Internship Task Report: Deploying WordPress Using Monolithic and Microservices Architectures

Introduction

This report details the steps taken to deploy WordPress using two different architectures: monolithic and microservices. The task involves configuring EC2 instances on AWS, deploying WordPress and MySQL, and setting up a welcome page as the homepage.

Architectures Overview

Monolithic Architecture

In a monolithic architecture, all components of the application run on a single server. For this task, we deployed both WordPress and MySQL on a single EC2 instance.

Microservices Architecture

In a microservices architecture, different components of the application are isolated into separate services. For this task, we deployed WordPress on one EC2 instance and MySQL on another.

EC2 Instances Configuration

Instance Details

- Instance Type: t2-micro
- AMI: Ubuntu 20.04 LTS (ami-0a91cd140a1fc148a)

Security Groups

- Allow HTTP (port 80) for WordPress.
- Allow MySQL (port 3306) only from the WordPress instance for enhanced security.

- Allow SSH (port 22) for administration purposes.

Deployment Steps

Monolithic Architecture

1. **Launch EC2 Instance**
    - Create a t2-micro EC2 instance with the specified AMI (Ubuntu 20.04 LTS).
    - Assign a security group allowing HTTP, MySQL, and SSH access.

2. **Install and Configure Apache**
   ->sudo apt update
   ->sudo apt install apache2
   ->sudo systemctl start apache2
   ->sudo systemctl enable apache2

3. **Install and Configure MySQL**
   ->sudo apt install mysql-server
   ->sudo mysql_secure_installation
   ->sudo mysql
   ->CREATE DATABASE wordpress;
   ->CREATE USER 'wp_user'@'localhost' IDENTIFIED BY 'password';
   ->GRANT ALL PRIVILEGES ON wordpress.* TO 'wp_user'@'localhost';

->FLUSH PRIVILEGES;

4. **Install PHP and WordPress**

->sudo apt install php libapache2-mod-php php-mysql
->cd /tmp
->wget https://wordpress.org/latest.tar.gz
->tar -xzvf latest.tar.gz
->sudo mv wordpress /var/www/html/
->sudo chown -R www-data:www-data /var/www/html/wordpress
->sudo chmod -R 755 /var/www/html/wordpress
->sudo systemctl restart apache2

5. **Configure WordPress**

- Navigate to http://your_instance_ip/wordpress in a web browser.
- Complete the WordPress setup wizard, connecting it to the MySQL database created earlier.
- Create a welcome page and set it as the homepage.

Microservices Architecture

1. Launch EC2 Instances

- Create two t2-micro EC2 instances with the specified AMI (Ubuntu 20.04 LTS).
- One instance for WordPress (WP-Instance) and another for MySQL (DB-Instance).

- Assign appropriate security groups.

2. Configure MySQL on DB-Instance

->sudo apt update

->sudo apt install mysql-server

->sudo mysql_secure_installation

-> sudo mysql

->CREATE DATABASE wordpress;

->CREATE USER 'wp_user'@'%' IDENTIFIED BY 'password';

->GRANT ALL PRIVILEGES ON wordpress.* TO 'wp_user'@'%';

-> FLUSH PRIVILEGES;

->sudo ufw allow 3306

3. Install and Configure Apache and WordPress on WP-Instance

->sudo apt update

->sudo apt install apache2 php libapache2-mod-php php-mysql

->cd /tmp

>wget https://wordpress.org/latest.tar.gz

->tar -xzvf latest.tar.gz

->sudo mv wordpress /var/www/html/

->sudo chown -R www-data:www-data /var/www/html/wordpress

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

->sudo systemctl start apache2

->sudo systemctl enable apache2

4. Configure WordPress to Use Remote MySQL Database

- -Update the WordPress configuration to point to the DB-Instance.

-> // In /var/www/html/wordpress/wp-config.php

->define('DB_NAME', 'wordpress');

->define('DB_USER', 'wp_user');

->define('DB_PASSWORD', 'password');

->define('DB_HOST', 'DB-Instance_IP');

5. Complete WordPress Setup

- Navigate to `http://WP-Instance_IP/wordpress` in a web browser.
- Complete the WordPress setup wizard, connecting it to the remote MySQL database.
- Create a welcome page and set it as the homepage.

Conclusion

The deployment of WordPress using both monolithic and microservices architectures demonstrates different approaches to application architecture. The monolithic setup consolidates all services onto a single instance, simplifying management but potentially limiting scalability. The microservices setup separates the database and application server, enhancing scalability and fault isolation at the cost of increased complexity. Both setups were configured to ensure security and functionality, culminating in the creation of a welcome page on WordPress.

This task provided valuable insights into the strengths and trade-offs of different architectural styles, preparing for more advanced deployment scenarios in real-world environments.