Ashish Anil Khopade

8766528526

ashishakhopade@gmail.com

Introduction

This project represents the Infrastructure as Code (IaC) setup to migrate a Kubernetes cluster from Exoscale to AWS using Terraform and Helm. The goal is to provision the required AWS infrastructure and deploy applications and services onto EKS clusters, following best practices for high availability, security, scalability, and compliance.

Project Structure

The project is organized into two main parts: **Terraform** for AWS infrastructure provisioning and **Helm** for Kubernetes deployments.

Copy code

```
├── terraform
│   ├── modules
│   │   ├── vpc
│   │   ├── eks
│   │   ├── storage
│   │   ├── security
│   │   └── networking
│   ├── environments
│   │   ├── production
│   │   └── staging
│   └── ...
├── helm
│   ├── charts
│   │   ├── nextjs-website
│   │   ├── react-dashboard
│   │   ├── nextjs-api
```

```
│   │   ├── postgresql
│   │   ├── redis
│   │   ├── monitoring
│   │   └── ...
└── README.md
```

 Terraform: Contains the .tf files for AWS provisioning, including VPC, EKS, and S3 storage setup.

 Helm: Contains Helm charts for each application and service, including Next.js websites, React.js dashboards, PostgreSQL, and Redis.

 modules: Reusable Terraform modules for VPC, EKS, and Storage.

 environments: Separate Terraform configurations for production and staging environments.

 charts: Contains parameterized Helm charts for Kubernetes deployments.

Terraform Configuration

1. VPC and Networking

- **VPC**: The VPC is configured with public and private subnets across multiple Availability Zones (AZs) for high availability.
- **Internet Gateway & NAT Gateway**: Used to route traffic from the public subnets to the internet and enable access to the internet for private subnets.
- **Security Groups**: Configured to follow best practices, allowing only necessary traffic.
- **Route Tables**: Defined for both public and private subnets, ensuring proper routing between them.

2. EKS Cluster

- **EKS Cluster**: Configured using managed node groups with multi-AZ deployment for high availability.

- **Cluster Autoscaler**: Automatically adjusts the size of the EKS cluster based on application demand.
- **IAM Roles for Service Accounts (IRSA)**: Enables fine-grained permissions for Kubernetes workloads using AWS IAM.

3. Storage

- **EBS Volumes**: Configured as persistent storage for stateful applications like PostgreSQL and Redis.
- **S3 Buckets**: Replaces Exoscale SOS storage for object storage. Configured with proper IAM policies to ensure secure access.

4. Security

- **IAM Policies**: Implemented with the principle of least privilege, ensuring minimal permissions for services.
- **KMS Encryption**: Configured for encryption at rest using AWS KMS for sensitive data.
- **Network ACLs**: Provide an additional layer of security by controlling traffic at the subnet level.

5. Route 53 and Cloudflare Integration

- **Route 53**: Configured for DNS management. Integrated with Cloudflare for improved CDN performance and security.
- **SSL/TLS**: Ensured secure communication between services using certificates managed by AWS.

Helm Charts

1. Applications

- **Next.js Website**: Deployed in both production and staging namespaces using Helm. Ingress resources are managed via AWS Load Balancer Controller.
- **React.js Dashboard**: Same deployment strategy as Next.js website with multi-replica setup for high availability.
- **Next.js API**: Followed the same deployment patterns with health checks and autoscaling enabled.

2. Databases

- **PostgreSQL**: Deployed as a StatefulSet with high availability configurations, including read replicas, automatic failover, and backups to S3.
- **Redis**: Deployed with Redis Sentinel for automated failover. Data persistence is ensured using EBS volumes.

3. Ingress Controller

- **AWS Load Balancer Controller**: Deployed to manage Ingress resources for the applications, seamlessly integrated with Cloudflare for DNS resolution.

4. Monitoring and Alerting

- **Prometheus Operator**: Configured to scrape metrics from Kubernetes applications and services.
- **Grafana**: Provides dashboards for visualizing performance metrics.
- **AlertManager**: Configured to send alerts via Slack and email when thresholds are breached.

5. Autoscaling

- **Horizontal Pod Autoscaler (HPA)**: Configured to automatically scale pods based on CPU and memory usage.
- **Vertical Pod Autoscaler (VPA)**: Provides recommendations for resource requests and limits based on observed usage.
- **Cluster Autoscaler**: Adjusts the number of EKS nodes based on cluster-wide demand.

6. Security

- **Network Policies**: Restrict inter-service communication within the Kubernetes cluster, ensuring proper isolation.
- **Pod Security Policies**: Ensure only secure and compliant pods are deployed in the cluster.

**Terraform Setup **

Pre-requisites:

- Terraform 1.0 or later installed
- AWS CLI configured
- Access to an AWS account with sufficient permissions

**VPC and Networking **

- Create a **VPC** with public and private subnets across multiple Availability Zones (AZs).
- Configure:
    - **Internet Gateway (IGW)**: Allows public subnets to access the internet.
    - **NAT Gateway**: Routes traffic from private subnets to the internet through the IGW.
    - **Route Tables**: Define how traffic flows through public and private subnets.
    - **Security Groups**: Control ingress/egress traffic to/from the VPC.
    - **Network ACLs**: Set up stateless traffic rules.

File location: terraform/modules/vpc/main.tf

**EKS Cluster **

- Set up an **EKS Cluster** using AWS's managed service, enabling multi-AZ deployment for high availability.
- **Managed Node Groups**: Define your worker nodes that can auto-scale.
- Enable **IAM Roles for Service Accounts (IRSA)** for fine-grained control over EKS pods using AWS services.

- Integrate **Cluster Autoscaler** to automatically manage scaling of nodes.

File location: terraform/modules/eks/main.tf

**Storage Setup **

- **EBS Volumes**: Create EBS volumes for persistent storage for stateful services like PostgreSQL and Redis.
- **S3 Buckets**: Set up S3 buckets for object storage to replace Exoscale SOS. Ensure appropriate IAM policies are in place for access control.

File location: terraform/modules/storage/main.tf

**Security Configuration **

- Apply **IAM roles** and policies using the principle of least privilege.
- Use **AWS KMS** to encrypt sensitive data at rest (S3, EBS).
- Configure **Security Groups** and **Network ACLs** to allow only necessary traffic.

File location: terraform/modules/security/main.tf

**Route 53 and Cloudflare Integration **

- Set up **Route 53** for DNS management.
- Integrate seamlessly with your existing **Cloudflare** setup for improved performance and security.

File location: terraform/modules/networking/main.tf

**3. Helm Setup **

Pre-requisites:

- Helm 3 installed
- Access to Kubernetes cluster on EKS

**Next.js Website **

- Deploy the Next.js website across production and staging environments using multiple replicas.
- Define **Ingress resources** to expose the application via AWS Load Balancer Controller.

File location: helm/charts/nextjs-website

**React.js Dashboard **

- Same deployment strategy as the Next.js website with multiple replicas.

File location: helm/charts/react-dashboard

**Next.js API **

- Follow the same deployment and scaling setup as the Next.js website and React dashboard.

File location: helm/charts/nextjs-api

**PostgreSQL **

- Deploy **PostgreSQL** using StatefulSets for high availability.
- Configure read replicas, point-in-time recovery, and automated backups to S3.
- Implement **PgBouncer** for connection pooling and apply resource limits.

File location: helm/charts/postgresql

**Redis **

- Set up **Redis** with Redis Sentinel for high availability.
- Use EBS volumes for persistence and deploy using StatefulSets.
- Apply resource requests, limits, and anti-affinity rules.

File location: helm/charts/redis

**Monitoring and Autoscaling **

- Deploy **Prometheus Operator** for monitoring and Grafana for visualization.
- Set up **AlertManager** for sending alerts via Slack or email.
- Configure **Horizontal Pod Autoscaler (HPA)** for scaling based on resource usage.
- Enable **Cluster Autoscaler** at the node level.

File location: helm/charts/monitoring

**4. Folder Structure **

The folder structure is organized to separate Terraform and Helm files into their respective directories.

**5. Best Practices **

Terraform Best Practices:

- Run terraform validate and terraform fmt to ensure clean and consistent code.
- Use modules to keep your infrastructure DRY (Don’t Repeat Yourself).
- Implement terraform.tfvars to manage environment-specific configurations.

Helm Best Practices:

- Parameterize **values.yaml** for custom configurations across environments.
- Keep **templates** clean and reusable.
- Use resource requests and limits to ensure your applications run optimally.

6. Instructions to Use

Terraform:

1. Initialize the Terraform directory:

terraform init

2. Validate the Terraform code:

terraform validate

3. Plan the infrastructure changes:

terraform plan -var-file=terraform.tfvars

terraform apply -var-file=terraform.tfvars

Helm:

1. Package the Helm charts:

helm package ./charts/{chart-name}

2. Deploy the Helm chart:

helm install {release-name} ./charts/{chart-name} --namespace {namespace}

3. Upgrade the chart:

helm upgrade {release-name} ./charts/{chart-name}