Innovate Inc.

Cloud Infrastructure Design

This document outlines the cloud infrastructure design for Innovate Inc., detailing the architecture, network configuration, security measures and deployment process.

1. Cloud Environment Structure

Innovate Inc. will leverage AWS's scalable, reliable, and secure global infrastructure to create a well-architected cloud environment. The cloud infrastructure will be structured into two separate AWS accounts under a centralized AWS Organization, allowing for better isolation, security, and cost management.

Account 1 – Production: Host production workloads to ensure isolation and security.

Account 2 – Staging/Development: Separate environment for staging and development activities

Security Measures

In order to have better security, we will implement the following measures on both accounts:

- Implement AWS Identity and Access Management (IAM) policies to control access to AWS resources per account

- Enforce Two-Factor Authentication (2FA) for all IAM users to to protect against unauthorized access

- Enforcing policy for changing passwords and rotating AWS keys for both console and programmatic access

- VPN server will be deployed in the staging/development account to secure and hide development activities from external access.

2. Network Architecture

We will implement a VPC with multiple subnets to ensure isolation and security along with NAT gateway to allow outbound internet access. Subnets will be created In multiple Availability Zones for scaling and high availability.

- Public Subnet includes an Internet Gateway for Load Balancers to manage external traffic

- Private Subnet dedicated for Kubernetes (EKS) clusters to isolate compute workloads from public access

- Database Subnet for Database only ensuring restricted access and enhanced security

With security groups we will control inbound as well as outbound traffic to ensure only required ports should be open.

3. Compute Platform

We will deploy Kubernetes (EKS) to manage containerized workloads. Helm charts will be used for consistent, repeatable, and scalable deployments.

Node Groups and Scaling

Mixed Architecture (x86 and Arm64) – Two node groups will be deployed to separate application and management layers:

- Application Layer will runs frontend and backend services.

- Management Layer will hosts ArgoCD, monitoring tools and other management tools to prevent resource contention with application workloads.

Karpenter for Autoscaling

Karpenter will dynamically provision and scale nodes based on real-time demand, optimizing both cost and performance - vertical scaling.

- Production runs on-demand nodes for consistent performance.
- Development runs on spot instances for cost efficiency.

Also we will utilized HPA(horizontal pod autoscaling) depending on the load to support scaling in all directions.

From the start Frontend service will run on 2 pods to better utilize Application Loadbalancer.

Container Build and Deployment Strategy

Multi-Step Docker Builds – A two-stage Docker build process will be used:

- Stage 1 is build phase to compile and package code.

-Stage 2 is a lightweight runtime image for production to reduce the overall image size and improve deployment speed.

ECR Cost Optimization

- Untagged images will be deleted daily.

- Tagged images will be deleted after 14 days if the tag includes the TEST or DEV prefix.

CI/CD Pipeline with ArgoCD and GitHub Actions

The deployment process will be fully automated using GitOps and CI/CD best practices:

- Build Docker image using GitHub Actions.

- Configure AWS credentials and push the image to ECR.

- Update the values.yaml file with the new image tag.

- Push the updated Helm chart to the GitHub repository.

- ArgoCD will automatically detect changes and deploy the new version to Kubernetes.

4. Database

For Database we will go with Aurora PostgreSQL since we are having initial small load of users but in the future it can support even more without manually interventions.

Out of the box it provides automatic auto-scaling on both storage and compute. It has lower latency and it's overall faster performance than RDS PostgreSQL.

Provides automatic backups, encryption at rest and in transit, and multi-AZ deployment for high availability. Security patches are applied automatically.