**Innovate Inc. Cloud Infrastructure Design**

Overview

Innovate Inc. is building a web app with a Python/Flask REST API server-side, a React single-page app (SPA) client-side, and a PostgreSQL database. This report is the architecture design to deploy their app on AWS using managed Kubernetes (Amazon EKS) in order to scale, secure, and be cost-efficient.

*1. Cloud Environment Structure*

For isolation, cost control, and best security practices, the following AWS account structure is recommended:

Development Account: Staging and testing.

Production Account: Production deployment of the live application.

Shared Services Account: Logging, monitoring, and IAM roles, and CI/CD tools.

*2. Network Design*

VPC Architecture: A single dedicated Amazon VPC with public and private subnets across three availability zones for availability.

Security Measures:

Use Security Groups (as well as Network ACLs) to restrict access.

Provision an Application Load Balancer (ALB) in public subnets to handle frontend requests.

Host backend services and the PostgreSQL database on private subnets.

Add AWS WAF to protect against common web threats (if needed).

*3. Compute Platform*

Kubernetes Cluster (Amazon EKS)

Node Groups:

Managed node groups with CPU and memory autoscaling.

Dedicated node groups for frontend, backend.

Spot instances with on-demand instances as fallback for cost savings.

Scaling Strategy:

Use Karpenter for dynamic node scaling.

Containerization Strategy:

Containerize Python/Flask backend and React frontend in Docker.

Store container images in Amazon Elastic Container Registry (ECR).

Use any CI/CD tool like GitLab for CI/CD automation.

*4. Database*

Service:

Use Amazon RDS for PostgreSQL for managed database services with autoscaling and backups.

High Availability:

Deploy a Multi-AZ RDS instance.

Enable read replicas (in case of heavy read loads).

Backups & Disaster Recovery:

Enable automated backups and snapshots.

Employ AWS Backup for scheduled snapshots and long-term retention.

Implement cross-region replication for disaster recovery.

*5. Security Best Practices*

IAM Policies:

Implement least privilege access with IAM roles and policies.

Utilize AWS IAM Identity Center (SSO) for user access.

Implement multi-factor authentication (MFA) for all IAM users.

Encryption:

Enable TLS for service-to-service communication.

Enable database encryption with RDS native encryption.

Monitoring & Logging:

Use Amazon CloudWatch for application monitoring.

Store logs in Amazon S3 with lifecycle policies for cost savings.

Use AWS Security Hub for a centralized security monitor.

Use AWS Secrets Manager for safe credential storage.

*6. CI/CD Pipeline*

Use CI/CD tool with the following phases:

Build: Package and build the application.

Test: Automated testing with unit tests, sanity tests and integration tests.

Security Scanning:

Add security tools scan for vulnerabilities in container images.

Deploy:

Helm charts and Kubernetes manifests.

*7. Cost Optimization*

Use AWS Cost Explorer (or tools like CloudChipr) for cost monitoring.

Use EC2 Savings Plans and Reserved Instances for planned workloads.

Use AWS Spot Instances for non-production workloads.

Use AWS Budgets and Alerts to track and manage spending.