[Architecture Diagram: 1](#_Toc467703600)

[Infrastructure Components: 2](#_Toc1181104195)

[Deployment Process: 2](#_Toc342096566)

[Infrastructure Provisioning: 3](#_Toc627767159)

[Container Orchestration: 3](#_Toc1743810822)

[Security Measures: 3](#_Toc375144021)

[Auto-Scaling: 3](#_Toc1838307454)

**Infrastructure Deployment Documentation**

## **Architecture Diagram:**

## **Infrastructure Components:**

Amazon VPC (Virtual Private Cloud):

* A custom virtual network environment is created using Amazon VPC to launch AWS resources.
* Subnets, route tables, and internet gateways are configured to provide secure and isolated network access for the deployed services.

Amazon EKS (Elastic Kubernetes Service):

* AWS EKS is utilized as the container orchestration platform for managing containerized applications.
* Kubernetes clusters are deployed to manage the lifecycle of containers, enabling automatic scaling, updates, and self-healing capabilities.

Amazon ECR (Elastic Container Registry):

* Amazon ECR is employed as a fully managed Docker container registry for storing, managing, and deploying Docker container images.
* It provides secure, scalable, and highly available storage for Docker images, ensuring reliable access for containerized applications.

Amazon EFS (Elastic File System):

* Amazon EFS is utilized as a scalable and elastic file storage service for Kubernetes applications.
* It provides shared file storage for multiple Kubernetes pods, enabling data persistence and sharing across the cluster.

## **Deployment Process:**

### **Infrastructure Provisioning:**

* Terraform is used for infrastructure provisioning, leveraging its declarative configuration files to define the desired state of the infrastructure.
* AWS resources such as VPC, EKS clusters, ECR repositories, and EFS file systems are provisioned using Terraform scripts.
* Terraform state files are stored in an S3 bucket to maintain the state of the infrastructure and enable collaboration among team members.

### **Container Orchestration:**

* Kubernetes manifests and Helm charts are used to define the deployment, services, and other resources required to run containerized applications on AWS EKS.
* Helm charts provide a convenient way to package, version, and deploy Kubernetes applications, simplifying the management of complex application deployments.

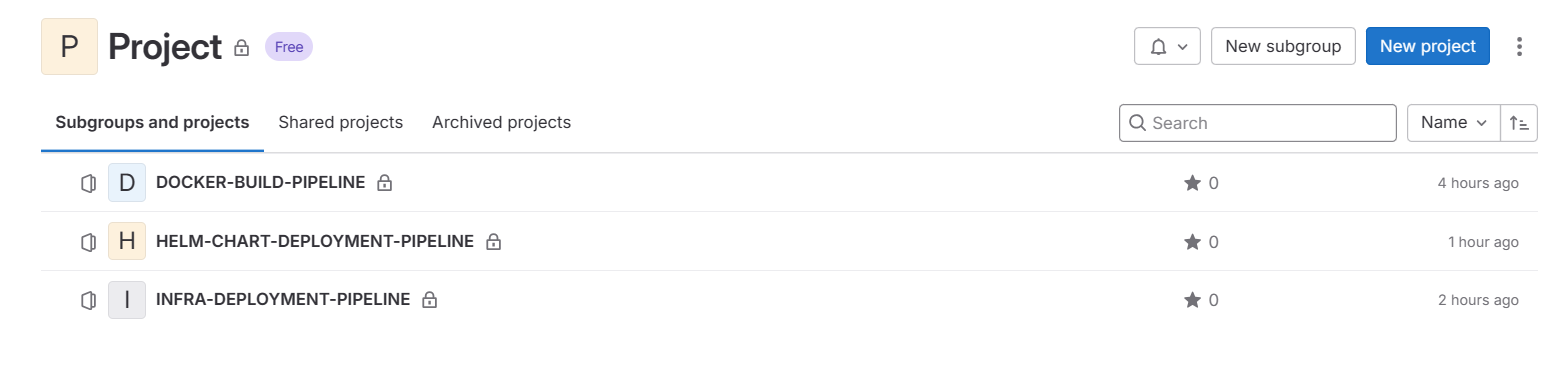
## **Security Measures:**

* Security best practices are implemented throughout the deployment process, including the use of private subnets, private EKS clusters, and rootless deployment for enhanced security.
* Role-based access control (RBAC) to control access to resources and restrict communication between pods and services.

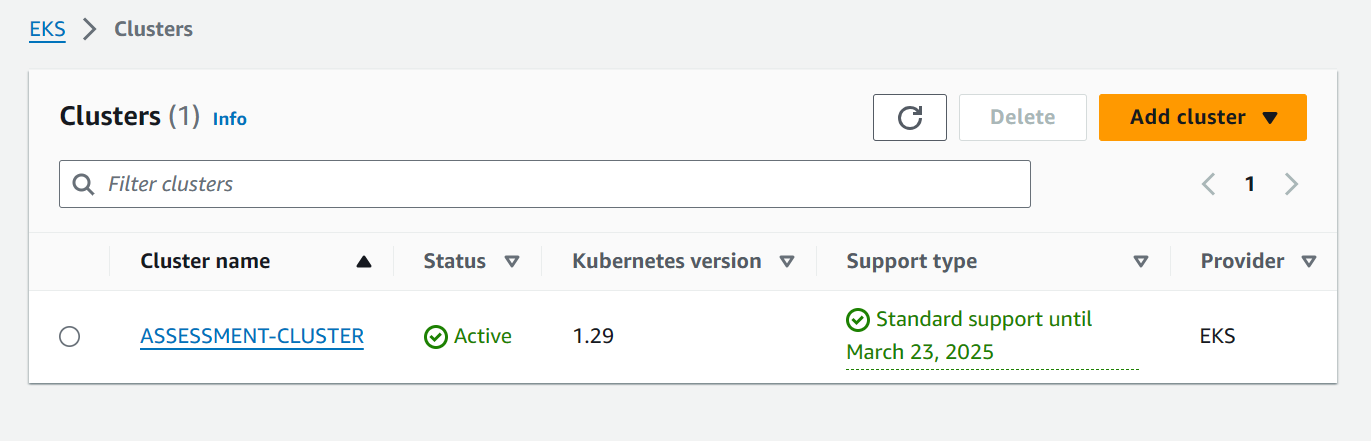
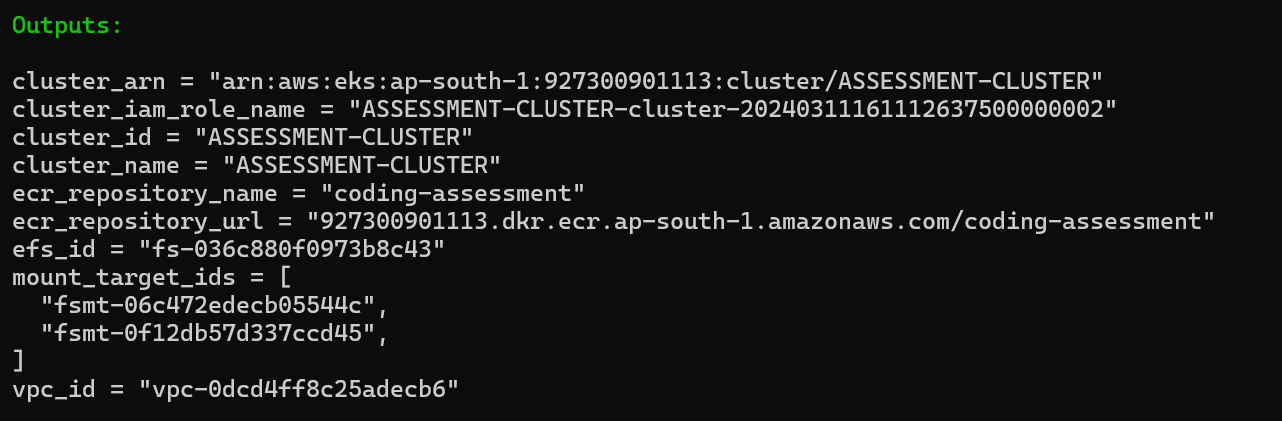
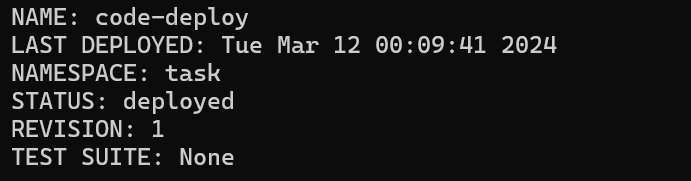
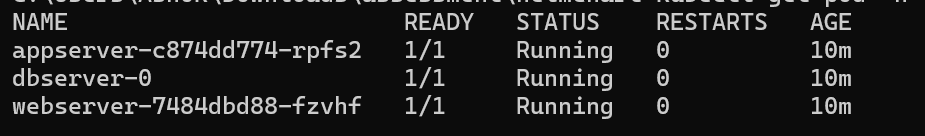
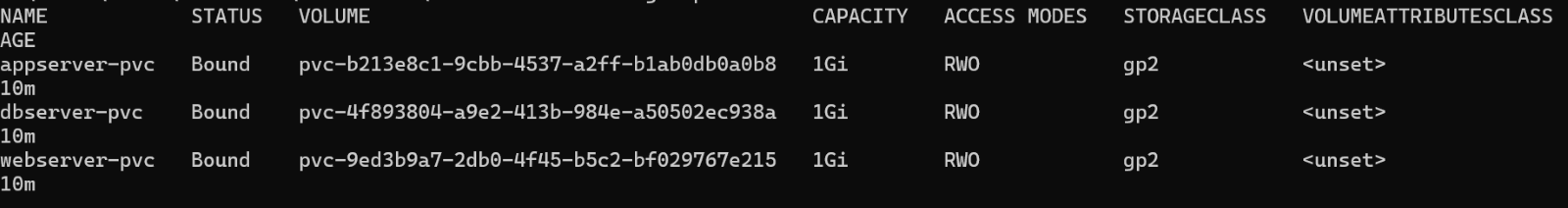
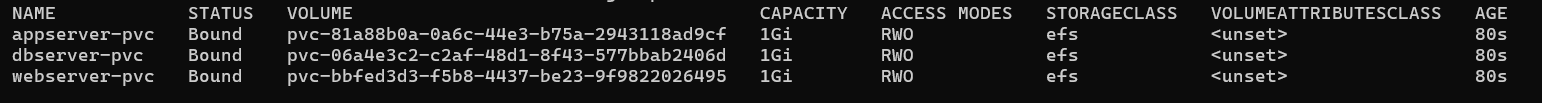
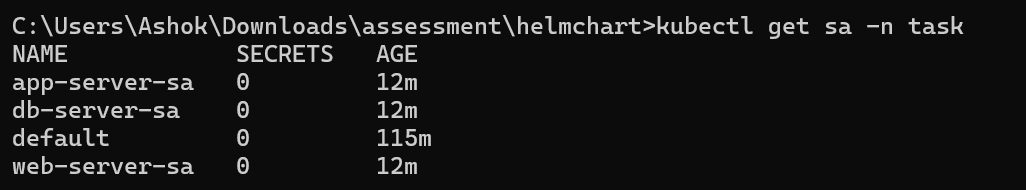
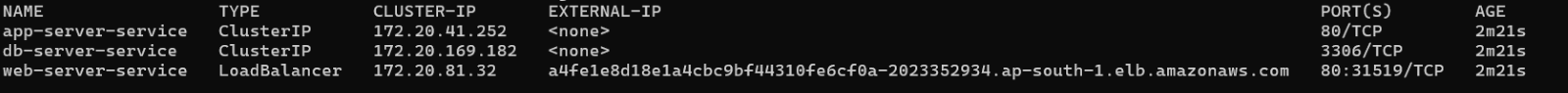
## **Auto-Scaling:**

* Horizontal pod autoscaling is configured to automatically adjust the number of pod replicas based on CPU utilization, ensuring optimal resource utilization and performance scalability.
* Auto-scaling enables the infrastructure to dynamically respond to changes in workload demand, ensuring high availability and reliability of the deployed applications.

**Pipeline Screenshots:**



**Deployment Screenshots:**

**Amazon EKS Cluster:**  
  
**Terraform Provisioning:**  
  
  
  
**Helm Chart Deployment:**  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
**Auto-Scaling Configuration:**  
  
