Project Summary / Documentation

Project Title:
WordPress Dev & Production Deployment on AWS with Monitoring and Scheduled Dev Environment

Objective:
To set up a production and development WordPress environment on AWS using CloudFormation, ensuring monitoring, cost-efficient scheduling, and operational readiness.

Problem Statement:

- • Configure a WordPress instance using AWS CloudFormation
- • Set up a live WordPress instance for production
- • Set up a development/testing instance that runs only during business hours (9 AM–6 PM)
- • Ensure monitoring of instance metrics

Tools & Technologies Used:

- • AWS Console - Manual verification, monitoring, deployment
- • AWS CloudFormation - IaC, reproducible infrastructure deployment
- • EC2 (t2.micro)
- • Amazon Linux 2023
- • PHP, Apache, MariaDB
- • CloudWatch (metrics) - Monitoring, scheduled actions
- • IAM roles - Secure access for monitoring and automation
- • Auto Scaling (for Prod & Dev environment)

Architecture

- Production Instance
  - 1 EC2 instance (t2.micro)
  - Apache + PHP + local MySQL
  - Public IP
  - CloudWatch monitoring
- Development Instance
  - Same setup
  - Auto start at 9 AM, stop at 6 PM (EventBridge Scheduler)
  - Not publicly indexed

Create CloudFormation Template

1. Open AWS Console → CloudFormation → Create Stack → “Template is ready”
2. Use the YAML template provided earlier (or the improved version)
3. Parameters to provide:
   - KeyPairName: Your existing EC2 KeyPair
   - AMI ID: Latest Amazon Linux 2 AMI (for your region)
4. Deploy stack for production first

Verify EC2 and Network Setup

1. Create Key-Pair

2	2	Go to EC2 → Instances
3	3	Check that your instance is running
4	4	Ensure Security Group allows: ◦ SSH (port 22) ◦ HTTP (port 80)
5	5	Check that Public IP is assigned

Choose an Amazon Machine Image (AMI)

Use SSM Parameter to Always Get Latest Amazon Linux 2 AMI

This avoids manually updating your template:

Sample code to get latest AMI, add below to Wordpress yaml template:

```
WordPressInstance:
  Type: AWS::EC2::Instance
  Properties:
    ImageId: !Sub "{{resolve:ssm:/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2}}"
    InstanceType: t2.micro
    KeyName: !Ref KeyPairName
    ...
```

- This dynamically resolves to the latest Amazon Linux 2 AMI for the region where you deploy.
- No need to hardcode AMI IDs.

Implementation Steps:

Step 1 — Launch WordPress Instances via CloudFormation

- CloudFormation stack created with parameters:
  - VPC and Public Subnet selection
  - Key Pair for SSH access
  - Latest Amazon Linux 2023 AMI via SSM Parameter
- Security Group configured for HTTP (80) and SSH (22)
- UserData script installs Apache, PHP, MariaDB, and WordPress
- IAM role attached for CloudWatch monitoring

Step 2 — Verify WordPress Deployment

- Prod and Dev instances reachable via:
- Prod Instance: http://34.204.50.59/wordpress/
- Dev-Instance: http://54.84.44.103/wordpress

Step 3 - IAM Roles attached to Dev and Prod instances

EC2-CloudWatch-Role:
Role is now attached to dev & prod EC2 instances.

Step 4 - Create an AMI of the WordPress Instance for Auto Scaling

Purpose: Capture a ready-to-use image of the instance (with WordPress, PHP, Apache, MariaDB configured) for auto-scaling or backups.

Steps:

1. Go to EC2 → Instances → Select your WordPress instance (Dev or Prod)
2. Click Actions → Image → Create Image

- AMIs created for both Dev and Prod instances:
  - Prod AMI: wordpress-prod-ami
  - Dev AMI: wordpress-dev-ami

Step 5 - Configure Auto Scaling to Launch New WordPress Instances

Purpose: Ensure your environment can scale automatically (for high availability or testing scaling behavior).

Steps:

1. 1. A. Create a Launch Template:
    - AMI: Select the AMI you just created
    - Instance Type: t2.micro (or your choice)
    - Key Pair: Same as your original instance
    - Security Group: Use the same WordPress SG
    - IAM Role: CloudWatch monitoring role
    - Optional: Add UserData script if you want extra configuration at boot

```
UserData:
  Fn::Base64: !Sub |
    #!/bin/bash
    dnf update -y
    dnf install -y httpd wget php-fpm php-mysqli php-json php php-devel
php-mysqlnd mariadb105-server mariadb105
    systemctl start httpd
    systemctl enable httpd
    systemctl start mariadb
    systemctl enable mariadb
    mysql -e "CREATE DATABASE wordpress;"
    mysql -e "CREATE USER 'wpuser'@'localhost' IDENTIFIED BY
'wppassword';"
    mysql -e "GRANT ALL PRIVILEGES ON wordpress.* TO
'wpuser'@'localhost'; FLUSH PRIVILEGES;"
    cd /var/www/html
    wget https://wordpress.org/latest.tar.gz
```

```
tar -xzf latest.tar.gz
chown -R apache:apache wordpress
```

1

2 2 B. Create an Auto Scaling Group (ASG):

- - VPC & Subnets: Same as your existing instance (public subnet for HTTP access)
- - Desired Capacity: 1 (start with single instance)
- - Min / Max Capacity: Min = 1, Max = 3 (for example)

3 3 Test Auto Scaling:

- - Increase desired capacity → new WordPress instance launches automatically
- - Verify it's accessible via the public IP or DNS

Prod Environment: Automatically deploy new WordPress instances using the prebuilt AMI.

C. Configure Dev Auto Scaling and enable automatic start/stop on schedule.

- Dev instance converted to Auto Scaling Group (ASG)
- Launch Template created from wordpress-dev-ami
- Scheduled Actions added:
  - - Start at 9 AM (desired capacity = 1)
  - - Stop at 6 PM (desired capacity = 0)
- - Ensures cost-efficient Dev environment

Step 6 - Configure Monitoring

- • IAM Role EC2-CloudWatch-Role attached to all

instances

- CloudWatch Agent installed on the Prod EC2 instance for memory and disk metrics
- Metrics verified:
  - CPUUtilization, DiskReadOps,, NetworkIn/Out
  - Memory and Disk %
-

Step 6 — Verification

- Both environments fully operational and monitored
- CloudWatch metrics visible for all EC2 instances
- Scheduled start/stop verified for Dev

- Verify EC2 Instance Launched by ASG
  - Terminate any running ASG-managed Prod instance (if needed)
  - ASG will automatically launch a new instance using latest Launch Template version
  - Confirm IAM Role column → should now show EC2-CloudWatch-Role

Course-End Project is fully functional:

- Dev and Prod WordPress instances up and running
- Database configured and WordPress accessible
- IAM role attached for CloudWatch
- CloudWatch monitoring working (CPU, memory, disk)
- Dev instance scheduled (either via EventBridge or ASG)
- Optional Auto Scaling implemented for Dev (and optionally Prod)
- AMIs created for rapid redeployment
- Security groups and networking correct