion**

The project demonstrates the real-world application of DevOps best practices including infrastructure provisioning, CI/CD pipelines, code quality checks, security scanning, high availability, and monitoring on the Azure cloud platform.

This setup ensures:

* **Resilience** with multi-region AKS clusters and Traffic Manager
* **Security** via secret scanning and image vulnerability checks
* **Scalability** with containerized architecture and Kubernetes
* **Visibility** through centralized monitoring and logging