Hosting a Scalable and Resilient Web Application on AWS with Automation and Backup

Anju Lakshmi S B

July 19, 2025

Project Overview

This project demonstrates the deployment of a highly available and fault-tolerant web application infrastructure on Amazon Web Services (AWS). The setup includes a dynamic website hosting environment with automated scaling, backup, monitoring, and notification mechanisms using various AWS services.

Key Features

- Deployed two web applications on EC2 instances in private subnets A and B.
- Used an Application Load Balancer (ALB) to route traffic to the web applications.
- Integrated EC2 Auto Scaling for handling high traffic and ensuring high availability.
- Created a secure Virtual Private Cloud (VPC) with public and private subnets, and NAT Gateway.
- Configured a MySQL RDS database in Multi-AZ mode for high availability.
- Automated the switch between web applications using AWS Lambda and CloudWatch Events.
- Implemented monitoring using Amazon CloudWatch (metrics, logs, alarms, dashboards).
- Set up notifications through Amazon SNS for web status changes and backup events.

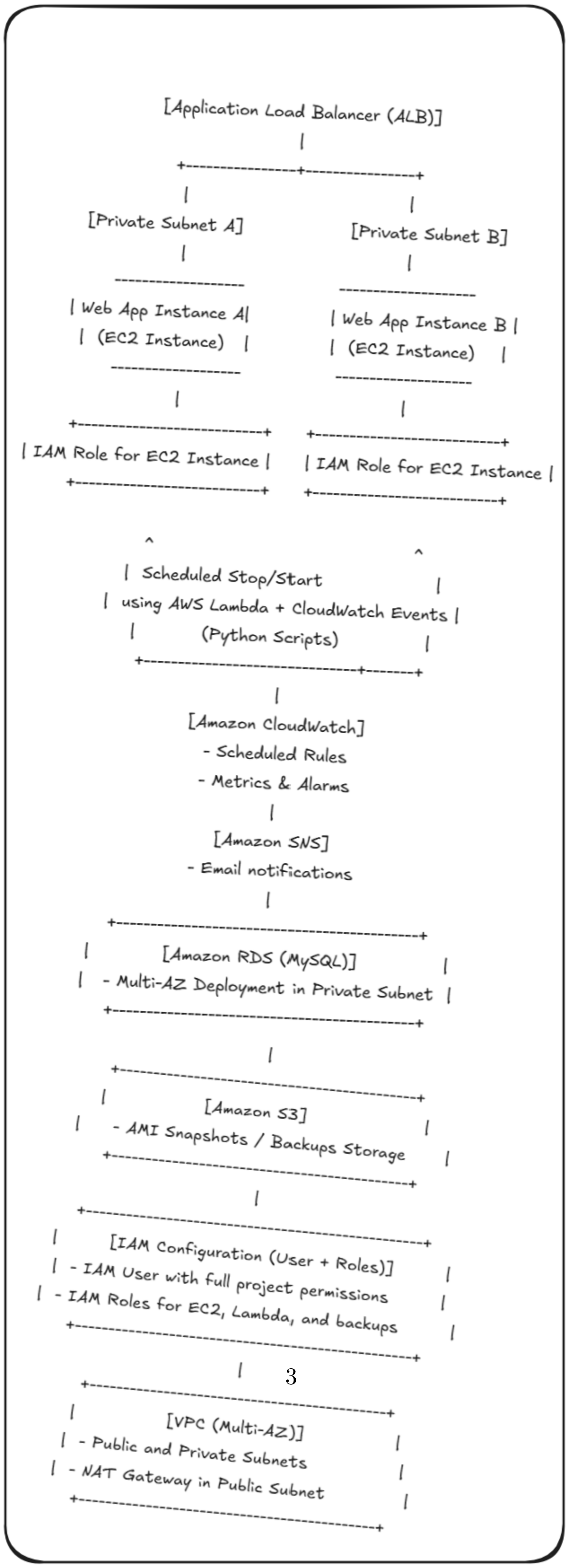
- Performed scheduled AMI snapshots and stored them in Amazon S3 for backup.
- IAM user and roles were created to securely manage resources with least-privilege access.

AWS Services Used

- **EC2** – Hosting the web applications.
- **ALB (Application Load Balancer)** – Load balancing traffic across EC2 instances.
- **VPC** – Secure and isolated networking with subnets and NAT gateway.
- **IAM** – User, role, and policy management.
- **RDS (MySQL)** – Managed relational database with Multi-AZ deployment.
- **S3** – Backup storage for AMI snapshots.
- **SNS** – Email notifications for infrastructure events.
- **CloudWatch** – Monitoring, logging, event triggering.
- **Lambda** – Automated logic to switch web applications and perform backups.

Architecture Overview

- Web App A runs first in Private Subnet A.
- After a specific time (e.g., 20 mins), CloudWatch triggers a Lambda function that stops Web App A and starts Web App B in Private Subnet B.
- SNS notifies the user via email when Web App B is online.
- The ALB continuously routes traffic based on which EC2 instance is active.
- Snapshots of the EC2 instance are regularly created and stored in S3 via automated Lambda functions.


[Application Load Balancer (ALB)]
[Private Subnet A]
[Private Subnet B]
Web App Instance A
(EC2 Instance)
Web App Instance B
(EC2 Instance)
IAM Role for EC2 Instance
IAM Role for EC2 Instance
Scheduled Stop/Start
using AWS Lambda + CloudWatch Events
(Python Scripts)
[Amazon CloudWatch]
- Scheduled Rules
- Metrics & Alarms
[Amazon SNS]
- Email notifications
[Amazon RDS (MySQL)]
- Multi-AZ Deployment in Private Subnet
[Amazon S3]
- AMI Snapshots / Backups Storage
[IAM Configuration (User + Roles)]
- IAM User with full project permissions
- IAM Roles for EC2, Lambda, and backups
[VPC (Multi-AZ)]
- Public and Private Subnets
- NAT Gateway in Public Subnet

Prerequisites

- AWS account with required service permissions
- Basic understanding of AWS networking and EC2
- Optional: Python knowledge for automation logic

1 IAM User Permissions Used in the Project

The following AWS service permissions were granted to the IAM user to build and manage the infrastructure for the project *"Hosting a Scalable and Resilient Web Application on AWS with Automation and Backup"*:

- **EC2 (Elastic Compute Cloud)** – For launching, stopping, and managing instances, AMIs, security groups, and networking.
- **Auto Scaling** – To manage Auto Scaling Groups and dynamically scale web instances based on demand.
- **VPC (Virtual Private Cloud)** – For creating isolated networks with subnets, route tables, internet/NAT gateways.
- **IAM (Identity and Access Management)** – Full access to create roles, attach policies, and assign trust relationships across AWS services.
- **RDS (Relational Database Service)** – To provision and manage the database in a private subnet for secure access.
- **S3 (Simple Storage Service)** – Used to store AMI backups, Lambda code, logs, and other project assets.
- **SSM (AWS Systems Manager)** – Enabled secure instance access and automation without needing SSH keys.
- **SNS (Simple Notification Service)** – Sent email notifications during automation workflows (e.g., instance switch alerts).
- **Lambda** – Executed automation scripts for backup and scheduled website switching.
- **CloudWatch** – Monitored metrics and logs, set alarms, and visualized dashboards.
- **CloudWatch Logs** – Collected and retained log streams from Lambda functions and other services.
- **EventBridge (CloudWatch Events)** – Scheduled Lambda functions using cron expressions for automation.

- **Elastic Load Balancing (ALB)** – Managed incoming HTTP traffic and routed it to healthy EC2 targets in different subnets.
- **EventBridge Schemas** – Accessed via `schemas:ListDiscoverers` to potentially support event-driven development (optional).

2 AWS Lambda Automation Scripts

As part of automation, Lambda functions were created to manage the lifecycle of EC2 instances and to send notifications using Amazon SNS when switching between the web servers hosted in Private Subnet A and Private Subnet B.

2.1 EC2 Start Script (Web Application A)

Listing 1: Lambda Function to Start EC2 Instances

```
import boto3

region = 'us-west-1'
instances = ['i-12345cb6de4f78g9h', 'i-08ce9b2d7eccf6d26']
ec2 = boto3.client('ec2', region_name=region)

def lambda_handler(event, context):
    ec2.start_instances(InstanceIds=instances)
    print('started your instances: ' + str(instances))
```

2.2 EC2 Stop Script (Web Application A)

Listing 2: Lambda Function to Stop EC2 Instances

```
import boto3

region = 'us-west-1'
instances = ['i-12345cb6de4f78g9h', 'i-08ce9b2d7eccf6d26']
ec2 = boto3.client('ec2', region_name=region)

def lambda_handler(event, context):
    ec2.stop_instances(InstanceIds=instances)
    print('stopped your instances: ' + str(instances))
```

2.3 EC2 Start Script for Web B with Email Notification

Listing 3: Start Web Application B and Send SNS Email Notification

```
import boto3

```

```
region = 'us-west-1'
instances = ['i-0abc123def456ghi7']
ec2 = boto3.client('ec2', region_name=region)
sns = boto3.client('sns')

def lambda_handler(event, context):
    ec2.start_instances(InstanceIds=instances)
    print('started your instances: ' + str(instances))

    # Send notification
    response = sns.publish(
        TopicArn='arn:aws:sns:us-west-1:123456789012:
            WebStatusTopic',
        Subject='Web Application B Started',
        Message='The second website (Web B) has been started
             successfully.'
    )
    print('Notification sent:', response)
```

3 CloudWatch Scheduled Events

Two EventBridge (CloudWatch) rules were created manually via the AWS Console:

- **Stop Web A (Website A)**
  Schedule: Every day at 11:55 AM
  Target Lambda Function: `StopWebAFunction`
- **Start Web B (Website B)**
  Schedule: Every day at 12:00 PM
  Target Lambda Function: `StartWebBFunction`

These automate the switch between two private EC2-hosted websites behind an Application Load Balancer.

4 Amazon S3

An S3 bucket was created to store backup data, logs, and snapshots triggered by Lambda automation.

- **Bucket Name:** `your-bucket-name`
- **Used For:**
  - Storing EC2 or RDS snapshot metadata (future extension)
  - Archiving log files or backups (optional)

- **Access Permissions:**
  - Read/Write access granted to the Lambda execution role
  - S3 permissions included in the IAM policy

S3 was integrated as part of the backup strategy to ensure availability and durability of stored data.

5 How to Set Up the Project

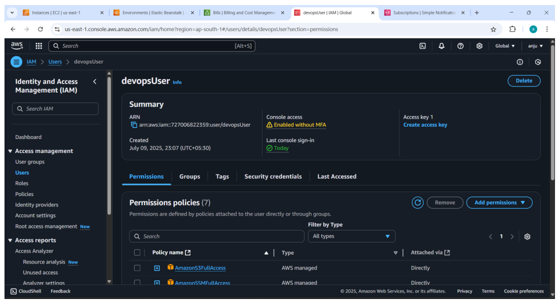
To recreate this project in your AWS account:

1. **Create a VPC** with both public and private subnets.
2. **Launch EC2 Instances**:
   - Web A (runs for 20 minutes)
   - Web B (runs for 5 minutes)
3. **Install Nginx** and configure static websites.
4. **Configure Auto Scaling Group (ASG)** to automatically handle traffic/load.
5. **Create an RDS Instance** in the private subnet for backend data storage.
6. **Deploy Application Load Balancer (ALB)** in the public subnet for traffic distribution.
7. **Create IAM Roles/Policies** for Lambda, EC2, RDS, S3, SNS, and CloudWatch.
8. **Develop Lambda Functions**:
   - `StopWebAFunction`
   - `StartWebBFunction` (includes SNS email notification)
9. **Schedule Events with CloudWatch (EventBridge)**:
   - Stop Web A daily at 11:55 AM
   - Start Web B daily at 12:00 PM
10. **Create an S3 Bucket** for backups/logs (optional).

6 Key Features

- Fully functional VPC with public/private subnets and route tables.
- Auto Scaling Group configured for EC2 instance scale-out/scale-in.
- RDS MySQL database connected securely from private EC2 instance.
- Application Load Balancer routing dynamic traffic.
- Scheduled Lambda functions to stop/start private websites.
- Email notification using SNS on EC2 instance events.
- IAM roles and policies customized for least-privilege.
- S3 integrated for log/archive purposes (optional).

7 Screenshot



us-east-1.console.aws.amazon.com/iam/home?region=ap-south-1#/users/details/devopsUser?section=permissions

IAM > Users > devopsUser

devopsUser

Summary

ARN: arn:aws:iam::727006822359:user/devopsUser
Console access: Enabled without MFA
Access key 1: Create access key
Created: July 09, 2025, 23:07 (UTC+05:30)
Last console sign-in: Today

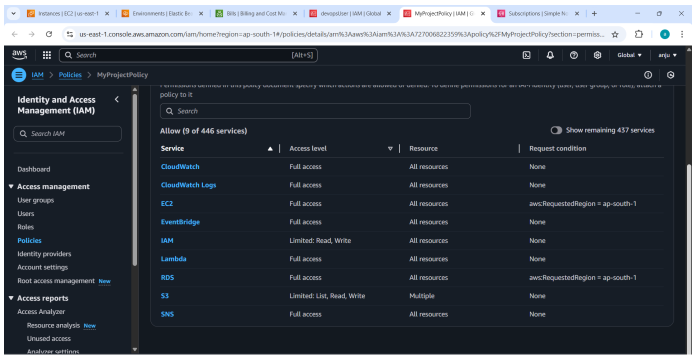
Permissions policies (7)

Permissions are defined by policies attached to the user directly or through groups.

Filter by Type: All types

Policy name	Type	Attached via
AmazonS3FullAccess	AWS managed	Directly
AmazonSSMFullAccess	AWS managed	Directly

Figure 1: IAM User



us-east-1.console.aws.amazon.com/iam/home?region=ap-south-1#/policies/details/arn%3Aaws%3Aiam%3A%3A727006822359%3Apolicy%2FMyProjectPolicy?section=permiss...

IAM > Policies > MyProjectPolicy

policy to it

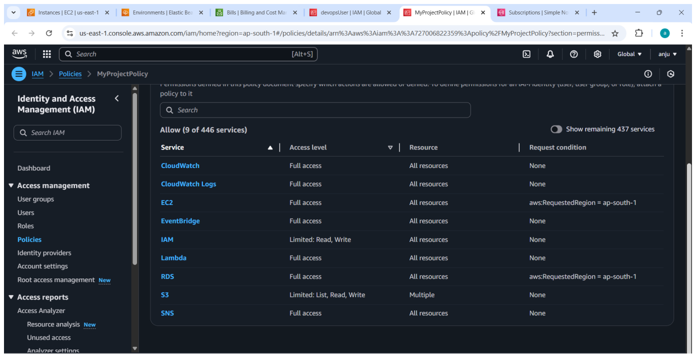
Allow (9 of 446 services)

Show remaining 437 services

Service	Access level	Resource	Request condition
CloudWatch	Full access	All resources	None
CloudWatch Logs	Full access	All resources	None
EC2	Full access	All resources	aws:RequestedRegion = ap-south-1
EventBridge	Full access	All resources	None
IAM	Limited: Read, Write	All resources	None
Lambda	Full access	All resources	None
RDS	Full access	All resources	aws:RequestedRegion = ap-south-1
S3	Limited: List, Read, Write	Multiple	None
SNS	Full access	All resources	None

Instances | EC2 | us-east-1
Environments | Elastic Bea
Bills | Billing and Cost Ma
devopsUser | IAM | Global
MyProjectPolicy | IAM | Gl
Subscriptions | Simple No
Search [Alt+S]
Global
anju
Identity and Access Management (IAM)
Search IAM
Dashboard
Access management
User groups
Users
Roles
Policies
Identity providers
Account settings
Root access management New
Access reports
Access Analyzer
Resource analysis New
Unused access
Analyzer settings
Search

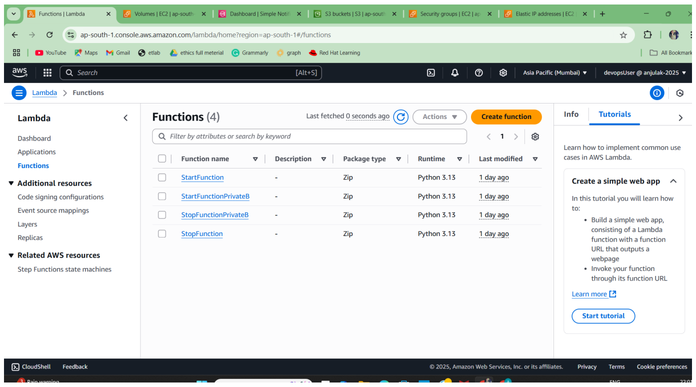
Figure 2: IAM User Policies attached



Allow (9 of 446 services)

Service	Access level	Resource	Request condition
CloudWatch	Full access	All resources	None
CloudWatch Logs	Full access	All resources	None
EC2	Full access	All resources	aws:RequestedRegion = ap-south-1
EventBridge	Full access	All resources	None
IAM	Limited: Read, Write	All resources	None
Lambda	Full access	All resources	None
RDS	Full access	All resources	aws:RequestedRegion = ap-south-1
S3	Limited: List, Read, Write	Multiple	None
SNS	Full access	All resources	None

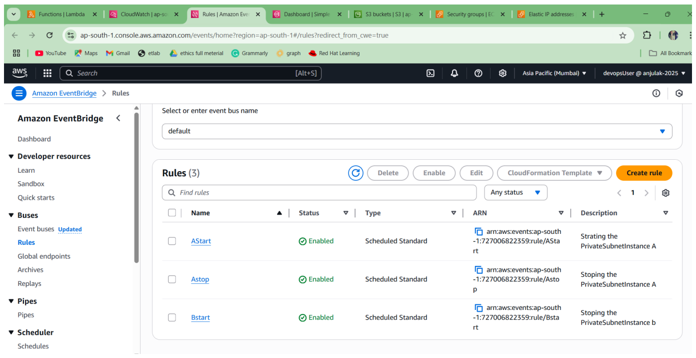
Figure 3: IAM User Policies attached



Functions (4)

Function name	Description	Package type	Runtime	Last modified
StartFunction	-	Zip	Python 3.13	1 day ago
StartFunctionPrivateB	-	Zip	Python 3.13	1 day ago
StopFunctionPrivateB	-	Zip	Python 3.13	1 day ago
StopFunction	-	Zip	Python 3.13	1 day ago

Figure 4: Lambda Function

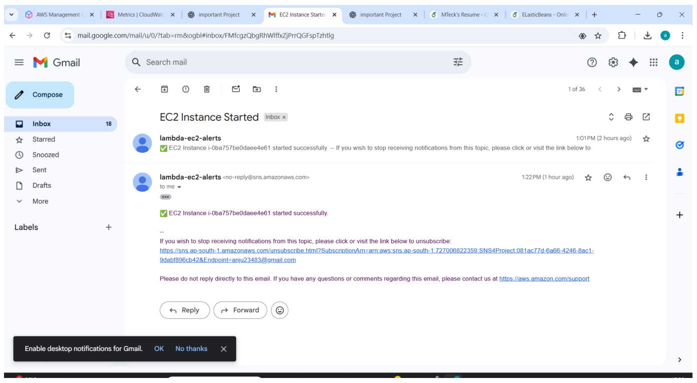


Select or enter event bus name: default

Rules (3)

Name	Status	Type	ARN	Description
AStart	Enabled	Scheduled Standard	arn:aws:events:ap-south-1:727006822359:rule/AStart	Strating the PrivateSubnetInstance A
Astop	Enabled	Scheduled Standard	arn:aws:events:ap-south-1:727006822359:rule/Astop	Stoping the PrivateSubnetInstance A
Bstart	Enabled	Scheduled Standard	arn:aws:events:ap-south-1:727006822359:rule/Bstart	Stoping the PrivateSubnetInstance b

Figure 5: Cloud Watch Schedule


EC2 Instance Started

lambda-ec2-alerts 1:01 PM (2 hours ago)
✅ EC2 Instance i-0ba757be0daee4e61 started successfully. -- If you wish to stop receiving notifications from this topic, please click or visit the link below to

lambda-ec2-alerts <no-reply@sns.amazonaws.com> 1:22 PM (1 hour ago)
to me

✅ EC2 Instance i-0ba757be0daee4e61 started successfully.

--
If you wish to stop receiving notifications from this topic, please click or visit the link below to unsubscribe:
https://sns.ap-south-1.amazonaws.com/unsubscribe.html?SubscriptionArn=arn:aws:sns:ap-south-1:727006822359:SNS4Project:081ac77d-6a66-4246-8ac1-9dabf896cb42&Endpoint=anju23483@gmail.com

Please do not reply directly to this email. If you have any questions or comments regarding this email, please contact us at https://aws.amazon.com/support

mail.google.com/mail/u/0/?tab=rm&ogbl#inbox/FMfcgzQbgRhWlffxZjPrrQGFspTzhtlg
Gmail
Compose
Inbox 18
Starred
Snoozed
Sent
Drafts
More
Labels
Reply
Forward
Enable desktop notifications for Gmail. OK No thanks

Figure 6: Mail notification when Private Web B is Started using Cloud Watch Schedule

8 Future Improvements

- Integrate Route 53 for custom domain support.
- Add Lambda backup scripts to export RDS/EC2 snapshots to S3.
- Improve security with tighter IAM role boundaries and VPC flow logs.
- Automate setup with Terraform or CloudFormation.

Author

Created by**Anju Lakshmi S B**
Contact: sblakshmianjusb.09@gmail.com