5.AWS Improvement Proposal Document

📘 AWS Improvement Proposal Document – English Version

Source:

🟦 AWS Improvement Proposal

Project: AWS Migration Project for SME Corporate Website

Author: Fuma

Created: 2025/11/22

Version: v1.0

💚 1. Purpose of This Proposal

This document outlines recommended improvements to the existing AWS environment from the perspectives of:

- **Availability**
- **Security**
- **Operations**
- **Cost Optimization**

The goal is to evolve the system into an environment that can **improve continuously during operation**.

💙 2. Summary of Improvement Proposals

No	Category	Proposal	Expected Benefit	Priority
1	Availability	NAT Gateway redundancy (multi-AZ)	Stronger AZ failure tolerance	Medium
2	Operations	Systems Manager Patch Manager	Automated patching & reduced workload	Medium
3	Automation	Terraform (IaC)	Improved reproducibility & maintainability	**High**
4	Deployment	Introduce CodePipeline (CI/CD)	Automated application deployment	Medium
5	Security	Implement AWS WAF	Enhanced protection (SQLi / XSS)	**High**
6	Cost	S3 lifecycle optimization	Reduce unnecessary long-term storage	Low
7	Monitoring	Additional ALB/RDS metrics	Improved failure detection	Medium

🟨 3. Detailed Improvement Proposals

🧩 3.1 NAT Gateway Redundancy (Multi-AZ)

Issue:

The NAT Gateway exists only in ap-northeast-1a.

If AZ 1a experiences an outage, all Private Subnet communication becomes unavailable.

Proposal:

- Add a second NAT Gateway in AZ 1c
- Separate routing tables per AZ
- Achieve true multi-AZ outbound redundancy

Expected Benefit:

- Higher fault tolerance during AZ outages
- Production-ready availability level

Priority: Medium

Cost Impact: +40–45 USD/month

🧩 3.2 AWS Systems Manager Patch Manager

Issue:

EC2 OS and middleware updates are done manually, increasing operational effort.

Proposal:

- Install and enable SSM Agent
- Apply patches automatically using Patch Manager
- Define monthly maintenance windows

Expected Benefit:

- Major reduction in operational workload
- Improved security baseline

Priority: Medium

🧩 3.3 Infrastructure as Code with Terraform

Issue:

The environment is currently GUI-driven, making reproducibility low.

Proposal:

- Manage VPC, EC2, ALB, RDS using Terraform
- Store all IaC code in GitHub for version control

Expected Benefit:

- Easy environment duplication and rebuilds
- Change tracking via Git
- Strong advantage in freelance and enterprise projects

Priority: High

🧩 3.4 CodePipeline (CI/CD) Integration

Issue:

Deployments to EC2 are currently manual, leading to human error and wasted effort.

Proposal:

- Implement CodeCommit → CodeBuild → CodeDeploy pipeline
- Enable automatic application updates with a simple git push

Expected Benefit:

- Eliminate deployment mistakes
- Improve deployment speed and consistency
- Better team scalability

Priority: Medium

🧩 3.5 AWS WAF Implementation

Issue:

Only ALB security groups protect the publicly exposed site.

Insufficient against common attack vectors.

Proposal:

- Attach AWS WAF (CommonRuleSet) to ALB
- Block SQL injection, XSS, bot attacks

Expected Benefit:

- Meets minimum web security requirements

- Considered essential for real client-facing systems

Priority: High

🧩 3.6 Fine-Tuned S3 Lifecycle Optimization

Issue:

Storage class transitions are not optimized for long-term cost efficiency.

Proposal:

- After 30 days → Standard-IA
- After 90 days → Glacier Instant Retrieval

Expected Benefit:

- Monthly cost reduction
- Smarter backup and archive management

Priority: Low

🧩 3.7 Additional Monitoring Items

Issue:

CloudWatch monitoring is currently minimal.

Proposal:

Add additional alarms for:

- ALB 4xx error rate
- RDS CPU / DB Connections / FreeStorage
- EC2 StatusCheckFailed
- S3 error notifications

Expected Benefit:

- Faster incident detection
- Higher operational quality

Priority: Medium

💛 4. Priority Summary

Priority	Items
High	Terraform (IaC), WAF
Medium	NATGW redundancy, CI/CD, SSM Patch Manager, Enhanced Monitoring
Low	S3 Lifecycle Optimization

💜 5. Conclusion

The current AWS environment is fundamentally solid, but by applying the following improvements, it can evolve from **“basic practical level” → “enterprise-ready level.”**

- **Availability:** Multi-AZ, NATGW redundancy
- **Security:** WAF, IAM hardening
- **Operations:** Patch automation, CI/CD
- **Automation:** Full IaC with Terraform

Implementing these enhancements sequentially will ensure a continuously improving and resilient AWS architecture.