# Technical Design Document

## Enterprise Hybrid Architecture & Disaster Recovery

**Client:** Enterprise Operations

**Version:** 1.0

**Status:** Final Draft

**Design Principles:** 99.9% Availability, Multi-Region DR, Zero-Trust

## Table of Contents

*Reference Figure 1.0: Enterprise Hybrid Architecture + Disaster Recovery Design (Attached Image Input)*

# 1. Infrastructure Design

The architecture adopts a **Hybrid Cloud strategy**, leveraging both an On-Premises Primary data center and AWS Cloud for scalability and disaster recovery. This design ensures data sovereignty while benefiting from cloud-native resilience.

## 1.1 Network Segmentation (VLAN / VPC)

- **AWS VPC:** Designed with three distinct tiers: Public (DMZ), Private (App), and Data (Database). Spans across 3 Availability Zones (AZs) for high availability.
- **On-Prem VLANs:** Segregated using hardware firewalls into DMZ, Kubernetes Management, and Backend Data zones.

## 1.2 DMZ and Load Balancing

- **External Layer:** AWS WAF and Shield protect the Application Load Balancer (ALB). On-prem, a hardware-based WAF/LB pair handles citizen and internal traffic.
- **Load Balancing Strategy:** Round-robin for stateless app clusters; sticky sessions (cookie-based) for legacy components if necessary. Health checks monitor endpoint vitality every 5 seconds.

## 1.3 Container Orchestration

- **AWS EKS:** Managed Kubernetes service for the cloud environment. AWS Fargate or Managed Node Groups are used to ensure node-level lifecycle management.
- **On-Prem Kubernetes:** A self-managed or vendor-supported K8s cluster (e.g., Rancher/OpenShift) to maintain consistency with cloud deployments.

## 1.4 Storage and Database Strategy

- **Database HA:** On-prem uses PostgreSQL Primary/Standby with synchronous replication. AWS uses **RDS Multi-AZ** for automated failover.
- **Storage Strategy: Amazon S3** for document storage with cross-region replication. On-prem uses localized NAS/SAN with backup links to S3 for hybrid data persistence.

- **Firewall & WAF:** Multi-layer defense with AWS Security Groups (SG), Network ACLs, and On-prem hardware firewalls (e.g., Palo Alto/Fortinet).

## 2. High Availability Design

This design targets a **99.9% SLA**, allowing for only ~8.77 hours of downtime per year.

### 2.1 Multi-AZ and Clustering

- The AWS environment utilizes 3 Availability Zones. If one AZ fails, traffic is automatically rerouted by Route 53 and ALB to the remaining healthy zones.
- Kubernetes clusters utilize **Pod Anti-Affinity** rules to ensure application replicas are distributed across different physical hosts/zones.

### 2.2 Database and Cache HA

- **PostgreSQL:** RDS Multi-AZ provides a standby instance in a different AZ with synchronous replication. Failover is automatic via DNS endpoint updates.
- **Redis HA:** Amazon ElastiCache (Redis) with Cluster Mode enabled. Primary and Replica nodes are spread across AZs to ensure sub-millisecond failover.
- **Stateless Scaling:** The React SPA, FastAPI, and Django services are stateless. HPA (Horizontal Pod Autoscaler) scales pods based on CPU/Memory utilization.

## 3. Disaster Recovery (DR) Design

| Metric | Target |
|---|---|
| RTO (Recovery Time Objective) | < 4 Hours |
| RPO (Recovery Point Objective) | < 15 Minutes |
| Strategy | Active-Passive (Warm Standby) |

### 3.1 Backup and Replication

- **Cross-Region Replication:** AWS S3 buckets are replicated to a DR region. RDS Cross-Region Read Replicas are maintained for near-real-time data availability.
- **Backup Frequency:** Daily snapshots for RDS with transaction logs backed up every 5 minutes. Continuous backup to S3.

### 3.2 Failover Process and Testing

- **Failover:** Initiated via Route 53 health checks or manual DNS cutover. The EKS cluster in the DR region (scaled down) is automatically scaled up via HPA/ Autoscaling groups.
- **Testing:** DR drills are conducted semi-annually. Each drill tests data integrity and application functionality in the DR region.

# 4. Security Architecture

## 4.1 Zero-Trust and IAM

- **Identity & Access:** Principle of Least Privilege applies. AWS IAM roles are used for service-to-service authentication (IRSA for EKS).
- **RBAC:** Kubernetes Role-Based Access Control defines specific permissions for developers, admins, and automated systems.

## 4.2 Data Protection

- **Encryption at Rest:** AES-256 encryption via AWS KMS (Key Management Service) for S3, RDS, and EBS volumes.
- **Encryption in Transit:** Mandatory TLS 1.3 for all endpoints. Internal communication between microservices uses mTLS (Service Mesh like Istio or Linkerd).
- **Secrets Management:** Integration with **AWS Secrets Manager** or HashiCorp Vault. No hardcoded secrets in code or environment variables.

## 4.3 Monitoring and Mitigation

- **WAF & DDoS:** Global protection via AWS Shield Advanced. WAF rules filter SQL injection, XSS, and bot traffic.
- **Audit Logging:** All API calls are logged via AWS CloudTrail. OS and K8s logs are forwarded to the ELK stack for compliance auditing.

# 5. CI/CD Pipeline Design

Automated lifecycle management from code commit to production deployment.

## 5.1 Pipeline Stages

- **Branching:** GitFlow model (Feature, Develop, Release, Main).
- **CI Tool: Jenkins** (as per diagram) or GitHub Actions. Jenkins triggers on commit, executes unit tests, and performs **Code Scanning** (SonarQube).
- **Docker Build:** Multi-stage Docker builds to keep image sizes small. Images are scanned for vulnerabilities (Trivy/Clair) before pushing to **Amazon ECR**.

## 5.2 Infrastructure and Deployment

- **IaC:** 100% template-based infrastructure using **Terraform**. Ansible handles configuration management for any non-containerized on-prem nodes.
- **Deployment Strategy: Rolling Updates** for minor fixes. **Canary Deployments** for major releases to test features on 5% of traffic before full rollout.

# 6. Monitoring & Observability

Full-stack visibility to detect and resolve issues before they impact users.

## 6.1 Toolset

- **Metrics: Prometheus** for time-series data collection from K8s nodes, pods, and databases.
- **Visualization: Grafana** dashboards for real-time monitoring of CPU, RAM, Latency, and Error rates.
- **Logging: ELK Stack** (Elasticsearch, Logstash, Kibana) for centralized log aggregation and searching.
- **Tracing:** Distributed tracing (e.g., Jaeger) to track requests across React, FastAPI, and Django layers.

## 6.2 Alerting and SLAs

- Alerts integrated with Slack/PagerDuty. SLA reporting is generated monthly to track 99.9% availability targets.

# 7. Capacity Planning

## 7.1 Assumptions for 1,000 Concurrent Users

- **Compute:** 3 x m5.xlarge nodes (4 vCPU, 16GB RAM) in EKS cluster. Total capacity: 12 vCPUs, 48GB RAM.
- **Database:** db.r5.large RDS instance (2 vCPU, 16GB RAM) to handle concurrent I/O.
- **Growth Model:** Projected 20% annual increase in user base. Horizontal scaling (HPA) will add nodes/pods dynamically.

## 7.2 Scaling Strategy

- **Horizontal Scaling:** Primary strategy. Adding more pods to the K8s cluster and scaling EKS nodes via Cluster Autoscaler.
- **Vertical Scaling:** Reserved for databases where write-scaling becomes a bottleneck (upgrading RDS instance size).

# 8. Cost Estimation (Cloud Only - Monthly)

| Component | Estimated Cost (USD) |
|---|---|
| Compute (EKS + EC2 Nodes) | $450 - $600 |
| Database (RDS Multi-AZ) | $300 - $450 |
| Storage (S3 + EBS) | $100 - $200 |
| DR Replication / Bandwidth | $150 - $300 |
| Security & Monitoring (WAF/CloudWatch) | $200 - $300 |
| **Total Estimated** | **$1,200 - $1,850** |

## 4. Constraints & Compliance

- **Scalability:** Supporting 5-year growth via horizontal K8s scaling.
- **SLA:** 99.9% uptime managed through Multi-AZ and failover scripting.
- **Compliance:** Full audit logging and RBAC for regulatory adherence.
- **Hybrid:** Integration via AWS Direct Connect with VPN backup for 100% connectivity.