**Set Up and Monitor a WordPress Instance**

Course-end Project 1

Description

To set up and monitor a WordPress instance for your organization to establish a reliable and secure online presence that supports your business or organizational goals

**Problem Statement:**

You are given a project. You should be able to configure a WordPress instance using AWS CloudFormation and monitor the instance

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Detailed Step-by-Step Process**

**Phase 1: Preparation**

1. **Set up AWS Account & Access**
   * Create or use an existing AWS account.
   * Configure IAM user with appropriate permissions.
   * Install and configure AWS CLI.
2. **Set up Git & GitHub**
   * Create a new repository (e.g., wordpress-deployment).
   * Store your CloudFormation templates, Dockerfiles, and Jenkinsfiles here.
3. **Prepare Docker Image**
   * Create a Dockerfile for WordPress setup (custom plugins, themes, configs).
   * Push Docker image to **Amazon ECR**.

**Phase 2: Infrastructure as Code (CloudFormation)**

1. **Write CloudFormation Template (YAML/JSON)**
   * Resources to define:
     + **VPC** with public/private subnets.
     + **Security Groups** (HTTP, HTTPS, SSH access).
     + **EC2 Instance(s)** for WordPress.
     + **RDS MySQL** instance for the database.
     + **Elastic IP** for production.
     + **Auto Scaling** (optional).
     + **CloudWatch Alarms** for monitoring.
   * Parameters for easy modification (Instance type, KeyName, etc.).
2. **Deploy the Stack**
3. aws cloudformation create-stack --stack-name wordpress-stack \
4. --template-body file://wordpress.yaml \
5. --capabilities CAPABILITY\_IAM
6. **Verify Resources**
   * Access EC2, RDS, Security Groups in AWS Console.
   * Confirm WordPress is running via browser (use Elastic IP or Public DNS).

**Phase 3: Application Configuration**

1. **Access the EC2 Instance**
   * SSH into the instance and verify Docker + WordPress container running.
   * Connect WordPress to RDS DB.
   * Configure themes, plugins, and initial blog setup.
2. **Set Up Two Environments**
   * **Production Instance:** Full-time running instance.
   * **Development Instance:** Use AWS Lambda or Instance Scheduler to start/stop EC2 between 9 AM–6 PM.
3. **Automation for Dev Instance**
   * Create a **Lambda Function** with CloudWatch Event rule:
     + Start EC2 at **9 AM**.
     + Stop EC2 at **6 PM**.
   * Example (Python boto3 script in Lambda):
   * import boto3
   * ec2 = boto3.client('ec2')
   * ec2.start\_instances(InstanceIds=['i-xxxxxxxxxx'])

**Phase 4: CI/CD Integration**

1. **Set Up Jenkins**
   * Install Jenkins on a separate EC2 or Docker container.
   * Configure GitHub Webhook → Jenkins job trigger.
2. **Create Jenkins Pipeline**
   * Clone GitHub repo.
   * Build Docker image.
   * Push to ECR.
   * Deploy updated container to EC2 (via SSH or Ansible script).
3. **Sample Jenkinsfile**
4. pipeline {
5. agent any
6. stages {
7. stage('Clone') {
8. steps {
9. git 'https://github.com/yourrepo/wordpress-deployment.git'
10. }
11. }
12. stage('Build Image') {
13. steps {
14. sh 'docker build -t wordpress-custom .'
15. }
16. }
17. stage('Push to ECR') {
18. steps {
19. sh './push-to-ecr.sh'
20. }
21. }
22. stage('Deploy') {
23. steps {
24. sh 'ssh ec2-user@prod-server "docker pull <ECR\_URI> && docker run -d -p 80:80 <ECR\_URI>"'
25. }
26. }
27. }
28. }

**Phase 5: Monitoring and Alerts**

1. **Set Up CloudWatch Metrics**
   * Monitor EC2:
     + CPU utilization
     + Disk read/write
     + Network I/O
   * Set thresholds (e.g., CPU > 70% → send alert).
2. **Create CloudWatch Alarms**
   * Send notifications via **SNS** topic (email/SMS).
3. **Enable Logging**
   * Send EC2 system logs and Docker container logs to **CloudWatch Logs**.
   * Enable RDS performance insights.
4. **Optional**: Set up a **Dashboard**
   * Use CloudWatch Dashboard to visualize uptime, performance, and resource usage.

**Phase 6: Validation & Reporting**

1. **Validate Deployment**
   * Access both production and dev URLs.
   * Test CI/CD deployment with sample update.
   * Test monitoring alerts.
2. **Document Setup**
   * Include:
     + Architecture diagram.
     + Steps for deployment.
     + Monitoring configuration.
     + Cost optimization notes (dev shutdown schedule).