DevOps to scale Health Insurance Platform

Introduction:

The project involved leveraging DevOps to address the needs of a Healthcare Insurance Platform. This platform is TypeScript and Node.js based and has been designed with the intent to revolutionize how users access and manage health insurance plans. The platform extends beyond traditional medical coverage by integrating holistic services like teleconsultation, prescription medication delivery, diagnostic tests and outpatient/inpatient reimbursements. It allows policyholders to include family members under a single plan, making it a comprehensive solution for modern healthcare needs.

Policyholders can:

- Purchase insurance plans with extensive benefits, including hospitalization, surgeries and daily health needs.
- Upload prescriptions for online medication orders and leverage elastic search for seamless product selection.
- Claim reimbursements for OPD and IPD services by uploading required documents like bills and prescriptions.
- Manage multiple family coverages (parents, spouse, children, siblings, in-laws).

The platform's user-friendly design consolidates various healthcare services, ensuring accessibility and convenience, particularly for individuals with chronic conditions or mobility issues.

Client Details:

Name: Confidential | **Industry:** Healthcare, Software | **Location:** India

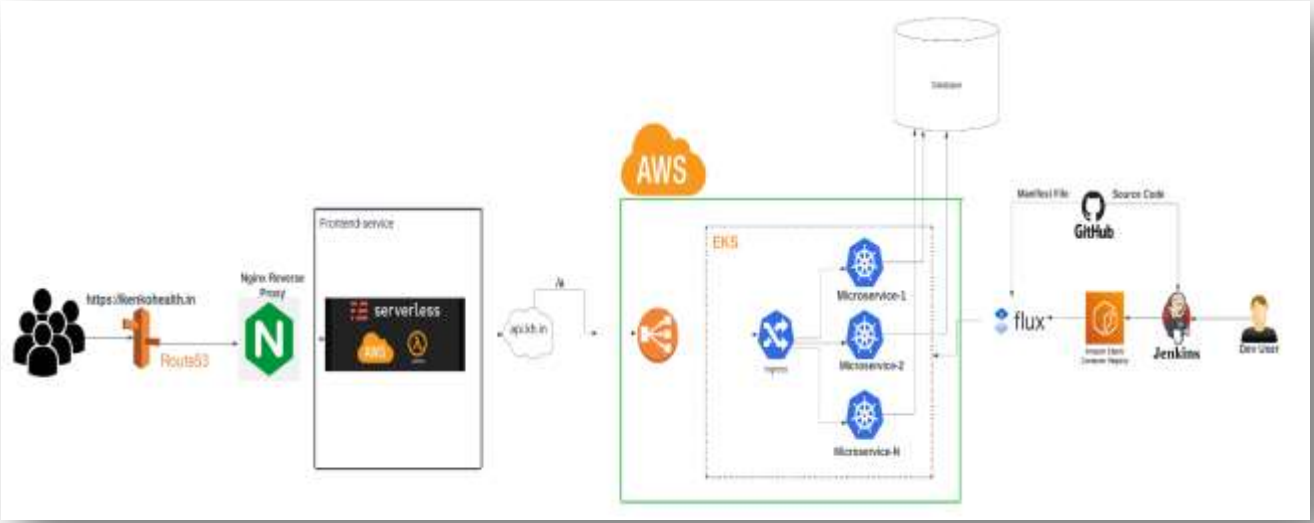
Technologies Used:

- **Cloud Platform**: AWS
- **Repository**: GitHub (Source Code), ECR (Container Registry)
- **CI/CD**: Jenkins
- **Containerization**: Docker, Kubernetes (EKS)
- **Hosting**: AWS Lambda (Serverless), Nginx (Web Server on EKS)
- **DNS**: Route 53
- **Database**: RDS

DevOps to scale Health Insurance Platform

Infrastructure Design and Traffic Flow:

The attached image illustrates the robust infrastructure architecture supporting the platform, built on AWS's scalable cloud ecosystem. Below is a detailed breakdown:



A. Traffic Flow

1. **User Interaction:**
   - Users access the platform via the custom domain (**https://ABC.com**) managed through AWS **Route53** for DNS resolution.
2. **Nginx Reverse Proxy:**
   - Requests are routed to the **Nginx Reverse Proxy**, which serves as the entry point for frontend and backend services, ensuring traffic is efficiently directed.
3. **Frontend Service:**
   - The frontend is hosted using **AWS Lambda** with the Serverless framework. This provides a highly scalable and cost efficient solution for handling user requests.
   - API requests from the frontend are directed to the backend server as the entry point.
4. **Backend Processing (EKS):**
   - Requests hitting **api.kh.in** are forwarded to the **AWS Application Load Balancer (ALB)**.
   - ALB routes traffic to the **Elastic Kubernetes Service (EKS)** cluster where multiple microservices (e.g., Microservice-1, Microservice-2, etc.) are deployed.

DevOps to scale Health Insurance Platform

These microservices handle core functionalities like Policy Management, Prescription processing and Claims validation.

5. **5 Ingress Controller:**
    - Inside EKS, the **Nginx Ingress Controller** manages routing between various microservices, ensuring seamless communication and load distribution.

B. CI/CD Pipeline

1. **Development Workflow:**
    - Developers push source code and Kubernetes manifest files to **GitHub**.
    - **Jenkins** automated build and deployment pipelines, creating Docker images and pushing them to **Amazon Elastic Container Registry (ECR)**.
2. **GitOps with Flux CD/Argo CD:**
    - **Flux CD/Argo CD** synchronizes the manifests stored in GitHub with the Kubernetes cluster, ensuring that any updates to the codebase are automatically reflected in the infrastructure.

C. Database Layer

Persistent data, such as user profiles, policy details, prescription records and reimbursement claims, is stored in a managed **AWS RDS database** for high availability and durability.

Infrastructure Components:

1. **Route53:**
    - Handles DNS management for the custom domain, ensuring low latency access.
2. **Nginx Reverse Proxy:**
    - Enhances performance by managing HTTP/HTTPS traffic and balancing load.
3. **Serverless Framework with Lambda:**
    - Efficiently handles frontend service requests with minimal infrastructure overhead.
4. **EKS (Elastic Kubernetes Service):**
    - Orchestrates containerized microservices, ensuring high availability and scalability.
5. **Application Load Balancer (ALB):**
    - Balances incoming traffic across EKS pods, optimizing performance.
6. **Flux CD/Argo CD:**
    - Implements GitOps principles for streamlined CI/CD processes.
7. **Jenkins:**
    - Facilitates automated build, test and deployment workflows.
8. **Amazon RDS:**
    - Ensures reliable and secure data storage for healthcare related information.

DevOps to scale Health Insurance Platform

Key Benefits of Infrastructure Design:

- **Scalability:** The use of AWS Lambda and EKS ensures the platform can handle increased traffic and workloads without performance degradation.
- **Resilience:** Redundancy in Route53, ALB and EKS ensures high availability and minimal downtime.
- **Automation:** CI/CD pipelines streamline deployments, reducing manual intervention and improving development agility.
- **Security:** APIs and services are protected through Nginx reverse proxy, ALB security groups and AWS managed policies.
- **User Experience:** Elastic search and optimized backend services deliver fast and reliable performance.

This infrastructure design provides a robust foundation for the platform, ensuring it meets the demands of modern healthcare while delivering a seamless user experience.