**AWS Cloud Deployment Documentation for Nana Cafe Ordering System**

**1. Overview**

Nana Cafe's online ordering system has been migrated from a local on-premises setup to AWS for better **scalability, security, and availability**. The cloud solution leverages **EC2, RDS, S3, Auto Scaling, Load Balancing, CloudWatch, and AWS Systems Manager (SSM)** to ensure optimal performance while maintaining cost efficiency.

This document outlines the **AWS services implemented, the setup process, modifications made to the original code, and security best practices**.

**2. AWS Cloud Architecture**

**Cloud Components**

* **Amazon EC2:** Hosts the PHP-based website.
* **Amazon RDS (MySQL):** Manages the relational database for storing orders and user data.
* **Amazon S3:** Stores static assets such as images and CSS files.
* **Application Load Balancer (ALB):** Distributes traffic across multiple EC2 instances.
* **Auto Scaling Group:** Ensures scalability by launching or terminating instances based on demand.
* **AWS Systems Manager (SSM):** Securely stores database credentials.
* **CloudWatch:** Monitors performance and sends alerts for critical issues.
* **AWS WAF (Web ACL):** Protects against security threats such as SQL injection and DDoS attacks.

**Cloud Diagram**

Internet

│

┌─────────▼─────────┐

│ Route 53 (DNS) │

└─────────┬─────────┘

│

┌─────────▼─────────┐

│ Load Balancer │ ← Web ACL (Security)

└─────────┬─────────┘

│

┌──────────┴──────────┐

│ Auto Scaling │

├────────┬────────────┤

│ EC2-1 │ EC2-2 ...│

└────────┴────────────┘

│ │

▼ ▼

┌────────────────────┐

│ RDS MySQL Database │

└────────────────────┘

│

▼

┌──────────────────┐

│ S3 (Static Files)│

└──────────────────┘

**3. AWS Services Implementation**

**3.1 EC2 Instance (Web Server)**

**Steps Performed:**

1. Launched an **Amazon Linux 2** instance.
2. Installed required packages:

sudo yum update -y

sudo yum install -y httpd php php-mysqli php-json php-xml unzip

1. Started and enabled Apache:

sudo systemctl start httpd

sudo systemctl enable httpd

1. **Deployed PHP Application**:
   * Copied files to /var/www/html/ using SCP:

scp -i your-key.pem -r nana-cafe/ ec2-user@your-ec2-public-ip:/var/www/html/

* + Adjusted permissions:

sudo chown -R apache:apache /var/www/html/

sudo chmod -R 755 /var/www/html/

**3.2 Amazon RDS (MySQL)**

1. **Created a MySQL RDS instance.**
2. **Configured Security Groups** to allow access **only from EC2 instances**.
3. **Disabled Public Access** to enhance security.
4. **Retrieved RDS Endpoint:**

aws rds describe-db-instances --query "DBInstances[\*].Endpoint.Address"

1. **Connected from EC2 and imported database:**

mysql -h your-rds-endpoint -u your-username -p < wings.sql

**3.3 AWS SSM Parameter Store (Secure Credentials)**

To **avoid storing credentials in code**, database credentials were stored securely in AWS SSM.

sh

CopyEdit

aws ssm put-parameter --name "/nana-cafe/endpoint" --value "your-rds-endpoint" --type "String"

aws ssm put-parameter --name "/nana-cafe/username" --value "your-db-username" --type "String"

aws ssm put-parameter --name "/nana-cafe/password" --value "your-db-password" --type "SecureString"

aws ssm put-parameter --name "/nana-cafe/database" --value "wings" --type "String"

**Updated get-parameters.php to fetch credentials dynamically:**

use Aws\Ssm\SsmClient;

require 'vendor/autoload.php';

$ssm\_client = new SsmClient([

'version' => 'latest',

'region' => 'us-east-1',

]);

$result = $ssm\_client->getParametersByPath([

'Path' => '/nana-cafe/',

'WithDecryption' => true

]);

foreach ($result['Parameters'] as $p) {

$values[str\_replace('/nana-cafe/', '', $p['Name'])] = $p['Value'];

}

$ep = $values['endpoint'] ?? '';

$un = $values['username'] ?? '';

$pw = $values['password'] ?? '';

$db = $values['database'] ?? '';

**3.4 Load Balancer & Auto Scaling**

1. **Created an Application Load Balancer (ALB).**
2. **Registered EC2 instances to the ALB Target Group.**
3. **Configured Health Checks** for instance monitoring.
4. **Created an Auto Scaling Group**:
   * Min: **1**, Max: **4** instances.
   * Configured a Launch Template with **User Data**:

#!/bin/bash

# Update system

sudo yum update -y

# Install necessary packages

sudo yum install -y httpd php php-mysqli php-json php-xml unzip aws-cli cronie

# Configure AWS CLI with static credentials (not recommended in production)

aws configure set aws\_access\_key\_id "YOUR\_ACCESS\_KEY"

aws configure set aws\_secret\_access\_key "YOUR\_SECRET\_KEY"

aws configure set aws\_session\_token "YOUR\_SESSION\_TOKEN"

aws configure set default.region us-east-1

# Start and enable Apache

sudo systemctl start httpd

sudo systemctl enable httpd

# Sync files from S3 to the web server

cd /var/www/html

aws s3 sync s3://tp070587bucket/ /var/www/html/ --delete

# Set permissions

sudo chown -R apache:apache /var/www/html/

sudo chmod -R 755 /var/www/html/

# Add cron job to sync files from S3 every 5 minutes

(crontab -l 2>/dev/null; echo "\*/5 \* \* \* \* aws s3 sync s3://tp070587bucket/ /var/www/html/ --delete") | crontab -

# Start and enable the cron service

sudo systemctl start crond

sudo systemctl enable crond

# Restart Apache

sudo systemctl restart httpd

**3.5 S3 for Static Assets**

**Steps Performed:**

1. **Created an S3 bucket** for images & stylesheets.
2. **Uploaded style.css and images**.
3. **Updated paths in the code to use S3 URLs:**

body {

background: url(https://tp070587bucket.s3.us-east-1.amazonaws.com/body1.jpg) repeat-x center top;

}

**3.6 CloudWatch (Monitoring)**

1. **Enabled CloudWatch Alarms** for:
   * High CPU Usage.
   * RDS connection failures.
2. **Configured Log Monitoring** for Apache logs.

**3.7 AWS Web ACL (Security)**

* **Enabled AWS WAF (Web Application Firewall).**
* **Rules Implemented:**
  + Block SQL Injection.
  + Block Malicious IPs.
  + Allow only specific geolocations.

**4. Modifications to the Original Code**

* **Replaced MySQL with MySQLi for compatibility.**
* **Removed hardcoded database credentials** (Now retrieved from SSM).
* **Updated PHP Session Management** to handle multiple EC2 instances.
* **Integrated S3 for static files.**
* **Updated Auto Scaling User Data** to sync latest website files.

**5. AWS Cost Estimation Breakdown**

To estimate the monthly cost of running Nana Cafe’s online ordering system in AWS, we consider the following services:

**AWS Services Used**

|  |  |  |
| --- | --- | --- |
| Service | Instance/Usage | Estimated Cost (USD/Month) |
| EC2 Instances | 2 t2.micro (On-Demand, Linux) | $16.94 ($8.47 \* 2) |
| Amazon RDS | 1 db.t3.micro (MySQL) – 20 GB GP2 SSD (Free Tier) | $0 ✅ (750 hrs/month Free) |
| RDS Storage | (Free Tier) | $0 ✅ |
| Application Load Balancer | 1 ALB + 1 Target Group | $16.00 (Refined LCU pricing) |
| åAuto Scaling | Scaling between 1–2 instances | Included in EC2 costs |
| Amazon S3 | 20GB Standard Storage | $0.47 |
| AWS Systems Manager (SSM) | Parameter Store usage | Free (Standard Tier) |
| CloudWatch | Basic Monitoring Only | $3.00 |

**Total Estimated Cost: $36.41 per month**

**6. Cost Optimization Recommendations**

1. **Use EC2 Spot Instances for Auto Scaling:** Can reduce EC2 costs by 50–70%.
2. **Enable RDS Auto Stop Feature:** Stops the database during inactive hours.
3. **Move Static Files to CloudFront CDN:** Reduces S3 and ALB request costs.
4. **Monitor Usage with CloudWatch Alarms:** Helps optimize instance sizes.