**Design CRM – System Design**

**CRM Overview**:  
**CRM** is a set of practices, strategies, and technologies used to manage and analyse customer interactions and data throughout the customer lifecycle. The primary goal of CRM is to improve customer relationships, streamline communication, and enhance customer satisfaction.

Before diving into the design, lets outline the functional and non-functional requirements.

**Functional Requirements:**

- **Customer Onboarding**: Register new users (mobile, broadband, etc.)  
- **Customer Profile Management**: View and update customer information (name, address, ID, contact), Link multiple services (SIMs, devices) to a single customer ID.  
- **Plan & Service Management**: View available plans (data, voice, roaming), Subscribe, upgrade, downgrade, or cancel plans  
- **Ticketing & Complaint Handling**: Create and manage support tickets (billing issue, SIM not working, etc.)  
- **Self-Service Integration**: Allow customers to log in to portal/mobile app, View profile, plans, active tickets, usage

### ****Non-Functional Requirements:****

* **Scalability**: Handle thousands of concurrent agent and customer sessions
* **High Availability**: -Redundant nodes and multi-zone deployment  
   - 99.99% uptime with zero-downtime upgrades
* **Low Latency** : Fast UI rendering for agent screens
* **Security**: RBAC (Role-Based Access Control) for agents  
   - IDP and api gateway for authentication and session management
* **Monitoring & Maintainability**: Logging: agent activity, ticket status, API calls  
   - Metrics: average resolution time, ticket volume, active sessions  
   - Use tools like Prometheus, ELK, and Grafana for visibility

2 . **Architecture Principles & Patterns**  
 1. **High Availability (HA)**  
 - **Multi-AZ Deployment** — Resources like EKS node groups, RDS (Primary/Secondary), DocumentDB, Kafka, and OpenSearch are **spread across AZs (1a, 1b, 1c)** for fault tolerance

* **Load Balanced APIs** behind ALB for availability.
* **RDS with failover**, DocumentDB clusters with HA

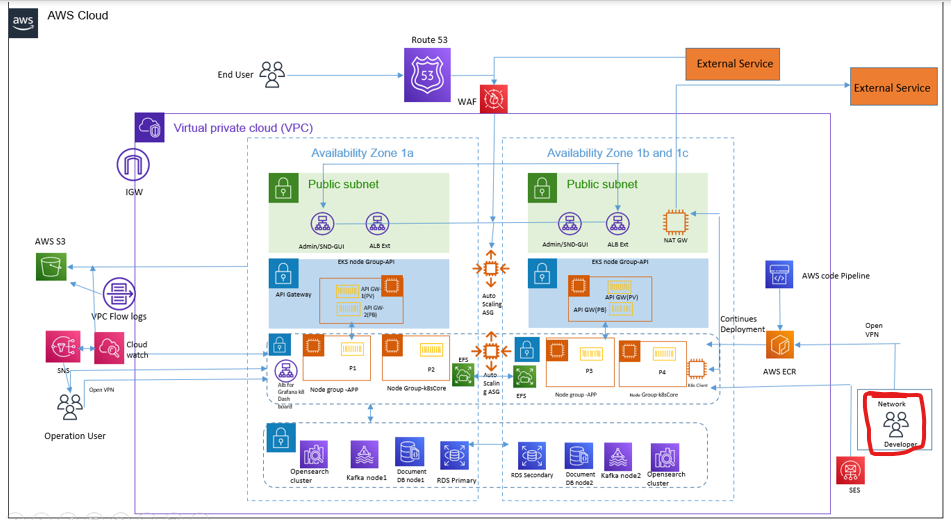
2. **Scalability:**  
 - **Auto Scaling Groups (ASG)** on EKS node groups allow your workloads to scale based on demand  
 - **EKS (Kubernetes)** handles pod-level horizontal scaling.  
3. **Separation of Concerns (SoC):**  
 - Node groups are **logically separated**: app, store, multiCare, etc.

4. **Secure by Design:**  
 - **WAF + Route 53** protects APIs and websites from common attacks.  
 - **NAT Gateway + Private Subnets** for internal workloads (no direct internet access).  
 - Use of **IAM roles**, **VPC flow logs**, and **CloudWatch** for monitoring.  
 - **ALB for HTTPS termination**, possibly offloading TLS.

5. **Observability:**  
 - **CloudWatch** for logs & metrics  
 - **SNS** for alerts  
 - **VPC Flow Logs** for network observability  
 - Prometheus/OpenSearch stack for app metrics and search

3 . Structure with Clear Architecture Layers

CRM platform is infrastructure agnostic, it supports the containerized deployment (with Amazon EKS) on AWS public cloud. Below diagram depict the standard AWS architecture. Solution is configured in Multi AZ environment on AWS.

AWS Architect :  


|  |  |
| --- | --- |
| **Service Name** | **Purpose** |
| IAM | User Access Management |
| VPC | Virtual Private Cloud for Internal/External Communication |
| EKS | K8s Cluster Creation on aws to run the MS |
| ECR | Repository for Hosting BSS Docker Images |
| S3 | Repository for Hosting Logs Backup |
| EC2 | Elastic compute for run standalone server |
| ELB | Use to segregate traffic basis on request type |
| NAT Gateway | Use for network translation for private network |
| EFS | Used for creating PV and for batch |
| RDS | MySQL database for storage of user profile, data, transaction details etc. |
| MSK | Kafka for messaging |
| Opensearch | For application log monitoring |
| SES | For send notifications |
| DocumentDB | Used by MongoDb apis |

4. Operational Excellence  
 1. Monitoring & Observability

- **CloudWatch** for logs & metrics  
 - **SNS** for alerts  
 - **VPC Flow Logs** for network observability  
 - Prometheus/OpenSearch stack for app metrics and search

2. Backup & Disaster Recovery (DR)  
 - RDS snapshots (daily + PITR)

- S3 versioning + replication

- EKS manifests stored in Git for rehydration

3. Security & Compliance Operations  
 - Enable **AWS WAF logs** and **guardrails**

- Use **AWS Config** for drift detection

- Enable **IAM Access Analyzer**

4 . Cost Monitoring & Right-Sizing  
 - Idle EKS nodes or underutilized pods

- Unused ALBs or NAT gateways

- S3 storage classes (move infrequent data to Glacier or IA)

- Use **Cost Explorer + Budgets**