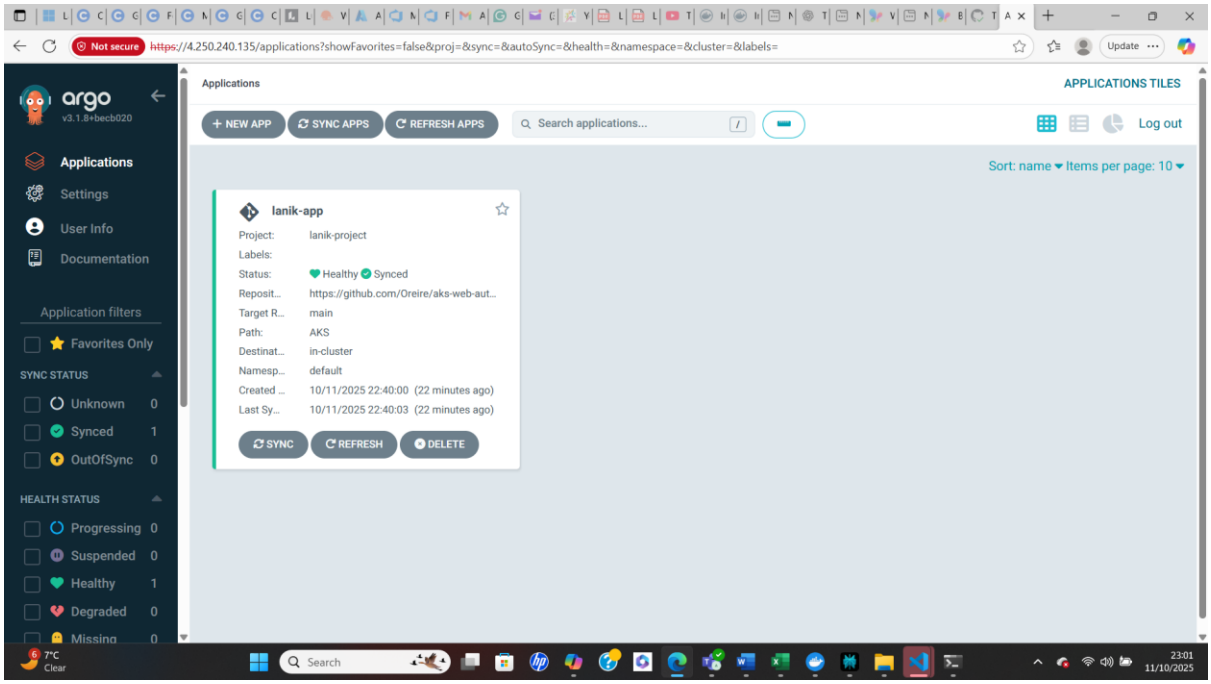
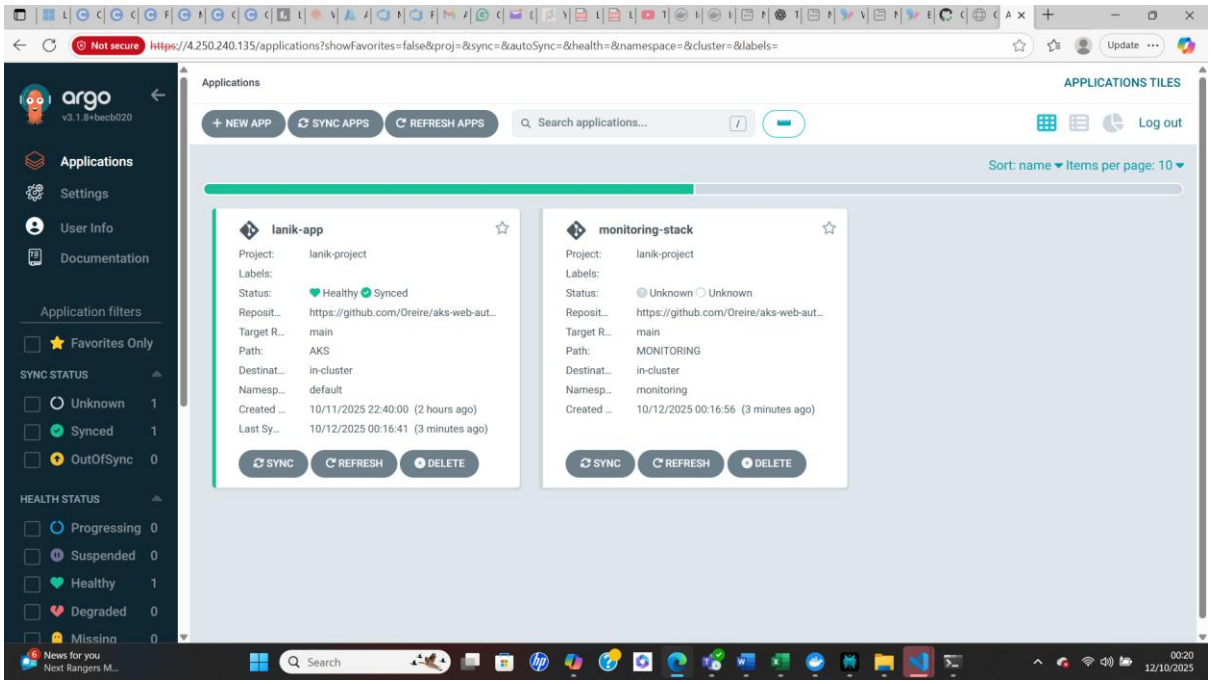
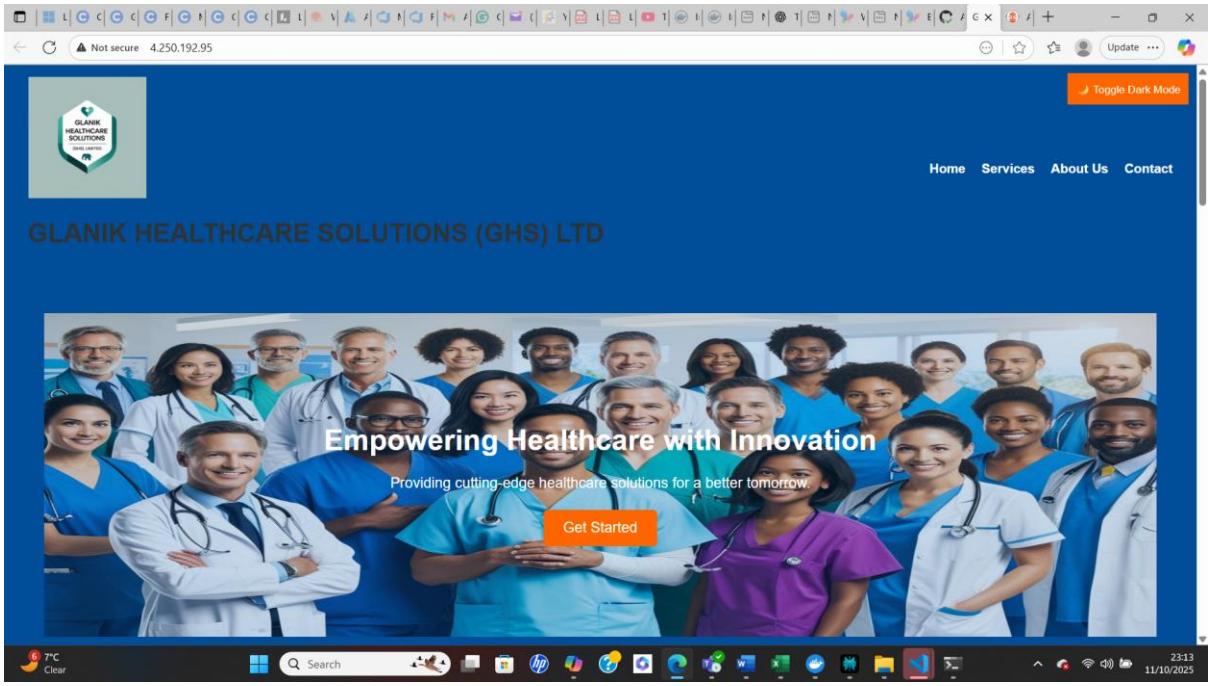
ArgoCD Implementation

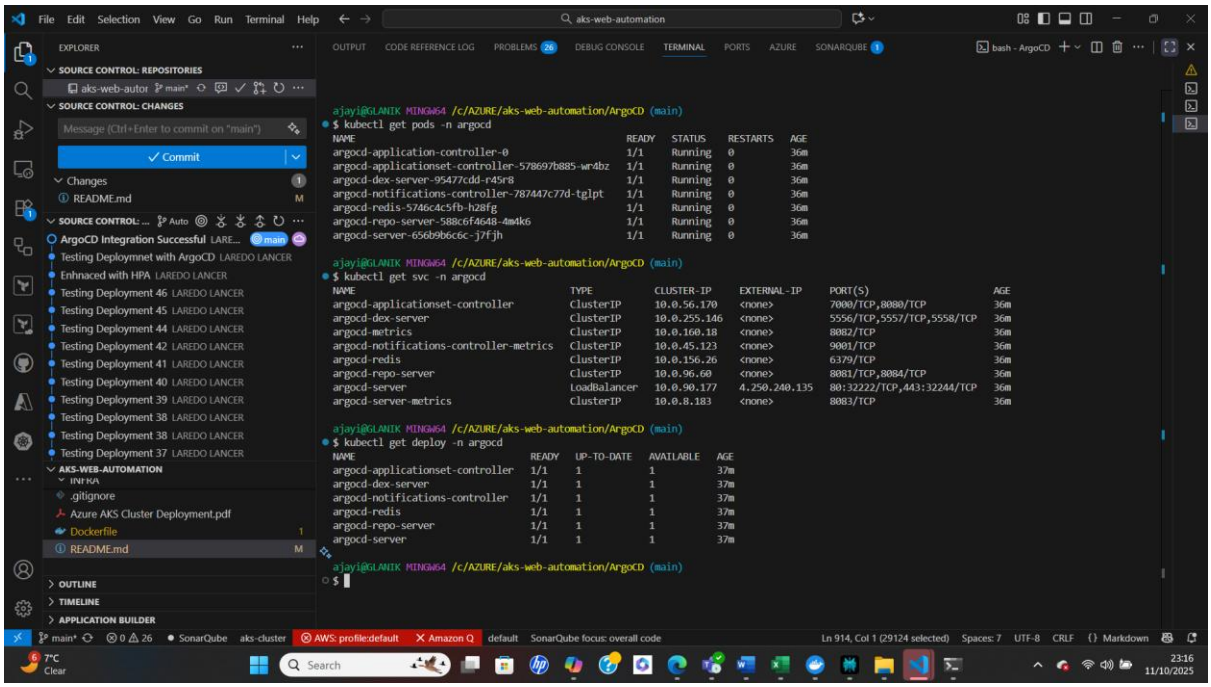




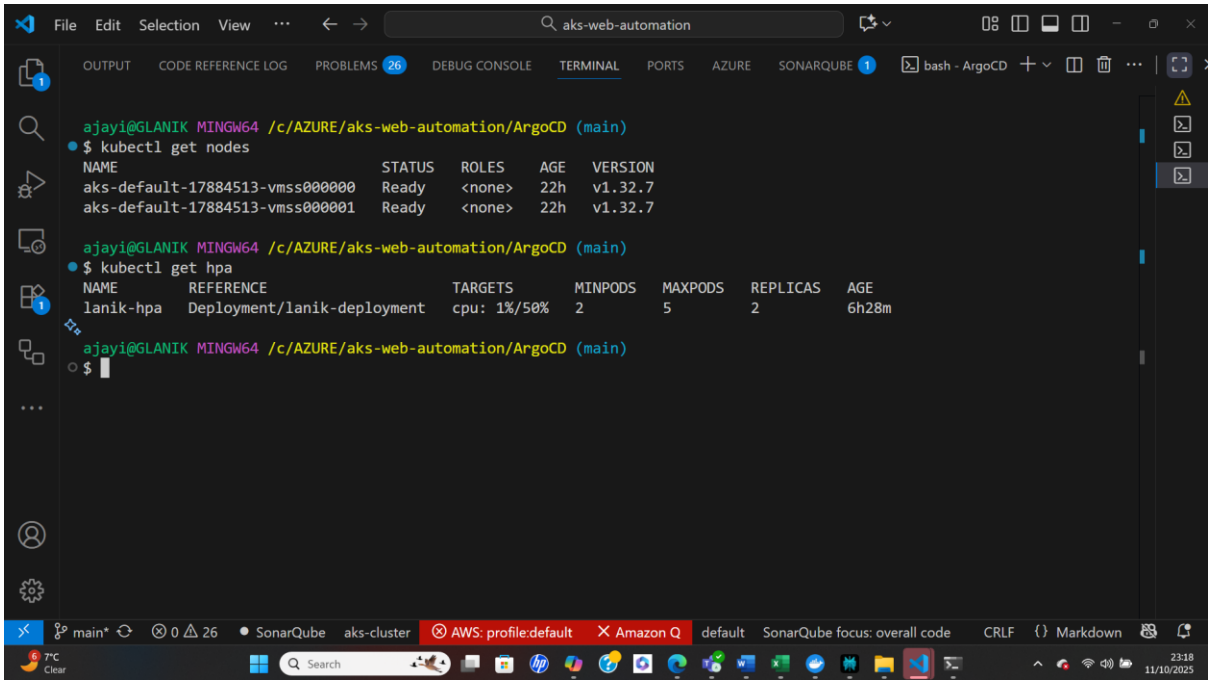
Web Application



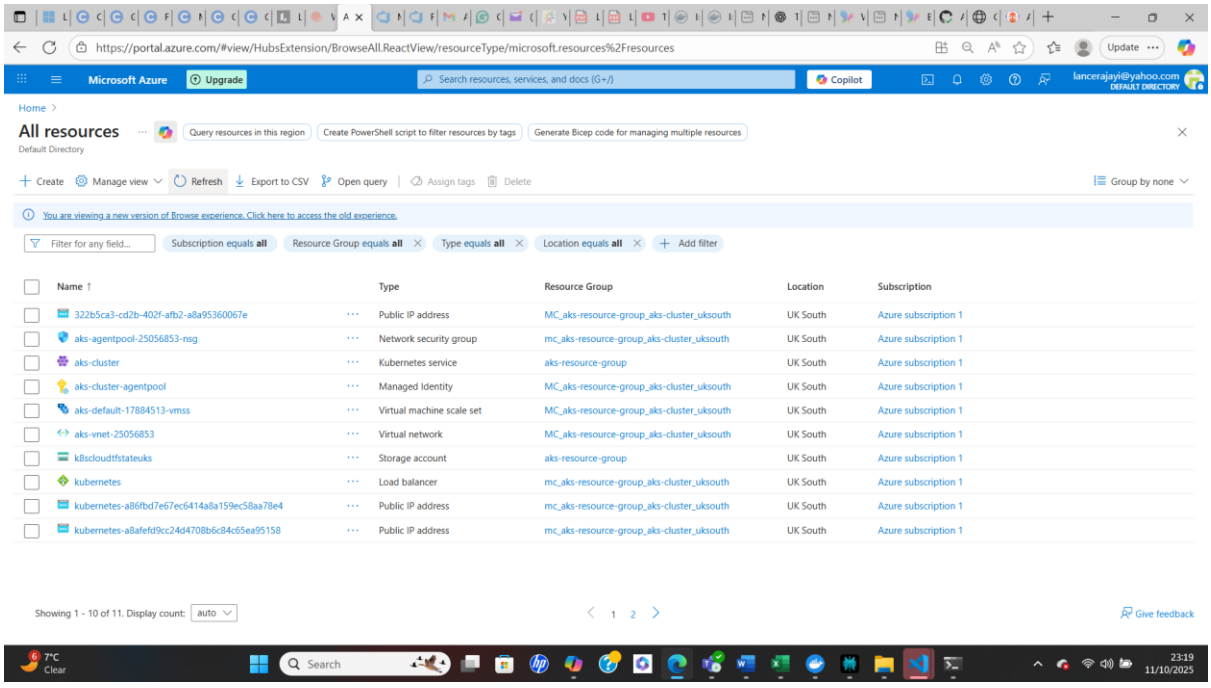
ArgoCD Resources



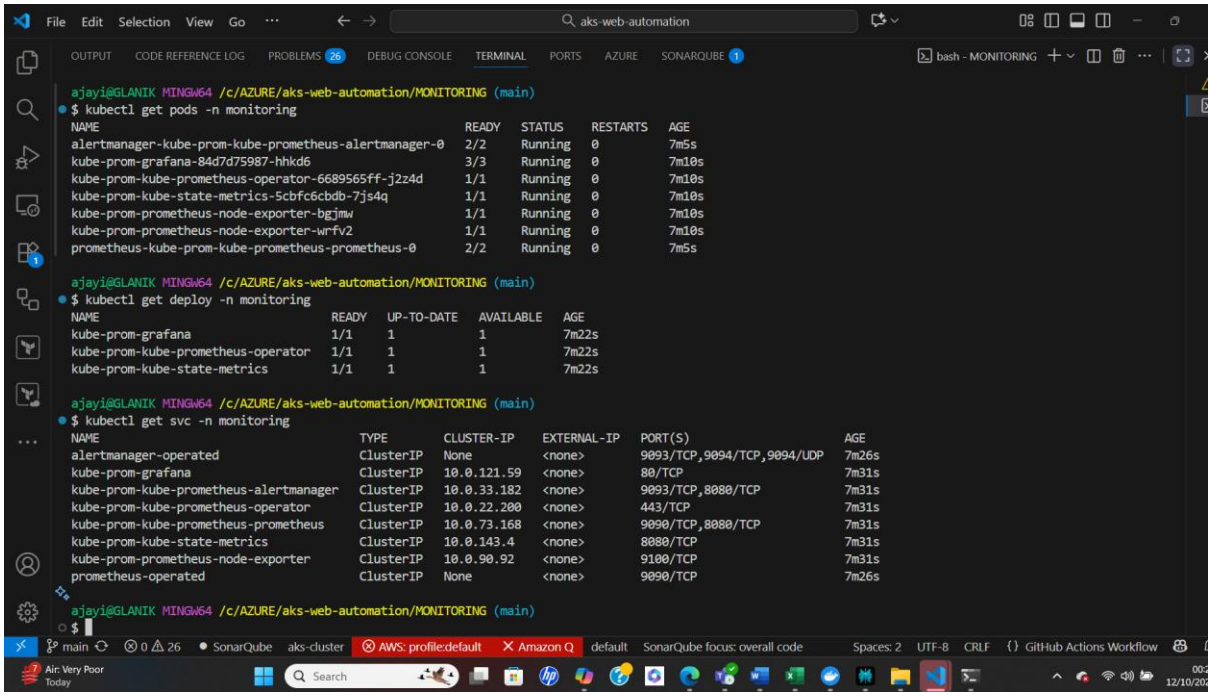
Azure AKS Cluster and HPA



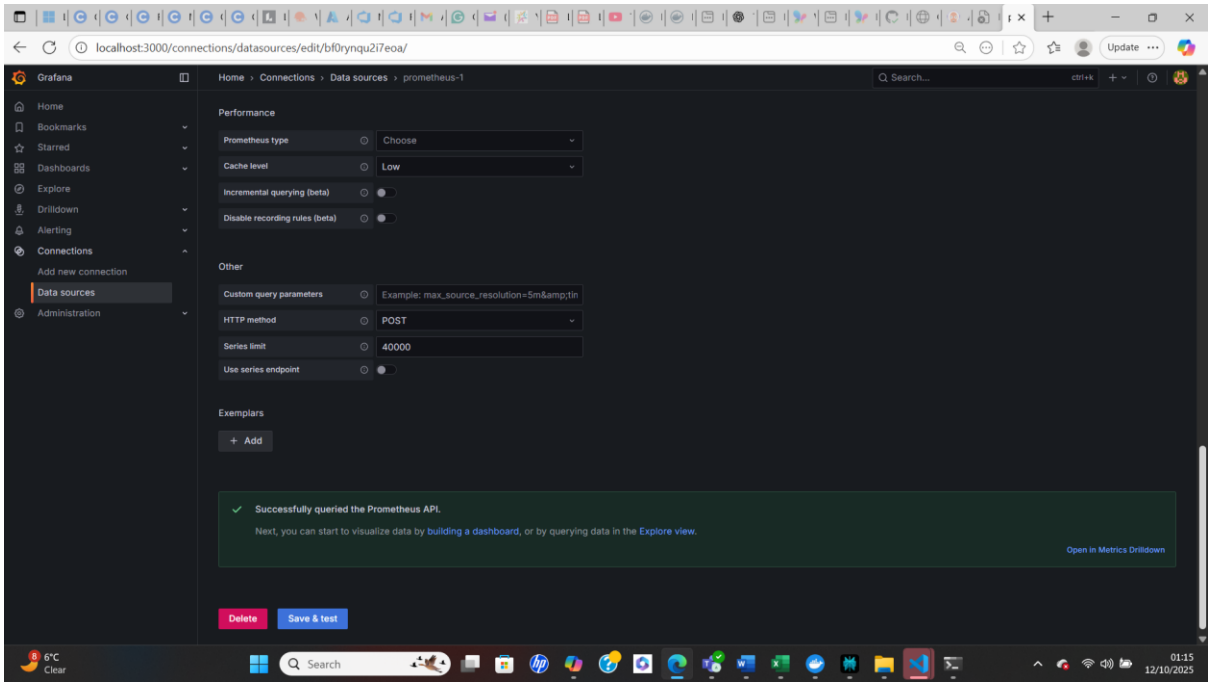
Azure Console Resources



Monitoring Stack



Grafana Dashboard



Next Steps

DevSecOps Integration Strategy within the Workflow

Different Layers would be considered including:

1️⃣ Code & Dependency Security (before build)

Objective: Detect vulnerabilities, secrets, or policy violations before provisioning or deployment.

- ***Static Application Security Testing (SAST)***
- ***Secret Scanning / Credential Leakage***

Check for accidentally committed secrets.

2️⃣ Infrastructure-as-Code (Terraform) Security

Objective: Prevent insecure infrastructure definitions before they reach Azure.

- ***Static IaC Scan (Terraform)***

Use **Checkov** or **tfsec** to enforce cloud security best practices.

This checks for: Publicly exposed AKS or storage accounts, missing encryption, insecure network rules & lack of RBAC / logging

3️⃣ Container Image Security

Objective: Scan your Docker image for OS/package vulnerabilities *before deployment*.

- ***Container Vulnerability Scan***

Integrate **Aqua Trivy** (free and popular) **before pushing** or **before deploying to AKS**.

4️⃣ Dependency & Supply-Chain Security

Objective: Secure your build environment and dependencies.

- Enable **Dependabot** for automatic dependency updates:
- Usage of **signed commits** and enable **branch protection rules** (required reviews, CI checks, etc.)

5️⃣ Runtime & Kubernetes Security

Objective: Validate your AKS manifests and cluster configuration.

- ***Kubernetes Manifest Scanning***

Use **kube-linter** or **kubescape** before kubectl apply to flag containers running as root, missing resource limits, unencrypted secrets & insecure host paths

- ***Post-Deployment Checks (Optional)***

After deployment, integration of **kubescape** or **OPA Gatekeeper** for runtime policy compliance and **Azure Defender for Kubernetes** (native AKS integration)

6️⃣ Compliance & Reporting

Generate a summary or artifact of all security checks.

Optionally, integrate results into **GitHub Security Dashboard** (with CodeQL + Dependabot).

⚙️ Putting It All Together: Enhanced Workflow Structure

You'd end up with a 4-stage pipeline:

Stage	Job	Key Tools
🧠 **Pre-Provision Security**	security-scan	CodeQL, Semgrep, TruffleHog
🌍 **Infra Provisioning**	provision	Terraform + Checkov
🐳 **Container Build & Scan**	image-scan	Docker + Trivy
🚀 **Deploy to AKS**	deploy	Kubectl + Kube-linter

Each stage can use needs: to enforce order:

security-scan → provision → image-scan → deploy

🔐 Bonus DevSecOps Enhancements

Category	Best Practice
Secrets Management	Use GitHub Actions Secrets or Azure Key Vault (not plaintext in workflow).
Policy as Code	Integrate OPA / Conftest to enforce compliance rules.
Auditing	Enable workflow artifact retention and audit logs in GitHub.
Least Privilege	Use minimal RBAC for your Azure SPN credentials (no Owner role).
Encryption	Ensure state files (Terraform backend) and images are encrypted at rest.