TECHNICAL DESIGN SPECIFICATION

Integrated Management Information System (MIS)

Enterprise Hybrid Architecture & Disaster Recovery

Version: 1.0

Status: Final Draft

Role: DevOps / Infrastructure Architect

Table of Contents

1. Executive Summary
2. Scenario Overview & Key Assumptions
3. High-Level Architecture Design
4. Hybrid Connectivity & Disaster Recovery
5. High Availability & Scalability

6. Security Architecture
7. CI/CD Pipeline Design
8. Monitoring & Observability
9. Capacity Planning
10. Conclusion

1. Executive Summary

This document outlines the technical design for a government-scale Integrated Management Information System (MIS). The architecture is designed to support 1,000+ concurrent users with a 99.9% uptime SLA, operating across a Hybrid Infrastructure (On-Premise and AWS Cloud). The system handles mission-critical citizen-facing portals and internal department functions, requiring robust security, high availability, and proactive disaster recovery. Key design highlights include:

- **Hybrid Architecture:** Seamless integration between On-Prem high-performance physical hardware and AWS scalable cloud services.
- **Kubernetes Orchestration:** Standardized containerized deployments using K8s (On-Prem) and EKS (AWS).
- **Disaster Recovery:** An Active-Passive multi-region/multi-site strategy ensuring low RTO/RPO.
- **Security First:** Zero-trust principles, WAF/DDoS protection, and end-to-end encryption.

2. Scenario Overview & Key Assumptions

The organization requires a scalable, mission-critical MIS. **User Load:** 1,000+ Concurrent high-activity users spanning multiple government departments. **Application Stack:**

- Backend: Containerized FastAPI/Django services.
- Frontend: React SPA.
- Data: PostgreSQL (Primary DB), Redis (Caching), Celery (Async Tasks).
- Storage: S3 (Cloud) and NAS (On-Prem) for document management.

Operational Targets:

- Uptime: 99.9% (approx. 8.77 hours downtime/year max).
- Compliance: Strict data protection (citizen data residency).
- Integrations: External Web + API integrations via secure gateways.

3. High-Level Architecture Design

The architecture leverages leading industry standards to provide a unified platform across distributed environments.

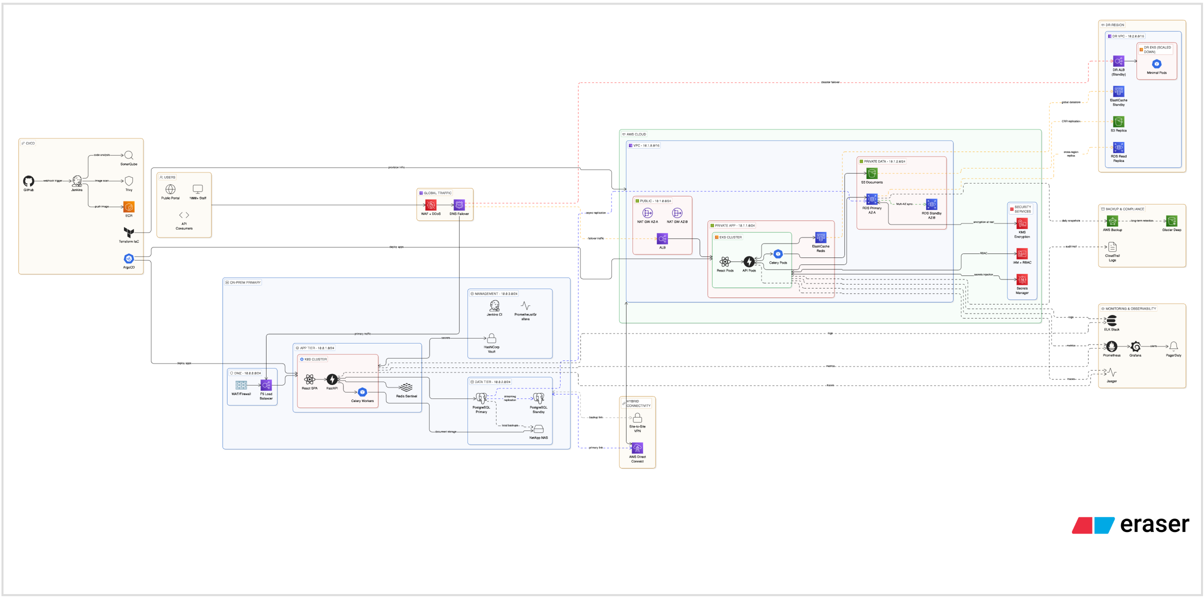

Figure 1: Enterprise Hybrid Architecture Design

3.1 On-Prem Architecture

The On-Premise site serves as a primary data center or secondary processing hub depending on specific regulatory requirements.

- **DMZ:** Segregated network for external traffic entry, housing hardware firewalls (FortiGate/Cisco) and F5 Load Balancers.
- **App Tier:** Self-managed Kubernetes cluster (K3s/RKE2) running FastAPI and React pods.
- **Data Tier:** High-performance PostgreSQL clusters with physical streaming replication to standby nodes.
- **Storage:** NetApp NAS for reliable document and file storage.

3.2 Cloud Architecture (AWS)

The AWS environment provides elasticity and global reach.

- **VPC Design:** Multi-tier VPC across availability zones. Public subnets for ALBs/NAT Gateways, Private subnets for Compute/EKS, and Isolated subnets for Databases.
- **Compute:** Amazon EKS (Elastic Kubernetes Service) for automated node management.
- **Data:** Amazon RDS for PostgreSQL with Multi-AZ enabled for automatic failover.

4. Hybrid Connectivity & Disaster Recovery

4.1 Hybrid Connectivity

Connectivity between On-Premise and AWS Cloud is established via:

- **AWS Direct Connect:** 1Gbps/10Gbps dedicated line for low latency and consistent throughput.
- **Site-to-Site VPN:** Encrypted backup tunnel to ensure connectivity remains if Direct Connect fails.

4.2 Disaster Recovery Strategy

A pilot-light/warm-standby Disaster Recovery (DR) Region is maintained in a separate AWS geographic location.

Parameter	Target	Strategy
RTO (Recovery Time Objective)	< 4 Hours	Automated failover of DNS and DB promotion.
RPO (Recovery Point Objective)	< 15 Mins	Cross-Region Replication (CRR) for RDS and S3.
Model	Active-Passive	DR region runs minimal compute footprints (scaled-down K8s).

Failover Process: Triggered via Amazon Route 53 Health Checks. RDS Standby is promoted to Primary, and EKS clusters are auto-scaled from minimum pods to production levels.

5. High Availability & Scalability

5.1 99.9% Uptime Achievement

Redundancy is implemented at every layer of the stack:

- **Infrastructure:** Multi-AZ deployment within Cloud Regions; diverse hardware racks for On-Prem.
- **Database:** RDS Multi-AZ synchronously replicates data; On-Prem uses PostgreSQL Streaming Replication with Repmgr for auto-healing.
- **App Layer:** K8s Horizontal Pod Autoscaler (HPA) scales pods based on CPU/Memory metrics.
- **Cache:** Redis Sentinel (On-Prem) and Amazon ElastiCache (Multi-AZ) handle session and transient data.

6. Security Architecture

6.1 Identity & Access

- **RBAC:** Kubernetes Role-Based Access Control and AWS IAM roles for service-to-service communication.
- **Secrets:** HashiCorp Vault (On-Prem) and AWS Secrets Manager (Cloud) for dynamic secret injection.

6.2 Edge Protection

- **WAF & DDoS:** AWS WAF and Cloudfront (or On-Prem F5 AFM) protect against SQLi, XSS, and Layer 7 attacks.
- **Zero-Trust:** Implementation of mTLS within the service mesh (Istio/ Linkerd) for encrypted inter-pod communication.

6.3 Data Security

- **In-Transit:** TLS 1.3 enforced for all internal and external communication.
- **At-Rest:** AES-256 encryption using AWS KMS and LUKS for On-Premise physical volumes.

7. CI/CD Pipeline Design

A unified DevOps pipeline ensures rapid, secure deployments using a 'GitOps' approach.

- **Source:** GitHub Enterprise with protected branches.
- **CI Tool:** Jenkins/GitHub Actions executing build scripts and unit tests.
- **Security Scanning:** Trivy for container image scanning and SonarQube for static code analysis.
- **Deployment:** ArgoCD for GitOps-style deployments to K8s clusters.
- **Strategy:** Blue-Green deployments to minimize downtime during releases and allow instant rollbacks.
- **IaC:** Terraform for provisioning VPCs, RDS, and EKS clusters, ensuring environment parity.

8. Monitoring & Observability

Comprehensive observability stack for proactive issue resolution:

- **Metrics:** Prometheus for time-series data collection with Grafana dashboards for visualization.
- **Logs:** ELK Stack (Elasticsearch, Logstash, Kibana) for centralized log aggregation.
- **Tracing:** Jaeger/AWS X-Ray for distributed tracing across microservices.
- **Alerting:** PagerDuty integration for critical incident alerts via Slack and SMS.

9. Capacity Planning

9.1 Sizing for 1,000 Concurrent Users

Based on standard MIS benchmarks:

- **Web Tier:** ~10-15 Pods (2 vCPU, 4GB RAM each) to handle peaks.
- **API/Backend:** ~20 Pods (4 vCPU, 8GB RAM each) for compute-intensive logic.
- **Database:** RDS db.m6g.4xlarge (16 vCPU, 64GB RAM) with IOPS optimized storage.

9.2 5-Year Growth Model

Targeting 20% annual data volume growth. We propose Horizontal scaling for applications and storage expansion for databases using Amazon Aurora or RDS Storage Autoscaling.

10. Conclusion

The proposed design offers a resilient, high-performance architecture suited for government-scale operations. By combining the security of On-Premise infrastructure with the flexibility of AWS Cloud, the MIS system is prepared for both day-to-day excellence and emergency disaster recovery scenarios.