# 2.AWS Build Procedure Document

## 📘 AWS Build Procedure Document – English Version

*Source:*

---

## 🟦 AWS Build Procedure Document

Project: Corporate Website Migration to AWS

Author: Fuma

Created: 2025/11/22

Version: 1.0

---

## 🟩 1. Purpose & Overview

This document describes the full build procedure for a **3-tier architecture (Web / AP / DB)** deployed on AWS.

It is designed to ensure **reproducibility**, clarify design intentions, and serve as a foundation for future IaC adoption (Terraform).

---

## 🟨 2. Environment Information

| Item | Details |
|---|---|
| Region | ap-northeast-1 (Tokyo) |
| AZs | 1a / 1c |
| OS | Amazon Linux 2023 |
| DB Engine | MySQL 8.0 |
| Account | Personal testing environment |
| Build Period | 2025/11/11 – 2025/11/17 |

| Item | Details |
|---|---|
| Purpose | Verification of migrating on-prem 3-tier architecture to AWS |

# 🟧 3. Chapter Overview (Build Structure)

1. Network (VPC / Subnet / Routing / NATGW)
2. Security (IAM / SG / Key Pair)
3. Compute (EC2 build & configuration)
4. Database (RDS build & connectivity)
5. Storage (S3 creation & integration)
6. Load Balancer (ALB configuration & target registration)
7. DNS / Certificate (Route53 / ACM)
8. Monitoring (CloudWatch / SNS)
9. Testing & Validation
10. Issues & Improvements

# 🟩 4. Detailed Build Procedures

## 🧩 4.1 Network Build

### Purpose

To design a secure and scalable VPC with separated Public / Private / DB layers.

### Procedure

1. Create VPC (10.10.0.0/16)
2. Create Public Subnets (10.10.0.0/24, 10.10.1.0/24)
3. Create Private-App Subnets (10.10.10.0/24, 10.10.11.0/24)
4. Create Private-DB Subnets (10.10.20.0/24, 10.10.21.0/24)
5. Create IGW & attach to VPC

6. Create NATGW in 1a and assign Elastic IP
7. Configure Route Tables (Public → IGW, Private → NATGW)

### Verification

✅ Confirm that all configuration parameters match the design

---

## 🧩 4.2 Security Configuration

### Purpose

Implement least-privilege IAM and ensure proper network segmentation.

### Procedure

1. Create IAM user **admin-**(MFA enabled)
2. Create IAM Role **EC2InstanceRole-Web**
    - Policies: `AmazonS3ReadOnlyAccess`, `CloudWatchAgentServerPolicy`
3. Create Security Groups:
    - **ALB-SG:** Allow 80/443 from 0.0.0.0/0
    - **Web-SG:** Allow 80 from ALB-SG, 22 from admin IP (or Session Manager)
    - **DB-SG:** Allow 3306 from Web-SG

### Verification

✅ Confirm correct port rules and trust relationships

---

## 🧩 4.3 RDS Build (DB Layer)

### Purpose

Run MySQL securely and reliably isolated from the Web layer.

### Procedure

1. Create RDS MySQL (db.t3.micro, Multi-AZ enabled)
2. Assign Private-DB subnet group

3. Configure Multi-AZ deployment
4. Set automated backup to 7 days

### Verification

✅ RDS running normally

✅ Multi-AZ status confirmed

✅ Automated backups generated

---

## 🧩 4.4 S3 Configuration (Storage Layer)

### Purpose

Use S3 for static file storage and backup support.

### Procedure

1. Create bucket: `***-*****`
2. Enable versioning
3. Add lifecycle rule (transition to IA after 90 days)
4. Grant S3 access to EC2 IAM role

### Verification

✅ File upload and retrieval successful

---

## 🧩 4.5 ALB Configuration (Load Balancer Layer)

### Purpose

Enable redundancy + HTTPS communication.

### Procedure

1. Create ALB in Public Subnets
2. Register EC2 instances in the Target Group
3. Configure health check (`/index.php`)

### Verification

✅ Both EC2 instances are in **Healthy** state

---

## 🧩 4.6 EC2 Configuration (Web / AP Layer)

### Purpose

Deploy Nginx + PHP environment and run the web application.

### Procedure

1. Launch EC2 (Amazon Linux 2023, t3.small)
2. Configure via user data:
   - Install Nginx
   - Install PHP / PHP-FPM
   - Deploy `index.php`
3. Configure Nginx ( `/etc/nginx/conf.d/default.conf` )
4. Configure PHP-FPM ( `/etc/php-fpm.d/www.conf` )
5. Run & enable services: `nginx` , `php-fpm`

### Verification

✅ EC2 launched successfully

✅ ALB health checks passing

✅ Web page displays correctly

✅ EC2 can connect to RDS

---

## 🧩 4.7 Route53 / ACM Configuration

### Purpose

Enable custom domain + HTTPS termination.

### Procedure

1. Create Route53 Public Hosted Zone
2. Create A-record (Alias → ALB DNS)
3. Issue ACM certificate for `.example.com`

4. Add HTTPS listener to ALB
5. Attach certificate to ALB

### Verification

✅ Access via https://example.com successful

✅ SSL certificate valid

---

## 🧩 4.8 CloudWatch / SNS (Monitoring & Alerts)

### Purpose

Centralize monitoring, logging, and alert notifications.

### Procedure

1. Create CloudWatch alarm (CPUUsage > 80%)
2. Create SNS topic and configure email subscription

### Verification

✅ Alarm tested and triggered

✅ Email notification received

---

## 🧩 4.9 Test & Validation Results

| Test Item | Result | Notes |
|---|---|---|
| ALB access | ✅ OK | HTTPS normal |
| DB connection | ✅ OK | MySQL connected |
| S3 access | ✅ OK | Upload successful |
| CloudWatch alerting | ✅ OK | Mail notification received |

---

## 🧩 4.10 Issues & Improvement Points

| Item | Issue | Improvement |
|---|---|---|
| NATGW single-AZ | No redundancy | Add second NATGW (cost-dependent) |
| Patch management | Manual | Use SSM Patch Manager |

| Item | Issue | Improvement |
|---|---|---|
| Deployment | Manual upload | Introduce CodePipeline |
| Monitoring | Minimal | Add RDS / ALB metrics |

---

## 🟦 5. Attachments

- Screenshots (ALB display, RDS connection, SNS notifications)
- Architecture diagram (draw.io PNG)
- Cost estimation (AWS Pricing Calculator)

---

## 🧠 6. Notes & Learnings

- Faced issues with routing through NATGW
- IAM role changes require EC2 restart to apply
- ALB → EC2 health check is more stable using `/index.php`

---

## 🏁 7. Summary

Through this build, the full lifecycle of AWS design, redundancy, security, and monitoring was understood.

Future steps include **IaC (Terraform)** and **CI/CD implementation**.

---