Azure DevOps Capstone Project

Product Management System

Abstract

This project delivers a multi-region, production-grade infrastructure on Microsoft Azure with high availability, disaster recovery (DR), and robust CI/CD practices using Infrastructure-as-Code (IaC).

Pratheek U B

289226

Pratheek U B
289226

Agenda

---

1. Project Introduction & Architecture

- Goals: High Availability, Disaster Recovery (DR), Scalability
- Multi-Region Strategy (Central Australia & Japan West)
- Traffic Manager for Global Routing

2. Infrastructure Provisioning (IaC)

- Terraform for Region 1
- Bicep for Region 2

3. Core Azure Resources

- AKS Cluster Setup with NGINX / Web App Routing
- Failover Group (FOG)
- Azure Container Registry (ACR)
- Azure Key Vault for Secret Management

4. CI/CD Automation (Azure DevOps)

- Trigger from GitHub
- Build: Frontend (React) & Backend (Spring Boot)
- Security Scans: SonarCloud, Snyk, Trivy
- Docker Build & Push to ACR
- Deployment to Primary & DR AKS Clusters

5. Disaster Recovery & Failover

- DR Region Setup via Bicep
- Traffic Manager Routing Policies
- Simultaneous Deployment to Both Clusters

6. Monitoring & Observability

- Prometheus for Metrics
- Grafana Dashboards (Node, Pod, API Server)

7. Security & Compliance

- NSGs
- Trivy, Snyk for vulnerability scanning tools
- Key Vault + Secret Rotation

8. Site Reliability Engineering (SRE) Practices

- SLA, SLO, SLI

Pratheek U B
289226

Project Overview

This project delivers a multi-region, production-grade infrastructure on Microsoft Azure with high availability, disaster recovery (DR), and robust CI/CD practices using Infrastructure-as-Code (IaC).

Key Features:

Global Traffic Management: Azure Traffic Manager routes user requests to the most optimal regional deployment.

Region 1 (Central Australia): Provisioned via Terraform.

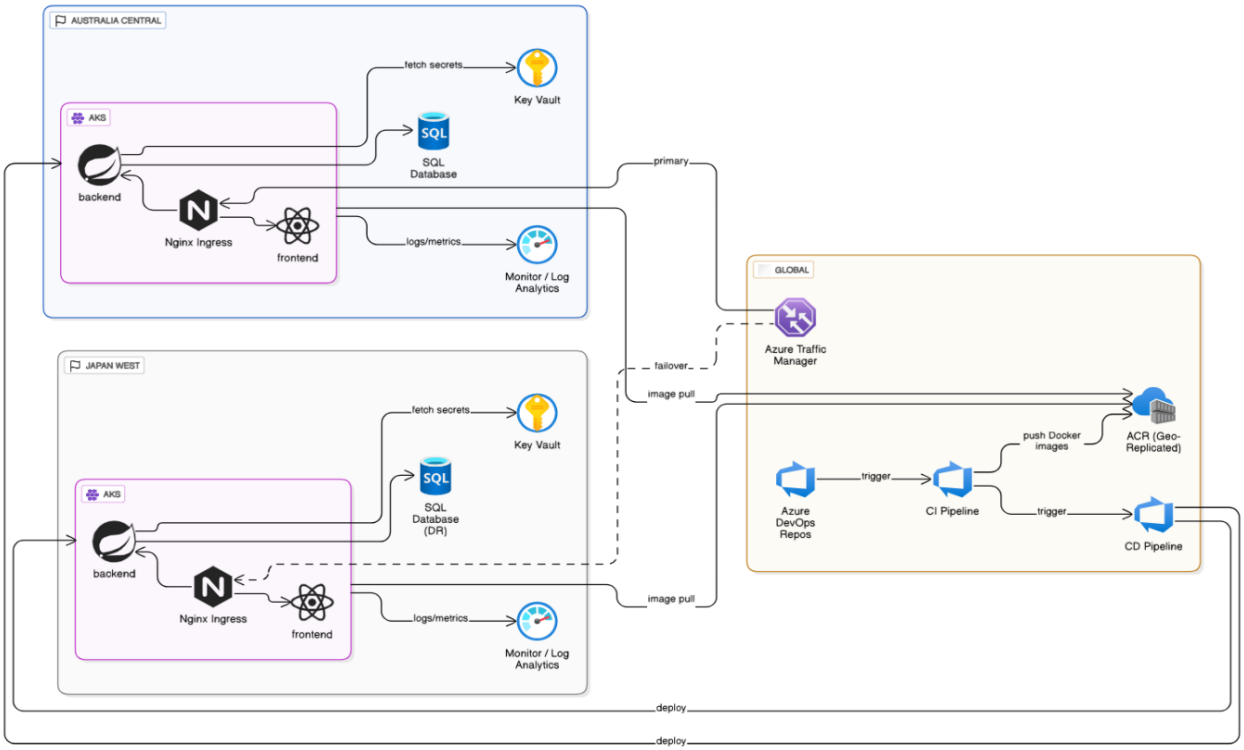
Region 2 (Japan West).

Kubernetes Deployments: Frontend and backend apps deployed using AKS with different ingress strategies.

CI/CD Pipelines: Azure DevOps automates provisioning and deployments.

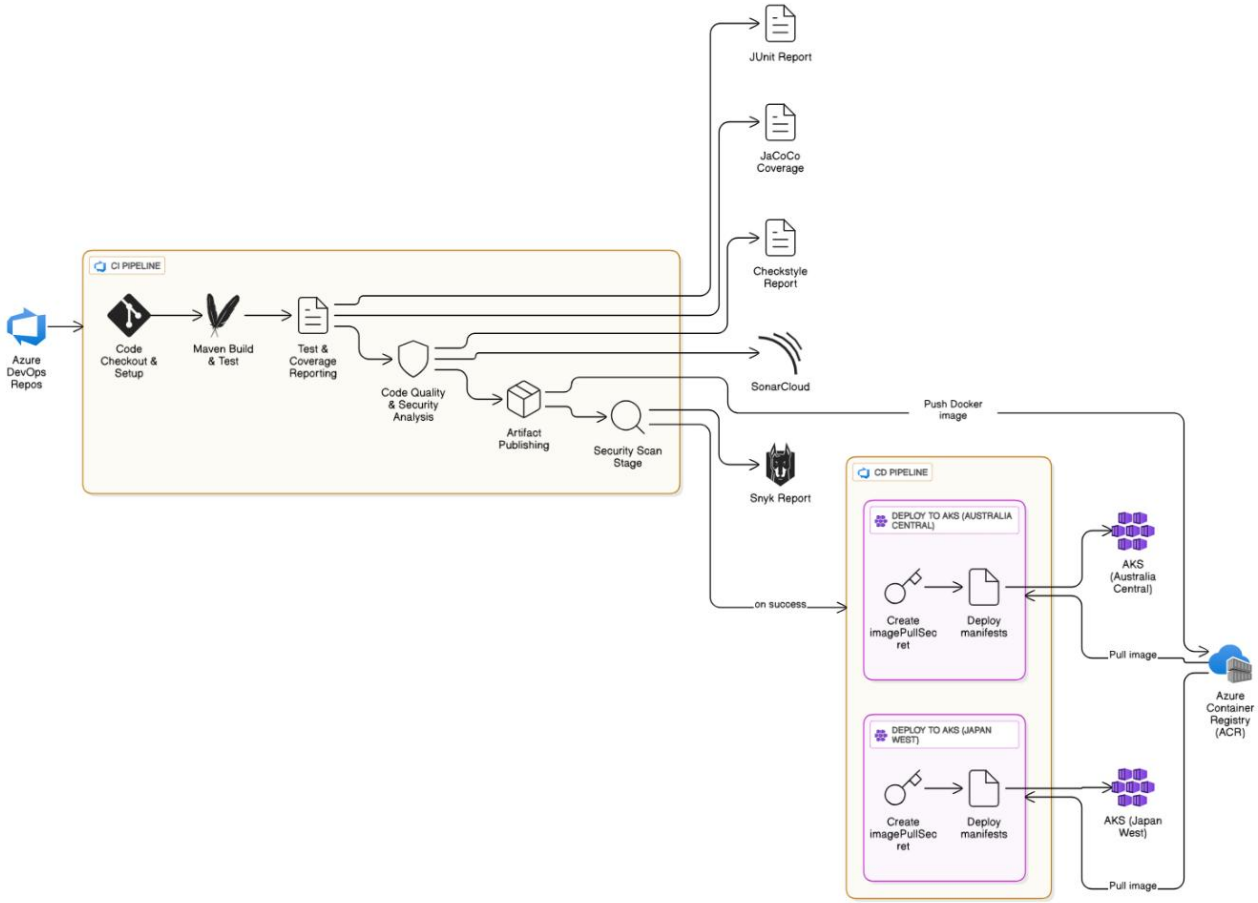
Database Resiliency: Azure SQL Failover Group ensures geo-redundancy.

Architecture:



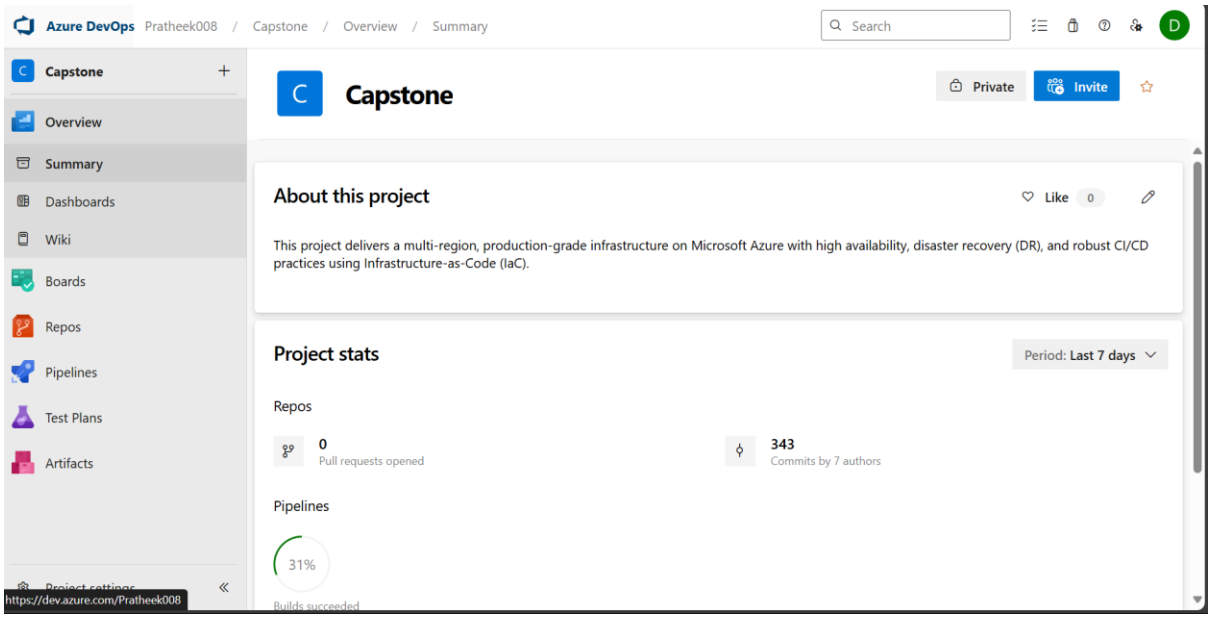
Pratheek U B
289226

CI/CD:



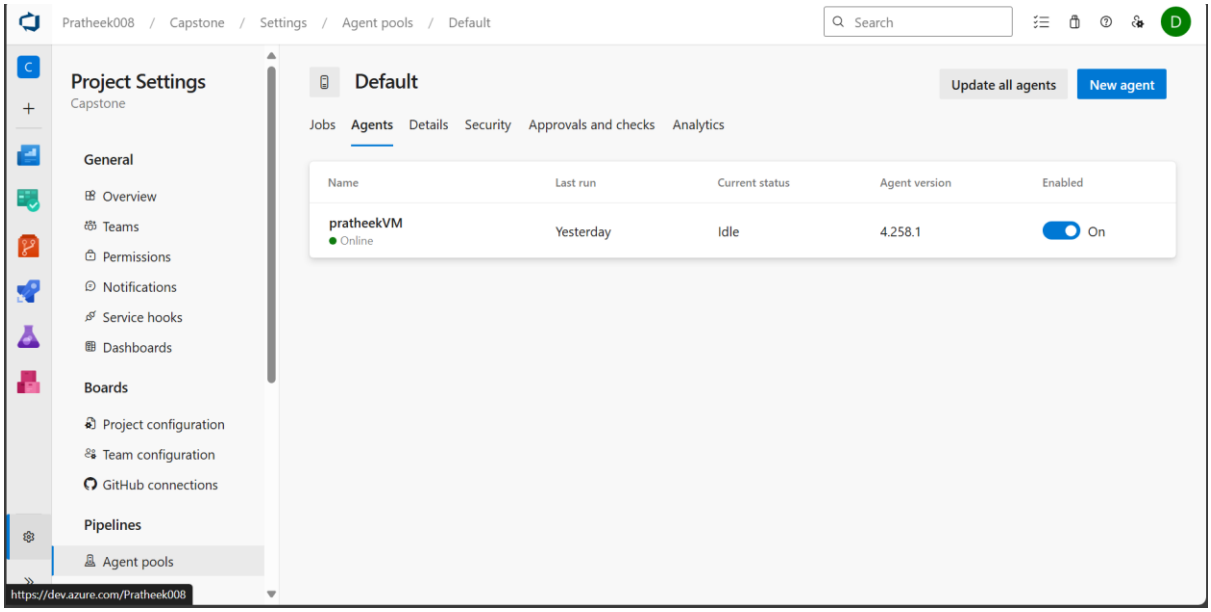
Pratheek U B
289226

1. Created a project in azure devops -

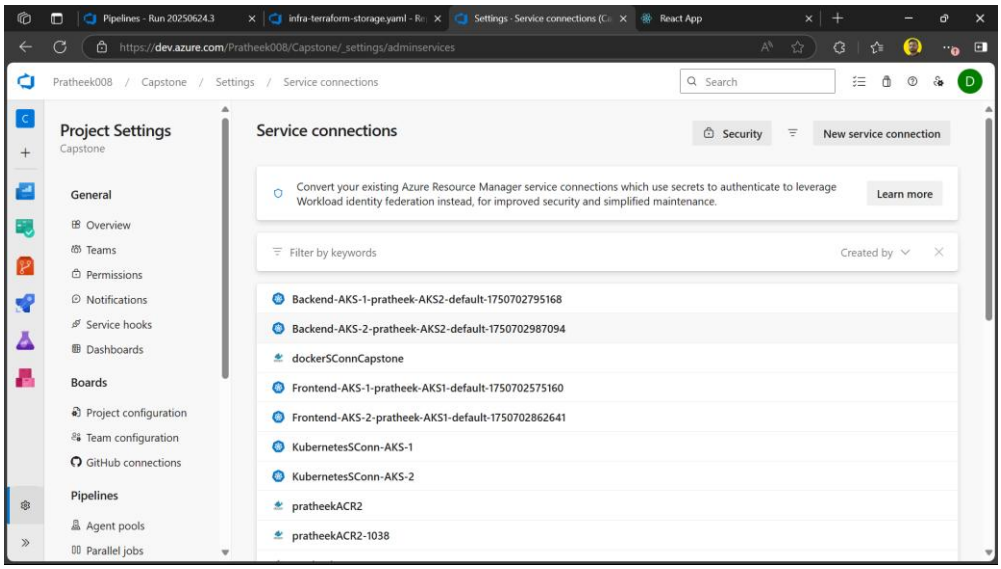


2. created an agent pool

a. projectsettings-> agent pools -> add pool

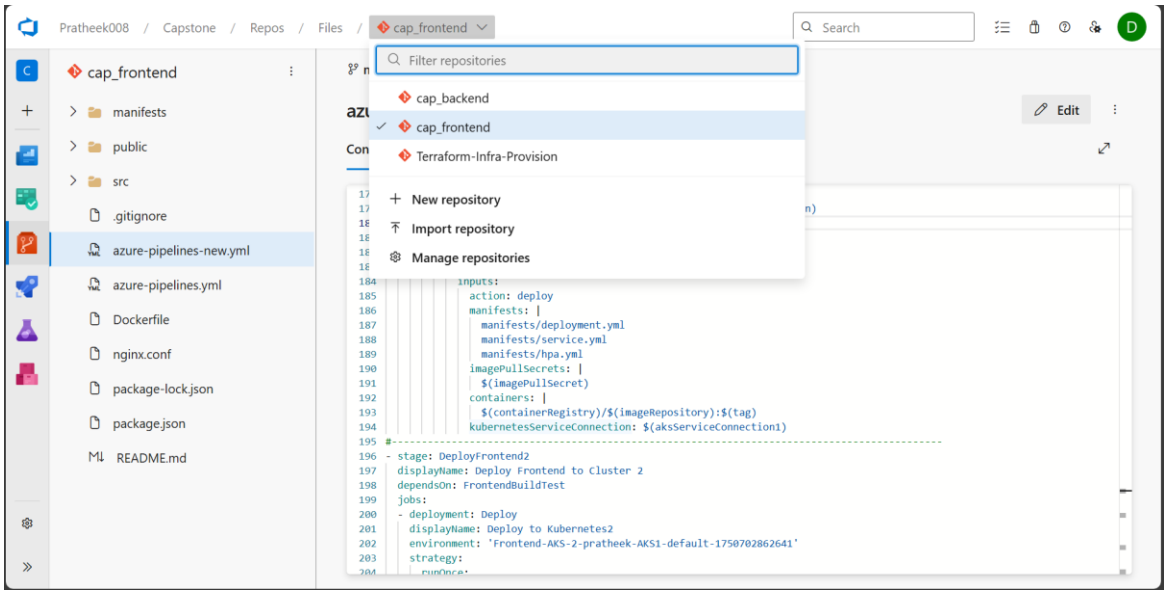


3. **Service connections:**
   a. **Azure connection**
   b. **Docker registry connection (ACR)**
   c. **Kubernetes registry connection**
   d. **Sonarqube connection**



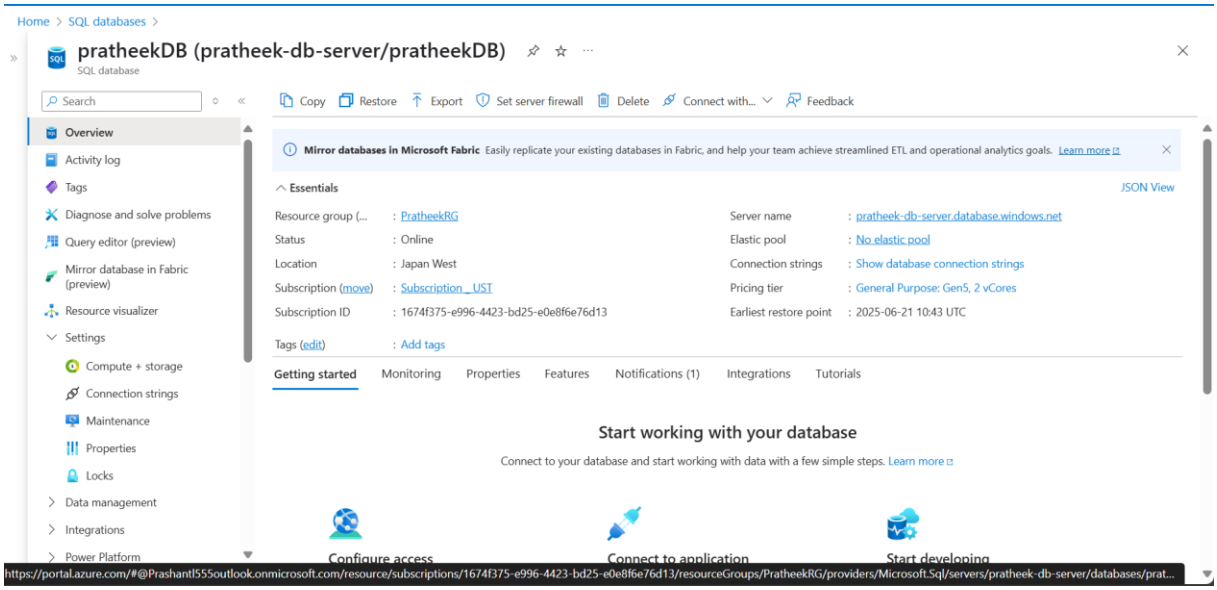
---

4. **Different repository for each services in azure devops**
   - **Cap_frontend**
   - **Cap_backend**
   - **Terraform-infra-provision**



---

5. **Created the database:**



---

6. My infrastructure:

Region 1: Central Australia (Provisioned via Terraform)

Terraform Backend Configuration

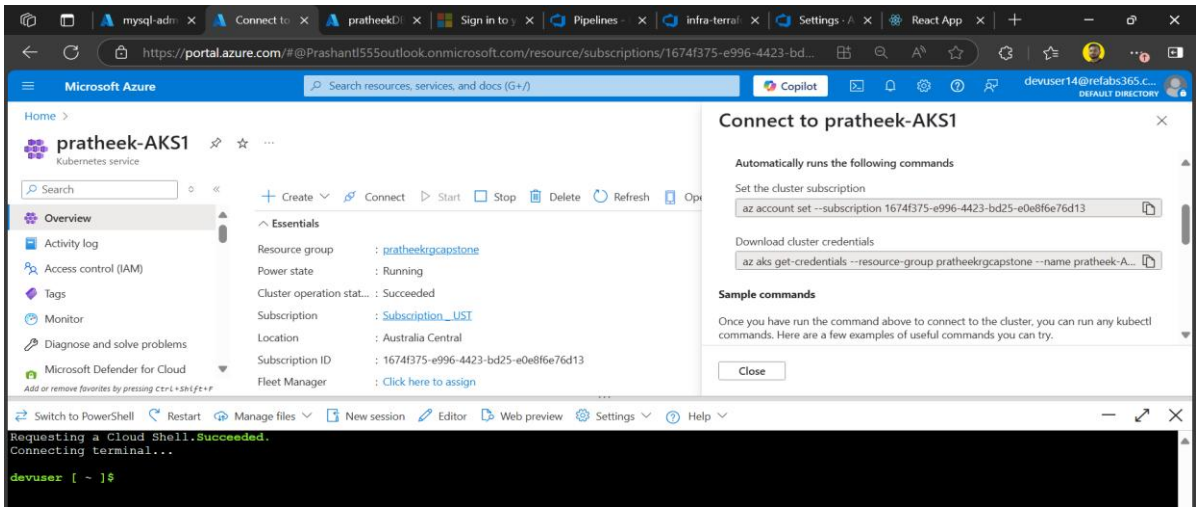
infra-terraform-storage.yaml

Contents History Compare Blame

```yaml
trigger: none  # Run manually only

variables:
  azureServiceConnection: 'pratheekSConnNew'

stages:
  - stage: CreateTerraformBackend
    displayName: "Create Terraform Backend Resources"
    jobs:
      - job: CreateBackend
        displayName: "Provision Resource Group, Storage Account, and Container"
        pool:
          name: Default
        steps:

          - task: AzureCLI@2
            displayName: "Create Resource Group"
            inputs:
              azureSubscription: $(azureServiceConnection)
              scriptType: 'bash'
              scriptLocation: 'inlineScript'
              inlineScript: |
                az group create \
                  --name pratheekSTORAGE-rg \
                  --location japanwest

          - task: AzureCLI@2
            displayName: "Create Storage Account"
            inputs:
              azureSubscription: $(azureServiceConnection)
              scriptType: 'bash'
              scriptLocation: 'inlineScript'
              inlineScript: |
                az storage account create \
                  --name pratheekstorage0008 \
                  --resource-group pratheekSTORAGE-rg \
                  --location japanwest \
                  --sku Standard_LRS

          - task: AzureCLI@2
            name: GetKey
            displayName: "Get Storage Account Key"
            inputs:
              azureSubscription: $(azureServiceConnection)
              scriptType: 'bash'
              scriptLocation: 'inlineScript'
              inlineScript: |
                ACCOUNT_KEY=$(az storage account keys list \
                  --resource-group pratheekSTORAGE-rg \
                  --account-name pratheekstorage0008 \
                  --query '[0].value' \
                  --output tsv)
                echo "##vso[task.setvariable variable=ACCOUNT_KEY;issecret=true]$ACCOUNT_KEY"

          - task: AzureCLI@2
            displayName: "Create Blob Container for Terraform State"
            inputs:
              azureSubscription: $(azureServiceConnection)
              scriptType: 'bash'
              scriptLocation: 'inlineScript'
              inlineScript: |
                az storage container create \
                  --name tfstate \
                  --account-name pratheekstorage0008 \
```

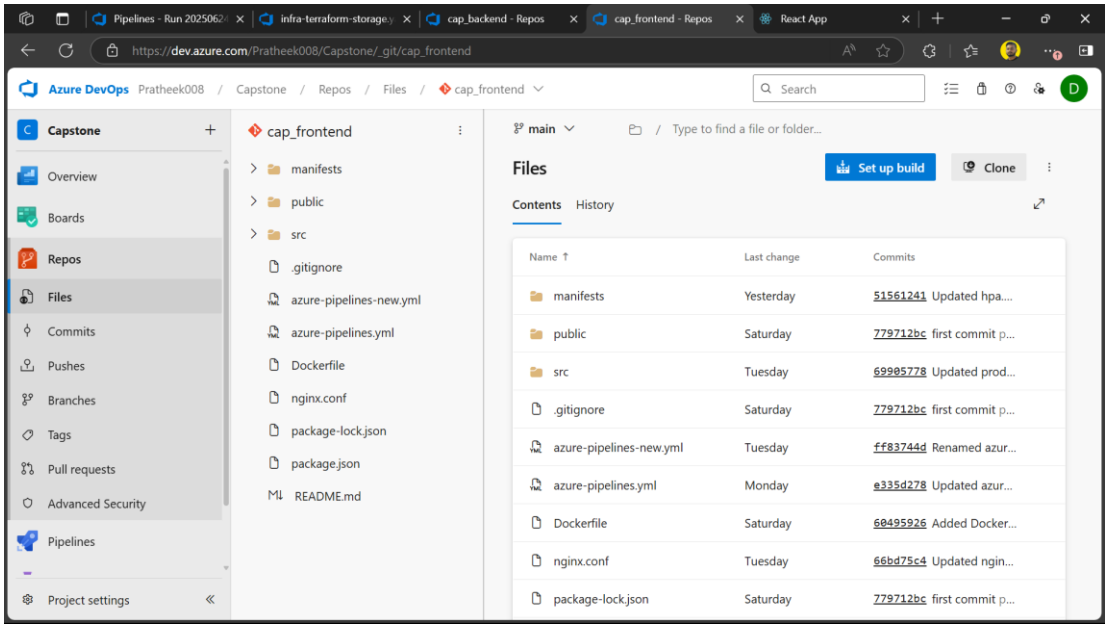
Core Infrastructure Resources

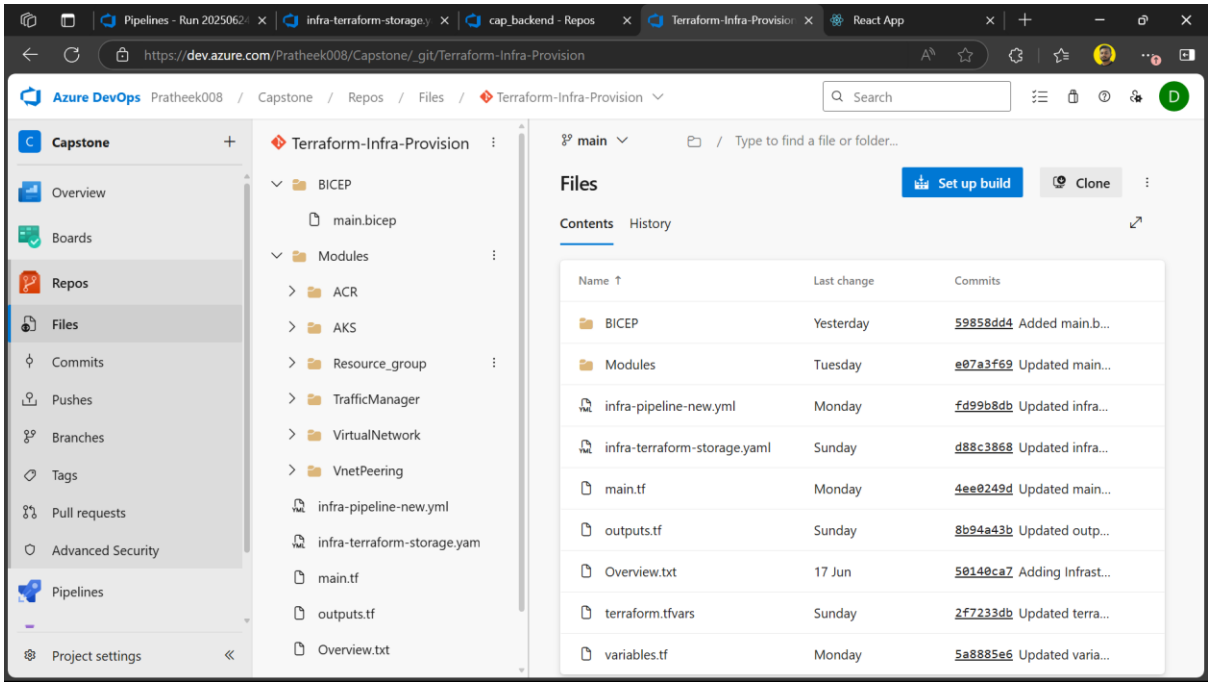
- **Resource Group**: pratheekRGcapstone
- **Virtual Network**: aks-vnet-1
  - **Subnet**: aks-subnet-1 (used by AKS)
- **Azure SQL**
  - **Server**: pratheekDB
  - **Database**: springbootdb
  - **Private Endpoint**: Enabled with Private DNS zone integration
- **AKS Cluster**: pratheekAKS1
  - **Node Pool**: 3 nodes (Standard_D2s_v3)
  - **Network Plugin**: Azure CNI
  - **Subnet Integration**: Connected to aks-subnet-1
- **Ingress Controller**: NGINX
  - Installed via Helm
  - Static Public IP provisioned using Terraform
- **Azure Container Registry (ACR)**: pratheekACRcapstone
  - AcrPull role assigned to AKS Managed Identity (MSI)
- **Azure Traffic Manager**
  - **Profile Name**: pratheek-tm
  - **Routing Method**: Priority
  - **Endpoint**: Region 1 Ingress Public IP



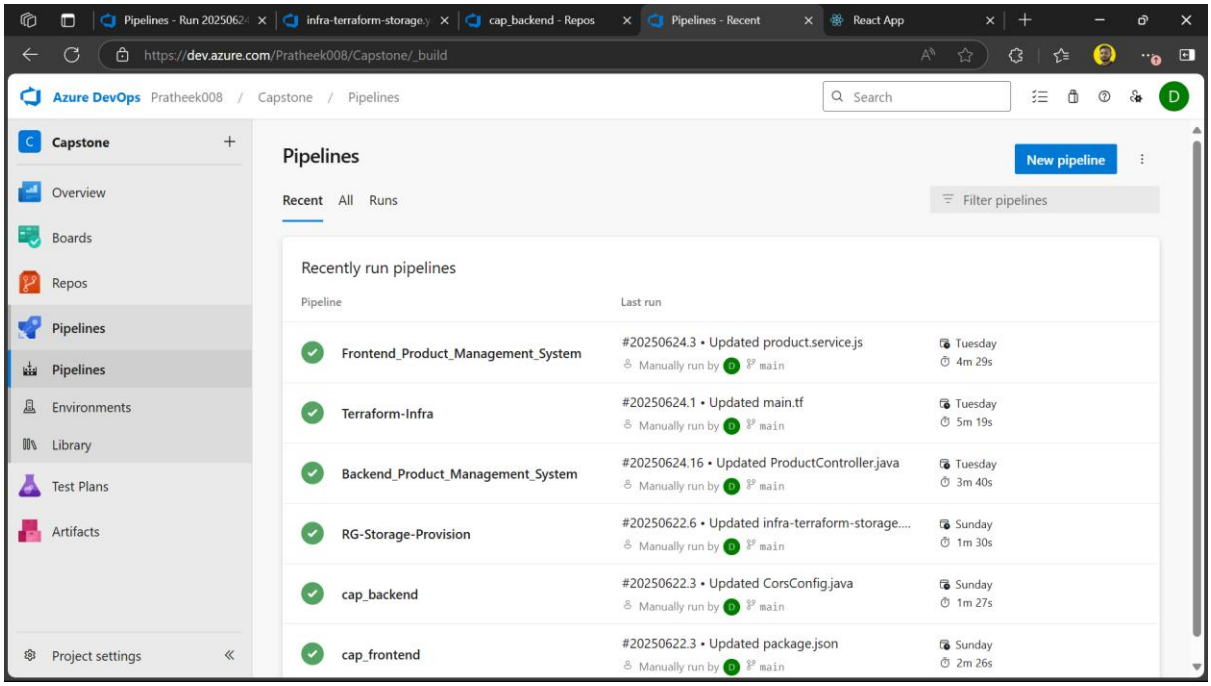
Region 2: Japan West (Provisioned via Azure Bicep)

- **AKS Cluster**
  - **Name**: pratheekAKS2
  - **Region**: japanwest
  - **VNet Integration**: Connected to pratheek-vnet1
  - **RBAC**: Enabled
  - **Identity**: System-assigned managed identity
  - **Agent Pool**: systempool with 3 nodes
- **Azure SQL Database with Geo-Redundancy**
  - **Primary Region (Central Australia)**
    - SQL Server and Database
    - Private Endpoint configured
  - **Secondary Region (Japan West)**
    - Geo-replica (read-only)
    - **Failover Group (FOG)**: Provides automatic regional failover
    - Uses **public endpoint DNS** for read access




Pipelines - Run 20250624
infra-terraform-storage.y
cap_backend - Repos
Terraform-Infra-Provision
React App
https://dev.azure.com/Pratheek008/Capstone/_git/Terraform-Infra-Provision
Azure DevOps Pratheek008 / Capstone / Repos / Files / Terraform-Infra-Provision
Search
Capstone
Overview
Boards
Repos
Files
Commits
Pushes
Branches
Tags
Pull requests
Advanced Security
Pipelines
Project settings
Terraform-Infra-Provision
BICEP
main.bicep
Modules
ACR
AKS
Resource_group
TrafficManager
VirtualNetwork
VnetPeering
infra-pipeline-new.yml
infra-terraform-storage.yam
main.tf
outputs.tf
Overview.txt
main
Type to find a file or folder...
Files
Set up build
Clone
Contents
History
Name
Last change
Commits
BICEP Yesterday 59858dd4 Added main.b...
Modules Tuesday e07a3f69 Updated main...
infra-pipeline-new.yml Monday fd99b8db Updated infra...
infra-terraform-storage.yaml Sunday d88c3868 Updated infra...
main.tf Monday 4ee0249d Updated main...
outputs.tf Sunday 8b94a43b Updated outp...
Overview.txt 17 Jun 50140ca7 Adding Infrast...
terraform.tfvars Sunday 2f7233db Updated terra...
variables.tf Monday 5a8885e6 Updated varia...

7. Pipelines


Pipelines - Run 20250624
infra-terraform-storage.y
cap_backend - Repos
Pipelines - Recent
React App
https://dev.azure.com/Pratheek008/Capstone/_build
Azure DevOps Pratheek008 / Capstone / Pipelines
Search
Capstone
Overview
Boards
Repos
Pipelines
Pipelines
Environments
Library
Test Plans
Artifacts
Project settings
Pipelines
New pipeline
Recent
All
Runs
Filter pipelines
Recently run pipelines
Pipeline
Last run
Frontend_Product_Management_System #20250624.3 • Updated product.service.js Manually run by main Tuesday 4m 29s
Terraform-Infra #20250624.1 • Updated main.tf Manually run by main Tuesday 5m 19s
Backend_Product_Management_System #20250624.16 • Updated ProductController.java Manually run by main Tuesday 3m 40s
RG-Storage-Provision #20250622.6 • Updated infra-terraform-storage.... Manually run by main Sunday 1m 30s
cap_backend #20250622.3 • Updated CorsConfig.java Manually run by main Sunday 1m 27s
cap_frontend #20250622.3 • Updated package.json Manually run by main Sunday 2m 26s

Pratheek U B
289226

8. Azure DevOps CI/CD Pipeline

Trigger

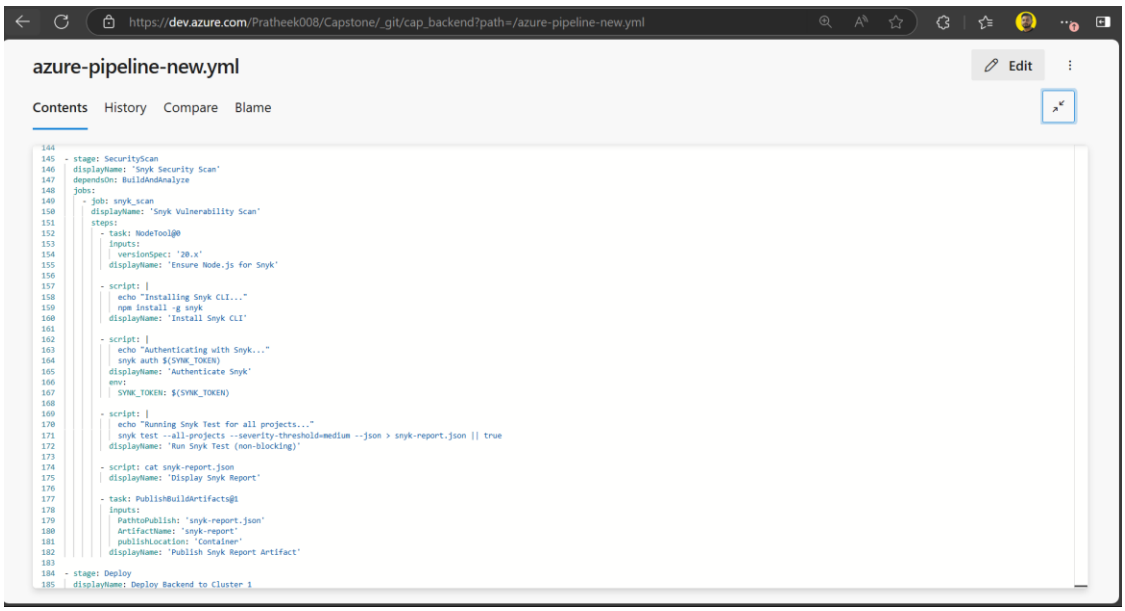
- Push to main branch on GitHub

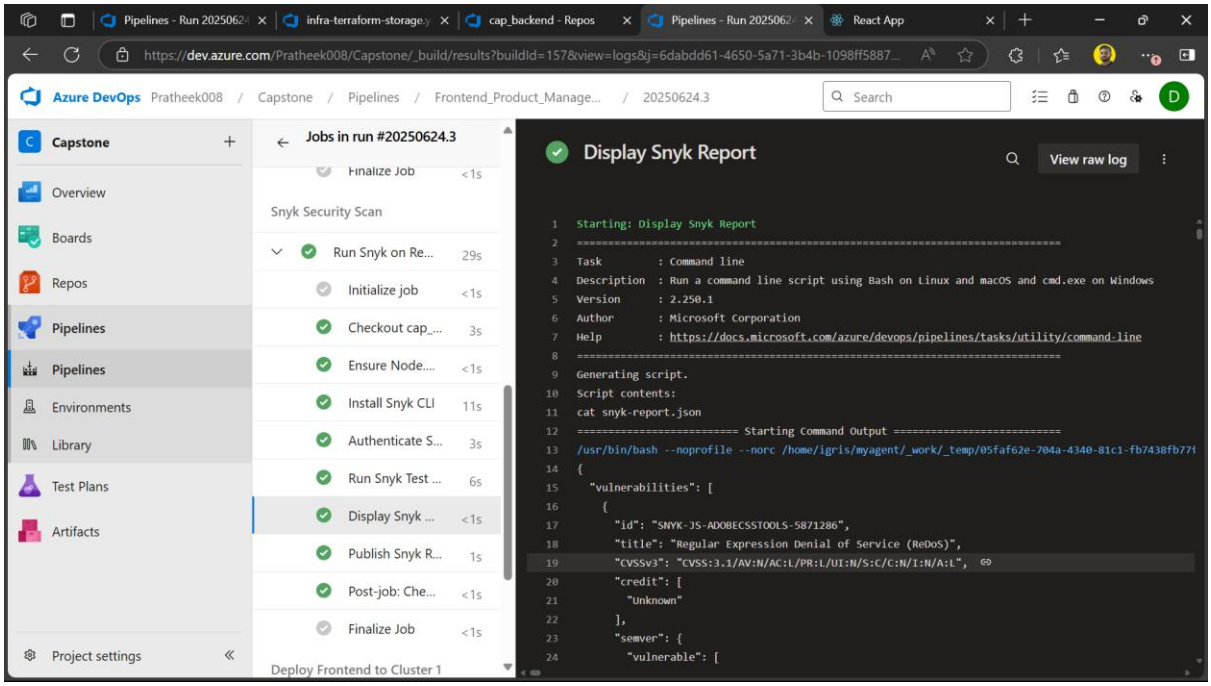
Setup

- Install Node.js and dependencies for React frontend

Testing & Code Analysis

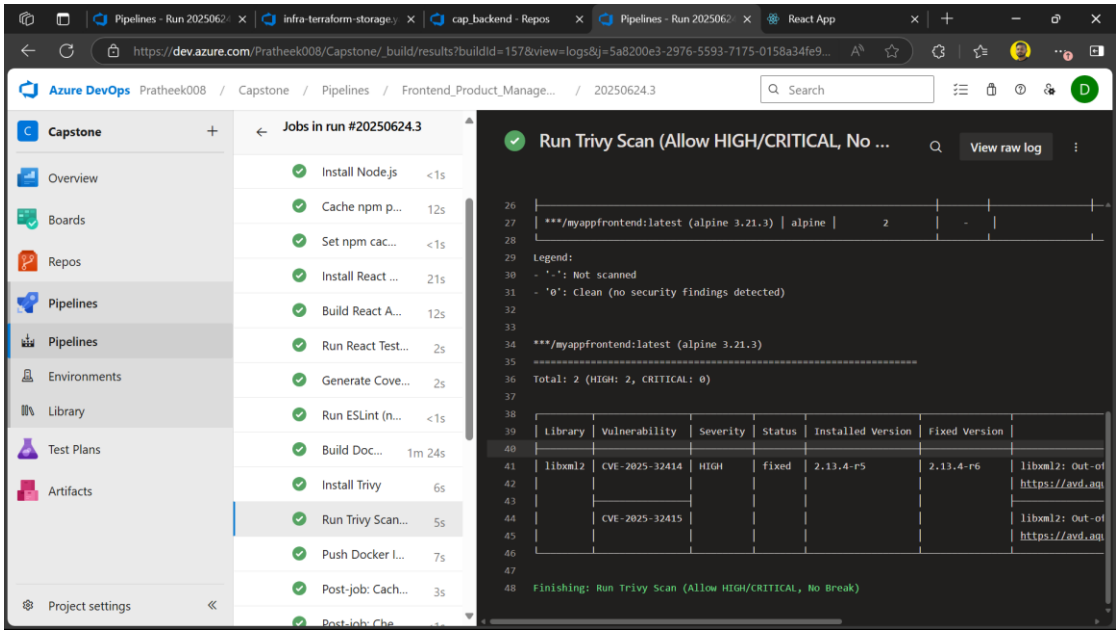
- **SonarCloud**: For static code analysis
- **Snyk**: For scanning NPM vulnerabilities





Build Stage

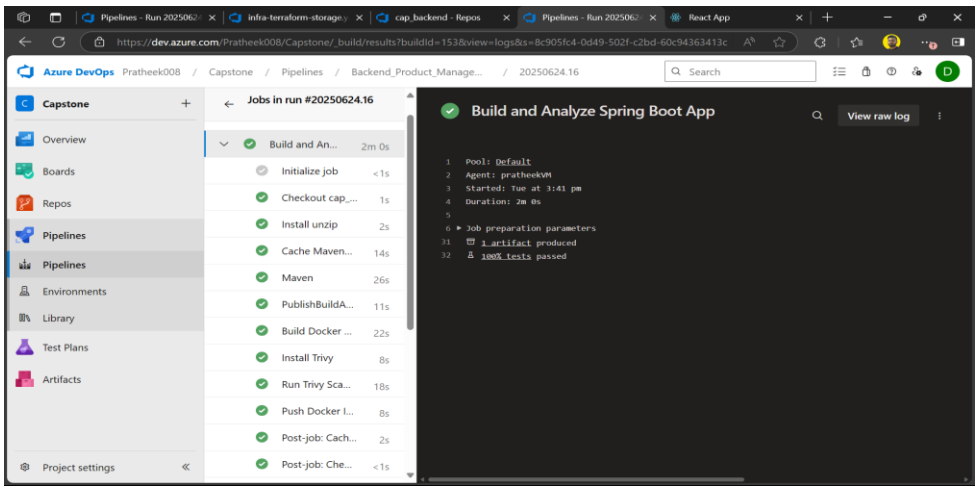
- Compile Spring Boot backend using Maven
- Run unit tests and calculate code coverage
- Perform **Snyk** and **Trivy** scans for vulnerabilities
- Build Docker images and push to **ACR**

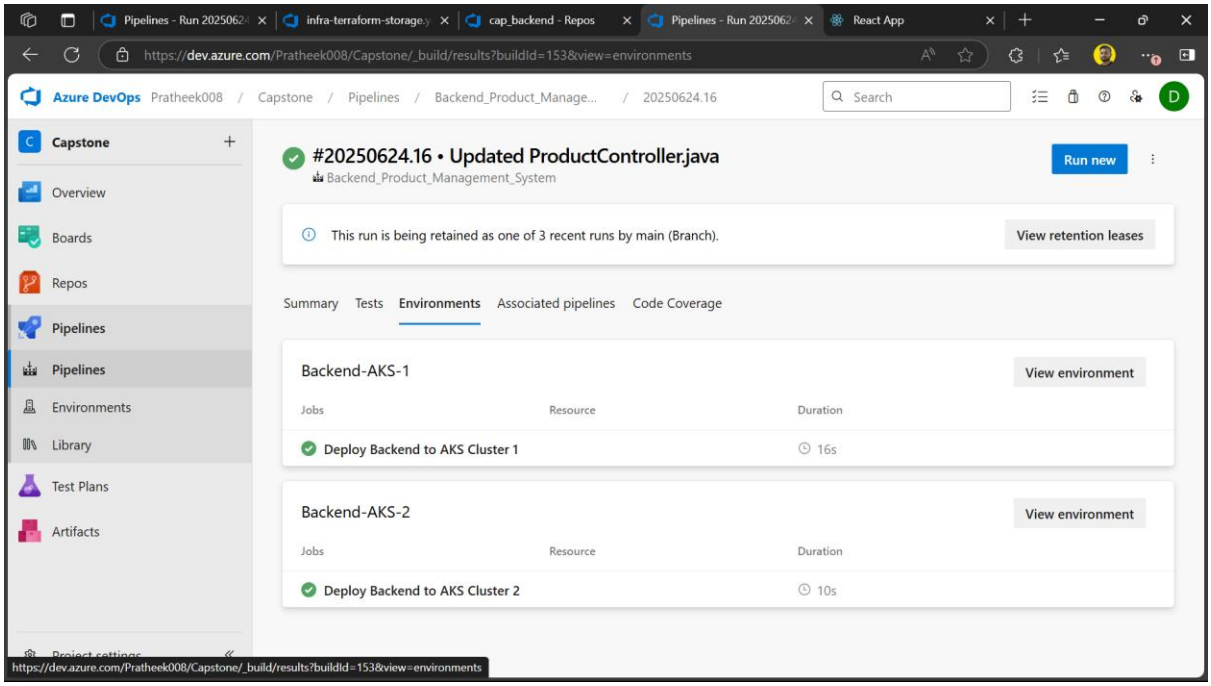


9. Deployment

Primary Region (pratheekAKS1)

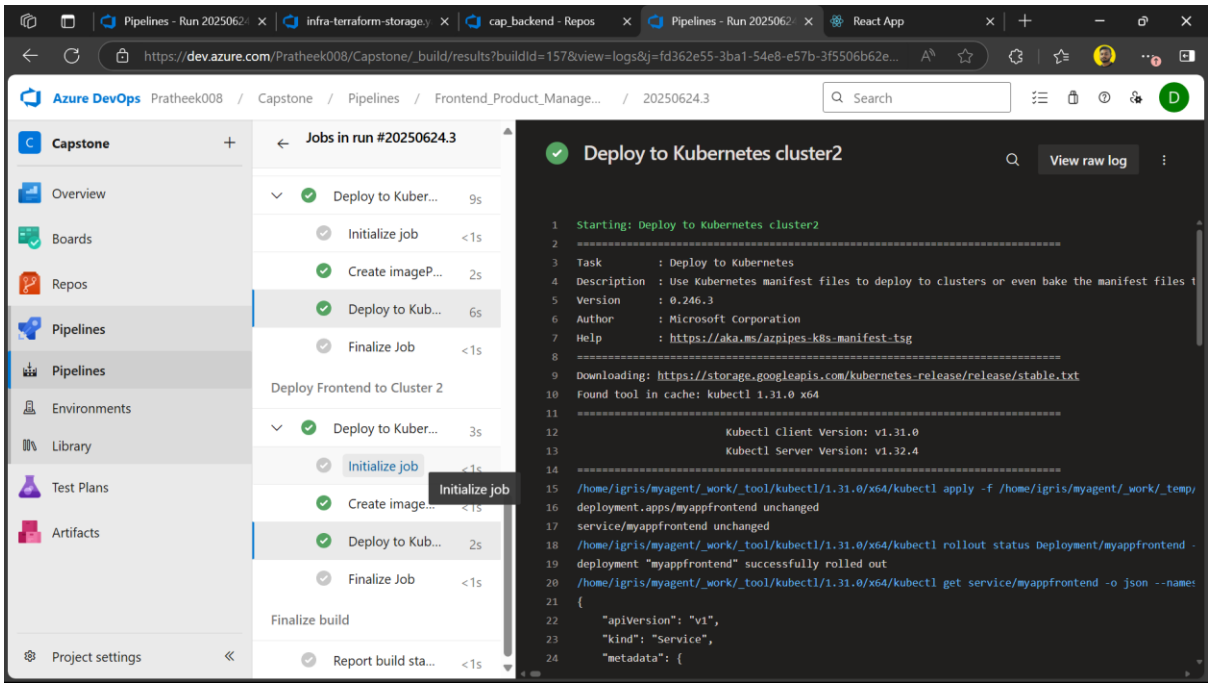
- Retrieve SQL credentials from **Azure Key Vault**
- Create Kubernetes secrets for sensitive data
- Deploy backend and frontend using Helm/manifests



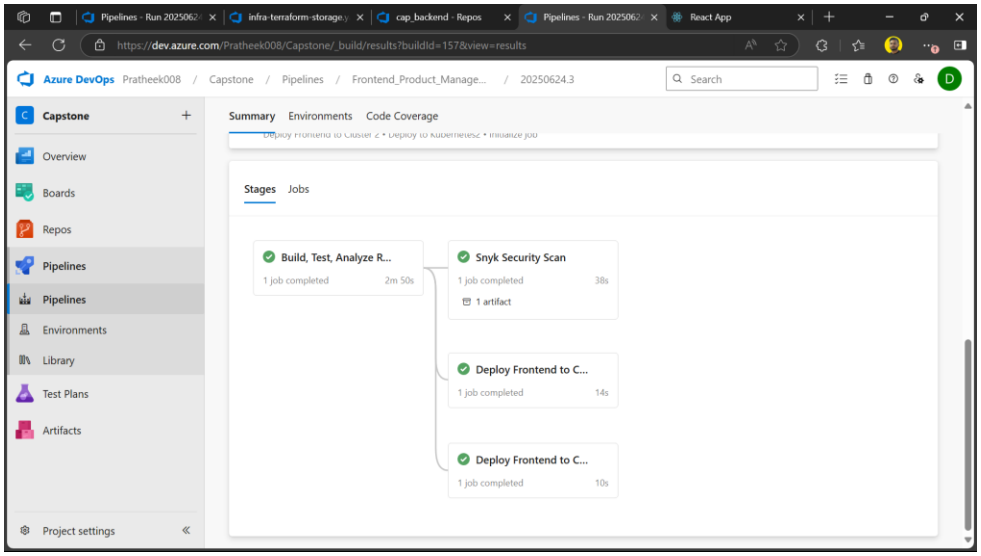


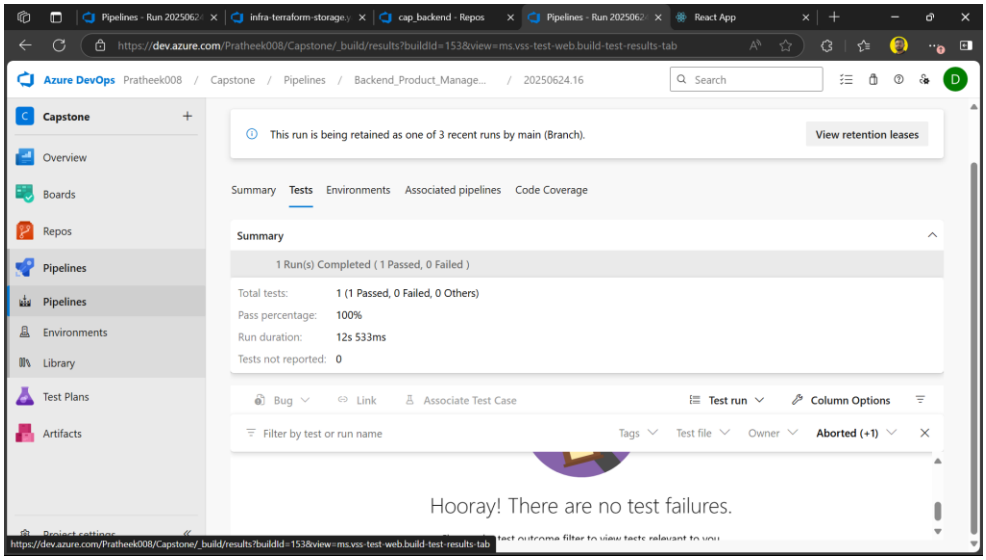
Backup Region (pratheekAKS2)

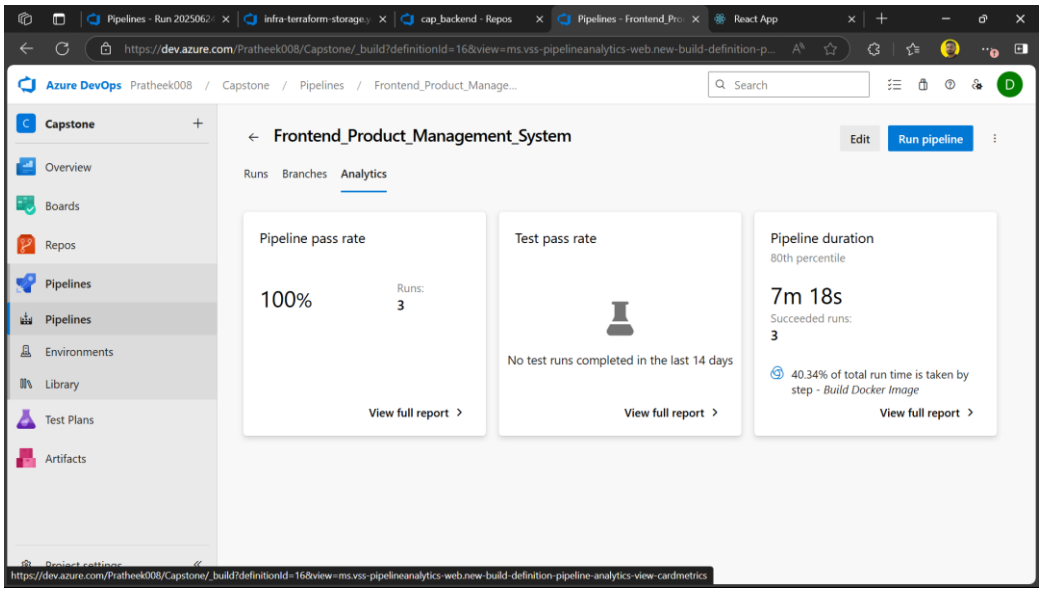
- Use capstone_backup agent pool
- Connect to pratheekAKS2 cluster
- Reuse the same Docker image and manifests for deployment



Pratheek U B
289226







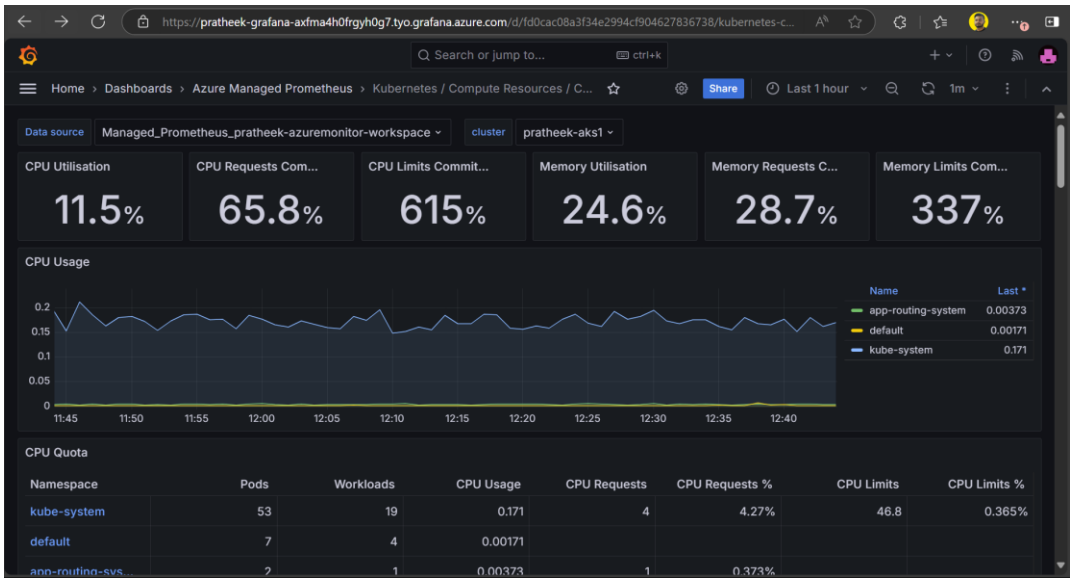
10. Monitoring Setup (Both AKS Clusters)

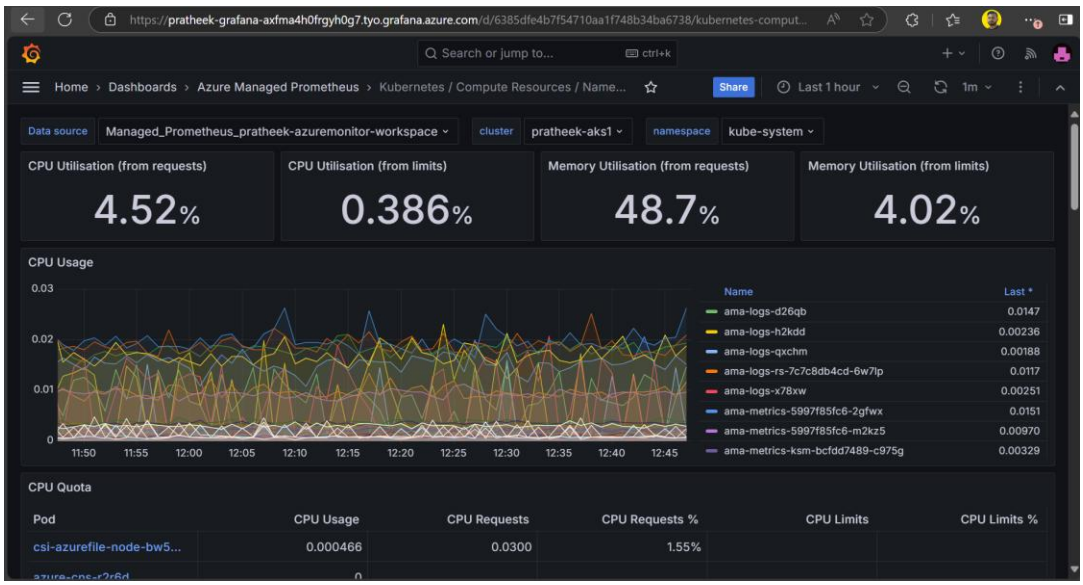
- **Prometheus**: Metrics collection and alerting
- **Grafana**: Visualization dashboards

Configured Grafana for aks1

Steps:

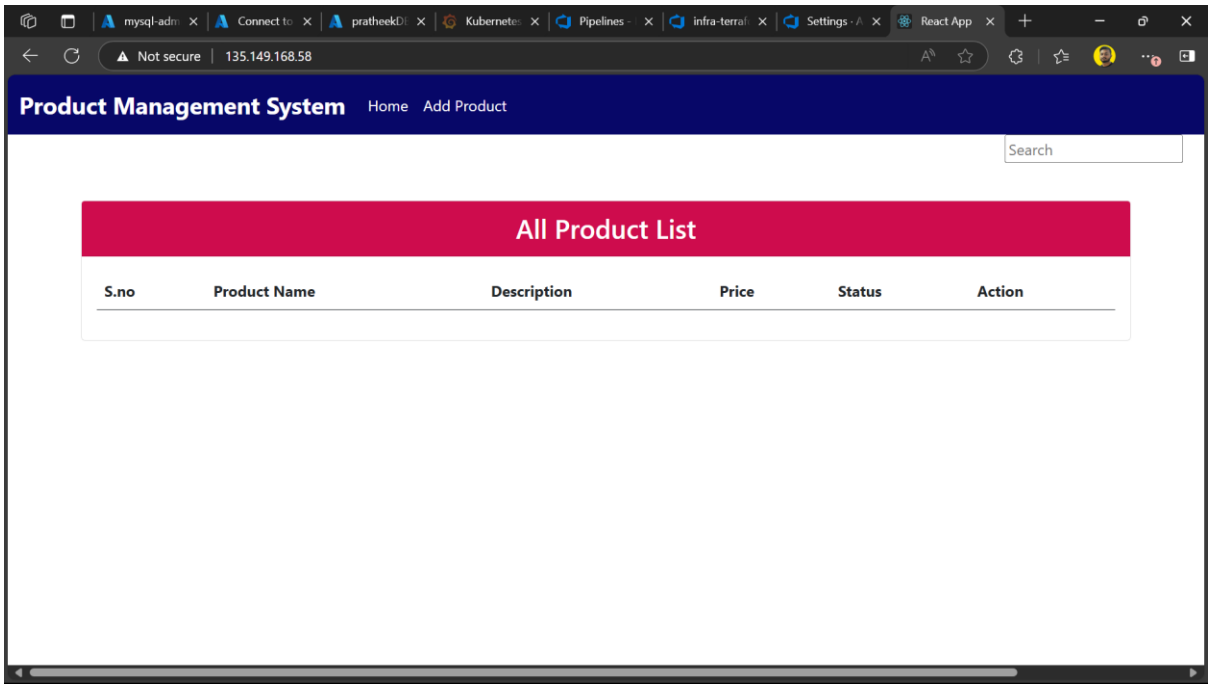
- Azure Managed Grafana
- Azure Monitor workspaces (configure Grafana with workspace)
- Kubernetes services (configure workspace with k8s service)
- Grafana

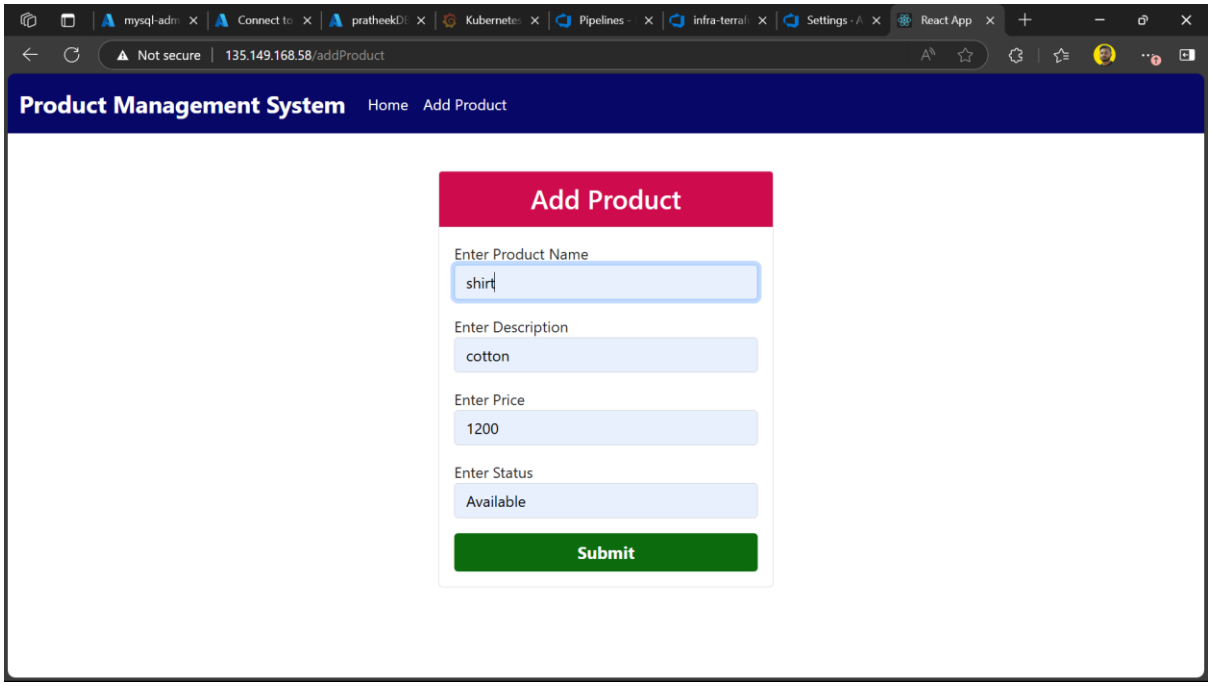




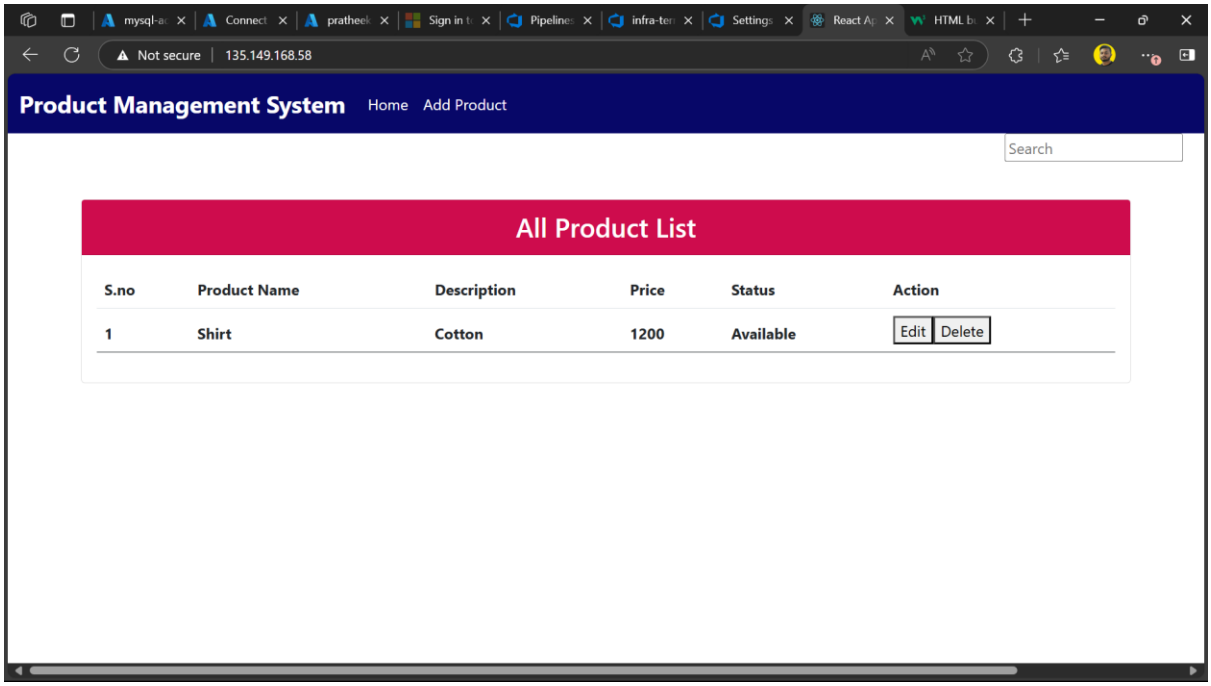
11. Accessing application:

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





Pratheek U B
289226



12. Conclusion

This Capstone project showcases a comprehensive, real-world implementation of DevOps and Cloud Engineering principles on the Azure platform. The solution follows industry best practices across the entire software delivery lifecycle — from infrastructure provisioning to deployment, security, and observability.

Through this project, the following outcomes were achieved:

- **High Availability & Resilience:**
  Multi-region deployment of Azure Kubernetes Service (AKS) clusters with Active-Passive failover using Azure Traffic Manager ensures business continuity and disaster recovery readiness.
- **Scalable & Modular Architecture**:
  Microservices are containerized and deployed on Kubernetes, allowing horizontal scaling and easier maintainability.
- **Automated Infrastructure Provisioning**:
  Terraform is used to define and manage cloud infrastructure as code, enabling repeatable, version-controlled deployments across environments.
- **Robust CI/CD Pipelines:**
  Azure DevOps Pipelines automate build, test, security scanning, and multi-region deployment, enabling faster, safer delivery of application changes.
- **Security & Compliance:**
  Azure Defender, Key Vault, NSGs, and Azure Policy enforce strong security postures, including secret management, image scanning, and port hardening.
- **Comprehensive Monitoring & Logging:**
  Application Insights and Azure Monitor deliver full-stack observability — capturing logs, metrics, dependencies, crash analytics, and SLO breaches.

Github:

pratheek08/Capstone-Frontend

pratheek08/Capstone-Backend

pratheek08/Capstone-Infra