ESCUELA COLOMBIANA DE INGENIERÍA JULIO GARAVITO

UNIVERSIDAD

Computing Networks
Laboratory No. 5
Cloud

Students:
Andrea Camila Torres González
Jorge Andrés Gamboa Sierra

Presented to:
Fabian Eduardo Sierra Sánchez
Semester 2024-2

ESCUELA COLOMBIANA DE INGENIERÍA JULIO GARAVITO

UNIVERSIDAD

Table of Contents

Abstract 3

Objective 3

Tools to be used 3

Theorical framework 3

Introduction 3

Installation of Base Software 4

1. Installation of Web Service in the Cloud 4

Research 5

Configuration 6

Configuring Web Service 15

Adding content to the web server that connects to Amazon database instance 21

Conclusions 38

Bibliography 38

Abstract

This laboratory exercise aims to teach the installation and configuration of cloud-based web server software using Amazon EC2. Students will deploy a web server on an EC2 instance, set up a Virtual Private Cloud (VPC), and understand cloud infrastructure basics. The lab covers theoretical concepts like EC2 instances, VPCs, and Amazon RDS, emphasizing best practices in cloud security and scalability. Through hands-on experience, students will gain practical skills in managing and optimizing cloud resources.

Keywords: *Cloud Computing, Amazon EC2, Virtual Private Cloud, Web Server, AWS, Cloud Infrastructure, Scalability, Security*

Objective

- Install and configure base software – Web servers

Tools to be used

- Computers
- Internet access

Theorical framework

- **Cloud Computing:** Cloud computing is the delivery of computing services such as servers, storage, databases, networking, software, analytics, and intelligence over the Internet ("the cloud"). It offers faster innovation, flexible resources, and economies of scale. Instead of owning their own IT infrastructure or data centers, companies can rent access to anything from applications to storage from a cloud service provider.

- **Amazon EC2 (Elastic Compute Cloud):** Amazon Elastic Compute Cloud (EC2) is a web service that provides scalable computing capacity in the Amazon Web Services (AWS) cloud. EC2 instances are virtual machines that allow you to run applications in the cloud, adjusting resources based on demand. EC2 offers different instance types optimized for various workloads.

- **VPC (Virtual Private Cloud):** A Virtual Private Cloud (VPC) is a virtual network dedicated to your AWS account. It allows you to provision a section of the AWS cloud where you can launch AWS resources in a user-defined virtual network. A VPC offers complete control over your network environment, including the selection of your own IP address range, the creation of subnets, and the configuration of route tables and network gateways.

- **AMI (Amazon Machine Image):** An Amazon Machine Image (AMI) is a template that contains a software configuration (including an operating system, applications, and server settings) used to create EC2 instances. AMIs allow you to launch instances with predefined configurations.

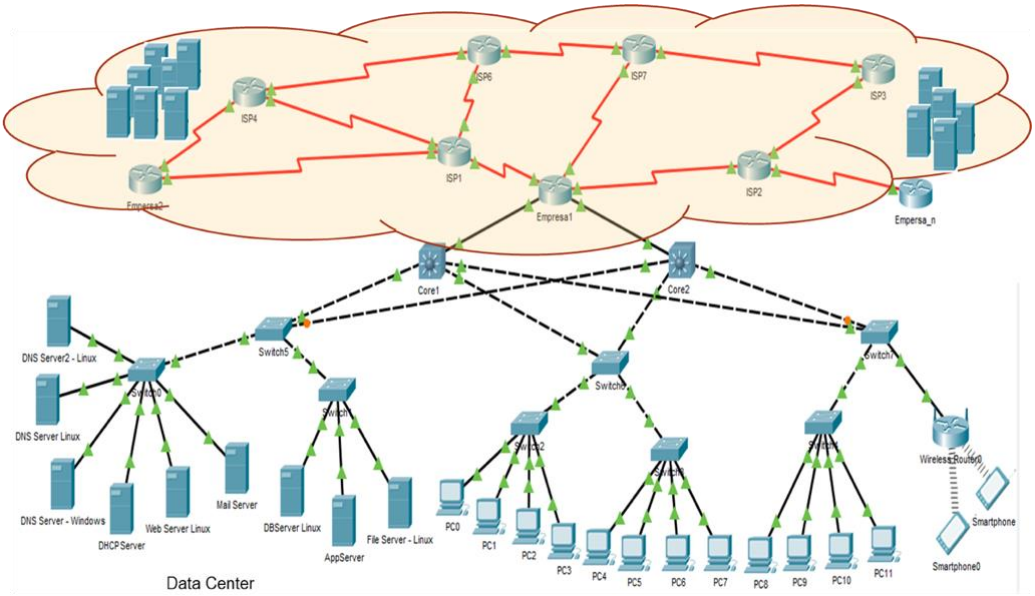
- **Amazon Linux 2:** Amazon Linux 2 is a Linux distribution optimized to run on Amazon EC2. It provides a stable, secure, and high-performance execution environment, with regular updates and long-term support.

Introduction

We continue working on the infrastructure of a company, which typically includes various IT infrastructure services. It consists of wired and wireless user workstations and servers (both physical and virtualized), all connected through

switches (Layer 2 and Layer 3), wireless equipment, and routers that connect to the Internet. It is also common to have cloud infrastructures from which resources are provisioned according to the organization's needs. Within the servers, you can find web services, DNS, email, database, storage, and applications, among others. Let's remember the base configuration we are using:

In this lab, we will focus on the server infrastructure.



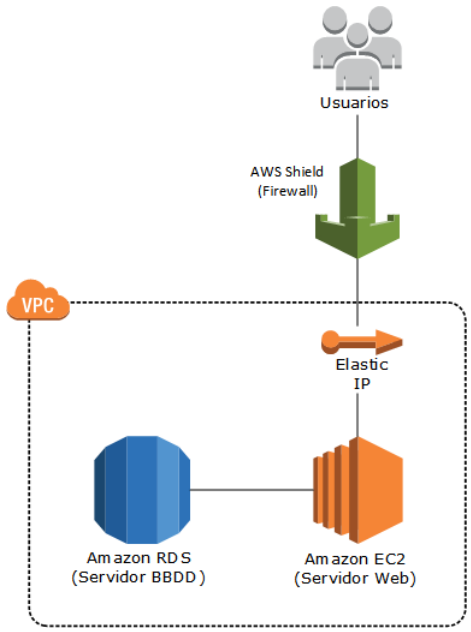
Installation of Base Software

Part of the foundational platform of a computing infrastructure for an organization involves web services, which may be hosted within the company's data center or on a server located in the cloud. These services store the organization's web pages and are used by various clients. In this lab, we will implement this service.

1. Installation of Web Service in the Cloud

Amazon Elastic Compute Cloud (Amazon EC2) offers the broadest and deepest computing platform, with over 500 instance types and the ability to choose the latest processor, storage, networking, operating system, and purchasing model to tailor it to the needs of your workload.

For this lab, we will have an introduction to Amazon Cloud, where the task is to deploy a server, in this case, EC2, and install a web server.



Research

a) What is an EC2 instance and what is it used for?

An Amazon EC2 (Elastic Compute Cloud) instance is a virtual machine that allows users to run applications in the AWS cloud. This service provides scalable computing capacity, enabling the adjustment of computing resources according to demand. EC2 instances are essential for running web applications, performing data analysis, hosting databases, and managing high-performance workloads. The flexibility and scalability of EC2 allow businesses to quickly adapt to the changing needs of their applications.

b) What is a VPC, how should I configure it, and what are the best practices to consider?

A VPC (Virtual Private Cloud) is a private virtual network within the AWS cloud that offers complete control over the network environment, including subnets, route tables, and gateways. To configure a VPC, the following steps must be followed:

- **Create the VPC**: Define the IP address range in CIDR format.
- **Configure subnets**: Create subnets in different availability zones to ensure high availability and redundancy.
- **Assign an internet gateway**: Allow instances within the VPC to access the internet.
- **Configure route tables**: Define routes for network traffic.
- **Configure security groups and access control lists (ACLs)**: Control traffic to and from the instances.

Best practices include segmenting applications into public and private subnets, using multiple availability zones for high availability, implementing monitoring and logging, and applying strict security policies to protect network resources.

c) How can I run multiple systems in an Amazon EC2 environment?

To run multiple systems in an Amazon EC2 environment, you can launch multiple EC2 instances, each with its own operating system and specific configuration. It is possible to select different instance types according to the needs of each system. Additionally, you can configure the network to allow secure communication between instances. Elastic Load Balancing is used to distribute network traffic across multiple instances, and Auto Scaling automatically adjusts capacity based on application demand.

d) ¿How quickly can I scale my capacity (increase and decrease) on an EC2 instance?

The capacity of an EC2 instance can be quickly scaled using Amazon EC2 Auto Scaling, which automatically adjusts the number of instances based on workload. This scaling process can be done in minutes, allowing users to launch and stop instances in an agile and efficient manner. Additionally, it is possible to change the instance type to increase or decrease CPU and memory resources easily, thus adapting to the changing needs of applications.

e) How does this service differ from a standard hosting service?

Amazon EC2 differs from traditional hosting services in several key aspects:

- **Scalability**: EC2 allows resources to be scaled based on demand, something that is not possible with most traditional hosting services.
- **Pay-as-you-go**: EC2 charges only for the time the instances are in use, whereas traditional services often charge a fixed fee.
- **Flexibility**: EC2 offers a wide range of instance types and customizable configurations, while traditional services typically offer more limited plans.
- **High availability and redundancy**: AWS provides options to distribute instances across multiple regions and availability zones, enhancing the resilience and availability of hosted applications and services.

f) What is Amazon RDS?

Amazon RDS (Relational Database Service) is a managed database service that simplifies the setup, operation, and scaling of databases in the cloud. RDS supports multiple database engines, including MySQL, PostgreSQL, MariaDB, Oracle, SQL Server, and Amazon Aurora. This service provides features such as automated backups, replication, automatic scaling, and software patches, allowing users to focus on developing their applications without worrying about managing the database infrastructure. Amazon RDS handles complex administrative tasks, enhancing operational efficiency and the reliability of databases.

Configuration

a) We log in to the AWS Management Console at awsacademy.instructure.com and search for the course Lab 5 / Laboratory. We go to Modules and click on Learner Lab, accept the terms and conditions, and then click on “Start Lab.”

Illustration 1 initialization laboratory

b) When it loads, we click on the top left corner where it says "AWS"; this will redirect us to the AWS Console.

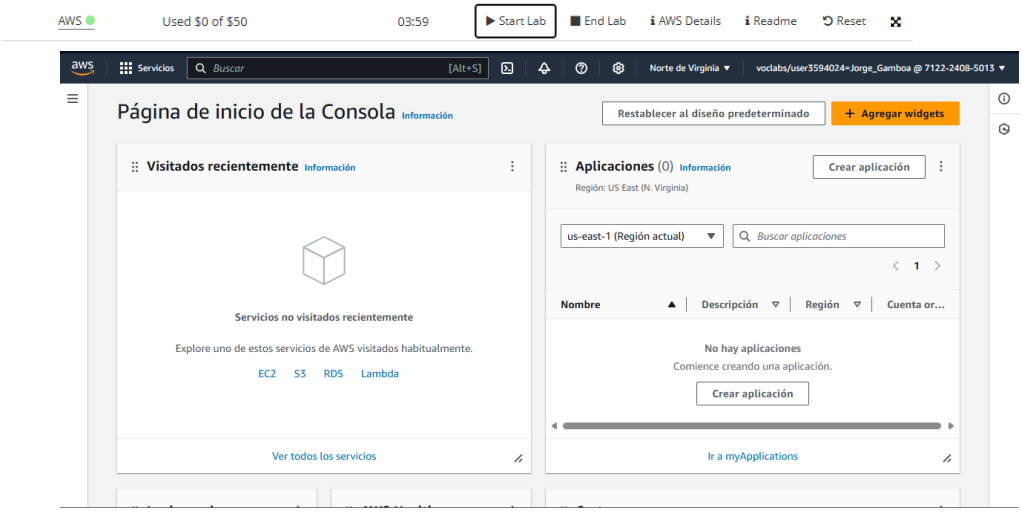


Illustration 2 console home page

c) We click on EC2

Illustration 3 Choose the EC2 option

d) We choose EC2 Dashboard and then look for the Launch Instance option.

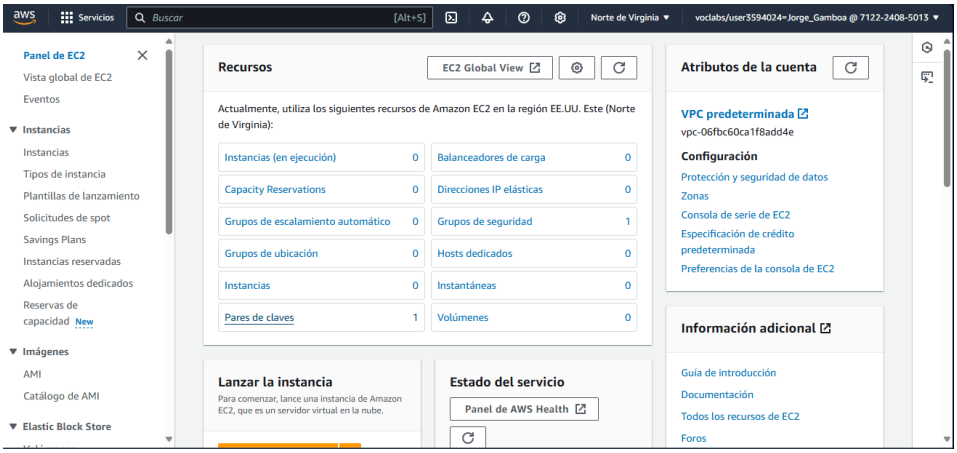


Illustration 4 Search to launch instance

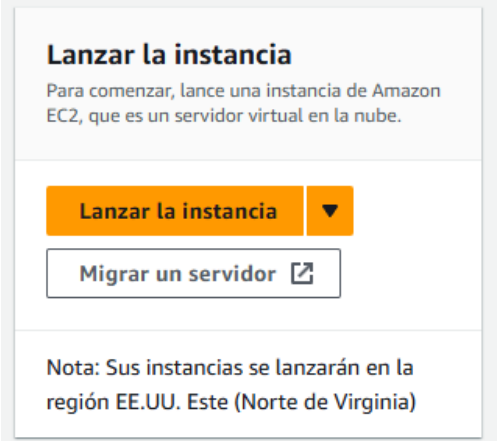


Illustration 5 Launch instance button

e) For this exercise, we need to select the Amazon Linux 2 AMI option..

Illustration 6 AMI selection

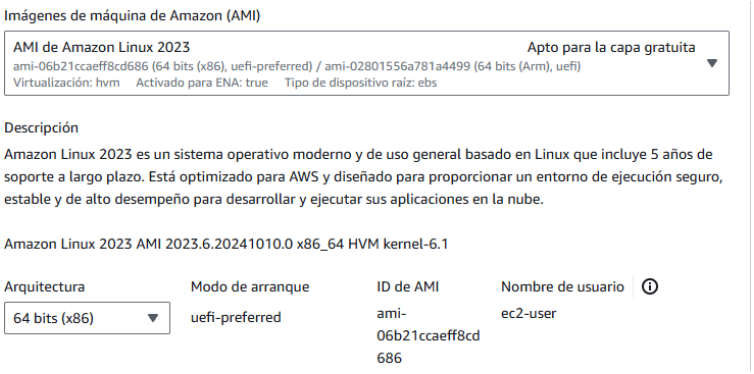


Illustration 7 AMI characteristics

f) We choose the t2.micro instance type.

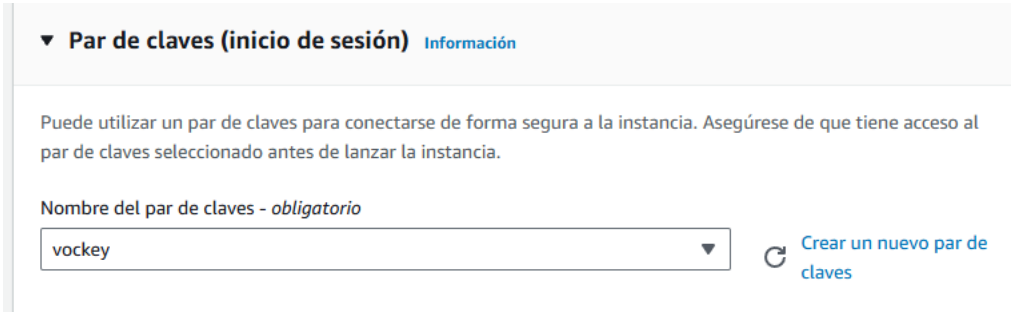


Illustration 8 Key pair configuration

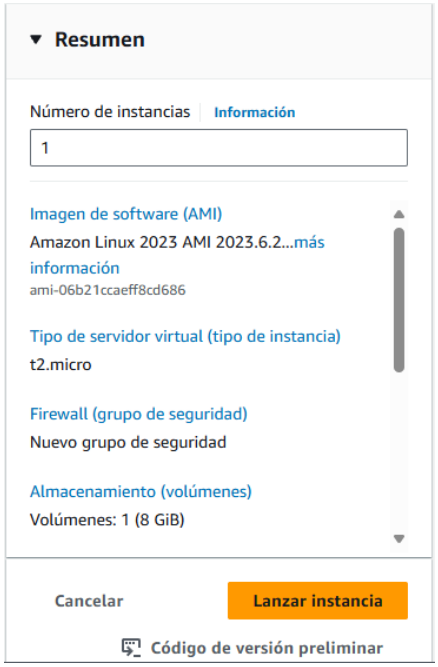


Illustration 9 Instance summary

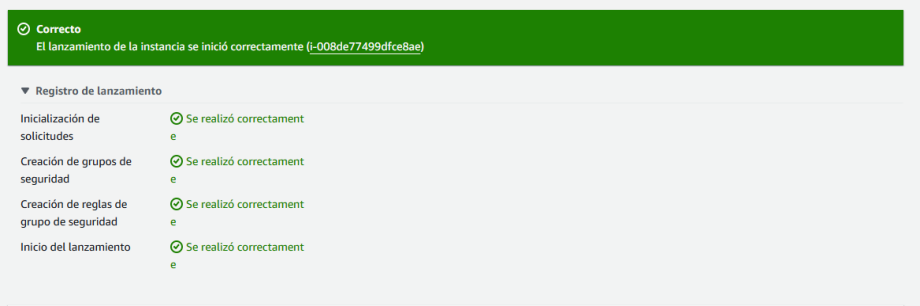


Illustration 10 Instance launch

- What are the instance types in Amazon EC2? Why do you think we chose t2.micro?

1. **General Purpose Instances:** General purpose instances provide a balanced mix of computers, memory, and networking resources, making them a versatile option for a variety of applications. They are ideal for web servers, development applications, and small databases, delivering balanced performance that adapts to diverse workloads.

    - **Examples:** M5 | M5a | M5ad | M5d | M5dn | M5n | M5zn | M6a | M6g | M6gd | M6i | M6id | M6idn | M6in | M7a | M7g | M7gd | M7i | M7i-flex | M8g | Mac1 | Mac2 | Mac2-m1ultra | Mac2-m2 | Mac2-m2pro | T2 | T3 | T3a | T4g

2. **Compute Optimized Instances** These instances are designed to deliver optimized performance for workloads that require high computing capacity. They provide a greater amount of vCPUs compared to memory, making them ideal for data analysis, batch processing, and applications that demand high computational performance.

    - **Examples s:** C5 | C5a | C5ad | C5d | C5n | C6a | C6g | C6gd | C6gn | C6i | C6id | C6in | C7a | C7g | C7gd | C7gn | C7i | C7i-flex | C8g

3. **Memory Optimized Instances**: Memory optimized instances offer a high RAM capacity relative to the number of vCPUs, making them the perfect choice for workloads that require significant memory usage. They are suitable for in-memory databases, big data analytics, and memory-intensive applications, ensuring fast access to large volumes of data.

    - **Examples**: R5 | R5a | R5ad | R5b | R5d | R5dn | R5n | R6a | R6g | R6gd | R6i | R6idn | R6in | R6id | R7a | R7g | R7gd | R7i | R7iz | R8g | U-3tb1 | U-6tb1 | U-9tb1 | U-12tb1 | U-18tb1 | U-24tb1 | U7i-12tb | U7in-16tb | U7in-24tb | U7in-32tb | X1 | X2gd | X2idn | X2iedn | X2iezn | X1e | X8g | z1d

4. **Storage Optimized Instances**: Storage optimized instances are designed to provide high-speed storage and quick access to data, thanks to their ability to include local NVMe storage. They are ideal for NoSQL databases and applications that require optimal performance in data access, making them perfect for data analytics and I/O-intensive workloads..

    - **Examples:** D2 | D3 | D3en | H1 | I3 | I3en | I4g | I4i | Im4gn | Is4gen

5. **Accelerated Computing Instances**
   Accelerated computing instances include specialized hardware resources, such as GPUs, for tasks

that require parallel processing and high performance. They are suitable for machine learning applications, high-performance graphics, and complex simulations, enabling efficient execution of intensive computations.

- **Examples:** DL1 | DL2q | F1 | G4ad | G4dn | G5 | G5g | G6 | G6e | Gr6 | Inf1 | Inf2 | P2 | P3 | P3dn | P4d | P4de | P5 | P5e | Trn1 | Trn1n | VT1

6. **High-Performance Computing Instances** Designed for scientific and technical computing workloads, these instances deliver maximum performance with many vCPUs and optimized memory. They are ideal for complex simulations and scientific data analysis, providing the necessary power for applications that require intensive computational resources.

- **Examples:** Hpc6a | Hpc6id | Hpc7a | Hpc7g

The choice of the t2.micro instance from Amazon EC2 is due to its low cost (approximately $0.0116 per hour) and its inclusion in the AWS Free Tier, making it ideal for beginners. With 1 vCPU and 1 GiB of RAM, it provides sufficient performance for lightweight applications, such as microservices and development environments. Its ability to burst performance allows for temporary scaling during load spikes without the need for a constantly larger capacity instance. This combination of cost-effectiveness, performance capacity, and flexibility makes it an excellent option for prototyping and testing.

a) We refer to the following documentation related to creating a VPC: https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_Tutorials.WebServerDB.CreateVPC.html#CHAP_Tutorials.WebServerDB.CreateVPC.VPCAndSubnets and create the VPC with the addresses assigned by our instructor.

VPC Creation

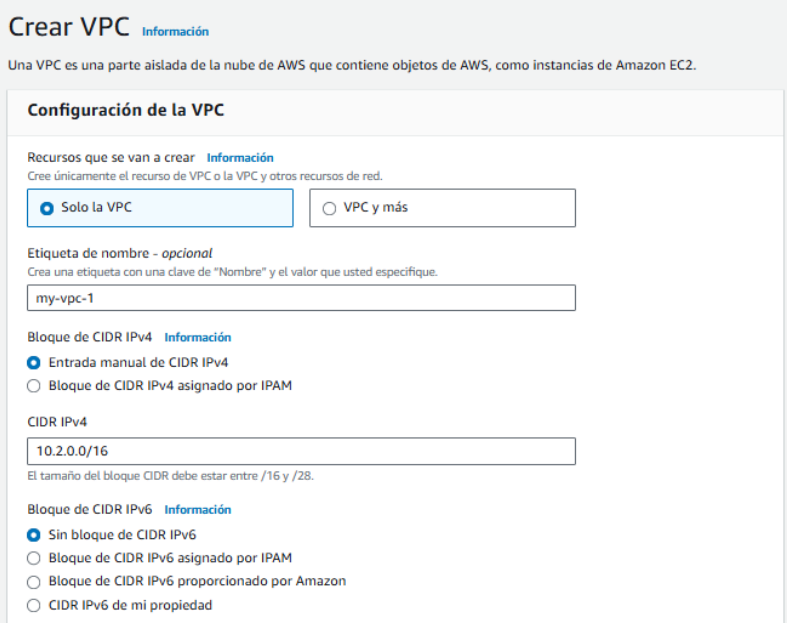


Illustration 11 VPC creation

Illustration 12 VPC summary

Subnet Creation

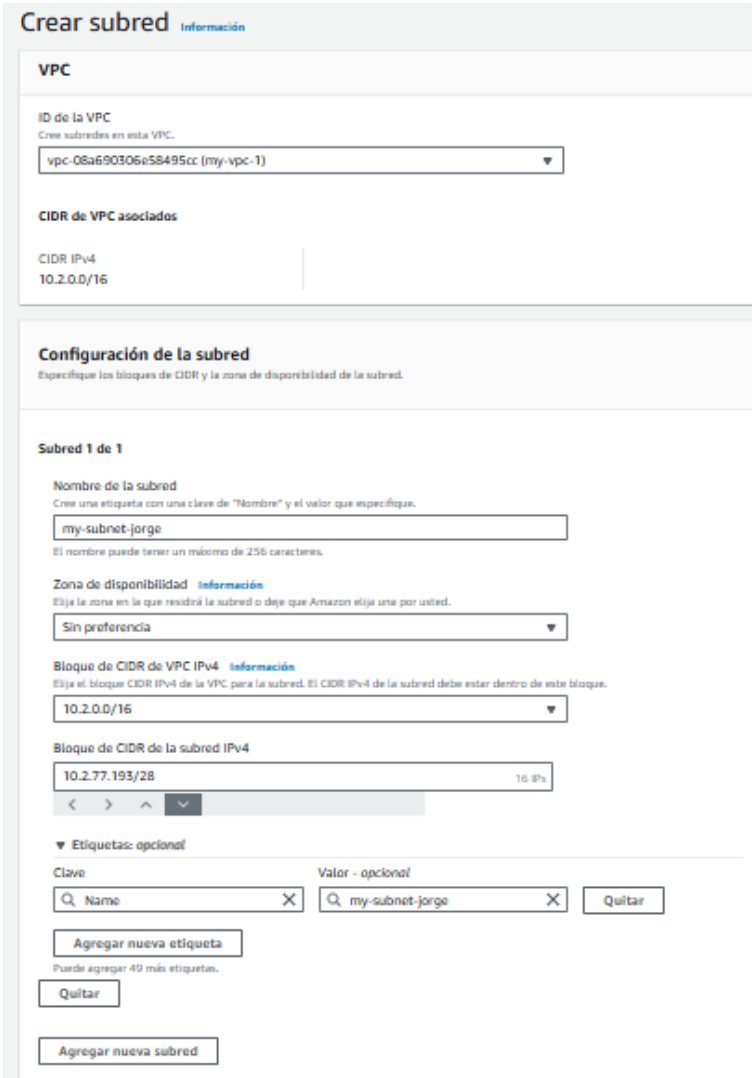


Illustration 13 Subnet configuration

b) On the Configure Instance Details page, we set the values created in the previous step

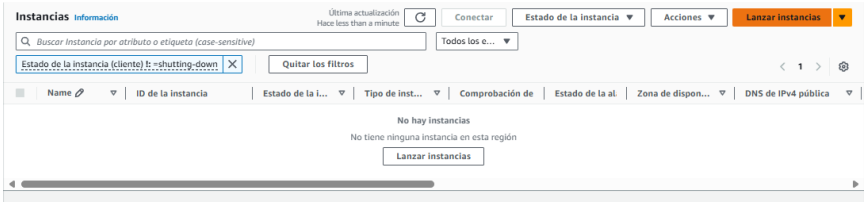


Illustration 14 Definition of values

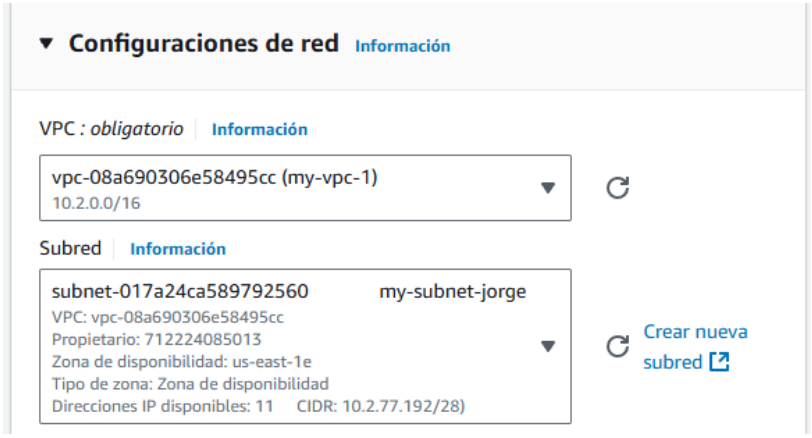


Illustration 15 Network configurations

c) In Auto-assign Public IP, we choose Enable.

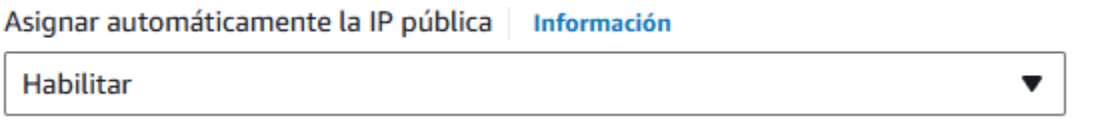


Illustration 16 Auto-assign Public IP

d) On the Add Storage page, we leave the default values and move to the next step: Add Tags.

e) On the Add Tags page shown below, we choose Add Tag and then type Name for Key and type tutorial-web-server for Value.

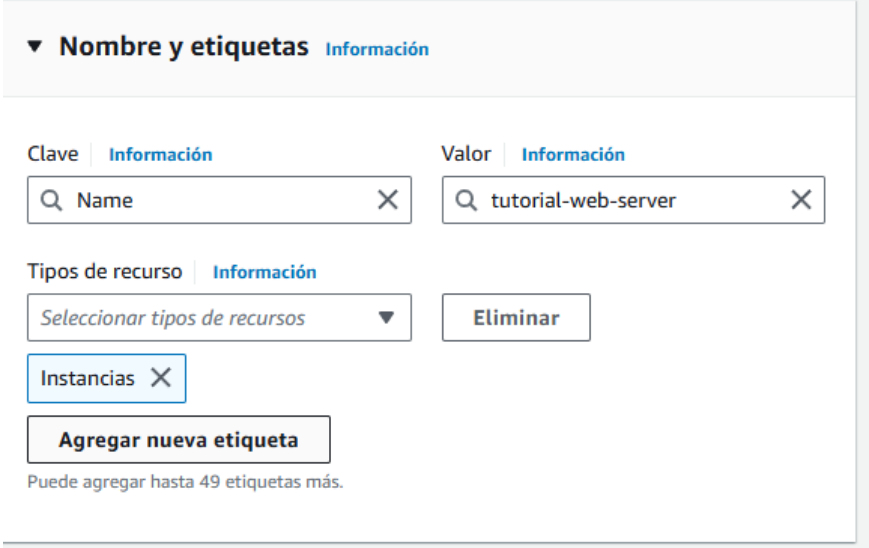


Illustration 17 Name and tag configuration

f) On the Configure Security Group page, we choose Select an existing security group. What are security groups? What should we consider when creating a security group for a public web server?

A security group in Amazon Web Services (AWS) acts as a virtual firewall that controls incoming and outgoing network traffic to and from EC2 instances. These security groups allow us to define rules to permit or deny traffic based on different criteria, such as IP addresses, ports, and protocols. Security group rules are applied at the instance level and provide a way to protect and control access to our resources in the cloud.

Considerations When Creating a Security Group for a Public Web Server

When creating a security group for a public web server, it is essential to consider several factors to ensure both accessibility and security of the server:

1. **Inbound Rules:**
   - **HTTP (Port 80):** We allow HTTP traffic from any IP address (0.0.0.0/0) so that the web server can receive standard web requests.
   - **HTTPS (Port 443):** We allow HTTPS traffic from any IP address (0.0.0.0/0) to receive secure web requests.
   - **SSH (Port 22):** We only allow SSH access from specific IP addresses for remote administration. It is good practice to limit SSH access to the IP addresses of administrators.

2. **Outbound Rules:**
   - **Unlimited Access:** Generally, we allow all outbound traffic so that the web server can respond to user requests and access other Internet services as needed.

3. **Security Configuration:**
   - **IP Limits:** For administrative access (SSH), we should limit source IP addresses to a specific range or to individual IP addresses to reduce the risk of unauthorized access.
   - **Regular Review:** We should regularly review and update security rules to adapt to changes in security policies or network infrastructure.

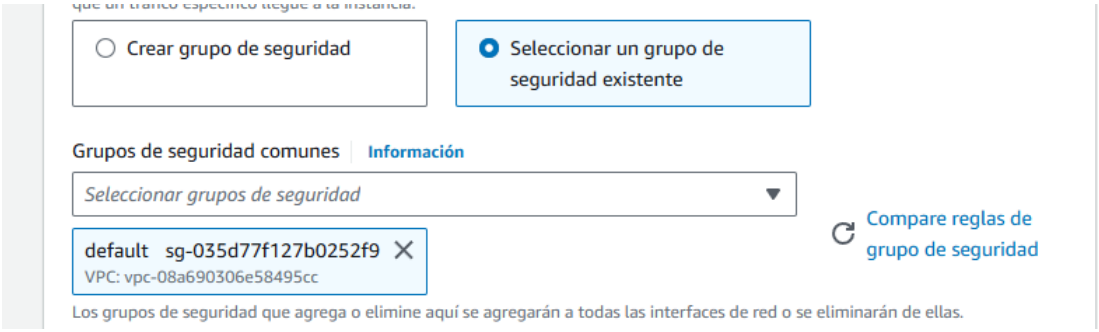


Illustration 18 Default group selection

g) On the **Review Instance Launch** page, we check the configuration and choose **Launch**.

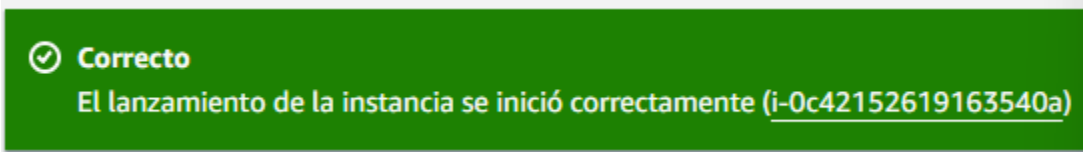


Illustration 19 Instance launch

Configuring Web Service

- We use PuTTy to connect to the created instance. First, we open 'PuTTYgen' and load the .pem key pair that was generated when we created the EC2 instance. To do this, we click on 'Load'

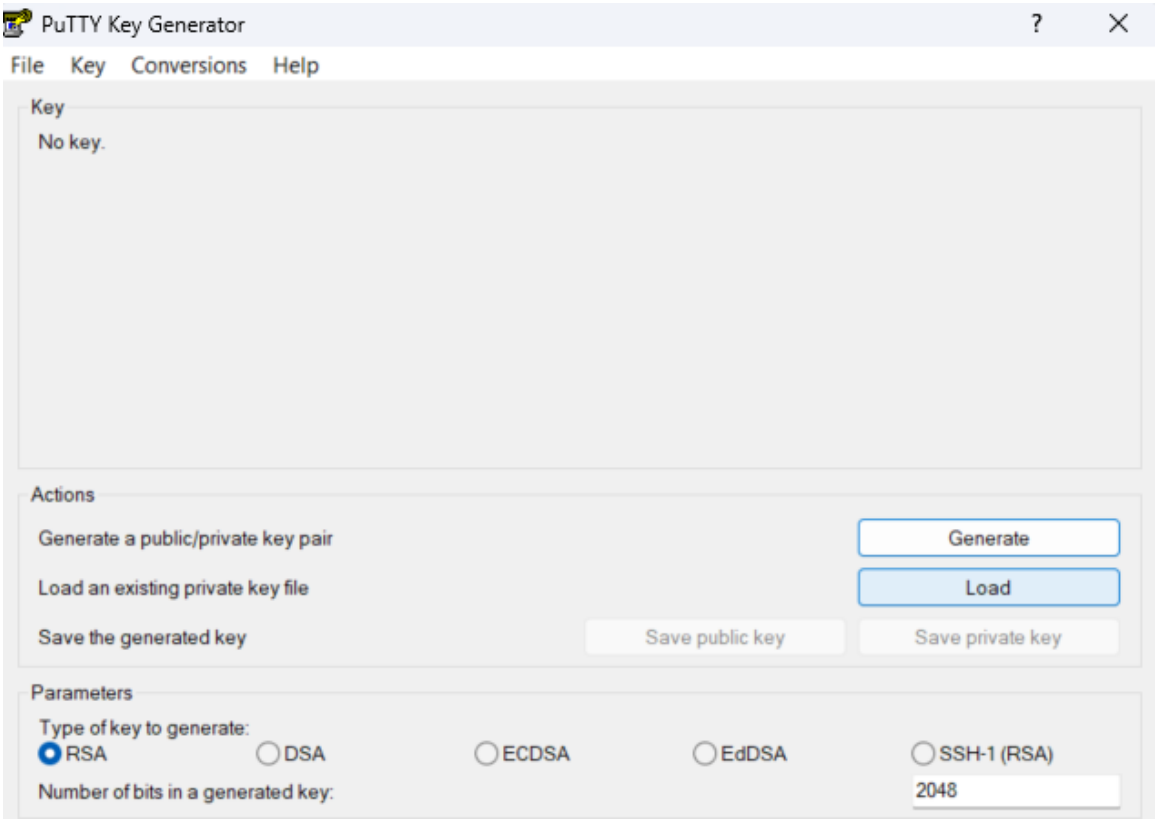


Illustration 20. Loading the key pair

- We click on 'Save private key' and save the key as a .ppk file so that it can be read by PuTTY

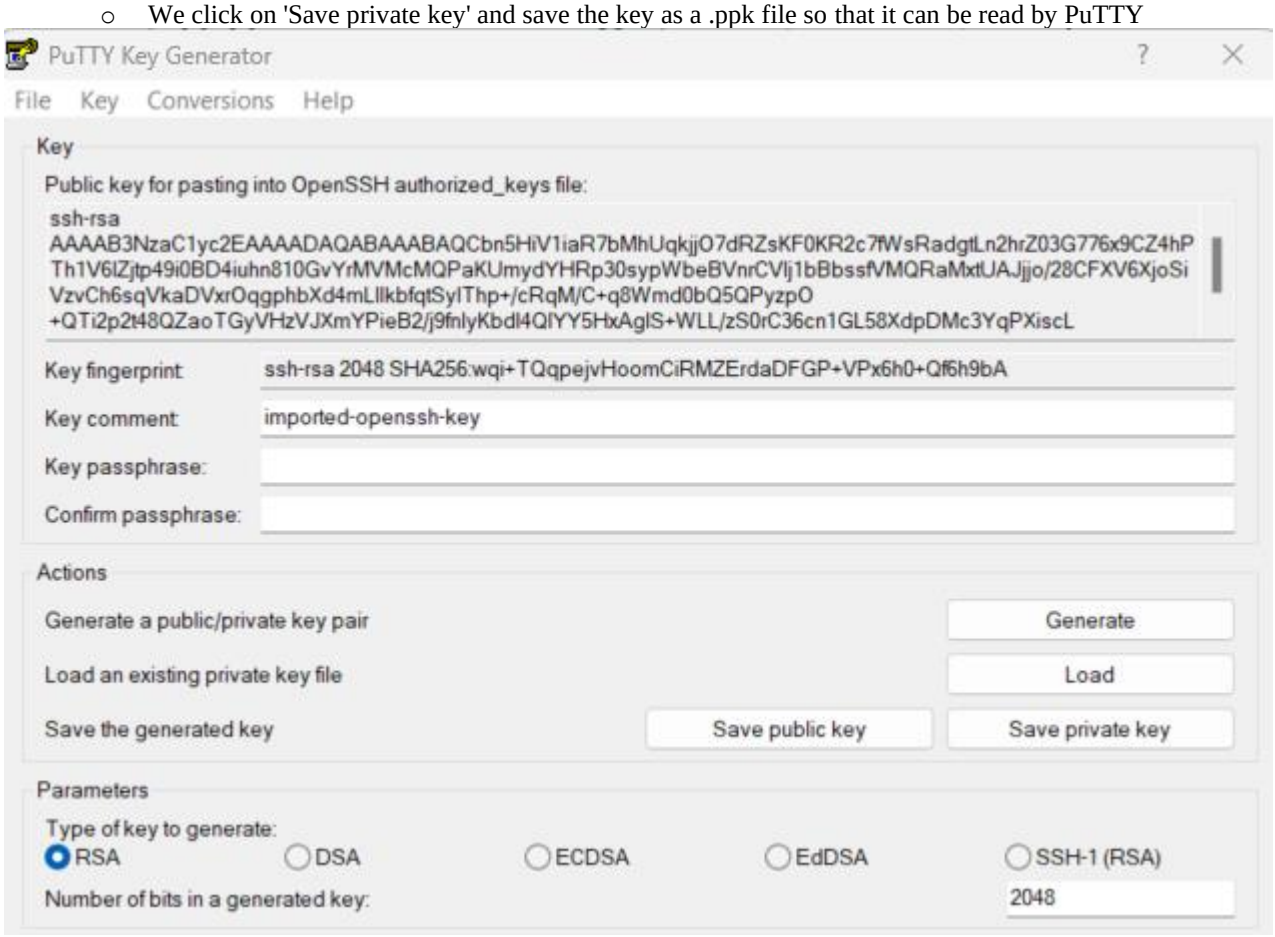


Illustration 21. Exporting the key in .ppk format

Illustration 22. Saving the .ppk file

- In AWS, we edit the inbound rules of the security group to allow SSH access on port

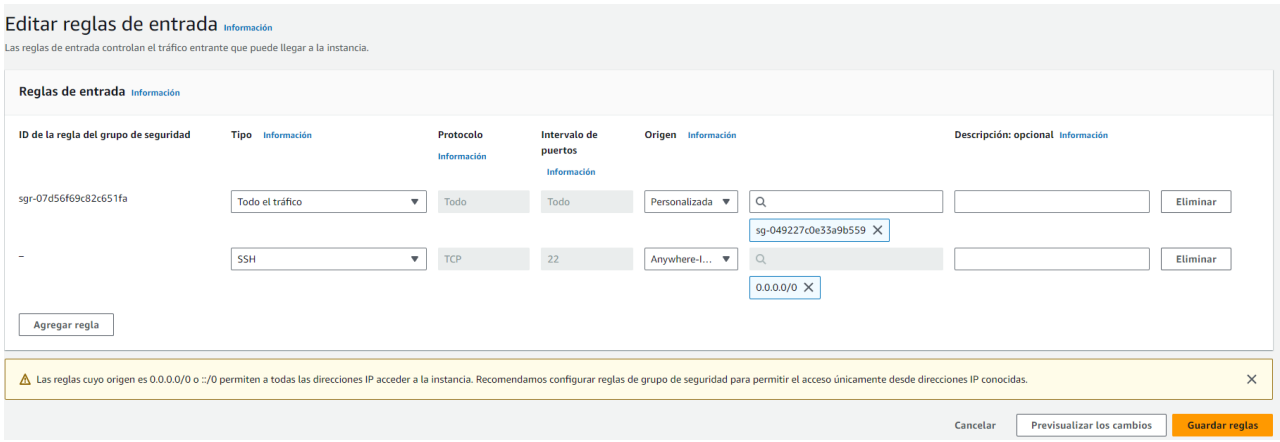


Illustration 23. Creating SSH Inbound Rule

- We open PuTTY and enter the IP address of the EC2 instance

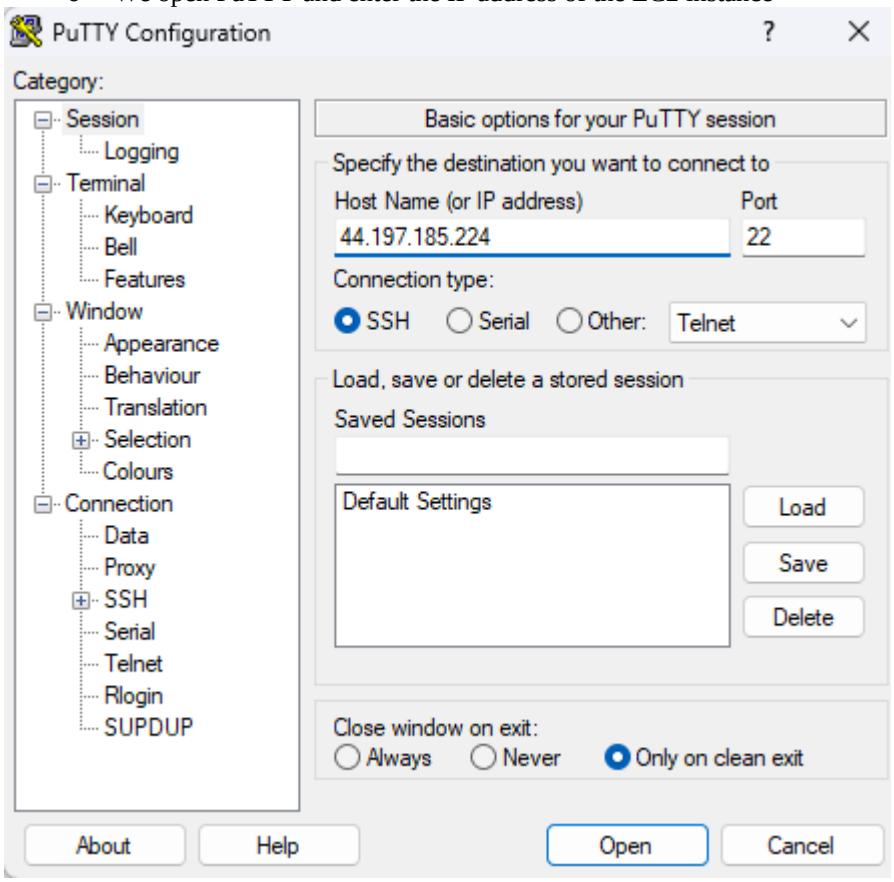


Illustration 24. Typing the IP address of the EC2 Instance

- In the left column, we go to SSH, then Auth and Credentials, and import the key that was saved earlier (*Image 22*)

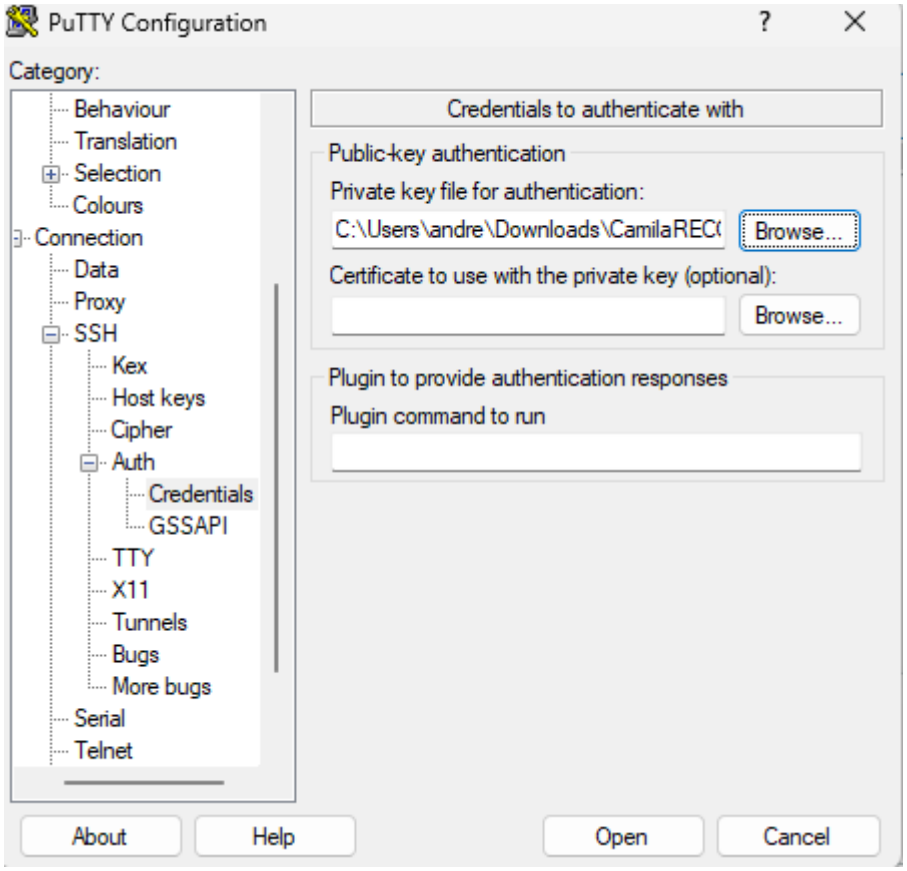


Illustration 25. Importing the key file

- We click on 'Open' and log in as ec2-use

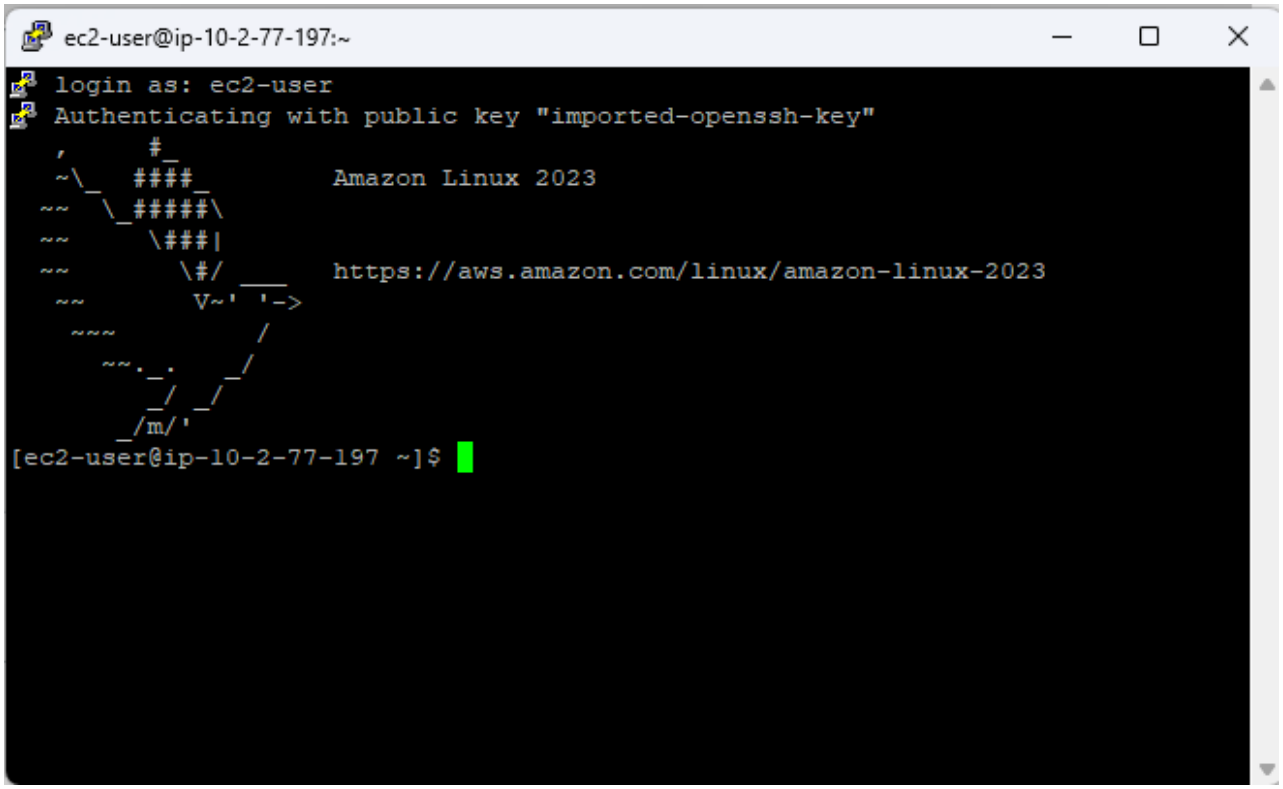


Illustration 26. Logging into Instance EC2

- We install httpd with the command 'sudo yum install httpd -y'

```
[ec2-user@ip-10-2-77-197 ~]$ sudo yum install httpd -y
Last metadata expiration check: 0:29:14 ago on Sat Oct 19 22:36:00 2024.
Dependencies resolved.
================================================================================
 Package               Arch     Version                    Repository     Size
================================================================================
Installing:
 httpd                 x86_64   2.4.62-1.amzn2023          amazonlinux    48 k
Installing dependencies:
 apr                   x86_64   1.7.2-2.amzn2023.0.2       amazonlinux   129 k
 apr-util              x86_64   1.6.3-1.amzn2023.0.1       amazonlinux    98 k
 generic-logos-httpd   noarch   18.0.0-12.amzn2023.0.3     amazonlinux    19 k
 httpd-core            x86_64   2.4.62-1.amzn2023          amazonlinux   1.4 M
 httpd-filesystem      noarch   2.4.62-1.amzn2023          amazonlinux    14 k
 httpd-tools           x86_64   2.4.62-1.amzn2023          amazonlinux    81 k
 libbrotli             x86_64   1.0.9-4.amzn2023.0.2       amazonlinux   315 k
 mailcap               noarch   2.1.49-3.amzn2023.0.3      amazonlinux    33 k
Installing weak dependencies:
 apr-util-openssl      x86_64   1.6.3-1.amzn2023.0.1       amazonlinux    17 k
 mod_http2             x86_64   2.0.27-1.amzn2023.0.3      amazonlinux   166 k
```

Illustration 27. Installing httpd

```
ec2-user@ip-10-2-77-197:~
  Verifying        : httpd-filesystem-2.4.62-1.amzn2023.noarch              7/12
  Verifying        : httpd-tools-2.4.62-1.amzn2023.x86_64                  8/12
  Verifying        : libbrotli-1.0.9-4.amzn2023.0.2.x86_64                 9/12
  Verifying        : mailcap-2.1.49-3.amzn2023.0.3.noarch                 10/12
  Verifying        : mod_http2-2.0.27-1.amzn2023.0.3.x86_64               11/12
  Verifying        : mod_lua-2.4.62-1.amzn2023.x86_64                     12/12

Installed:
  apr-1.7.2-2.amzn2023.0.2.x86_64
  apr-util-1.6.3-1.amzn2023.0.1.x86_64
  apr-util-openssl-1.6.3-1.amzn2023.0.1.x86_64
  generic-logos-httpd-18.0.0-12.amzn2023.0.3.noarch
  httpd-2.4.62-1.amzn2023.x86_64
  httpd-core-2.4.62-1.amzn2023.x86_64
  httpd-filesystem-2.4.62-1.amzn2023.noarch
  httpd-tools-2.4.62-1.amzn2023.x86_64
  libbrotli-1.0.9-4.amzn2023.0.2.x86_64
  mailcap-2.1.49-3.amzn2023.0.3.noarch
  mod_http2-2.0.27-1.amzn2023.0.3.x86_64
  mod_lua-2.4.62-1.amzn2023.x86_64

Complete!
[ec2-user@ip-10-2-77-197 ~]$
```

Illustration 28. Httpd Installation completed

- We go to AWS, select the instance, and add a new inbound rule to the security group for HTTP on port 80

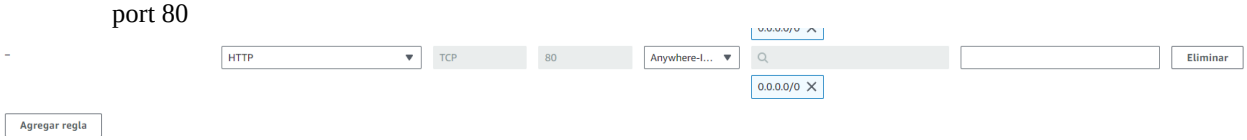


Illustration 29. Creating HTTP Inbound Rule

- We start the service with the command 'systemctl start httpd', then enable the service with 'systemctl enable httpd'. Finally, we insert an <h1> tag into the index.html that was generated when the service was created, located in the directory /var/www/html/index.html

```
ec2-user@ip-10-2-77-197 ~]$ sudo systemctl start httpd
ec2-user@ip-10-2-77-197 ~]$ sudo systemctl enable  httpd
ec2-user@ip-10-2-77-197 ~]$ echo "<h1>Hello world from $(hostname -f)</h1>" > /var/www/html/index.html
bash: /var/www/html/index.html: Permission denied
ec2-user@ip-10-2-77-197 ~]$ sudo echo "<h1>Hello world from $(hostname -f)</h1>" > /var/www/html/index.html
bash: /var/www/html/index.html: Permission denied
ec2-user@ip-10-2-77-197 ~]$ sudo echo "<h1>Hello world from $(hostname -f)</h1>" | sudo tee /var/www/html/index.html
h1>Hello world from ip-10-2-77-197.ec2.internal</h1>
ec2-user@ip-10-2-77-197 ~]$
```

Illustration 30. Starting Web Service

- We make the service available upon instance restart with the command 'sudo chkconfig httpd on'

```
ec2-user@ip-10-2-77-197 ~]$ sudo chkconfig httpd on
ote: Forwarding request to 'systemctl enable httpd.service'.
ec2-user@ip-10-2-77-197 ~]$
```

Illustration 31. Configuring the Web Server to start when the instance restarts

- We open the browser, enter the IP address of our instance, and confirm that the web server is working, with the <h1> tag we added in the index.html visible

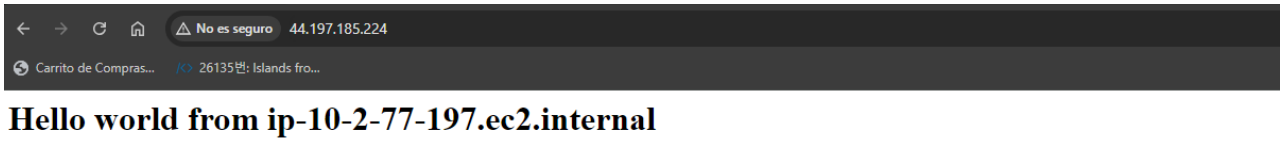


Illustration 32 Verifying that the web server is working

Adding content to the web server that connects to Amazon database instance

Illustration 33 creation second subnet

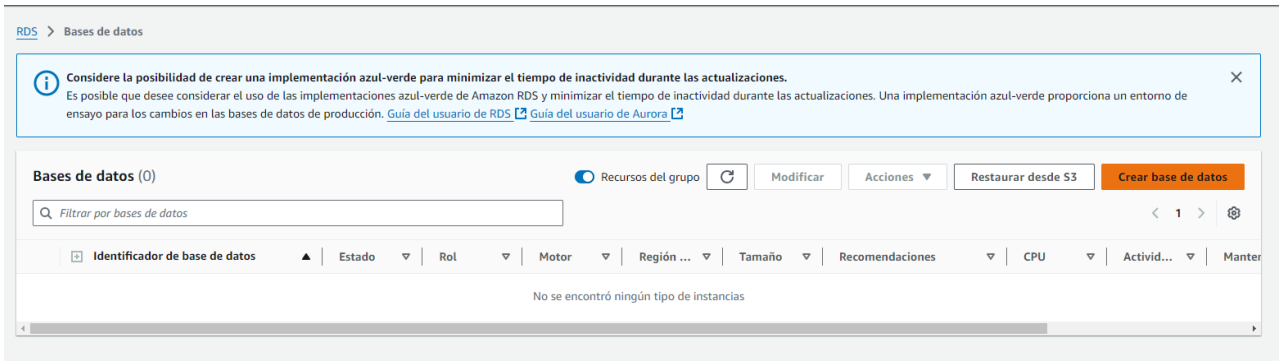


Illustration 34 creation database

We created a new standard and selected the MySQL engine.

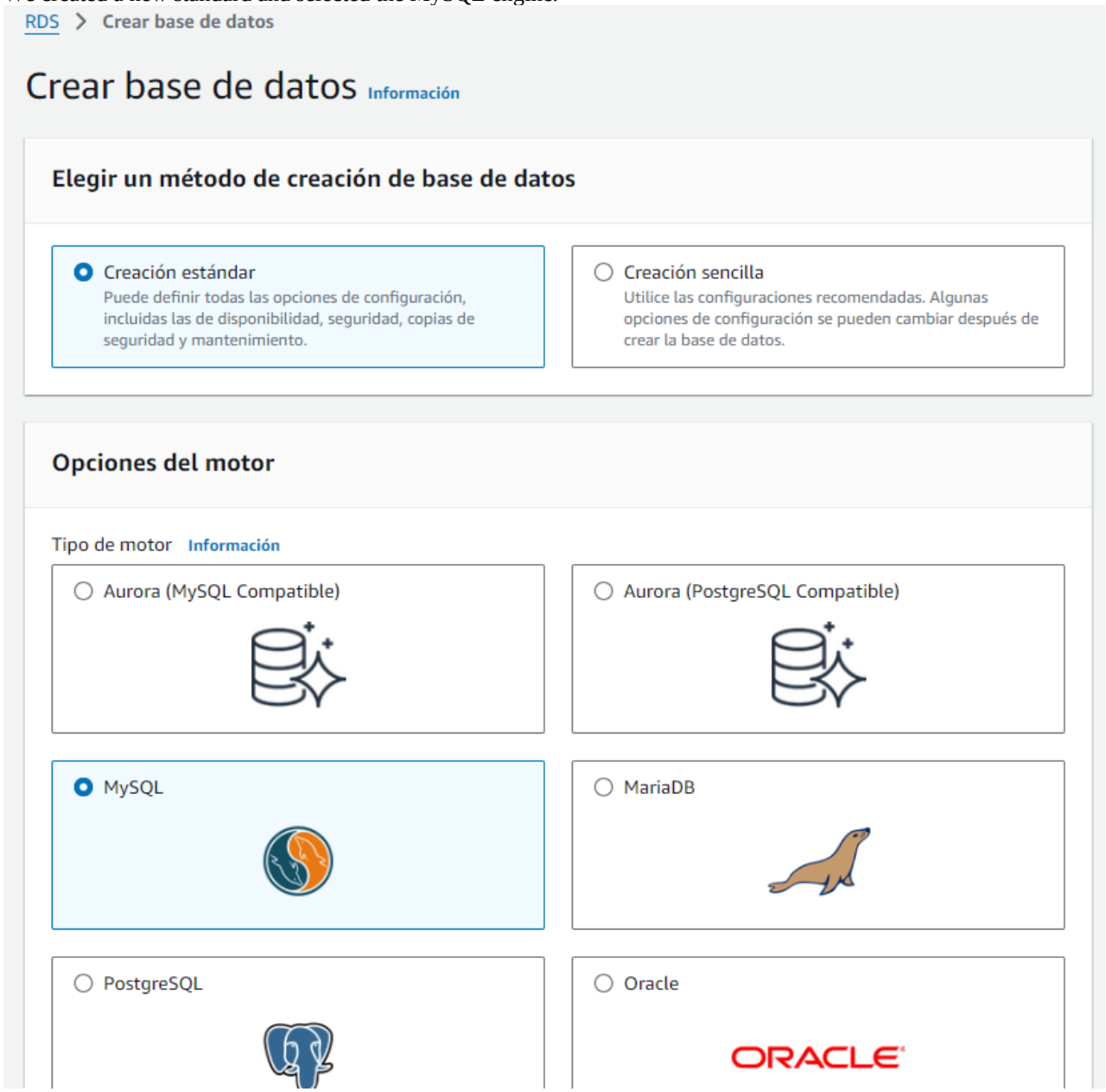


Illustration 35 selection of a database

We kept the MySQL engine and set the free template.

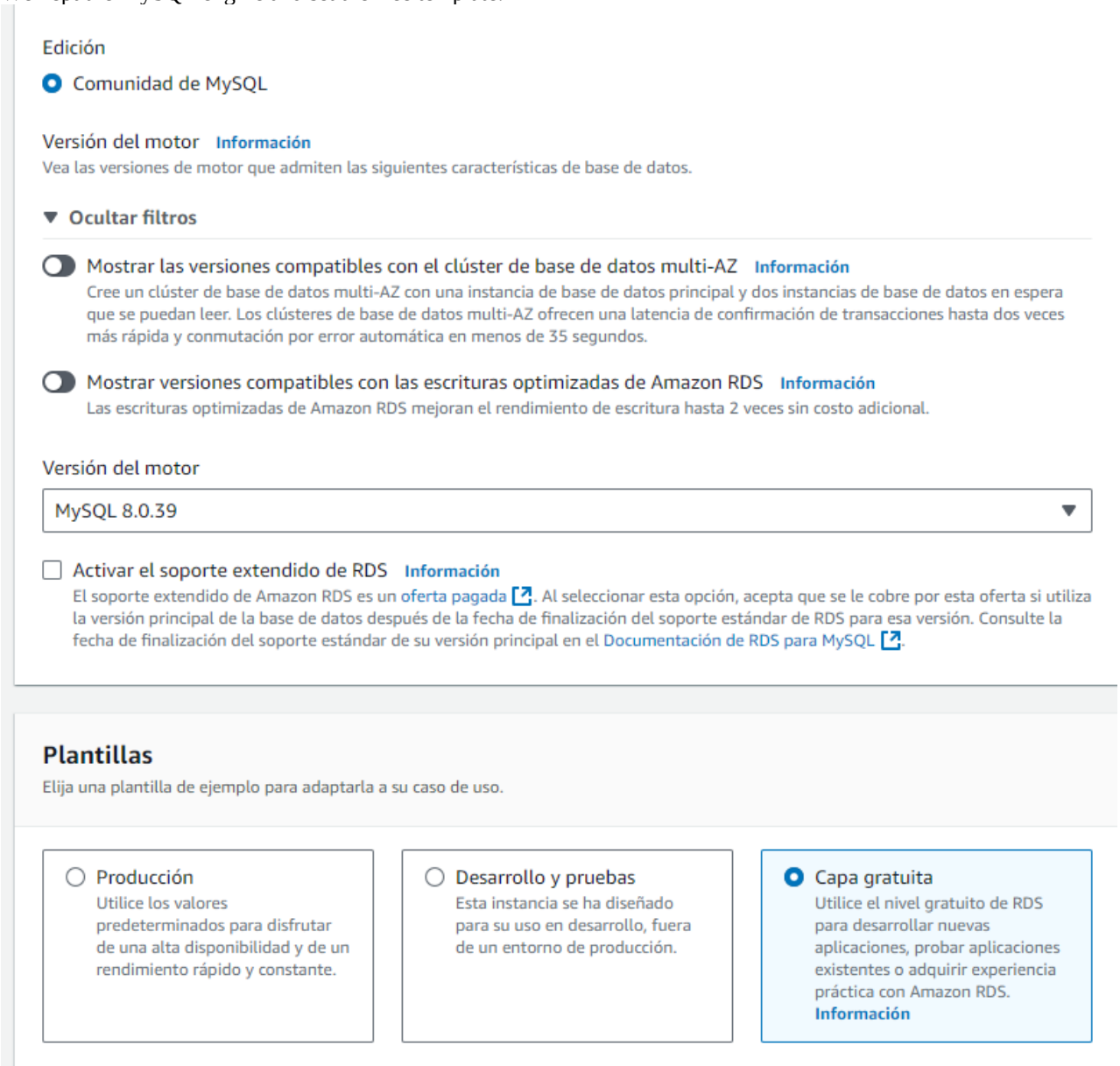


Illustration 36 Engine and template configuration

We configured the credentials and set a password for our database.

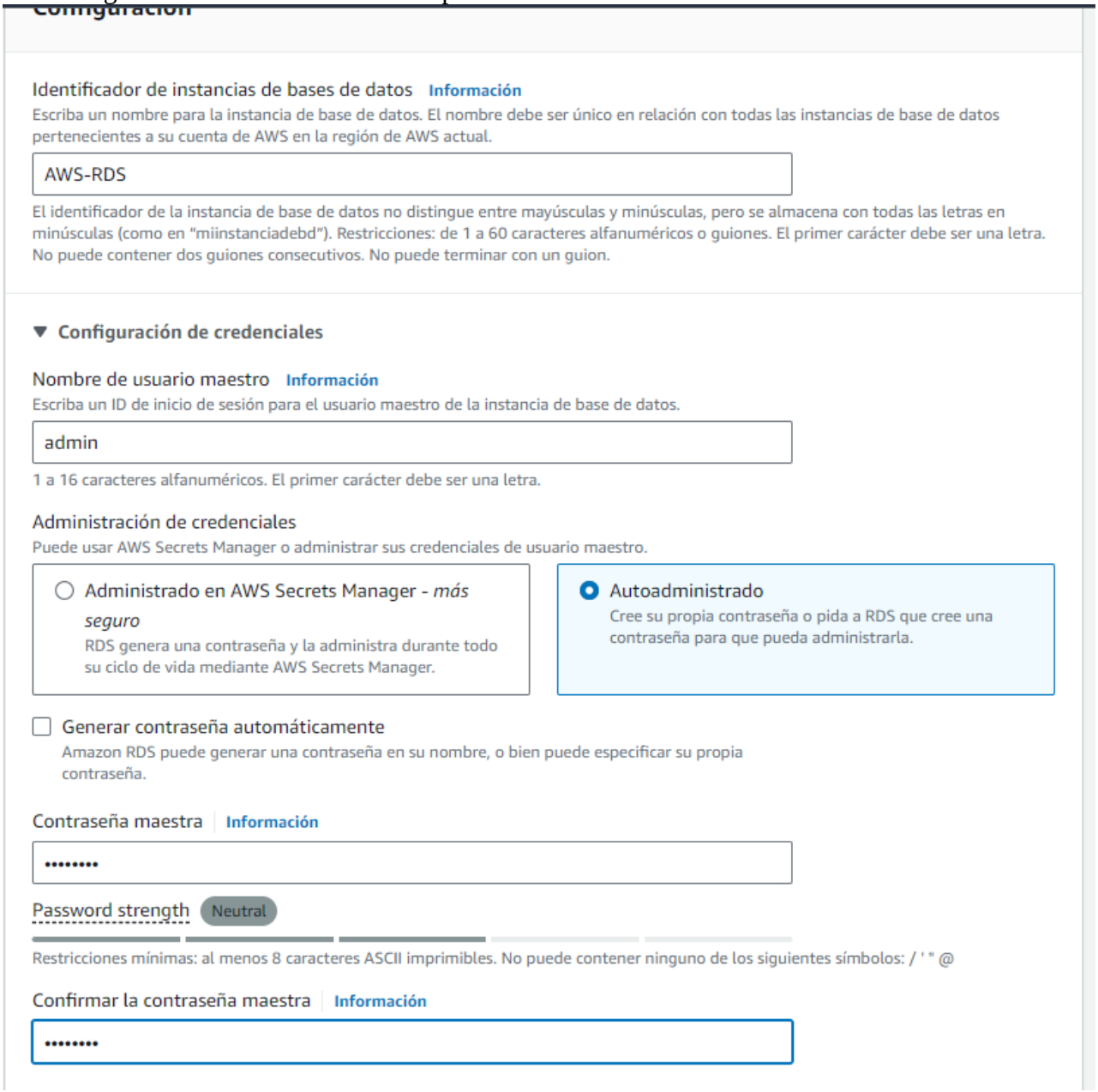


Illustration 37 password configuration

For the instance configuration, we chose the default options.

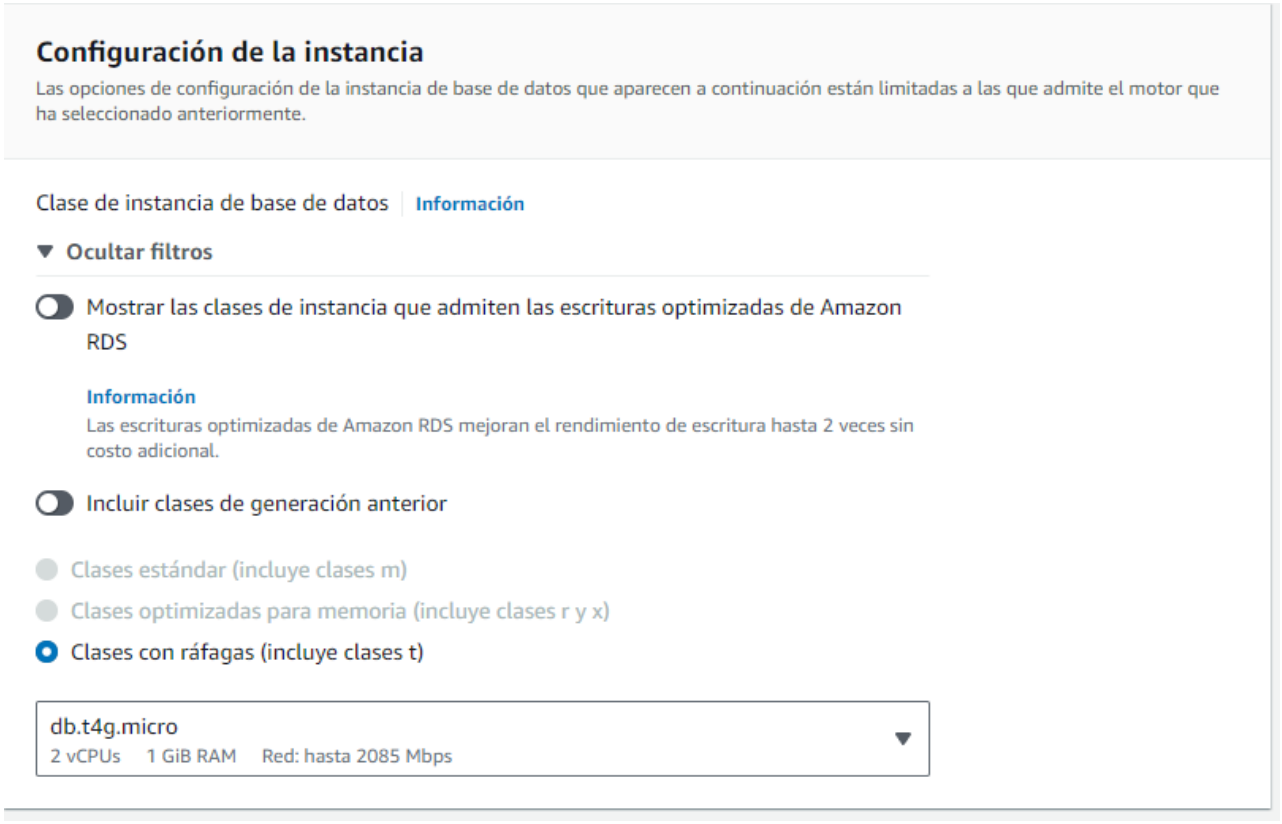


Illustration 38 database instance configuration

We left the storage configuration at the minimum default setting.

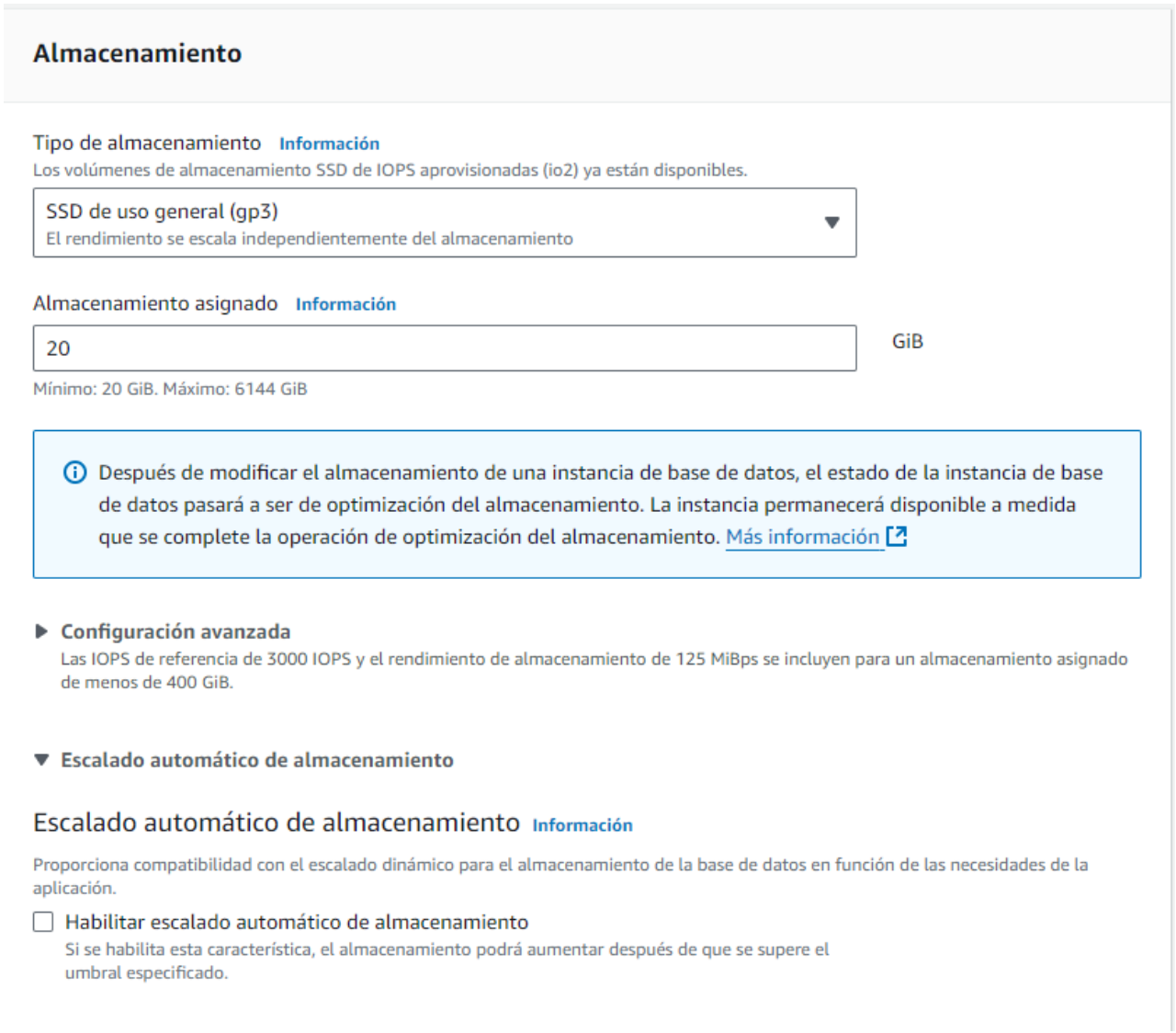


Illustration 39 ***storage configuration.***

We chose to connect to the VPC we created earlier, which now has two subnets to manage the database, and we selected a default subnet group.

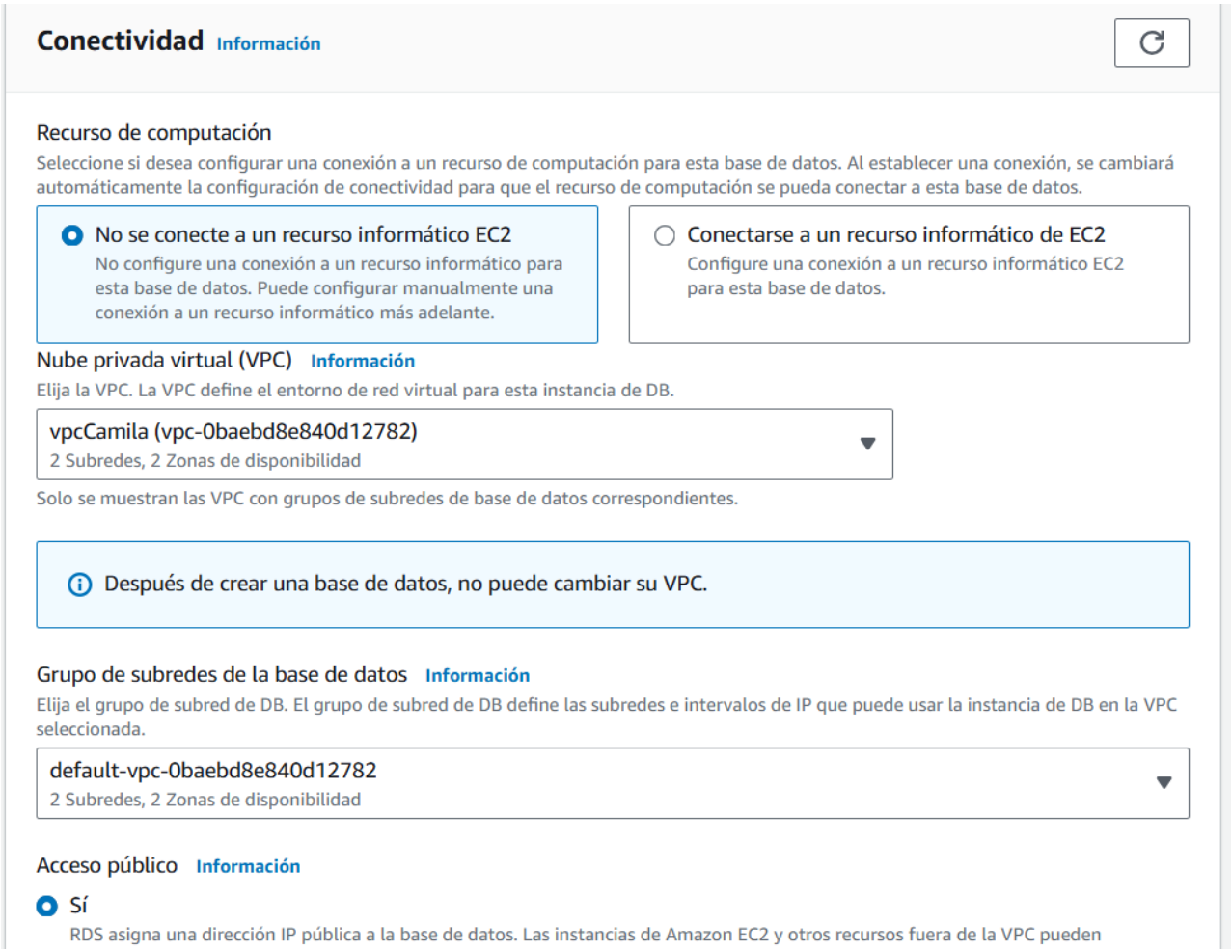


Illustration 40 ***connectivity configuration.***

We chose an existing security group, in this case, the default, and left the other configurations at their default settings.

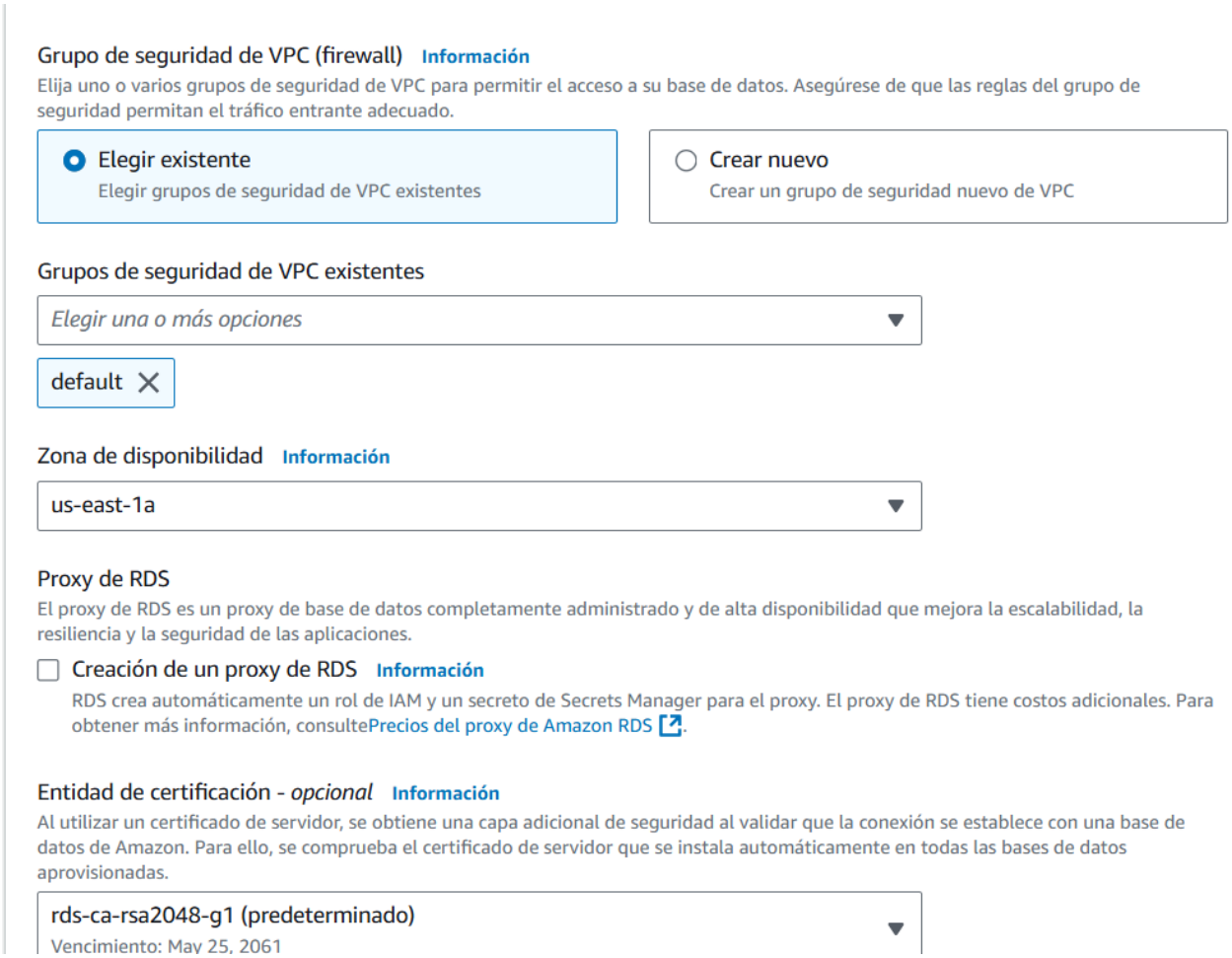


Illustration 41 security group configuration.

We reviewed the database port to connect to it later.

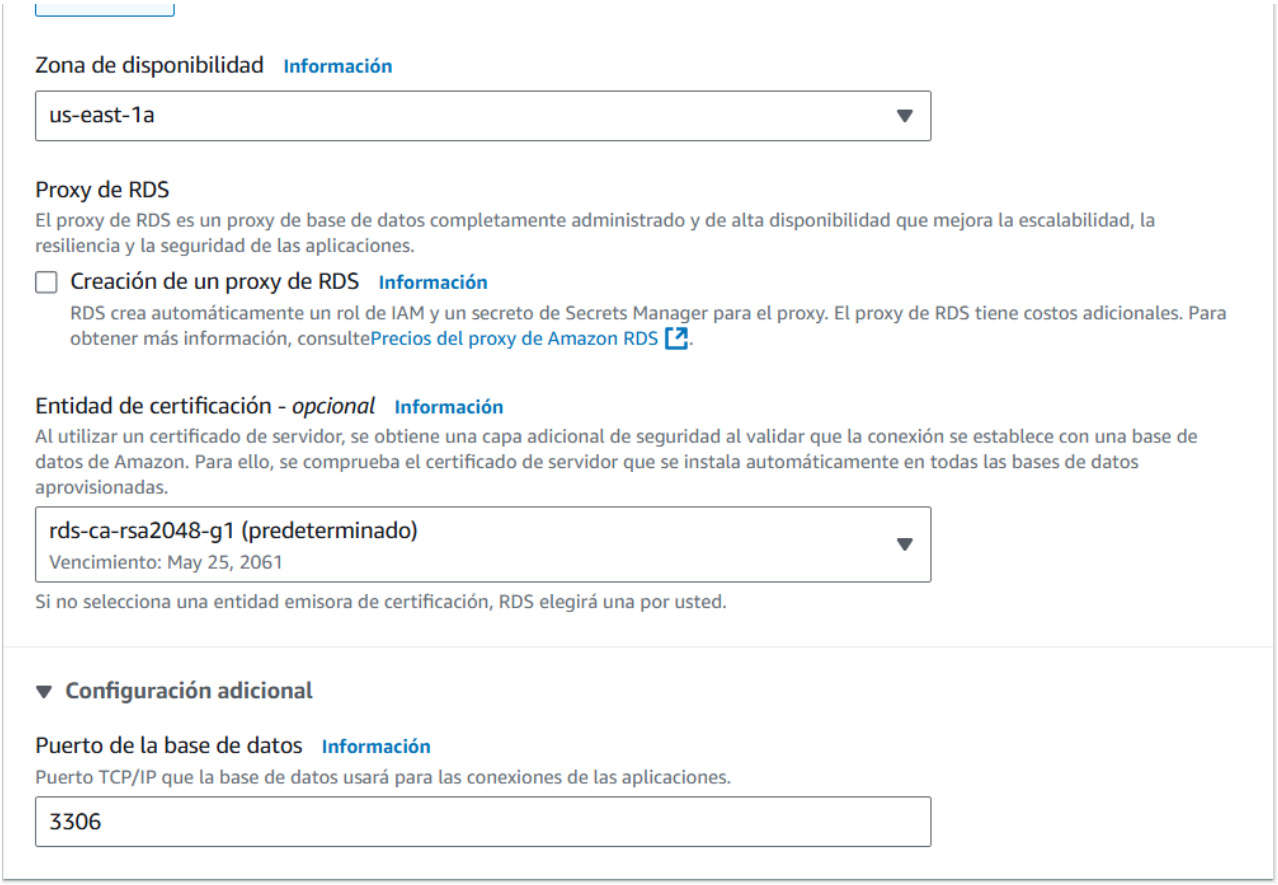


Illustration 42 database port configuration.

We left password authentication enabled to access the database.

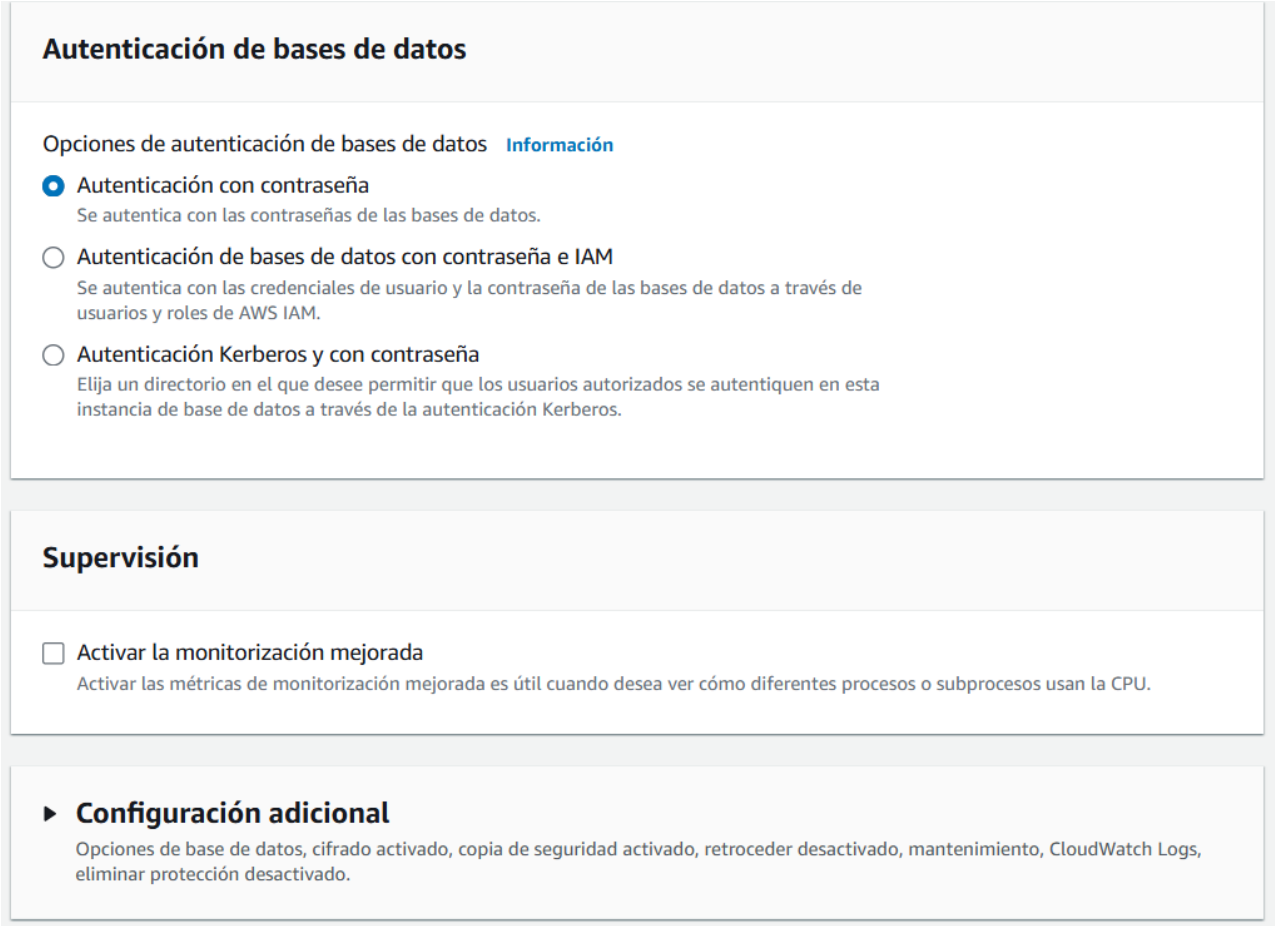


Illustration 43 authentication configuration.

We selected to create a database.

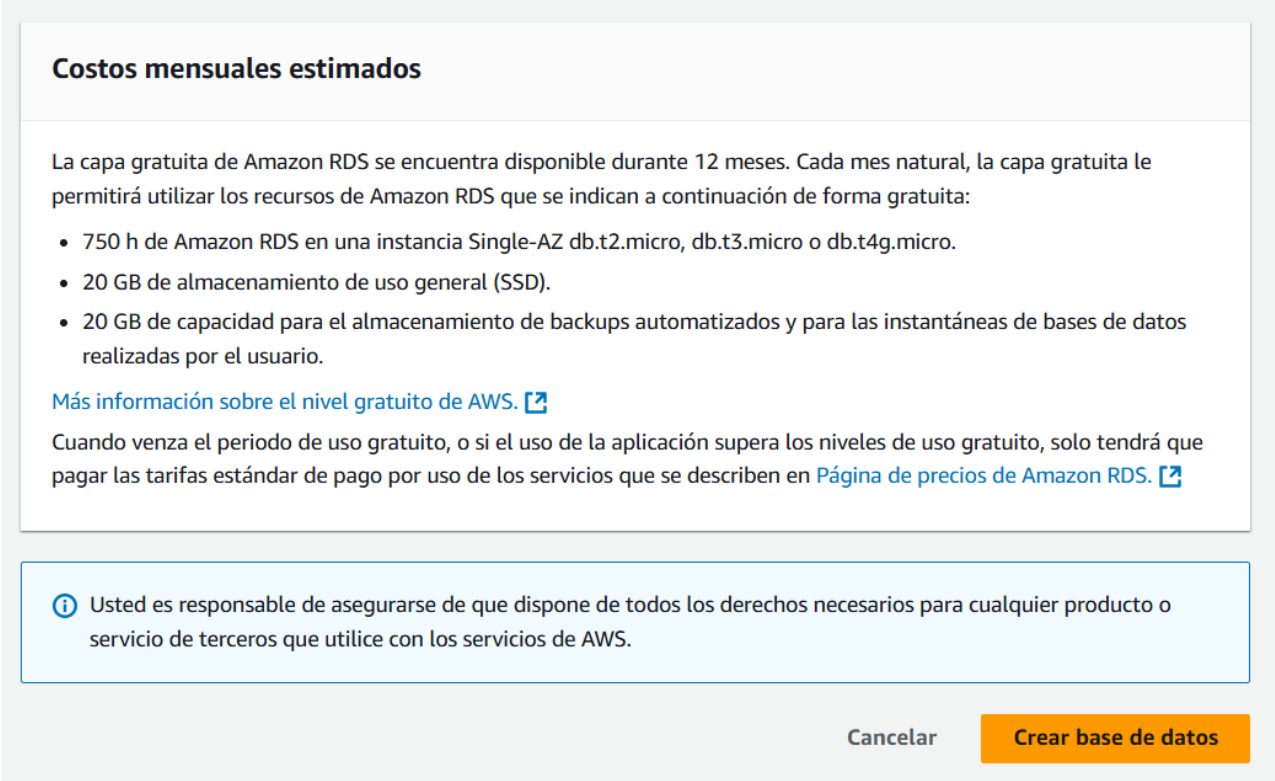


Illustration 44 database creation finished.

We configure the security group rules to allow traffic to the database.

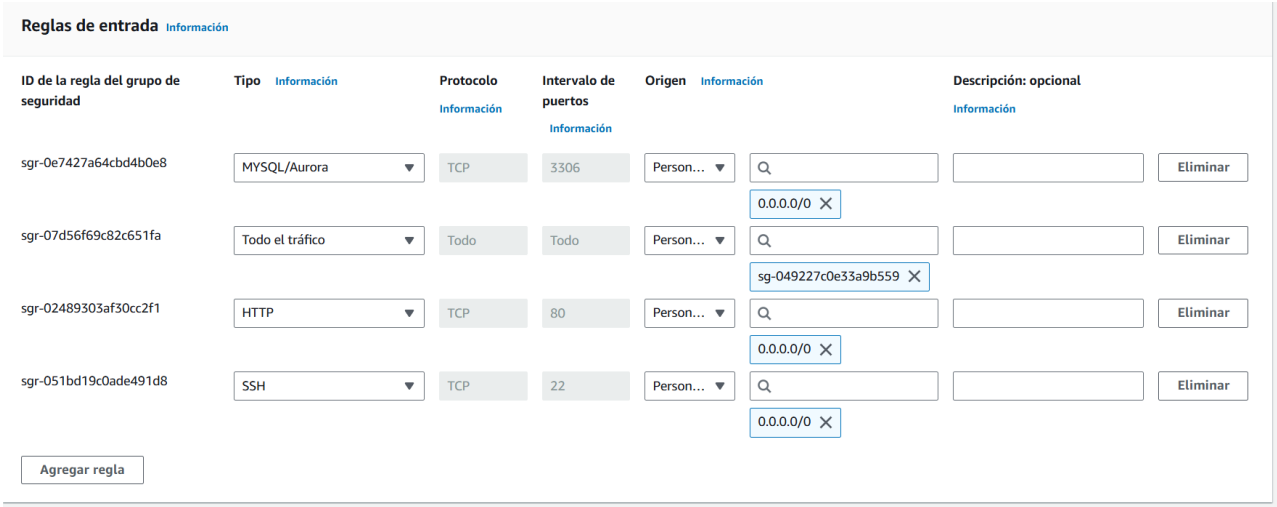


Illustration 45 security group rules configuration

We connected the database and the instance so they can interact.

Illustration 46 database and instance connection.

We reviewed that all the configurations were correct and accepted.

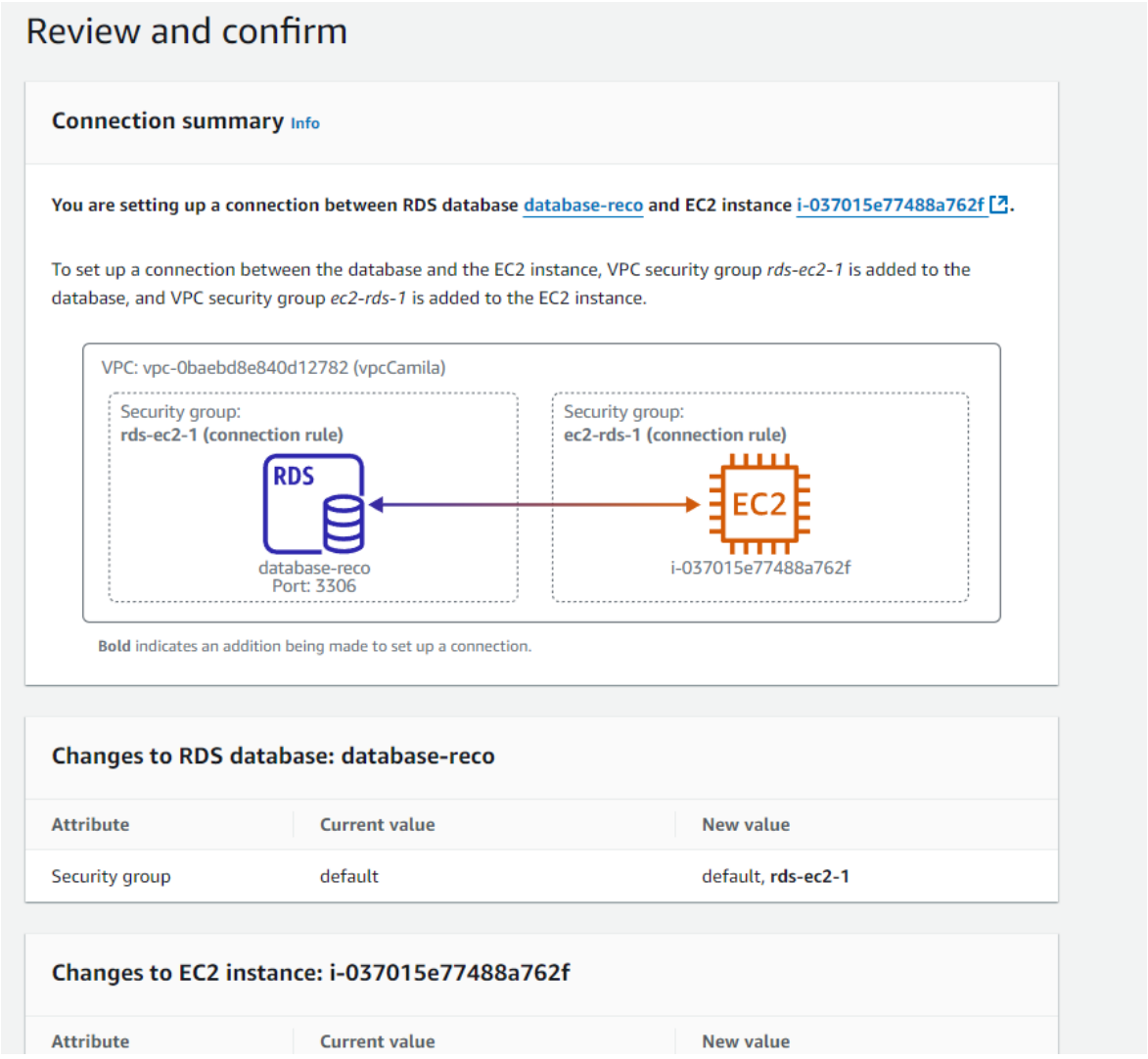


Illustration 47 connection confirmation.

A notification is displayed that the connection was successful.

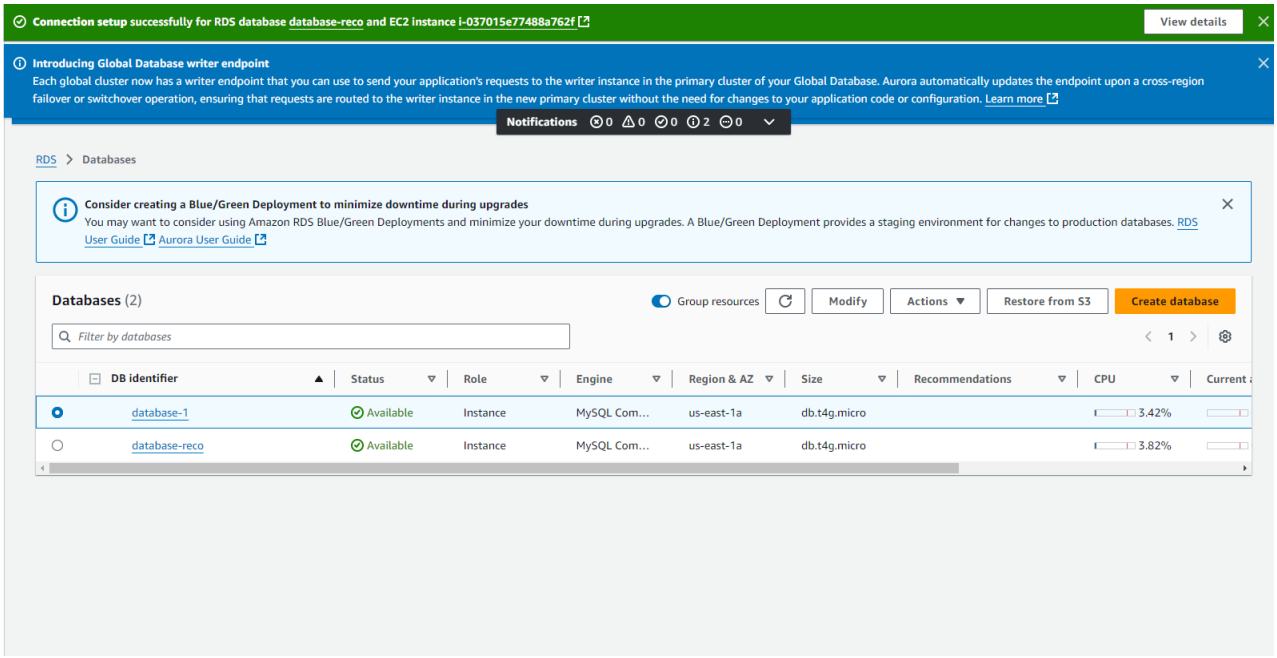


Illustration 48 notification launch.

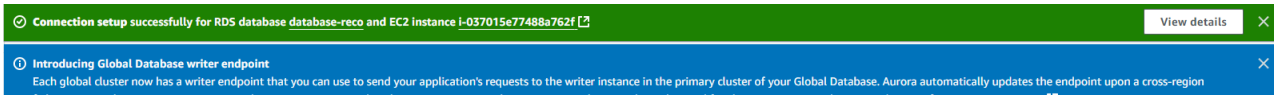


Illustration connection established message.

We installed the database client to interact with the database, using the command "sudo yum install mysql-community-server -y" for the installation.

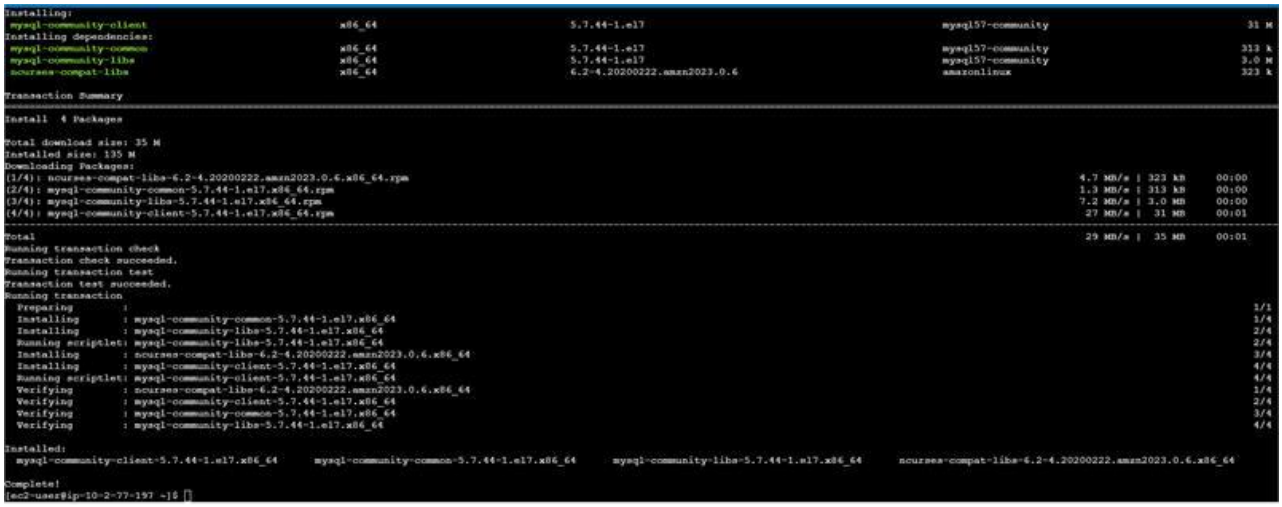


Illustration 49 client installation on the instance.

```
Complete!
[ec2-user@ip-10-2-77-197 ~]$ mysql --version
mysql  Ver 14.14 Distrib 5.7.44, for Linux (x86_64) using  EditLine wrapper
[ec2-user@ip-10-2-77-197 ~]$ mysql -h database-reco.cpaconw7a6nt.us-east-1.rds.amazonaws.com -P 3306 -u admin -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 28
Server version: 8.0.39 Source distribution

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

Illustration 50 installation completed

We created a database using the command USE [database name]. After that, we created an authors table and added three entries. To verify if it was done correctly, we executed a query on the table.

```
mysql> CREATE DATABASE nombre_de_tu_base_de_datos;
Query OK, 1 row affected (0.02 sec)

mysql> USE nombre_de_tu_base_de_datos;
Database changed
mysql> CREATE TABLE Authors (     AuthorID INT AUTO_INCREMENT PRIMARY KEY,     Name NVARCHAR(100) NOT NULL );
Query OK, 0 rows affected, 1 warning (0.03 sec)

mysql> INSERT INTO Authors (Name) VALUES
    -> ('John Doe'),
    -> ('Jane Smith'),
    -> ('Alice Johnson');
Query OK, 3 rows affected (0.00 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM Authors;
+----------+---------------+
| AuthorID | Name          |
+----------+---------------+
|        1 | John Doe      |
|        2 | Jane Smith    |
|        3 | Alice Johnson |
+----------+---------------+
3 rows in set (0.00 sec)

mysql>
```

Illustration 51. data creation and insertion.

Displaying table in Browser

- Now we're going to see our table on the browser. To do that, we must install PHP
  We enter the command **'sudo yum install -y php56 php56-mysqlnd'**

```
[ec2-user@ip-10-2-77-197 ~]$ sudo yum clean metadata
sudo yum install -y php php-mysqlnd
Cache was expired
22 files removed
Amazon Linux 2023 repository                    18 MB/s |  28 MB     00:01
```

Illustration 52. Starting PHP installation

```
  Verifying        : php8.3-opcache-8.3.10-1.amzn2023.0.1.x86_64          10/14
  Verifying        : php8.3-pdo-8.3.10-1.amzn2023.0.1.x86_64              11/14
  Verifying        : php8.3-process-8.3.10-1.amzn2023.0.1.x86_64          12/14
  Verifying        : php8.3-sodium-8.3.10-1.amzn2023.0.1.x86_64           13/14
  Verifying        : php8.3-xml-8.3.10-1.amzn2023.0.1.x86_64              14/14

Installed:
  libsodium-1.0.19-4.amzn2023.x86_64
  libxslt-1.1.34-5.amzn2023.0.2.x86_64
  nginx-filesystem-1:1.24.0-1.amzn2023.0.4.noarch
  php8.3-8.3.10-1.amzn2023.0.1.x86_64
  php8.3-cli-8.3.10-1.amzn2023.0.1.x86_64
  php8.3-common-8.3.10-1.amzn2023.0.1.x86_64
  php8.3-fpm-8.3.10-1.amzn2023.0.1.x86_64
  php8.3-mbstring-8.3.10-1.amzn2023.0.1.x86_64
  php8.3-mysqlnd-8.3.10-1.amzn2023.0.1.x86_64
  php8.3-opcache-8.3.10-1.amzn2023.0.1.x86_64
  php8.3-pdo-8.3.10-1.amzn2023.0.1.x86_64
  php8.3-process-8.3.10-1.amzn2023.0.1.x86_64
  php8.3-sodium-8.3.10-1.amzn2023.0.1.x86_64
  php8.3-xml-8.3.10-1.amzn2023.0.1.x86_64

Complete!
[ec2-user@ip-10-2-77-197 ~]$
```

Illustration 53 PHP installation completed

- Now, we open the file /var/www/html/conexion.php and insert the following code. In this code, we specify the endpoint, username, password, and database name. After that, we establish the connection to MySQL using these credentials and display a message indicating that the connection was successful

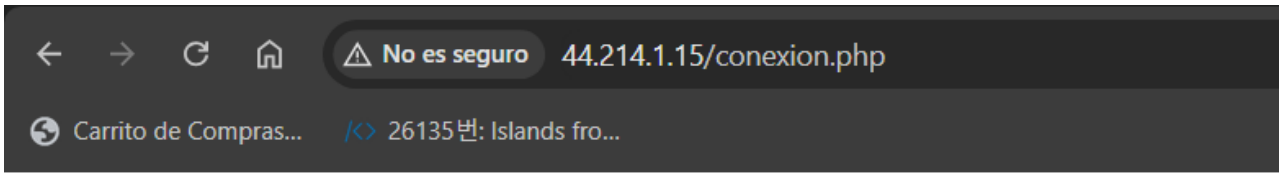
```php
<?php
$servername = "database-reco.cpaconw7a6nt.us-east-1.rds.amazonaws.com"
$username = "admin";
$password = "recO.80.";
$dbname = "nombre_de_tu_base_de_datos";

// Crear la conexión
$conn = new mysqli($servername, $username, $password, $dbname);

// Verificar la conexión
if ($conn->connect_error) {
    die("Conexión fallida: " . $conn->connect_error);
}
echo "Conexión exitosa";
?>
```

Illustration 54 Using a script to connect to MySQL with PHP

- We start the service with **sudo service httpd start**, then we open the browser, enter the IP address of the instance with the file that we created before



Conexión exitosa

Illustration 20. Connecting to the database in the browser

- Now we edit the file to display the table. Inside this file, we perform a query to the database and then display the table using <tr></tr>

```php
<?php
$servername = "database-reco.cpaconw7a6nt.us-east-1.rds.amazonaws.com";
$username = "admin";
$password = "recO.80.";
$dbname = "nombre_de_tu_base_de_datos";

// Crear la conexión
$conn = new mysqli($servername, $username, $password, $dbname);

// Verificar la conexión
if ($conn->connect_error) {
    die("Conexión fallida: " . $conn->connect_error);
}
echo "Conexión exitosa<br>";

// SQL para seleccionar datos de la tabla
$sql = "SELECT * FROM Authors";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
    // Crear tabla HTML
    echo "<table border='1'>
            <tr>
                <th>ID</th>
                <th>Name</th>
            </tr>";
    // Imprimir los datos de cada fila
    while($row = $result->fetch_assoc()) {
        echo "<tr>
                <td>" . $row["AuthorID"] . "</td>
                <td>" . $row["Name"] . "</td>
              </tr>";
    }
    echo "</table>";
} else {
    echo "No hay resultados";
}

$conn->close();
?>
```

Illustration 55 Displaying the table using PHP

- We open the browser again. We can see the table displayed in it

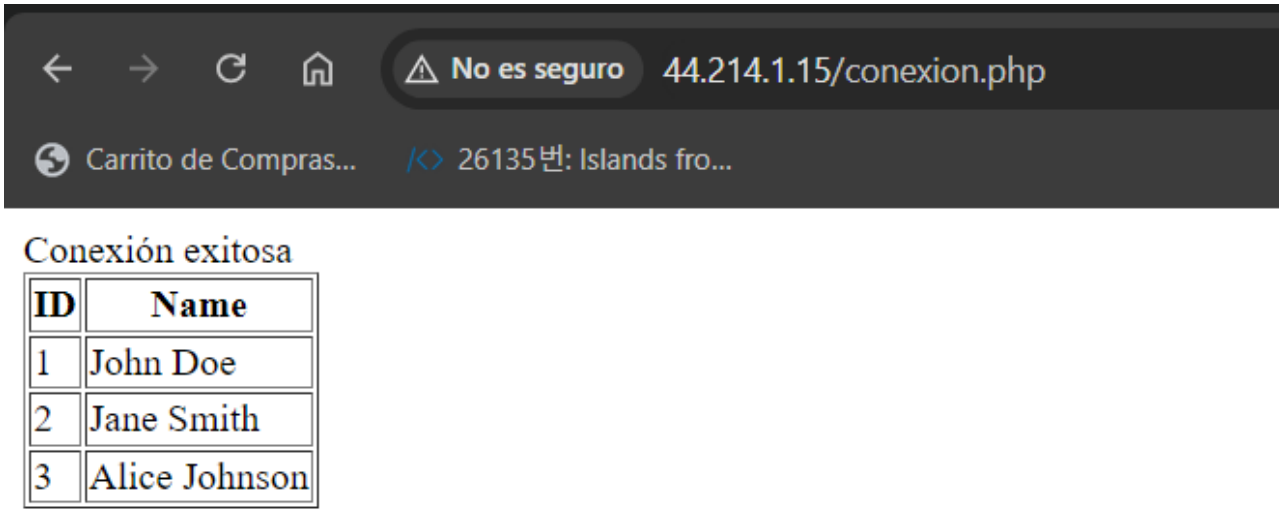


Illustration 56 Displaying database table in the browser

Conclusions

- The implementation of a web server on Amazon EC2 represents an effective and efficient solution for organizations looking to scale and manage their applications in a cloud environment. This strategy not only leverages the flexibility of cloud computing but also aligns with the pay-as-you-go model, resulting in a more rational use of financial resources.

- A fundamental part of this implementation is the proper configuration of a Virtual Private Cloud (VPC), which provides a controlled and secure network environment. The segmentation of resources and the application of security policies are crucial for protecting data and ensuring that only authorized users have access to critical resources.

- Additionally, the variety of instances available in EC2 allows users to select the most suitable options for their specific workloads. This not only optimizes performance but also contributes to more effective cost management, adapting to the changing needs of the organization.

- The ability to quickly scale EC2 instances, especially through the integration of services like Auto Scaling, offers businesses the agility needed to respond to market demand fluctuations. This feature ensures that resources are available when needed, minimizing downtime and improving user experience.

- Finally, the use of Amazon RDS simplifies database management by automating administrative tasks such as backups, software updates, and scaling. This allows developers and IT teams to focus on creating innovative applications and improving operational efficiency, rather than spending time on managing the underlying infrastructure.

Bibliography

- Nuvalab [@nuuvalab]. (n.d.). *Cómo Crear una VPC en AWS (Tutorial en Español)* 💻. Youtube. Retrieved October 19, 2024, from https://www.youtube.com/watch?v=b3nlvl2Rt4Q
- (N.d.-a). Amazon.com. Retrieved October 19, 2024, from https://aws.amazon.com/es/ec2/instance-types/
- (N.d.-b). Amazon.com. Retrieved October 19, 2024, from https://docs.aws.amazon.com/es_es/AWSEC2/latest/UserGuide/instance-types.html#AvailableInstanceTypes
- (N.d.-c). Amazon.com. Retrieved October 19, 2024, from https://docs.aws.amazon.com/ec2/latest/instancetypes/ec2-instance-type-specifications.html
- (N.d.-d). Amazon.com. Retrieved October 19, 2024, from https://aws.amazon.com/es/ec2/instance-types/t2/
- (N.d.-e). Amazon.com. Retrieved October 19, 2024, from https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_Tutorials.WebServerDB.CreateVPC.html#CHAP_Tutorials.WebServerDB.CreateVPC.VPCAndSubnets
- (N.d.-f). Amazon.com. Retrieved October 19, 2024, from https://docs.aws.amazon.com/es_es/AWSEC2/latest/UserGuide/ec2-security-groups.html