**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 Network ACLs and Security Groups 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.

Encrypt internal communication using VPC Peering or AWS PrivateLink.

Use AWS Transit Gateway to securely connect multiple VPCs.

Use AWS Network Firewall to implement security policies.

*3. Compute Platform*

Kubernetes Cluster (Amazon EKS)

Node Groups:

Managed node groups with CPU and memory autoscaling.

Dedicated node groups for frontend, backend, and database workloads.

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

Scaling Strategy:

Use Karpenter for dynamic node scaling.

Enable Cluster Autoscaler to dynamically scale resource usage.

Horizontal Pod Autoscaler (HPA) to dynamically scale the number of running pods.

Containerization Strategy:

Containerize Python/Flask backend and React frontend in Docker.

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

Use AWS CodePipeline or GitLab for CI/CD.

Execute lightweight workloads with AWS Fargate without node management.

*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 for scalability.

Employ Amazon Aurora Global Database for global low-latency access if needed.

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.

Is supported point-in-time recovery to recover to any point in the retention window.

*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.

Use AWS KMS encryption for data at rest.

Enable database encryption with RDS native encryption.

Monitoring & Logging:

Use Amazon CloudWatch for application monitoring.

Turn on AWS GuardDuty and AWS Config for security audit.

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 AWS CodePipeline (GitLab) with the following phases:

Source: GitLab or AWS CodeCommit.

Build: AWS CodeBuild to package and build the application.

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

Security Scanning:

Add security tools like Snyk or Trivy to scan for vulnerabilities in container images.

Deploy:

Helm charts and Kubernetes manifests deploy with AWS CodeDeploy.

Blue/Green Deployment:

Use AWS App Mesh or ALB for zero-downtime and secure rollouts.

*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.

Enable AWS Compute Optimizer to analyze cost-saving opportunities.

Use AWS Budgets and Alerts to track and manage spending.