AWS CLOUD INTERNSHIP TASK

Name:- Kumari Shweta Swasti(OL/TP2789)

Designation:- Cloud(AWS) Intern.

Date:- 08/11/2024

AWS Deployment of WordPress and MySQL: Monolithic vs. Microservices Architectures.

Table of Contents:-

1. Introduction

In today's cloud-native application landscape, developers can choose between a **monolithic architecture** or a **microservices architecture** for deploying web applications like WordPress. This document explores both approaches for deploying **WordPress** and **MySQL** using Amazon Web Services (**AWS EC2**).

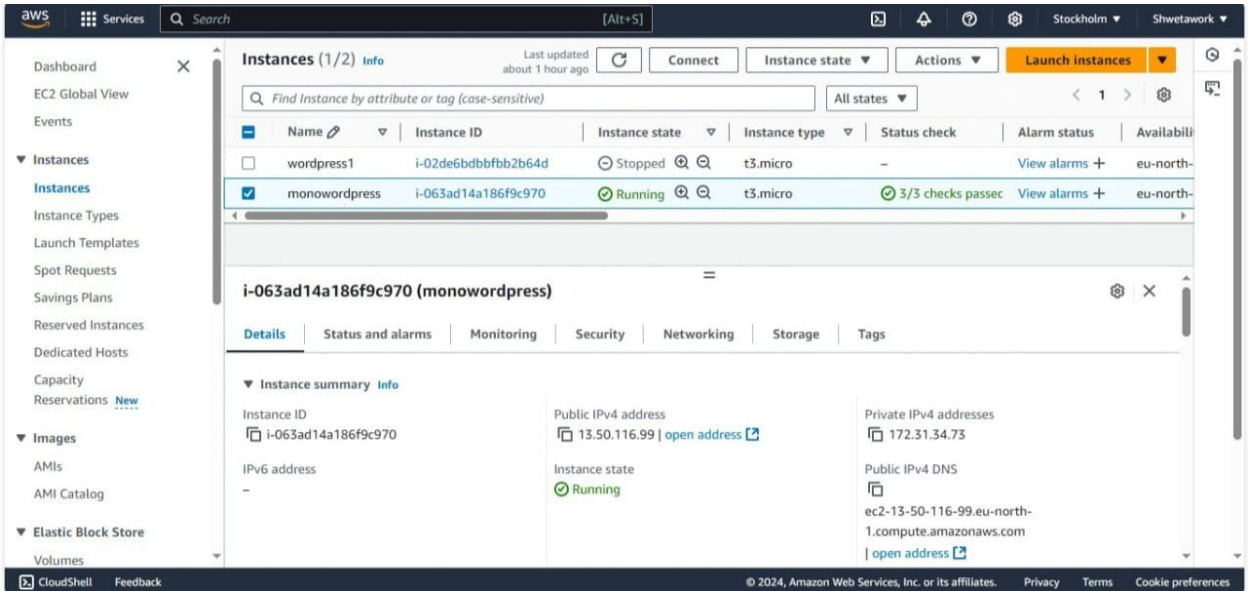
---

2. Steps to Deploy WordPress and MySQL

1 Monolithic Architecture Deployment

In the **monolithic architecture**, both WordPress and MySQL are deployed on a single EC2 instance. This is an easy-to-manage and cost-effective approach suitable for smaller applications. The steps for deployment include:

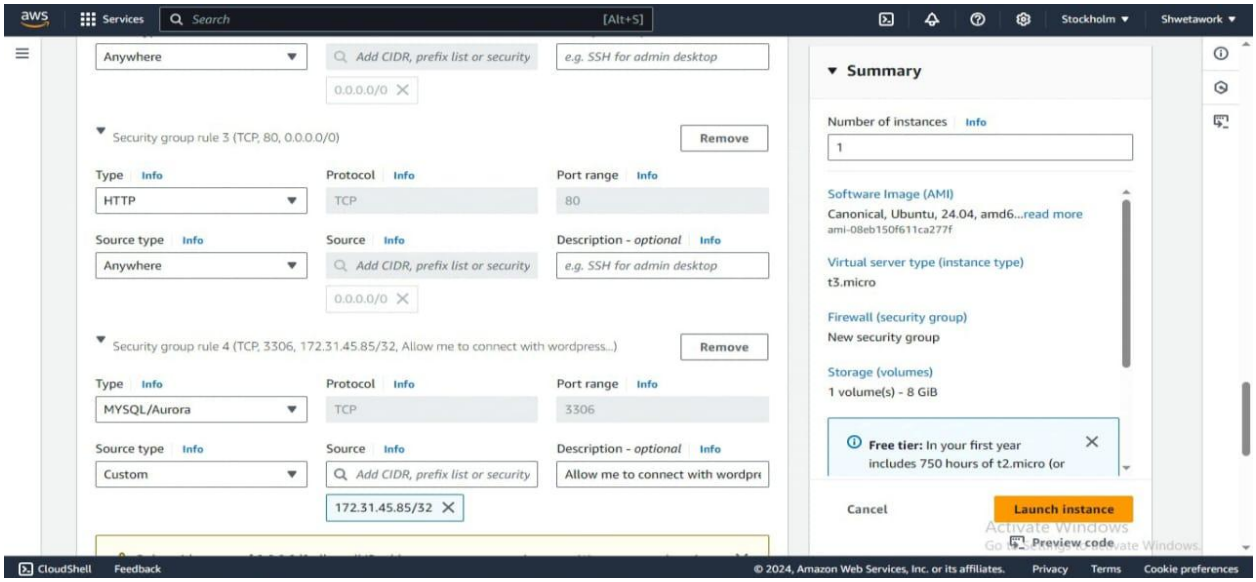
- **Launch an EC2 instance** with Ubuntu OS.
- **Install LAMP stack** (Linux, Apache, MySQL, PHP) on the same instance.
- **Install WordPress** and configure it to connect to the MySQL database.
- Set up the necessary **security groups** to allow HTTP (port 80) and MySQL (port 3306) access.
- Create a **welcome page** in WordPress and set it as the homepage.



2 Microservices Architecture Deployment

In the **microservices architecture**, WordPress and MySQL are deployed on separate EC2 instances. This approach offers greater flexibility and scalability but requires more configuration and management. The steps for deployment include:

- **Launch two EC2 instances**: One for WordPress and one for MySQL.
- **Set up MySQL** on its instance, creating the necessary databases and users.
- **Set up WordPress** on its instance and configure it to connect to the MySQL instance using the private IP address.
- Create appropriate **security groups** to restrict communication between instances (allow WordPress to connect to MySQL).
- Create a **welcome page** in WordPress and set it as the homepage.



---

3. Challenges and Solutions

Challenges in Monolithic Architecture:

- **Scalability**: As traffic grows, it becomes difficult to scale the entire application, as WordPress and MySQL are tightly coupled on the same server.
    - **Solution**: Horizontal scaling is possible but requires careful resource management. You may consider upgrading the EC2 instance type or adding more storage.
- **Resource Constraints**: Since both applications run on the same instance, resource contention (CPU, memory) can degrade performance.
    - **Solution**: Monitor resources closely and consider upgrading to a larger EC2 instance type as needed.

Challenges in Microservices Architecture:

- **Complex Configuration**: Managing multiple instances and networking between them adds complexity, particularly in handling the communication between WordPress and MySQL.
    - **Solution**: Use AWS security groups and IAM roles for better network management and inter-service communication.
- **Cost**: Running separate EC2 instances for WordPress and MySQL incurs higher costs compared to the monolithic approach.
    - **Solution**: Use **t2.micro** instances and ensure that both instances are configured correctly to minimize unnecessary costs.

---

4. Learning Outcomes

By the end of this deployment exercise, you should have a deeper understanding of:

- The steps required to deploy a **LAMP stack** and **WordPress** on **AWS EC2**.
- The differences in deployment and configuration between **monolithic** and **microservices** architectures.
- How to manage **security** using AWS **security groups**.
- The challenges of managing resources and scaling applications in both architectures.
- The importance of choosing the right architecture based on application size, scalability needs, and budget.

5. Comparative Analysis: Monolithic vs. Microservices Architecture

1 Pros and Cons of Monolithic Architecture

Pros:

- **Simpler to set up**: Both WordPress and MySQL are on the same server, making the deployment process faster and less complex.
- **Lower cost**: Since you're only running one instance, costs are lower, especially on the AWS Free Tier.
- **Easier management**: With everything on one instance, monitoring and maintaining the server can be simpler.

Cons:

- **Scalability issues**: As traffic increases, it’s harder to scale individual components. You must scale the entire instance.
- **Limited fault tolerance**: If the instance goes down, both WordPress and MySQL will be affected, causing downtime for the entire application.
- **Performance degradation**: With both WordPress and MySQL running on the same instance, there is the potential for resource contention, especially under heavy traffic.

2 Pros and Cons of Microservices Architecture

Pros:

- **Better scalability**: Each component (WordPress and MySQL) can be scaled independently based on resource needs.

- **Fault isolation**: If MySQL or WordPress goes down, only one part of the application is affected, improving overall application resilience.
- **Flexibility**: More control over individual components, allowing for easier updates and maintenance without affecting the whole system.

Cons:

- **Increased complexity**: Setting up and managing multiple EC2 instances can be more complex than a monolithic approach.
- **Higher cost**: Running multiple EC2 instances incurs more costs compared to the single-instance approach in the monolithic setup.
- **Networking issues**: Proper configuration of communication between the two EC2 instances is essential and requires careful management of security groups and IAM roles.

---

6. Resources Used

- **AWS EC2 Instances** (t2.micro) for both monolithic and microservices architectures.
- **Ubuntu** as the operating system on EC2 instances.
- **LAMP Stack** (Linux, Apache, MySQL, PHP) to support WordPress.
- **WordPress** as the CMS for creating and managing the website.

- **AWS Security Groups** for managing network access between instances.
- **AWS IAM** roles and policies for security management.

---

7. Conclusion

Both **monolithic** and **microservices** architectures have their unique advantages and challenges. The monolithic approach is simpler and more cost-effective for smaller applications, but it lacks the scalability and fault tolerance offered by microservices. On the other hand, the microservices architecture provides better scalability, flexibility, and fault isolation but comes with added complexity and cost.