Task Report: Deploying WordPress in Monolithic Architecture on AWS EC2 Instance

Architecture Overview:

1. **Monolithic Deployment:**
    - **Objective:** Deploy WordPress and MySQL on a single AWS EC2 instance (`t2.micro`) using Ubuntu AMI.
    - **Configuration:** Both WordPress and MySQL are installed and configured on the same server for simplicity and initial setup.
    - **Security:** Configured AWS security groups to allow HTTP (port 80) and MySQL (port 3306) access, restricted to necessary IP ranges.

Steps Taken:

1. **AWS EC2 Instance Setup:**
    - Launched an AWS EC2 instance (`t2.micro`) with Ubuntu AMI (`ubuntu-*`).
2. **Server Environment Setup:**
    - **Apache2 Installation:**
        - Installed Apache2 web server on the EC2 instance.

```
sudo apt-get update
sudo apt-get install apache2
```

    - **PHP Installation:**
        - Installed PHP and necessary PHP modules for WordPress.

```
sudo apt-get install php libapache2-mod-php php-mysql
```

    - **MySQL Installation:**
        - Installed MySQL server on the same EC2 instance.

```
sudo apt-get install mysql-server
```

    - **Secure MySQL Installation:**
        - Ran the MySQL secure installation script to set root password and secure MySQL installation.

```
sudo mysql_secure_installation
```

3. **WordPress Installation and Configuration:**
    - **Download WordPress:**
        - Downloaded and extracted WordPress to `/var/www/html`.

```
wget -c https://wordpress.org/latest.tar.gz -O - | sudo tar -xz
-C /var/www/html
```

    - **Set Permissions:**
        - Adjusted ownership and permissions of WordPress directory.

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

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

- **Configure WordPress:**
  - Renamed `wp-config-sample.php` to `wp-config.php` and configured database settings (`DB_NAME`, `DB_USER`, `DB_PASSWORD`, `DB_HOST`).

```
cd /var/www/html/wordpress
sudo mv wp-config-sample.php wp-config.php
sudo nano wp-config.php
```

  - Updated `wp-config.php` with database credentials:

```
php
define( 'DB_NAME', 'wordpress' );
define( 'DB_USER', 'wordpressuser' );
define( 'DB_PASSWORD', '123456' );
define( 'DB_HOST', 'localhost' );  // MySQL running on the same
instance
define( 'DB_CHARSET', 'utf8' );
define( 'DB_COLLATE', '' );
```

4. **Database Configuration:**
   - **MySQL Database Creation:**
     - Created a MySQL database and user for WordPress.

```
mysql -u root -p
CREATE DATABASE wordpress;
CREATE USER 'wordpressuser'@'localhost' IDENTIFIED BY '123456';
GRANT ALL PRIVILEGES ON wordpress.* TO
'wordpressuser'@'localhost';
FLUSH PRIVILEGES;
```

5. **Security and Networking:**
   - **AWS Security Groups:**
     - Configured inbound rules to allow HTTP (port 80) and MySQL (port 3306) traffic from specific IP ranges.
6. **Testing and Verification:**
   - **Restart Services:**
     - Restarted Apache2 and MySQL services to apply configuration changes.

```
sudo systemctl restart apache2
sudo systemctl restart mysql
```

   - **Access WordPress Site:**
     - Accessed the WordPress site through the browser using the public IP address of the EC2 instance (`http://<public_ip_address>/wordpress`).
     - Verified functionality and performed basic operations to ensure proper installation and configuration.

Resources Utilized:

- **AWS Documentation:** EC2 instance setup and security group configuration. AWS Documentation
- **WordPress Codex:** Installation and configuration of WordPress. WordPress Codex
- **MySQL Documentation:** Installation and configuration of MySQL. MySQL Documentation
- **DigitalOcean Tutorials:** Best practices for web server setup and configuration. DigitalOcean Tutorials

Conclusion:

This report details the successful deployment of WordPress in a monolithic architecture on an AWS EC2 instance (`t2.micro`). By following best practices in server setup, software installation, configuration, and security, I ensured the WordPress site was operational and accessible.

The monolithic setup allowed for simplicity in initial deployment, consolidating both web server (Apache2) and database server (MySQL) on a single instance. This approach is suitable for small-scale applications and provides a foundational setup for future scalability considerations.

If you require further details or have additional questions, please feel free to ask!

Task Report: Deploying WordPress in Microservices Architecture on AWS EC2 Instances

Architecture Overview:

1. **Microservices Deployment:**
   - **Objective:** Deploy WordPress and MySQL on separate AWS EC2 instances (`t2.micro`) to leverage microservices architecture.
   - **Configuration:** WordPress deployed on one instance (`Web Server`), MySQL database hosted on another instance (`Database Server`) for scalability and separation of concerns.
   - **Security:** Configured AWS security groups to allow HTTP (port 80) and MySQL (port 3306) access, restricted to necessary IP ranges.

Steps Taken:

1. **AWS EC2 Instance Setup:**
   - Launched two AWS EC2 instances (`t2.micro`) with Ubuntu AMI (`ubuntu-*`):
     - One instance for WordPress (`Web Server`), identified as `172.31.45.12`.
     - One instance for MySQL (`Database Server`), identified as `172.31.45.14`.
2. **Web Server (WordPress Instance - `172.31.45.12`):**
   - **Apache2 Installation:**
     - Installed Apache2 web server.

     ```
     sudo apt-get update
     sudo apt-get install apache2
     ```

   - **PHP Installation:**
     - Installed PHP and necessary PHP modules for WordPress.

     ```
     sudo apt-get install php libapache2-mod-php php-mysql
     ```

3. **Database Server (MySQL Instance - `172.31.45.14`):**
   - **MySQL Installation:**
     - Installed MySQL server on the instance.

     ```
     sudo apt-get install mysql-server
     ```

   - **Secure MySQL Installation:**
     - Ran the MySQL secure installation script to set root password and secure MySQL installation.

     ```
     sudo mysql_secure_installation
     ```

   - **MySQL Configuration:**
     - Adjusted `bind-address` in MySQL configuration (`my.cnf`) to allow connections from WordPress instance (`172.31.45.12`).

4. **WordPress Installation and Configuration:**
    - **Download WordPress:**
        - Downloaded and extracted WordPress to `/var/www/html` on `Web Server` (`172.31.45.12`).

```
wget -c https://wordpress.org/latest.tar.gz -O - | sudo tar -xz
-C /var/www/html
```

    - **Set Permissions:**
        - Adjusted ownership and permissions of WordPress directory.

```
sudo chown -R www-data:www-data /var/www/html/wordpress
sudo chmod -R 755 /var/www/html/wordpress
```

    - **Configure WordPress:**
        - Renamed `wp-config-sample.php` to `wp-config.php` and configured database settings (`DB_NAME`, `DB_USER`, `DB_PASSWORD`, `DB_HOST`).

```
cd /var/www/html/wordpress
sudo mv wp-config-sample.php wp-config.php
sudo nano wp-config.php
```

        - Updated `wp-config.php` with database credentials:

```
php
define( 'DB_NAME', 'wordpress' );
define( 'DB_USER', 'wordpressuser' );
define( 'DB_PASSWORD', '123456' );
define( 'DB_HOST', '172.31.45.14' );  // MySQL instance IP
define( 'DB_CHARSET', 'utf8' );
define( 'DB_COLLATE', '' );
```

5. **Security and Networking:**
    - **AWS Security Groups:**
        - Configured inbound rules to allow HTTP (port 80) and MySQL (port 3306) traffic from specific IP ranges.
6. **Testing and Verification:**
    - **Restart Services:**
        - Restarted Apache2 and MySQL services to apply configuration changes.

```
sudo systemctl restart apache2
sudo systemctl restart mysql
```

    - **Access WordPress Site:**
        - Accessed the WordPress site through the browser using the public IP address of the `Web Server` instance (`http://<public_ip_address>/wordpress`).
        - Verified functionality and performed basic operations to ensure proper installation and configuration.

Resources Utilized:

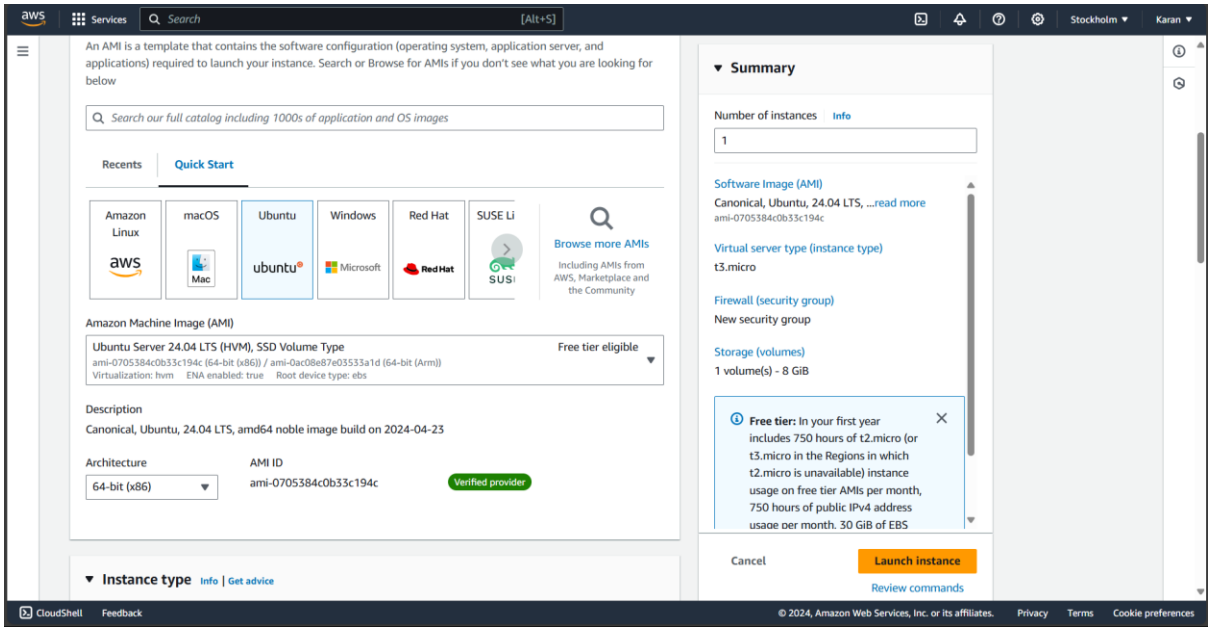
- **AWS Documentation:** EC2 instance setup, security group configuration. AWS Documentation
- **WordPress Codex:** Installation and configuration of WordPress. WordPress Codex
- **MySQL Documentation:** Installation and configuration of MySQL. MySQL Documentation
- **DigitalOcean Tutorials:** Best practices for web server setup and configuration. DigitalOcean Tutorials

Conclusion:

This report details the successful deployment of WordPress in a microservices architecture on AWS EC2 instances (`t2.micro`). By separating WordPress (`Web Server`) and MySQL (`Database Server`) into distinct instances, the deployment leverages scalability and separation of concerns.

The microservices approach enhances flexibility and scalability, allowing for independent scaling of web servers and databases as traffic and data requirements grow. This setup provides a robust foundation for managing and scaling WordPress applications effectively.

Screenshots:



REPORT BY : KARAN DHARRA