Identified Risks and Weaknesses
The table below maps each risk to its impact area:

Risk	**WAF Pillar Impacted**	**Potential Consequence**
Single AZ / Single Server Deployment	Reliability	Any hardware failure or AZ level event causes complete application outage with no automatic failover.
No Database Backup or DR Plan	Reliability	Data loss is permanent if the database server fails. RPO is effectively zero backups; RTO is undefined.
Open Security Groups (0.0.0.0/0 on SSH & DB)	Security	Brute force, credential stuffing, and direct database exfiltration attacks are possible from the internet.
No WAF or DDoS Protection	Security	The application is vulnerable to OWASP Top 10 attacks (SQLi, XSS) and volumetric DDoS without any mitigation layer.
No Data Encryption (at rest / in transit)	Security	Customer PII and sensitive business data is transmitted and stored in plaintext, violating compliance requirements.
Manual Deployments (No CI/CD or IaC)	Operational Excellence	Configuration drift, human error, and slow-release cycles reduce operational agility and increase mean time to recovery (MTTR).
No Monitoring or Alerting	Operational Excellence	Failures are discovered reactively (by users), not proactively; RCA is difficult without centralized logs.
Over Provisioned OnPrem Hardware	Cost Optimization	Fixed hardware costs cannot flex with demand; idle capacity generates waste with no ability to scale down.
No CDN or Caching Layer	Performance Efficiency	All requests hit the origin server; database roundtrips are repeated for cacheable data, increasing latency under load.

Task 2 AWS Well-Architected Framework (WAF) Assessment

The AWS Well-Architected Framework provides a consistent approach for evaluating cloud architectures against five pillars. Each pillar below includes a strength, a gap, a concrete improvement recommendation, and the supporting AWS service(s) that enable the improvement.

Pillar	**Observation (Strength / Gap)**	**Improvement Recommendation**	**Supporting AWS Service(s)**
Operational Excellence	**Strength:** Application code is modular and deployment scripts exist on premises, providing a foundation for automation. **Gap:** No CI/CD pipeline or infrastructure-as-code (IaC) in place; deployments are manual and error-prone.	Adopt IaC to automate infrastructure provisioning and establish a CI/CD pipeline for consistent, repeatable deployments.	AWS CodePipeline + AWS CodeDeploy for CI/CD; AWS CloudFormation / AWS CDK for IaC; Amazon CloudWatch for observability.
Security	**Strength:** Application has role-based access at the software layer. **Gap:** Security groups are overly permissive (0.0.0.0/0 on port 22/3306), database is exposed, no encryption at rest, and no WAF in front of the application.	Implement least privilege security groups, encrypt all data at rest and in transit, place the database in a private subnet, and add a web application firewall.	AWS WAF, AWS Shield, AWS KMS for encryption, AWS IAM with least privilege, AWS Secrets Manager for DB credentials, AWS Inspector for vulnerability scanning.
Reliability	**Strength:** Application tier is stateless, making horizontal scaling feasible. **Gap:** Single AZ deployment with no failover, no backup strategy for the database, and no health check automation.	Deploy across multiple Availability Zones with auto healing, configure automated database backups and enable Multi AZ for RDS, and add load balancer health checks.	Amazon RDS Multi AZ, AWS Elastic Load Balancing (ALB), Amazon Route 53 with health checks, AWS Backup, AWS Auto Scaling.
Performance Efficiency	**Strength:** On premises architecture keeps compute and database on fast internal networks. **Gap:** No CDN for static assets, no caching layer for the database, and instance sizing has not been right sized for cloud workloads.	Introduce a caching layer to reduce database load, serve static assets via CDN, and rightsized EC2 instances based on profiling.	Amazon CloudFront for CDN, Amazon ElastiCache (Redis) for caching, AWS Compute Optimizer for rightsizing, Amazon RDS Performance Insights.
Cost Optimization	**Strength:** Migrating to AWS eliminates hardware refresh costs and data centre overhead. **Gap:** No Reserved Instances or Savings Plans in use; EC2 instances are overprovisioned; no autoscaling to match demand.	Purchase Reserved Instances or Savings Plans for SteadyState workloads, implement Auto Scaling to match capacity to demand, and enable cost monitoring.	AWS Cost Explorer, AWS Budgets, EC2 Auto Scaling, EC2 Savings Plans / Reserved Instances, AWS Trusted Advisor.

Task 3: AWS Cloud Adoption Framework (CAF) Readiness Assessment

The AWS Cloud Adoption Framework (CAF) structures cloud adoption across six perspectives: Business, People, Governance, Platform, Security, and Operations. Each perspective identifies the organizational capabilities required for a successful migration. The analysis below assesses our organization's current readiness and defines key enablers for this two-tier web application migration.

Business: Executive sponsorship, TCO analysis, ROI business case, cloud KPIs, stakeholder alignment. Our business perspective ensures that cloud investments are aligned with organizational strategy and deliver measurable value. For this migration, the primary business driver is eliminating capital expenditure on aging on-premises hardware while improving application availability and responsiveness. The organization currently lacks a formal cloud business case and total cost of ownership (TCO) analysis. The migration must be justified with quantifiable ROI including reduced hardware refresh cycles, lower operational staffing costs (patching, hardware maintenance), and improved uptime that translates directly to revenue protection. Key actions include a) Conducting an AWS TCO Calculator analysis comparing current on-premises annual costs versus projected AWS monthly spend. b) Defining SLA commitments for the migrated application (e.g., 99.9% uptime). c) Establishing executive sponsorship from the CTO and CFO. d) Setting measurable KPIs such as mean time to recovery (MTTR), deployment frequency, and infrastructure cost per user. Without a clear business case, cloud adoption risks underfunding critical security and reliability investments. Leadership alignment is a prerequisite for all subsequent CAF perspectives.

People: Cloud training & certification, organizational change management, skills gap assessment. The People perspective addresses the human capital and organizational changes required to operate effectively in the cloud. Currently, our engineering team has deep expertise in on-premises infrastructure but limited AWS proficiency. A skills gap assessment must be conducted immediately to identify which team members require foundational cloud training (AWS Certified Cloud Practitioner), which require associate level certifications (Solutions Architect, SysOps, Developer), and which require advanced specializations in security and networking. Key actions should include a) A structured AWS learning path through AWS Skill Builder and APN Partner training. b) Establishing a Cloud Centre of Excellence (CCoE) a small cross functional team responsible for defining cloud best practices, guardrails, and governance policies for the broader organization. c)change management communications to reduce resistance and ensure all stakeholders understand the migration timeline and their new responsibilities.

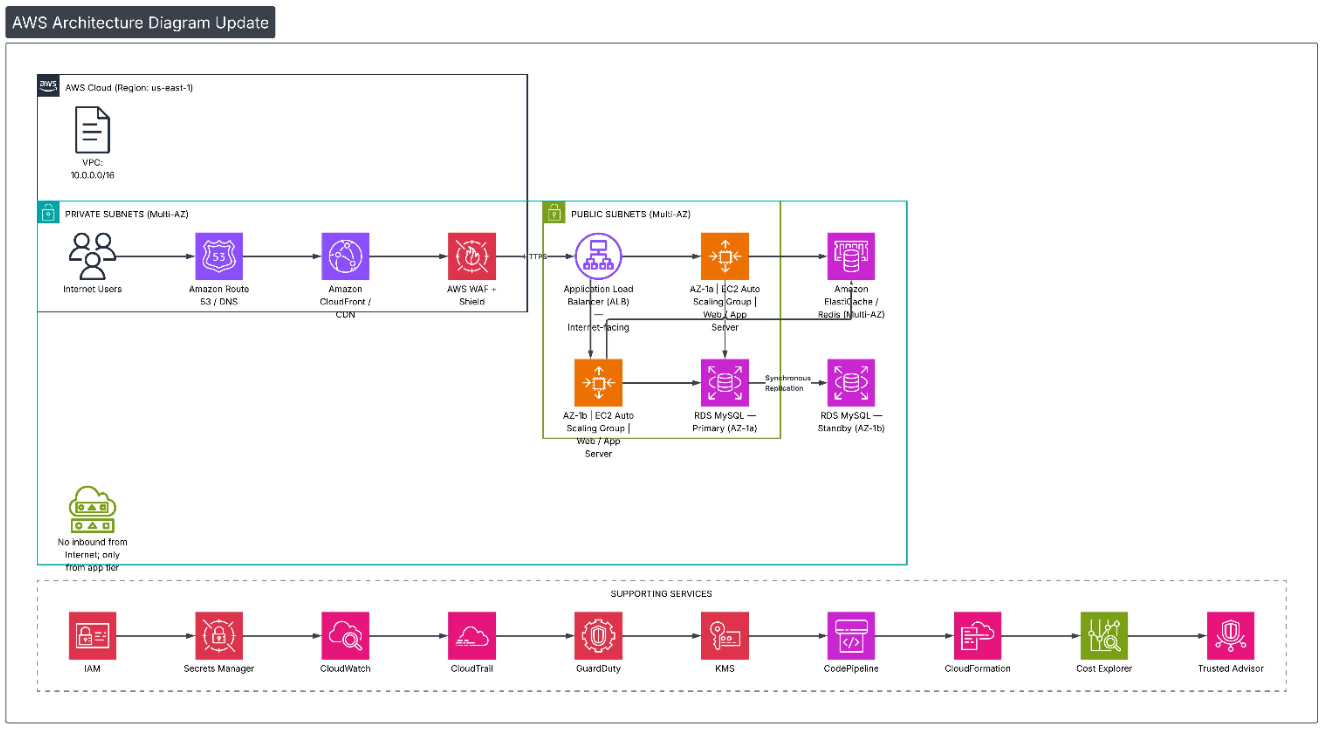
Governance: The Governance perspective ensures that cloud usage adheres to organizational policies, compliance requirements, and financial controls. Currently, the organization has no cloud governance policy, no resource tagging standard, and no mechanism to enforce compliance guardrails across AWS accounts. Without governance, cloud sprawl, uncontrolled spending, and compliance violations are inevitable outcomes. Key action should include a) Implementing a mandatory tagging policy (Environment, Cost Centre, Owner, Application) enforced by AWS Config rules. b) Mapping the application to applicable compliance frameworks (e.g., GDPR, PCIDSS, or ISO 27001) and configuring AWS Security Hub to monitor compliance posture continuously. c) defining cloud financial governance through AWS Budgets with alert thresholds and cost allocation tags linked to business units. d) establishing an approved services list (guardrails) through Service Control Policies (SCPs) in AWS Organizations to prevent unauthorized service usage. Governance must be established as the foundation before workloads are migrated, not retroactively applied afterward.

Platform: The Platform perspective focuses on designing and delivering the cloud infrastructure that will host the migrated workload. This includes VPC design, compute strategy, database platform selection, and the implementation of infrastructure-as-code (IaC) to ensure the environment is reproducible, version controlled, and auditable. Currently, the organization has no IaC practice; all infrastructure is provisioned manually, which introduces configuration drift and makes environment replication impossible. Key actions include a) designing a production-grade Amazon VPC with public subnets (for the Application Load Balancer and NAT Gateway) and private subnets (for EC2 application servers and RDS), spanning a minimum of two Availability Zones for fault tolerance. b) migrating the relational database to Amazon RDS with Multi AZ enabled, replacing the self managed on-premises database server and eliminating the need for manual patching and backups. b) adopting AWS CloudFormation or AWS CDK to codify the entire infrastructure stack, enabling consistent environment provisioning across development, staging, and production. c) publishing approved architecture patterns and AWS Service Catalog products so that new environments can be self served by development teams within preapproved guardrails. d) Integrating Amazon ElastiCache (Redis) for database query caching and Amazon CloudFront for CDN delivery of static assets.

Security The Security perspective defines the controls, processes, and capabilities required to protect the migrated workload in alignment with AWS security best practices and the shared responsibility model. The current on-premises environment has critical security gaps: open network access, no encryption, no secrets management, and no active threat detection. These must all be remediated as part of the migration. Key action includes a) enforcing least privilege, IAM policies using roles (not long-lived access keys), ensuring all EC2 instances use instance profiles for service-to-service authentication. b) storing all database credentials and API keys exclusively in AWS Secrets Manager with automatic rotation enabled, eliminating hardcoded credentials from application code. c) enabling encryption at rest for all RDS databases (AES256 via AWS KMS) and enforcing TLS 1.2+ for all data in transit. d) deploying AWS WAF in front of the Application Load Balancer with AWS Managed Rules (OWASP Top 10 ruleset) and enabling AWS Shield Standard for DDoS mitigation. e) activating Amazon GuardDuty for continuous threat detection, AWS CloudTrail for API audit logging, and AWS Config for configuration compliance monitoring. f) developing and testing an incident response runbook covering detection, containment, eradication, and recovery phases before going live.

Operations: The Operations perspective defines how the migrated workload will be monitored, managed, and continuously improved post migration. Our current state has no centralized monitoring, no alerting thresholds, no runbooks, and no formal incident management process. Operational maturity must be built in parallel with the technical migration to avoid post launch operational failure. Key actions include: a) implementing a full observability stack using Amazon CloudWatch for metrics, logs, and alarms with defined thresholds for CPU, memory, error rates, and database connection counts; configuring AWS XRay for distributed tracing of application requests; b) establishing a CI/CD pipeline using AWS CodePipeline, AWS CodeBuild, and AWS CodeDeploy that enforces automated testing gates (unit, integration, security scan) before any code reaches production; c) codifying all operational procedures as runbooks stored in AWS Systems Manager Run Command and OpsCenter, enabling one click remediation of common operational events; d) defining formal SLA targets (e.g., 99.9% monthly uptime) backed by Route 53 health checks and Auto Scaling group health replacement policies; e) configuring AWS Auto Scaling to automatically replace unhealthy instances and scale out compute capacity during peak demand, reducing on call burden; f) scheduling regular operational reviews using AWS Trusted Advisor and AWS Well-Architected Tool checks to drive continuous improvement.

Task 4: Improved Target AWS Architecture


AWS Architecture Diagram Update
AWS Cloud (Region: us-east-1)
VPC: 10.0.0.0/16
PRIVATE SUBNETS (Multi-AZ)
Internet Users
Amazon Route 53 / DNS
Amazon CloudFront / CDN
AWS WAF + Shield
HTTPS
PUBLIC SUBNETS (Multi-AZ)
Application Load Balancer (ALB) — Internet-facing
AZ-1a | EC2 Auto Scaling Group | Web / App Server
Amazon ElastiCache / Redis (Multi-AZ)
AZ-1b | EC2 Auto Scaling Group | Web / App Server
RDS MySQL — Primary (AZ-1a)
Synchronous Replication
RDS MySQL — Standby (AZ-1b)
No inbound from Internet; only from app tier
SUPPORTING SERVICES
IAM
Secrets Manager
CloudWatch
CloudTrail
GuardDuty
KMS
CodePipeline
CloudFormation
Cost Explorer
Trusted Advisor

AWS Architecture Diagram (Approach)

This diagram documents a two-tier web application reference architecture on AWS and serves as a communication artifact for both technical and non-technical stakeholders. The goal is to make the network boundaries, request flow, and nonnegotiable design principles immediately understandable.

Core architecture pattern (what the diagram represents)

The diagram is built around a standard single-VPC pattern:

- One VPC (e.g., 10.0.0.0/16).
- Two sibling subnet groups inside the VPC:
  - Public Subnets (Multi-AZ): the internet-facing entry tier.
  - Private Subnets (Multi-AZ): the data tier with no inbound internet access.

The end-to-end intent is Internet Users → Route 53 → CloudFront → WAF/Shield → ALB (public) → App tier (EC2 Auto Scaling) → Data tier (RDS/Redis in private)

Guiding framework: Design Principles (WAF-aligned)

i. High Availability: Multi-AZ everywhere; no single point of failure in the user to DB path.
ii. Defence-in-Depth: layered security at edge, WAF, SGs, and private subnets for data.
iii. Automation First: infrastructure as code + CI/CD for repeatability.
iv. Observability: metrics/logs/alarms and tracing for debugging.
v. Cost Consciousness: scale to demand + baseline commitments + tagging.

The diagram is organized so each principle can be traced to concrete components (e.g., ALB/ASG/RDS Multi-AZ for HA; WAF + subnet isolation + IAM/KMS/Secrets for DiD; CodePipeline/CloudFormation for Automation; CloudWatch/CloudTrail for Observability; Auto Scaling + cost tools/tags for Cost).

The diagram above represents the revised target architecture for migrating the two-tier web application to AWS. It is governed by five non-negotiable design principles, each directly traceable to a pillar of the AWS Well-Architected Framework. The table below maps each principle to the specific gap it closes from the original on-premises environment.

Principle	**Original Gap Identified**	**How the Diagram Resolves It**
High Availability (Reliability	Single-AZ deployment with no load balancer. Any server or zone failure caused a complete outage. No database failover or backup strategy; RPO and RTO were undefined.	ALB distributes traffic across EC2 Auto Scaling Groups in AZ-1a and AZ-1b simultaneously. RDS Multi-AZ (Primary + Standby with Synchronous Replication) ensures automatic DB failover within 2 minutes. ElastiCache Redis is also multi-AZ.
Defence-in-Depth (Security)	Open security groups allowed SSH (port 22) and database (port 3306) access from 0.0.0.0/0. No WAF, no DDoS protection, no encryption at rest or in transit, and database credentials were hardcoded.	Five layered controls: CloudFront absorbs DDoS; AWS WAF + Shield blocks OWASP Top 10; ALB enforces TLS via ACM; EC2 security groups accept only ALB traffic; RDS is in a private subnet accepting only EC2 traffic. KMS encrypts all data at rest; Secrets Manager rotates DB credentials automatically.
Automation First (Operational Excellence)	All infrastructure was provisioned manually through the console. Deployments required direct SSH file transfers. No CI/CD pipeline existed; no IaC templates; no test gates before production releases.	CloudFormation (IaC) codifies the entire architecture VPC, subnets, ALB, EC2, RDS, and all security rules enabling reproducible environments. CodePipeline automates every deployment: build, test, staging, approval gate, then Blue/Green production deploy with automatic rollback. SSH port 22 is closed; Systems Manager replaces all direct server access.
Observability (Operational Excellence)	No monitoring, alerting, or centralised logging. Failures were discovered reactively from user reports. No audit trail of infrastructure changes; no performance baselines; RCA after incidents was impossible.	CloudWatch collects metrics from every component (ALB, EC2, RDS, ElastiCache) with pre-configured alarms for CPU, error rates, and storage thresholds. CloudTrail logs every API call to an immutable S3 WORM archive. GuardDuty provides continuous threat detection. X-Ray enables distributed tracing for request-level debugging.
Cost Consciousness (Cost Optimization)	Fixed on-premises hardware forced over-provisioning for peak load 24/7. No Reserved Instances, no spend visibility, no cost allocation per team or service. No CDN meant all user requests hit origin servers, inflating compute requirements.	Auto Scaling dynamically adjusts EC2 count to match real demand, eliminating idle spend. CloudFront offloads 60-70% of requests to edge cache, allowing smaller origin instances. Reserved Instances cover the steady-state baseline at 40% savings. Cost Explorer and Trusted Advisor provide spend visibility; mandatory resource tagging enables cost attribution per team and application.

Reflection

Completing the lab shifted my mindset from focusing on individual cloud services to using structured architectural frameworks. Instead of simply migrating on-premises servers to EC2, the AWS Well-Architected Framework (WAF) encouraged evaluating decisions across all five pillars, exposing hidden risks and trade-offs. The key insight was how interconnected the pillars are choices in cost, security, or reliability directly impact operational excellence and overall system health. Additionally, the Cloud Adoption Framework (CAF) highlighted the importance of organizational readiness, including business alignment, people training, and governance. The author concludes that both frameworks should be used proactively at the start of architecture projects and continuously revisited using the AWS Well-Architected Tool to maintain workload health over time.