Report on Week 1 AWS Cloud Task: Deploying Application in Monolithic and Microservices Architectures

1. Introduction

This report details the steps taken to complete the Week 1 AWS Cloud Task of deploying an application using both monolithic and microservices architectures. The task involved setting up and configuring EC2 instances, deploying WordPress and MySQL, and ensuring proper security measures were in place. The report also includes resources consulted during the process and relevant screenshots of the AWS services used.

2. Prerequisites

- Created a free-tier AWS account to access the necessary AWS resources.
- Ensured that AWS CLI and necessary tools were set up for remote management of instances.

3. Task Overview

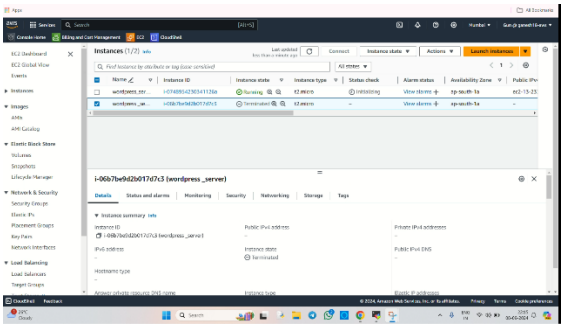
The task was divided into two parts:

1. **Monolithic Architecture**: Deploy WordPress and MySQL on a single EC2 instance.
2. **Microservices Architecture**: Deploy WordPress on one EC2 instance and MySQL on a separate EC2 instance.

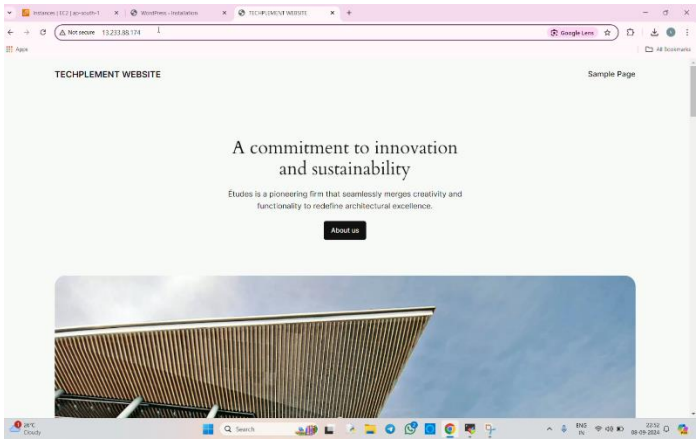
4. Monolithic Architecture Deployment

Steps Taken:

1. Launched an EC2 instance of type t2. micro using the **Ubuntu** AMI (ubuntu-*).
2. Configured the necessary security group to allow HTTP (port 80) and HTTPS (port 443) traffic.
3. Connected to the EC2 instance via SSH and installed the required packages:
    - Installed Apache, PHP, and WordPress.
    - Installed MySQL server on the same instance and configured it for local access.
4. Configured WordPress to connect to the locally installed MySQL database.
5. Created a welcome page in WordPress to serve as the homepage.
6. Tested the setup to ensure both WordPress and MySQL were running correctly on the same instance.



- *Screenshot of the EC2 instance details for monolithic architecture)*

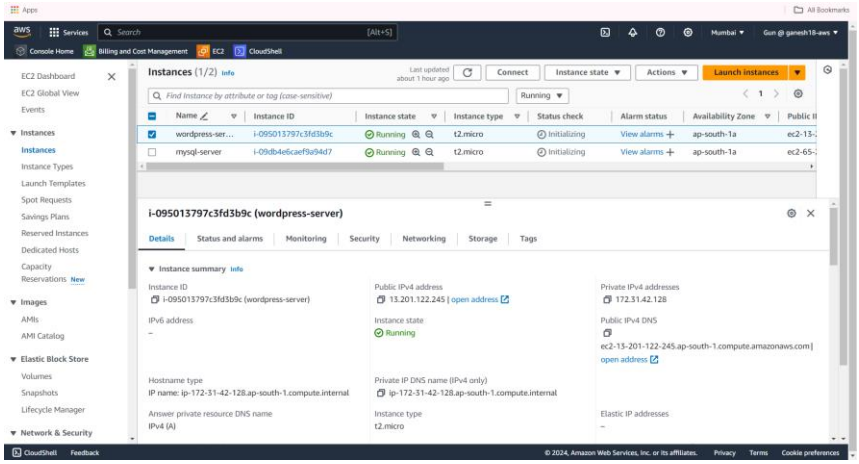


- *Screenshot of the WordPress welcome page for monolithic architecture)*

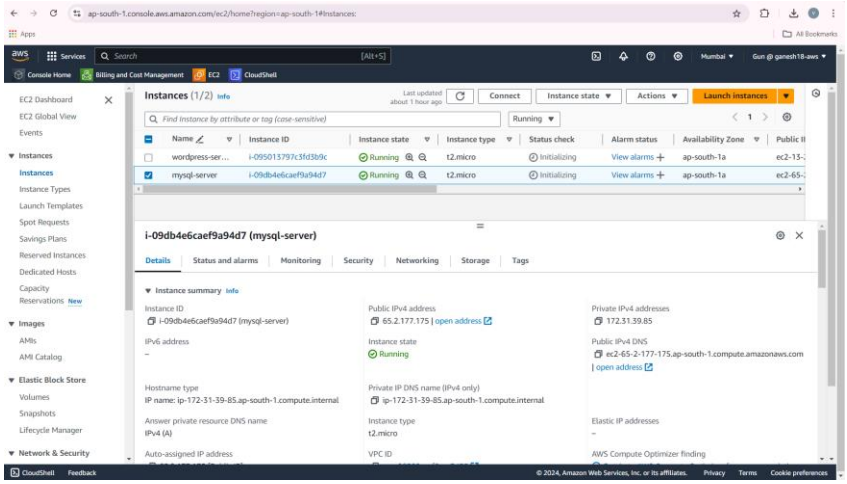
5. Microservices Architecture Deployment

Steps Taken:

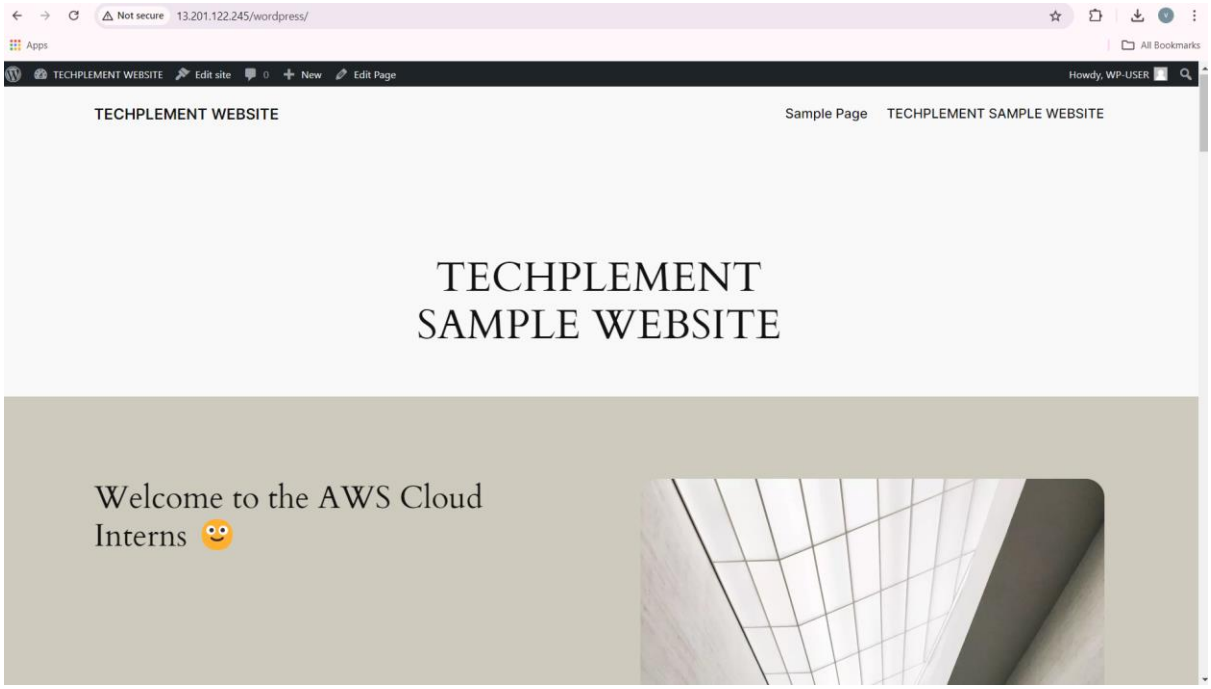
1. Launched two separate EC2 instances of type t2. micro using the **Ubuntu** AMI (ubuntu-*):
    - Instance 1: For WordPress.
    - Instance 2: For MySQL.
2. Configured security groups for both instances:
    - Instance 1 (WordPress): Allowed HTTP (port 80) traffic and HTTPS (port 443).
    - Instance 2 (MySQL): Allowed MySQL (port 3306) traffic and restricted access to the IP of the WordPress instance.
3. Connected to the EC2 instances via SSH:
    - On Instance 1, installed Apache, PHP, and WordPress.
    - On Instance 2, installed MySQL server and configured it to accept remote connections from the WordPress instance.
4. Updated the WordPress configuration on Instance 1 to connect to the MySQL database on Instance 2.
5. Created a welcome page in WordPress to serve as the homepage.
6. Tested the setup to ensure that WordPress on Instance 1 could successfully communicate with MySQL on Instance 2.



- *Screenshot of the EC2 instance details for microservices architecture - WordPress instance)*



- *Screenshot of the EC2 instance details for microservices architecture - MySQL instance)*

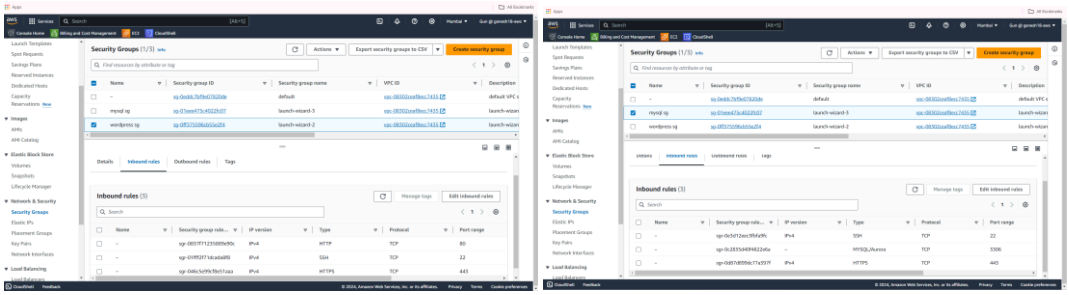


- *Screenshot of the WordPress welcome page for microservices architecture*

6. Security Group Configuration

Details:

- Configured security groups to allow appropriate traffic between the EC2 instances and to restrict unnecessary access:
    - Allowed SSH (port 22) access from my IP.
    - Allowed HTTP (port 80) access for web traffic.
    - Restricted MySQL (port 3306) access to only the IP address of the WordPress instance in the microservices setup.



- *Screenshots of security groups configuration for both monolithic and microservices setups)*

7. Conclusion

The deployment of the application using both monolithic and microservices architectures on AWS was successfully completed. This task provided valuable experience in setting up and configuring AWS resources, managing EC2 instances, and deploying applications using different architectural approaches.

8. Additional Notes

- All EC2 instances were stopped after verification to avoid incurring unnecessary costs.
- Kindly evaluate through recording videos, images in the report.
- All the commands executed were uploaded in the git repository.