5.改善提案書

🟦 AWS改善提案書

案件名：中小企業向けコーポレートサイトAWS移行プロジェクト

作成者：ふうま

作成日：2025/11/22

バージョン：v1.0

💚 1. 改善提案の目的

本改善提案書では、構築済みAWS環境に対して

「可用性・セキュリティ・運用・コスト最適化」 の観点から、

今後導入が望ましい改善ポイントをまとめる。

AWS環境を“運用しながら進化させられる”状態を目指す。

💙 2. 改善提案一覧（サマリー）

No	分野	提案内容	期待効果	実施優先度
1	可用性	NAT Gatewayの冗長化（AZ増設）	AZ障害に強くなる	中
2	運用	Systems Manager Patch Manager導入	パッチ自動化・運用負荷低減	中
3	自動化	TerraformによるIaC化	再現性・保守性向上	高
4	デプロイ	CodePipeline導入（CI/CD）	アプリ更新の自動化	中
5	セキュリティ	AWS WAF導入	攻撃対策（SQLi/XSS）強化	高

No	分野	提案内容	期待効果	実施優先度
6	コスト	S3ライフサイクル最適化	不要データの自動削減	低
7	監視	ALB / RDS メトリクス追加監視	障害検知精度向上	中

🟨 3. 改善提案（詳細）

🧩 3.1 NAT Gateway 冗長化

課題：

NATGWが ap-northeast-1a のみに存在しており、AZ障害時にPrivateSubnetの通信が停止する可能性がある。

提案内容：

- 1c に NATGW を追加し、Private Subnet のルーティングをAZごとに分離
- Multi-AZ構成による可用性向上

期待効果：

- AZ障害に強くなる
- 本番運用に耐えられる構成へ進化

優先度： 中

コスト： + 40〜45 USD/月

🧩 3.2 AWS Systems Manager Patch Manager導入

課題：

EC2のOS・ミドルウェアのパッチ適用が手動であり、工数が発生する。

提案内容：

- SSMエージェント導入
- Patch Managerで毎月自動適用
- メンテナンスウィンドウ設定

期待効果：

- 運用負荷の大幅削減

- セキュリティレベル向上

優先度： 中

🧩 3.3 Terraform（IaC）による環境コード化

課題：

GUI操作中心であり、環境再現性が低い。

提案内容：

- Terraformを使用してVPC〜EC2〜ALB〜RDSをコード管理
- GitHubでバージョン管理も実施

期待効果：

- 環境コピー・再構築が容易
- 修正履歴を追跡可能
- フリーランス案件で武器になる

優先度： 高

🧩 3.4 CodePipeline（CI/CD）導入

課題：

EC2へのデプロイが手動のため、作業ミスと工数が発生。

提案内容：

- CodeCommit → CodeBuild → CodeDeploy の構成で自動化
- pushするだけでアプリ更新可能に

期待効果：

- デプロイミス防止
- 運用効率UP
- チーム運用も楽になる

優先度： 中

🧩 3.5 AWS WAF導入

課題：

外部公開しているため、攻撃対策がALBセキュリティグループのみ。

提案内容：

- AWS WAF（CommonRuleSet）をALBにアタッチ
- SQL injection / XSS / Bots の防御強化

期待効果：

- Webセキュリティの最低要件を満たせる
- 実務案件でほぼ必須

優先度： 高

🧩 3.6 S3ライフサイクルの細分化

課題：

S3のストレージクラス管理が一律で、長期保管の最適化ができていない。

提案内容：

- 30日後：Standard-IA
- 90日後：Glacier Instant Retrieval

期待効果：

- 月額コスト削減
- バックアップ最適化

優先度： 低

🧩 3.7 監視項目の追加

課題：

CloudWatchの監視が最小構成。

提案内容：

- ALBリクエスト 4xx
- RDS CPU / Connection / FreeStorage
- EC2 StatusCheck
- S3エラー通知

期待効果：

- 障害の早期発見
- 運用品質向上

優先度： 中

💛 4. 優先度まとめ

優先度	改善内容
高	Terraform（IaC化） / WAF
中	NATGW冗長化 / CI-CD / SSM / 監視強化
低	S3ライフサイクル最適化

💜 5. 総括

AWS環境は基本構成としては完成しており、

以下の改善で “実務レベル → 企業利用レベル” へ進化できる：

- 可用性（Multi-AZ / NATGW冗長化）
- セキュリティ（WAF / IAM整備）
- 運用（Patch / CI/CD）
- 自動化（Terraform）

これらを順に導入することで、

継続的に強い構成へブラッシュアップ可能。
