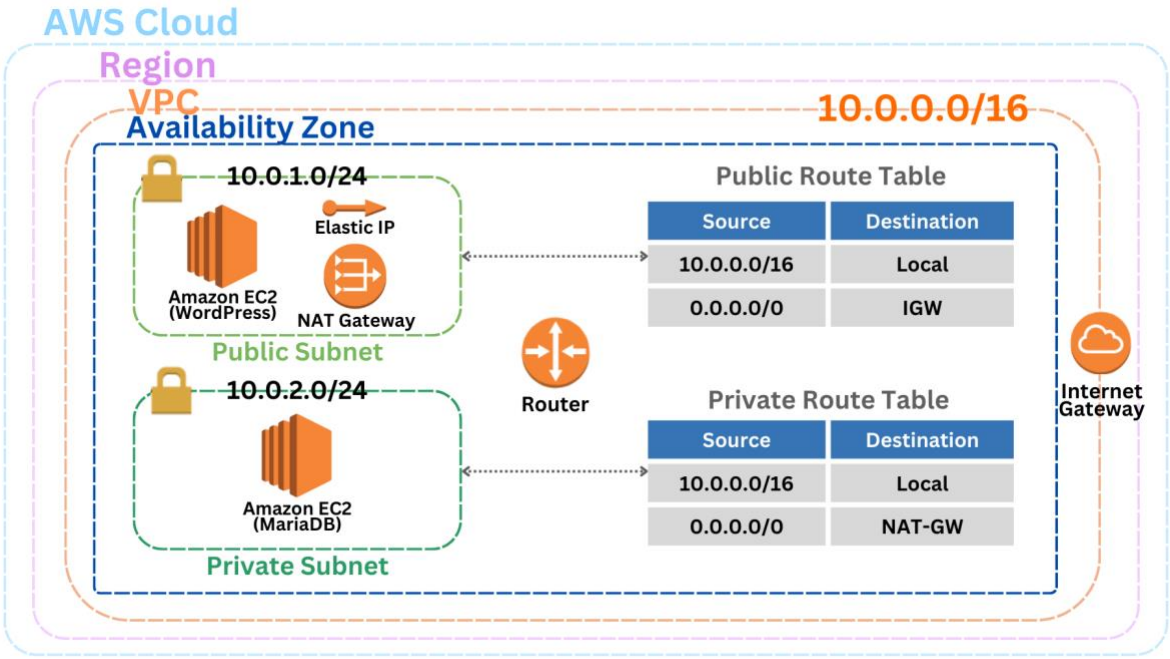
Activity 7 AWS Virtual Private Cloud Design and Implementation

Part I Design

Question 1.1

Assuming you want to set up a web application in **a single AZ** in **a single region** using Wordpress that requires use of a MariaDB database. Say your Wordpress software (google Wordpress if you do not know the requirements for this software) will be set up on an **EC2 instance**. And say the MariaDB database will be set up on **another EC2 instance**. Architect (draw) a VPC to enable connectivity for your application to work and identify how all VPC components will be set up, such as private subnet, public subnet, Internet gateway, NAT gateway, Elastic IP, etc. Look at the picture above to see if you have described every relevant component. Number your subnets (give them IP address ranges) and describe what is in your route tables. Note: As best practice for security, people must not be able to connect to the database instance from the Internet directly. Therefore, you should use appropriate gateways for your instances and consider what should be in public vs. private subnets.



The network architecture features a Virtual Private Cloud (VPC) with the address range of 10.0.0.0/16. Within this VPC, private and public subnets are configured, with private subnet 10.0.2.0/24 safeguarding backend system (MariaDB) and public subnet 10.0.1.0/24 facilitating web server system (WordPress). Routing within the private subnet is managed by a route table, directing traffic through either an Internet Gateway (IGW) or a Network Address Translation Gateway (NAT-GW) for outbound connectivity. Supporting components include routers, IGWs, NAT-GWs, elastic IPs for static public IP assignment, and the designation of a specific Availability Zone (AZ) for high availability.

Question 1.2

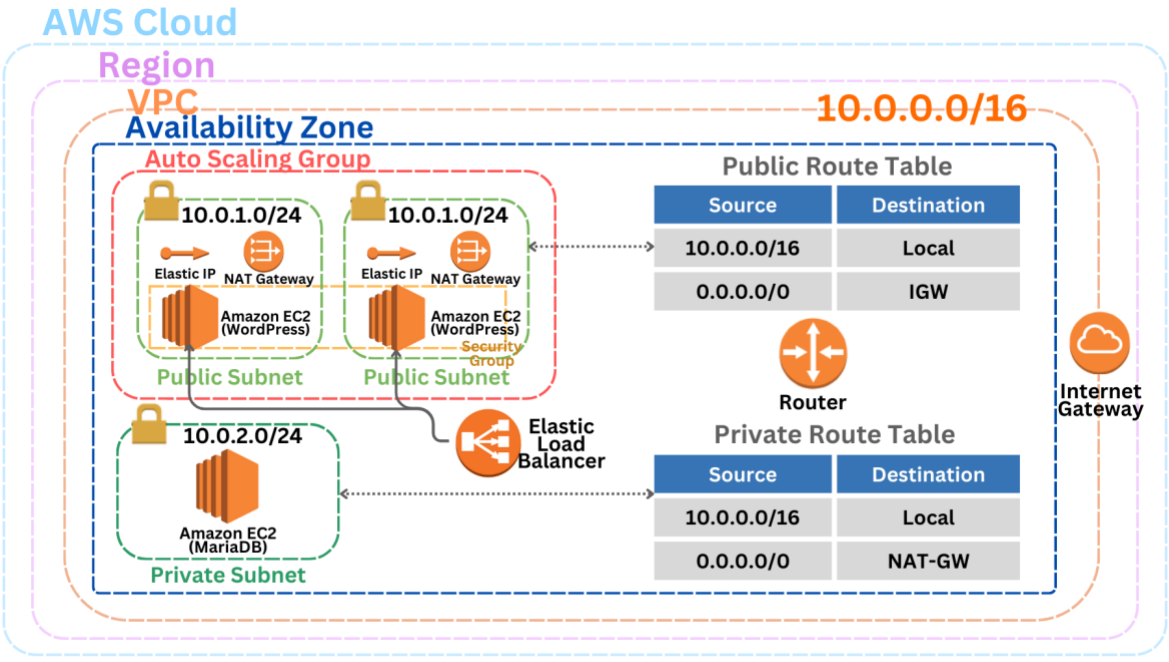
For the set up in Question 1, think about the security of the set up above. How would you use security groups and/or network access control lists with your set up above?

I apply **Security Groups** within my architecture as follows:

- For EC2 instances hosting WordPress in the public subnet, the security group allows inbound traffic only on HTTP port 80 for web servers and SSH port 22 for external communication from any IP address (0.0.0.0/0). Outbound traffic is generally allowed without restrictions.
- For EC2 instances hosting MariaDB in the private subnet, the security group allows inbound traffic only on MySQL port 3306 for database access and SSH port 22 only both for private IPv4 addresses of EC2 instances hosting WordPress. Outbound traffic is generally allowed without restrictions.

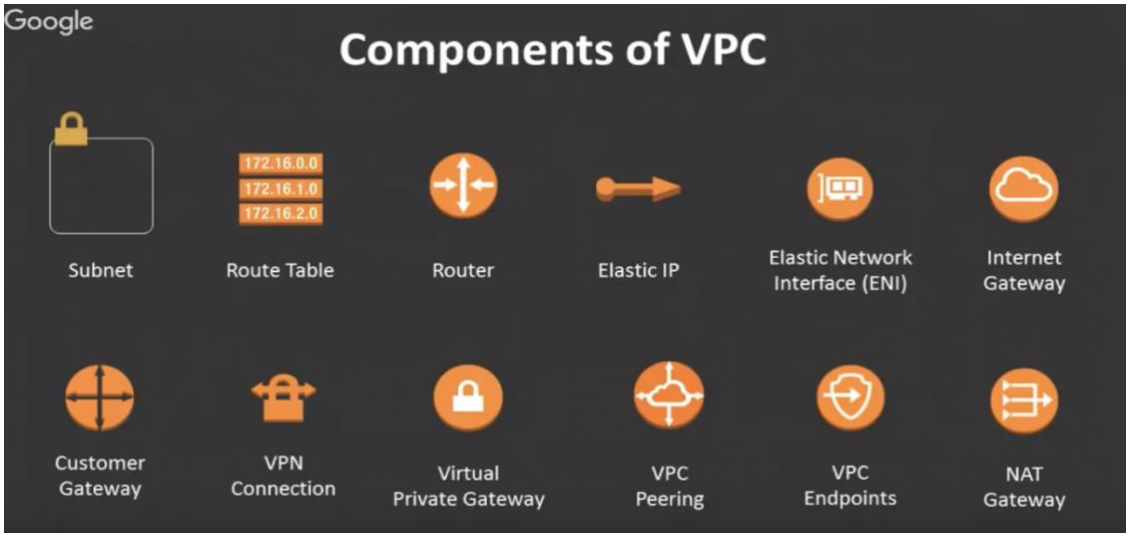
Question 1.3

To support scaling, you want to run multiple instances of Wordpress servers (multiple EC2 instances) and split up the requests to be serviced by different instances (say by URL). You will need to add a load balancer into your set up and multiple EC2 instances that will work as Wordpress servers. You will have only one database instance. Does this new requirement change your VPC architecture? Describe (draw) your new VPC and identify how all VPC components will be set up.



Yes, the new requirement change the VPC architecture. The updated architecture includes multiple EC2 instances, managed by auto-scaling groups and controlled by a load balancer.

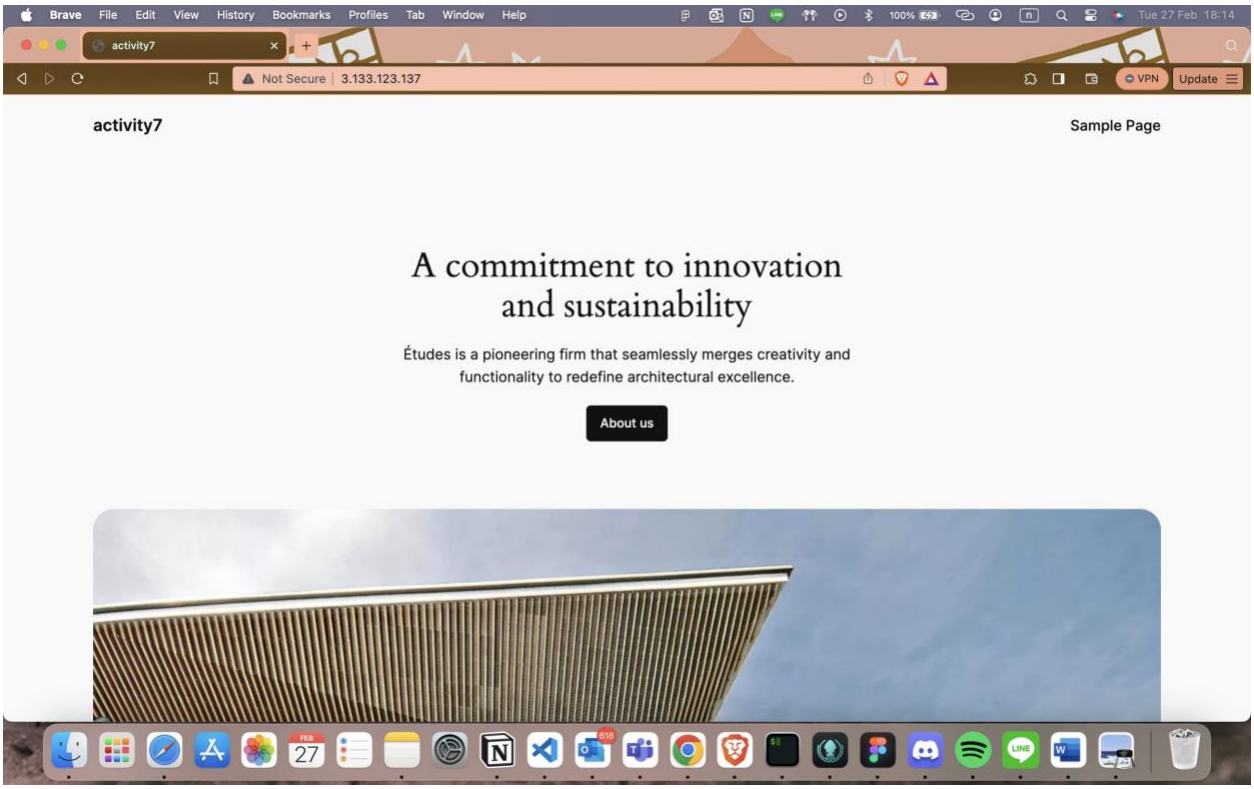
Part II Implementation

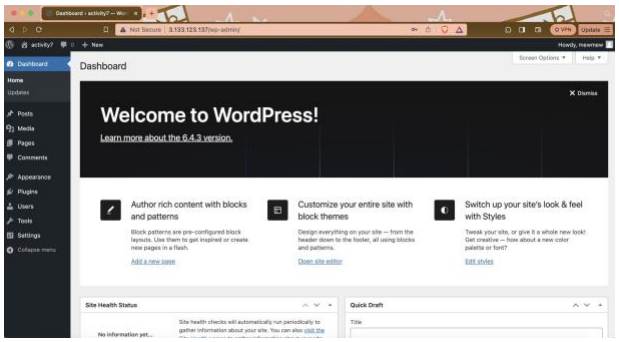


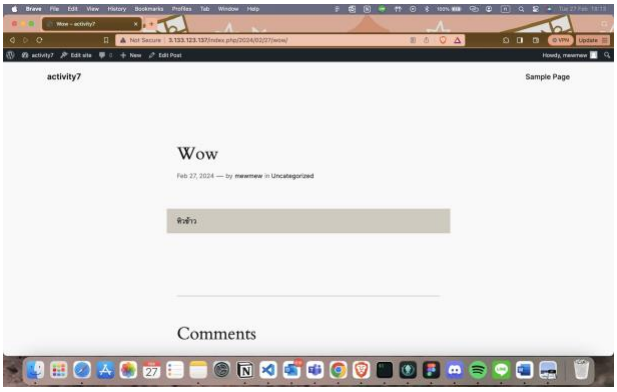
Implement your design from Questions 1.1 and 1.2 from Part I (no need to demo scaling in Question 1.3). You will implement your design on a single region in AWS and **create your own VPC**. Do not use the default VPC.

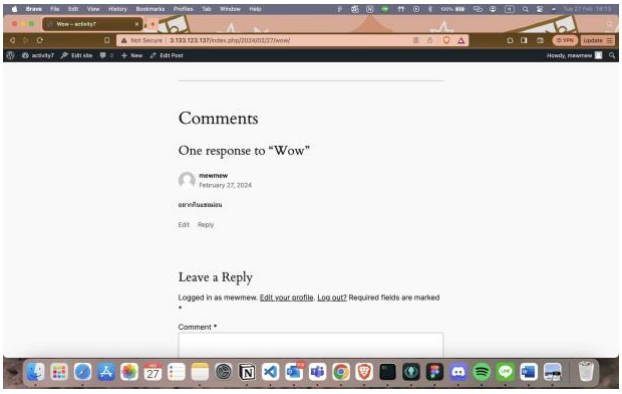
Question 2.1 Demo your work as follows:

- Take screenshots to demo that your Wordpress site works and it can connect to your MariaDB instances.

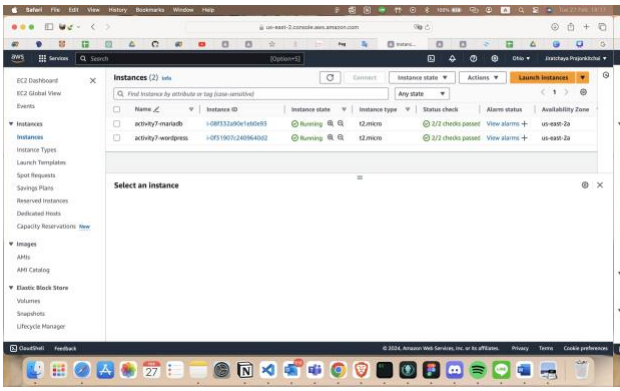




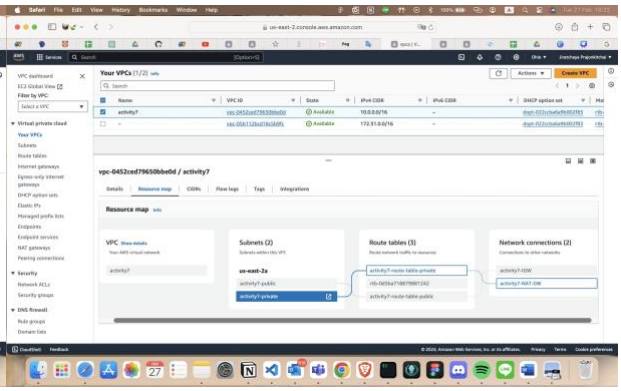




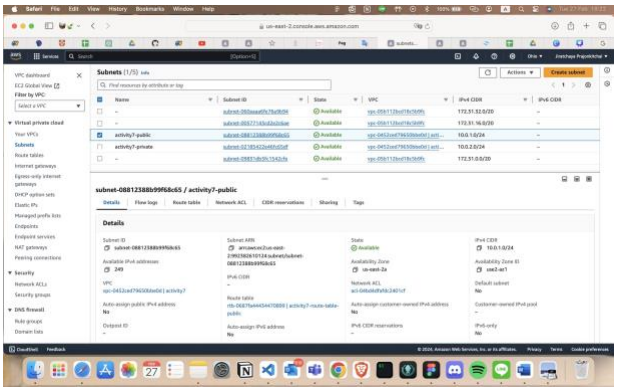
• Show your set up for all your network components (VPC, subnets, gateways, Elastic IP).



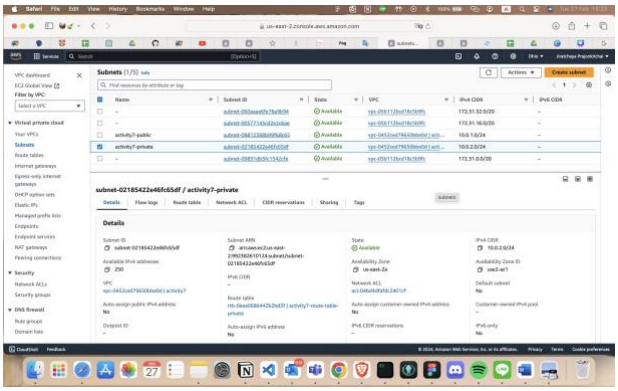
Instances



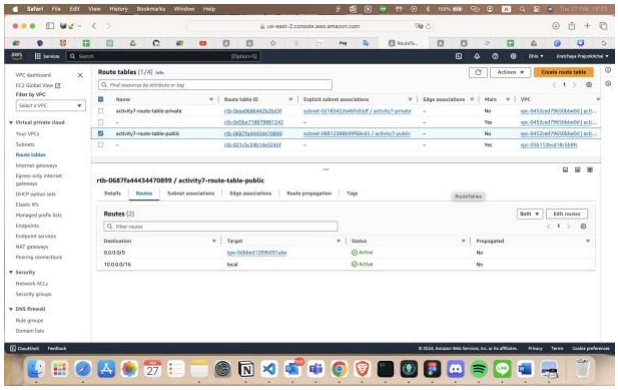
VPC



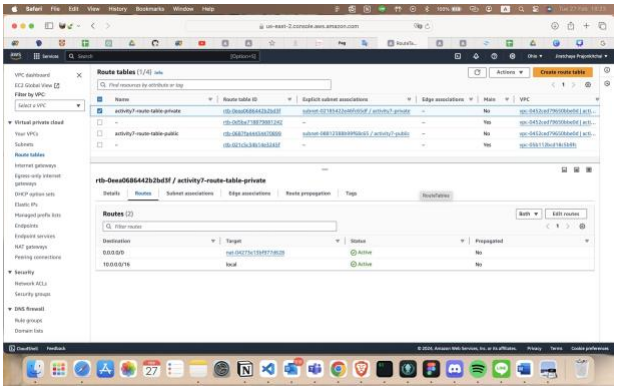
Public Subnet



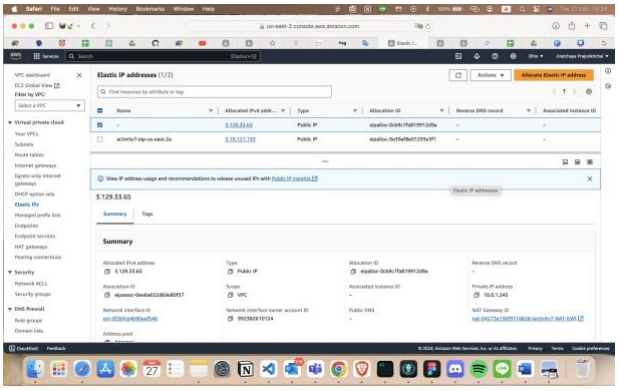
Private Subnet



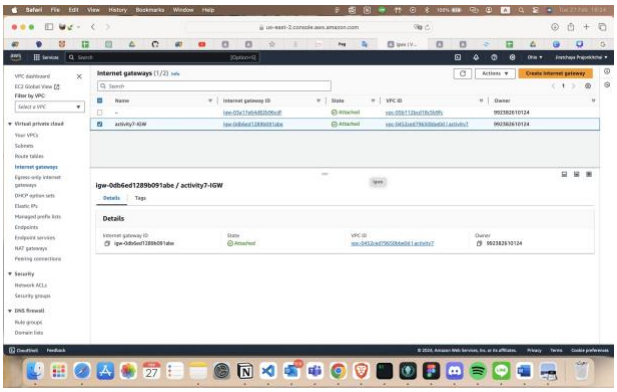
Public Route Table



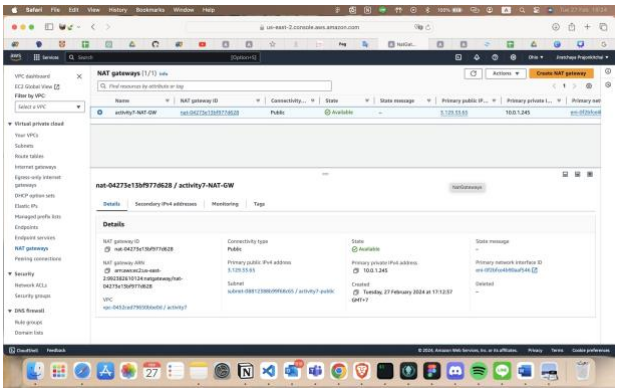
Private Route Table



Elastic IP address

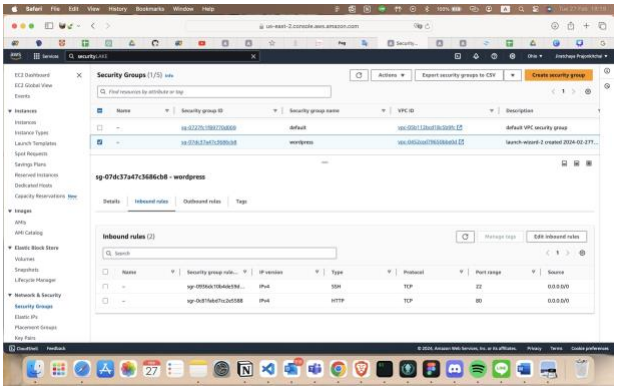


Internet Gateway

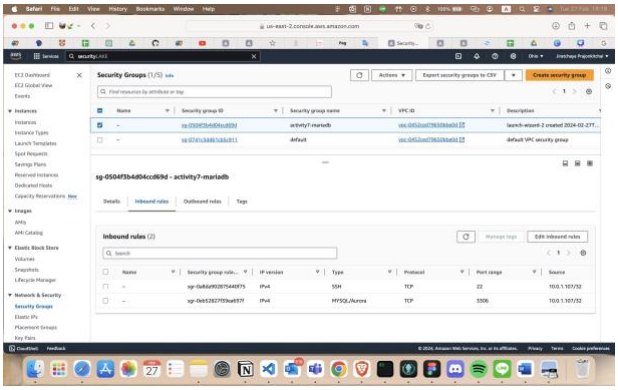


NAT Gateway

- Show your security and/or network access control list set ups too.



Security Group of WordPress instance



Security Group of MariaDB instance

Question 2.2

Did you have to make any changes to your design to make it work? Did you have any difficulties/challenges when implementing this? For example, how were you able to set up your MariaDB instance in the private subnet?

- No, I didn't change any of my design to make it work.
- I'm facing some challenges with my PHP version requirements. The software demands a minimum of PHP 7, prompting me to install a version higher than 7. Additionally, while using Chula WiFi, my website displays a '503 Service Unavailable' error. As a workaround, I've resorted to switching to my internet sharing option.
- I've set up my MariaDB by uploading a key pair (.pem file) to my WordPress instance, which is located in a public subnet. Then, I accessed the MariaDB instance using the following SSH command:

```
ssh -i <filename>.pem ec2-user@<private IPv4 of MariaDB instance>
```

This allows me to securely connect to the MariaDB instance.

Question 2.3

Let's say in the future, you want to repeat the steps to install Wordpress and MariaDB automatically by creating a provisioning script. You will not need to create the script now but write down all the commands that you used to install Wordpress and MariaDB so you can use these commands in the future to get a working Wordpress and MariaDB server.

All commands I've used is referenced from https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/hosting-wordpress.html.

To set up MariaDB (MariaDB instance)

```
[ec2-user ~]$ sudo yum install mariadb-server

[ec2-user ~]$ sudo systemctl enable mariadb

[ec2-user ~]$ sudo systemctl start mariadb

[ec2-user ~]$ mysqladmin -u root password '<password>'
```

To access MariaDB (MariaDB instance)

```
[ec2-user ~]$ mysql -u root -p
```

To create Database (MariaDB instance)

```
MariaDB[(none)]> CREATE USER 'wordpress-user'@'<private IPv4 of
WordPress instance>' IDENTIFIED BY '<password don't need to be the same>';

MariaDB[(none)]> CREATE DATABASE `wordpress-db`;

MariaDB[(none)]> GRANT ALL PRIVILEGES ON `wordpress-db`.* TO
"wordpress-user"@"<private IPv4 of WordPress instance>";

MariaDB[(none)]> FLUSH PRIVILEGES;
```

To install WordPress (WordPress instance)

```
[ec2-user ~]$ wget https://wordpress.org/latest.tar.gz

[ec2-user ~]$ tar -xzf latest.tar.gz
```

To create and edit the wp-config.php file (WordPress instance)

```
[ec2-user ~]$ cp wordpress/wp-config-sample.php wordpress/wp-
config.php

[ec2-user ~]$ nano wordpress/wp-config.php
```

Then change,

```
define('DB_NAME', 'wordpress-db');
define('DB_USER', 'wordpress-user');
define('DB_PASSWORD', '<password using from create database>');
```

To install HTTPD (WordPress instance)

```
[ec2-user ~]$ sudo yum install httpd

[ec2-user ~]$ sudo systemctl enable httpd

[ec2-user ~]$ sudo systemctl start httpd
```

To install PHP (WordPress instance)

```
[ec2-user ~]$ sudo amazon-linux-extras | grep php

[ec2-user ~]$ sudo amazon-linux-extras enable php8.2

[ec2-user ~]$ sudo yum clean metadata

[ec2-user ~]$ sudo yum install php

[ec2-user ~]$ sudo yum install php-
{pear,cgi,common,curl,mbstring,gd,mysqlnd,gettext,bcmath,json,xml,fp
m,intl,zip,imap}
```

To install WordPress files under the Apache document root (WordPress instance)

```
[ec2-user ~]$ sudo cp -r wordpress/* /var/www/html/
```

To restart HTTPD (WordPress instance)

```
[ec2-user ~]$ sudo systemctl restart httpd

[ec2-user ~]$ sudo systemctl start httpd
```