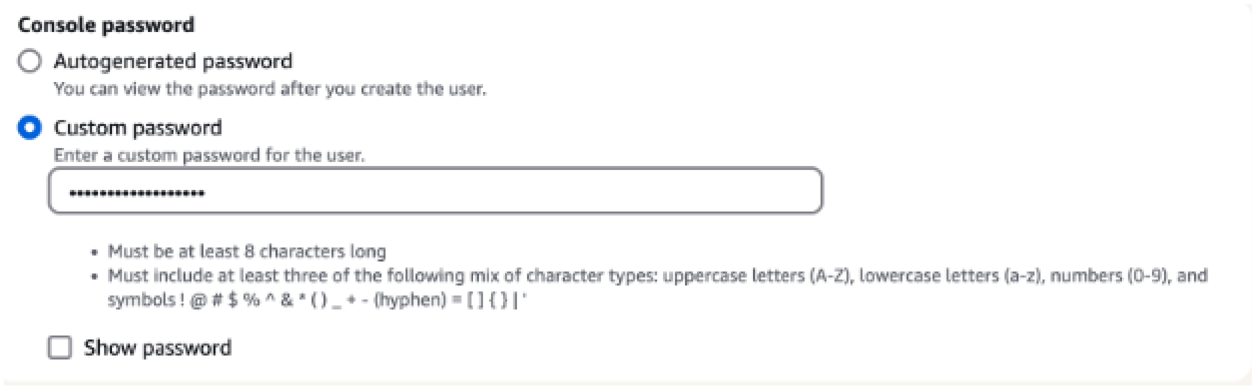
HOW TO CREATE AN AMAZON AURORA DATABASE AND CONNECT IT WITH AMAZON EC2

➢ CREATE AN IAM USER

- Head to your AWS Account as the root user.
- Open the **AWS IAM** console.
- From the left hand navigation panel, choose **Users.**
- Choose **Create user.**
- For the User name, use Yourname-IAM-Admin
- Make sure to select the checkbox next to **Provide user access to the AWS Management Console - optional.**
- If you're prompted with a pop up panel that says **Are you providing access to a person?**, choose **I want to create an IAM user.**
- For the console password, choose **Custom password.**
- Type in a password that you will be able to remember/access in the future.



Console password

○ Autogenerated password
You can view the password after you create the user.

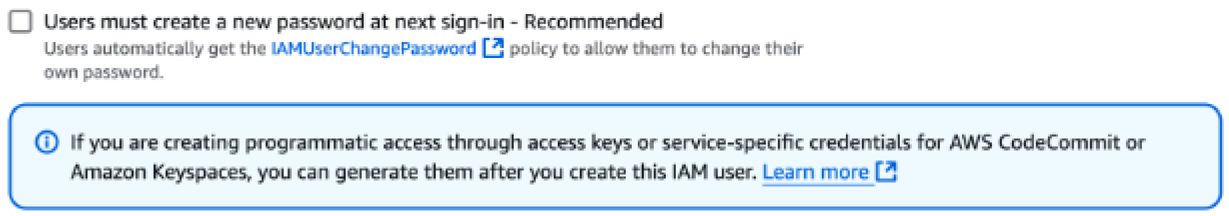
● Custom password
Enter a custom password for the user.

••••••••••••••••••

- Must be at least 8 characters long
- Must include at least three of the following mix of character types: uppercase letters (A-Z), lowercase letters (a-z), numbers (0-9), and symbols ! @ # $ % ^ & * ( ) _ + - (hyphen) = [ ] { } | '

☐ Show password

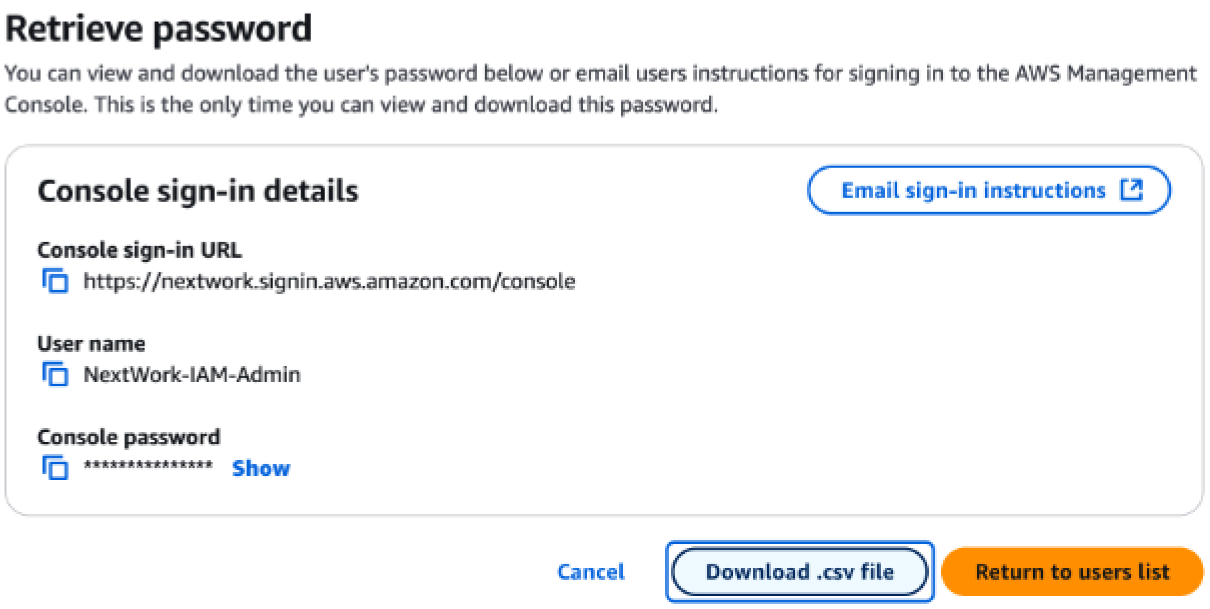
Deselect the checkbox for **Users must create a new password at next sign-in - Recommended.**



☐ Users must create a new password at next sign-in - Recommended
Users automatically get the IAMUserChangePassword policy to allow them to change their own password.

ⓘ If you are creating programmatic access through access keys or service-specific credentials for AWS CodeCommit or Amazon Keyspaces, you can generate them after you create this IAM user. Learn more

- Choose **Next.**
- In the permissions set up page, choose **Attach policies directly.**
- From the list of **Permissions policies**, select **AdministratorAccess.**
- Choose **Next.**
- Choose **Create user.**
- Choose **Download .csv file.**



Retrieve password

You can view and download the user's password below or email users instructions for signing in to the AWS Management Console. This is the only time you can view and download this password.

Console sign-in details

Console sign-in URL
https://nextwork.signin.aws.amazon.com/console

User name
NextWork-IAM-Admin

Console password
*************** Show

Copy the **Console sign-in URL.**

- Now you're ready to start using your IAM user.
- Log out of your root user's AWS Account.
- Paste and go to your copied console sign-in URL.
- Open your downloaded .csv file containing your user's access instructions.
- Log in using your IAM user's username and password in the .csv file.

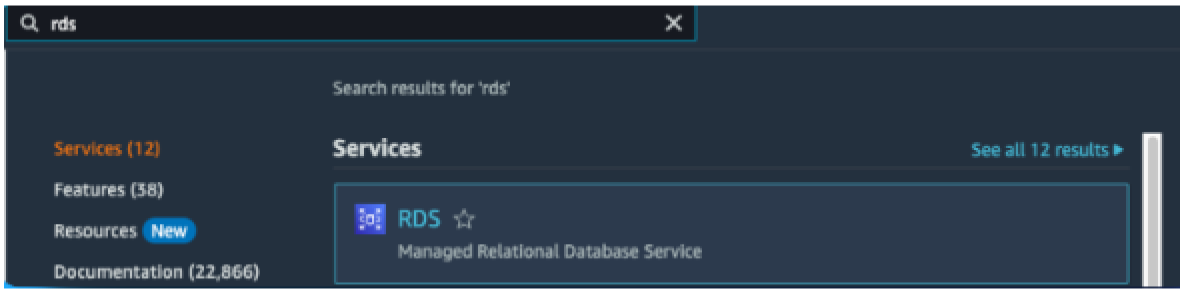
➢ CREATE AN AURORA MYSQL DATABASE

Amazon Aurora is an AWS relational database service which is great at handling high performance and high volume of data.

- Create an Aurora relational database from scratch

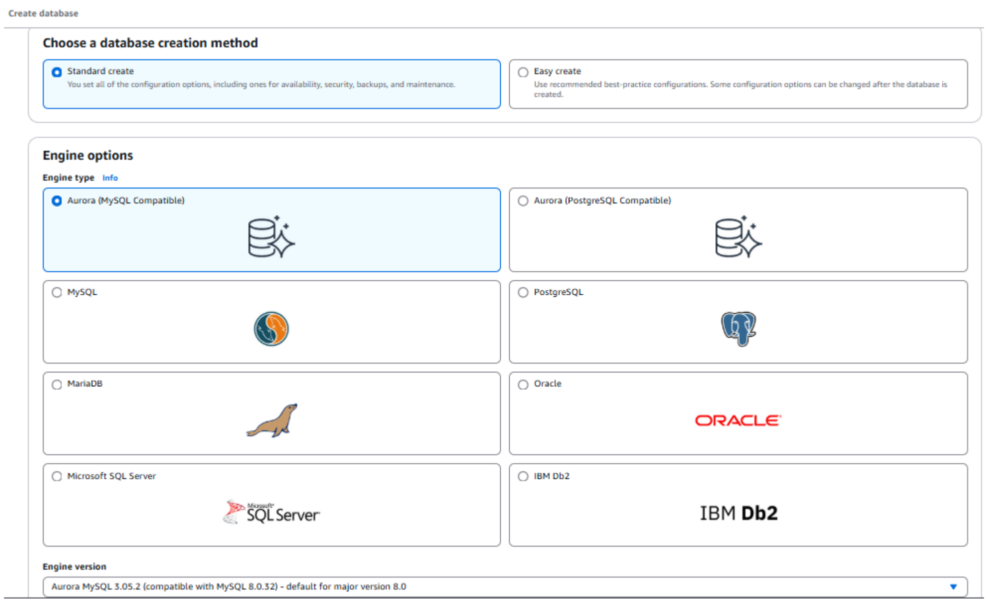
A relational database is a type of database that organizes data into tables, which are collections of rows and columns. Kind of like a spreadsheet. We call it relational because the rows *relate* to the columns and vice-versa.
When a database is relational we can query it using a special language called **SQL** (Structured Query Language)

- Take note of your **Region** in the top right of your AWS Console.
- Head to **RDS console** - search for RDS in search bar at the top of the screen.



- In the left navigation bar, select **Databases**.
- In the Database section, select **Create Database**.
- On the Create database page, choose **Standard Create**.
- In the Configuration section, make the following changes:
    - For **Engine type**, choose **Aurora (MySQL Compatible)**.

Aurora is a good choice when we need something large scale with peak performance and uptime. This is because Aurora databases use clusters.



Create database

Choose a database creation method

(● Standard create) You set all of the configuration options, including ones for availability, security, backups, and maintenance.

○ Easy create — Use recommended best-practice configurations. Some configuration options can be changed after the database is created.

Engine options

Engine type Info

[✓ Aurora (MySQL Compatible)] ○ Aurora (PostgreSQL Compatible) ○ MySQL ○ PostgreSQL ○ MariaDB ○ Oracle ○ Microsoft SQL Server ○ IBM Db2

Engine version

Aurora MySQL 3.05.2 (compatible with MySQL 8.0.32) - default for major version 8.0

- The engine type is like the core software that powers our database. Imagine it as the operating system of a computer, but for databases.
- For **Engine Version**, choose **Aurora MySQL 3.05.2 (compatible with MySQL 8.0.32) - default for major version 8.0**
  The engine version is like choosing the specific version of software that your database will use. For example, **Aurora MySQL 3.05.2 (compatible with MySQL 8.0.32)** is a specific version of Aurora that's compatible with a particular MySQL version (8.0.32)
- For **Templates**, choose **Dev/Test**
  Templates are pre-set settings that help you quickly set up your database environment according to your needs. The Dev/Test template is designed for development and testing environments, helping you pick lower cost options.

In the **Settings** section, set these values:

- **DB cluster identifier**: nextwork-db-cluster
- **Master username**: admin
- For **Credentials management** select **Self managed**.
- **Master password**: Type a password (eg. Technology)
- **Confirm password**: Retype the password.
- Make sure you save your database login details somewhere safe. You'll need them later on.

- Leave the **Cluster storage configuration** settings as default.
- In the **Instance configuration** section, set these values:
  - **Burstable classes (includes t classes)**
  - **db.t3.medium**

  Instance configuration is where we can select the type of virtual computer that our Aurora database will run on. Just like opening up a Word Document runs on your own computer, we need a computer for our Aurora database to run on too.

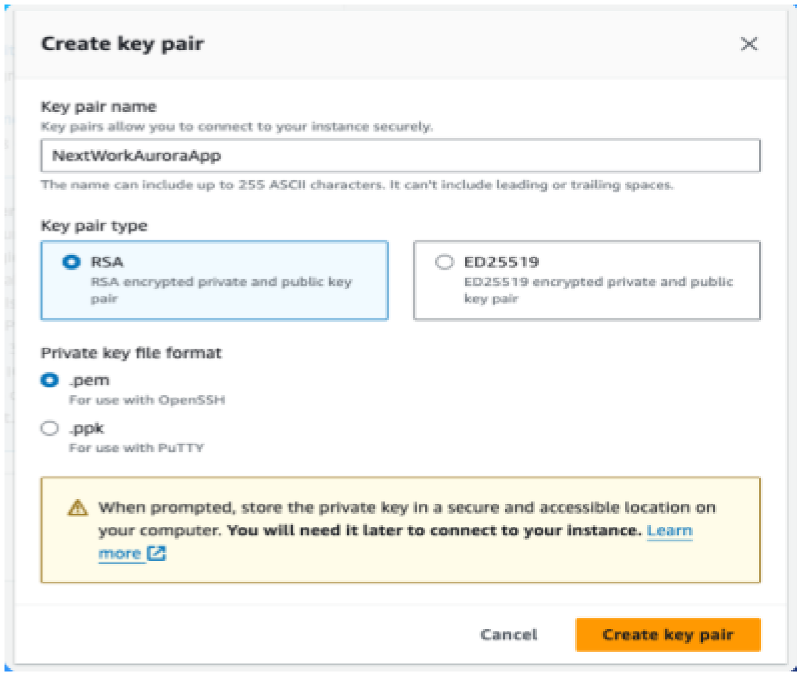
  The **instance type** that we chose determines how powerful, and how much memory, our virtual computer has. The more powerful it is, the better it will perform - but also the more expensive the price.
  The **Burstable classes (includes t classes)** are cost-effective types of database instances that are best when you have a consistent baseline level of traffic with occasional, random spikes in demand...a sudden "burst" of traffic. By choosing **burstable classes**, our database can perform well under high traffic, but also save us costs when things are quiet.
- In the **Availability and durability** section, use the default values.
- In the **Connectivity** section, the first thing it asks us is whether we need to connect to an EC2 instance
  We're trying to connect a **web app server** to our Aurora database. That's our cue to set up an EC2 instance to be that web app server.
  Any web app needs to run on a computer. But we are not going to go buy a physical computer. We're going to use a virtual computer that we rent through AWS. This is exactly what EC2 is here for. Think of an EC2 instance as a rented virtual computer that you use to run your web app. It's the place where all the processing, data handling, and user interactions happen.
- For **Compute resource**, choose **Connect to an EC2 compute resource**.
- Now select the drop down for **EC2 instance**
  We haven't even created an EC2 instance. Let's do that first, then come back to this.

➢ LAUNCH AN EC2 INSTANCE

We're going to create an Amazon EC2 instance using our default VPC and default subnets.

- Open a new tab in your web browser and go to your **AWS console** (this means we can keep our Aurora database set-up in progress!)
- In the upper-right corner of the AWS Management Console, make sure your AWS Region is the same as that in your Aurora database creation.
- In the AWS console search bar, search for **EC2**
- Select **Instances** in the left hand menu and choose **Launch instances**
- Choose the following settings in the **Launch an instance** page.
  - Under **Name and tags**, for **Name**, enter nextwork-ec2-instance-web-server

    - Under **Application and OS Images (Amazon Machine Image)**, choose **Amazon Linux**.
    - Choose the **Amazon Linux 2023 AMI**.
    - Keep the defaults for the other choices.
- Under **Instance type**, choose **t2.micro**.
- Under **Key pair (login)**, choose a **Create new key pair**
- A **key pair** in EC2 is quite literally the keys to access our EC2 instance.
  We need keys to our EC2 instance if we want to add, change, or update how our EC2 instance is running.
- For your **Key pair name**, enter NextWorkAuroraApp
- Leave your **Key pair type** as **RSA**
- Leave your **Private key file format** as **.pem** since we're using SSH later on to access our EC2 instance.
- Select **Create key pair**
  A new file will automatically download to your computer. This is your new **key pair**. We'll need it for later in the project to access our EC2 instance. Notice that it's a **.pem** file



Create key pair

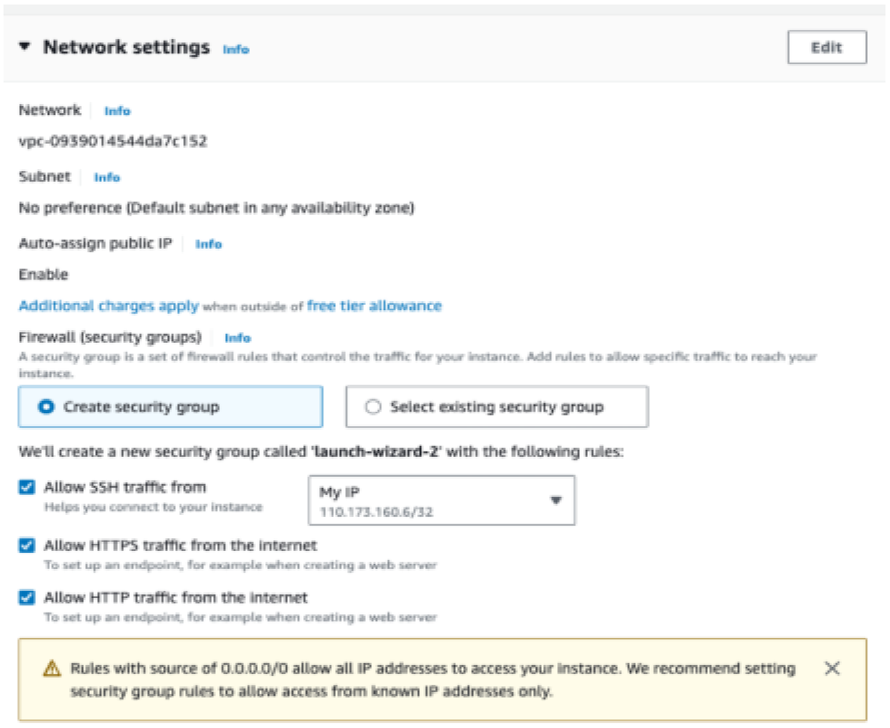
Key pair name
Key pairs allow you to connect to your instance securely.
NextWorkAuroraApp
The name can include up to 255 ASCII characters. It can't include leading or trailing spaces.

Key pair type
[✓ RSA] RSA encrypted private and public key pair
ED25519 ED25519 encrypted private and public key pair

Private key file format
[✓ .pem] For use with OpenSSH
.ppk For use with PuTTY

When prompted, store the private key in a secure and accessible location on your computer. **You will need it later to connect to your instance.** Learn more

- Back in our EC2 creation, under **Network settings**, set these values and keep the other values as their defaults:
    - For **Allow SSH traffic from**, choose your IP address if it's correct (you can check your IP by clicking here). Otherwise select **Anywhere**.
- Check the boxes for **Allow HTTP traffic from the internet**.



▼ Network settings Info

Edit

Network Info

vpc-0939014544da7c152

Subnet Info

No preference (Default subnet in any availability zone)

Auto-assign public IP Info

Enable

Additional charges apply when outside of free tier allowance

Firewall (security groups) Info

A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

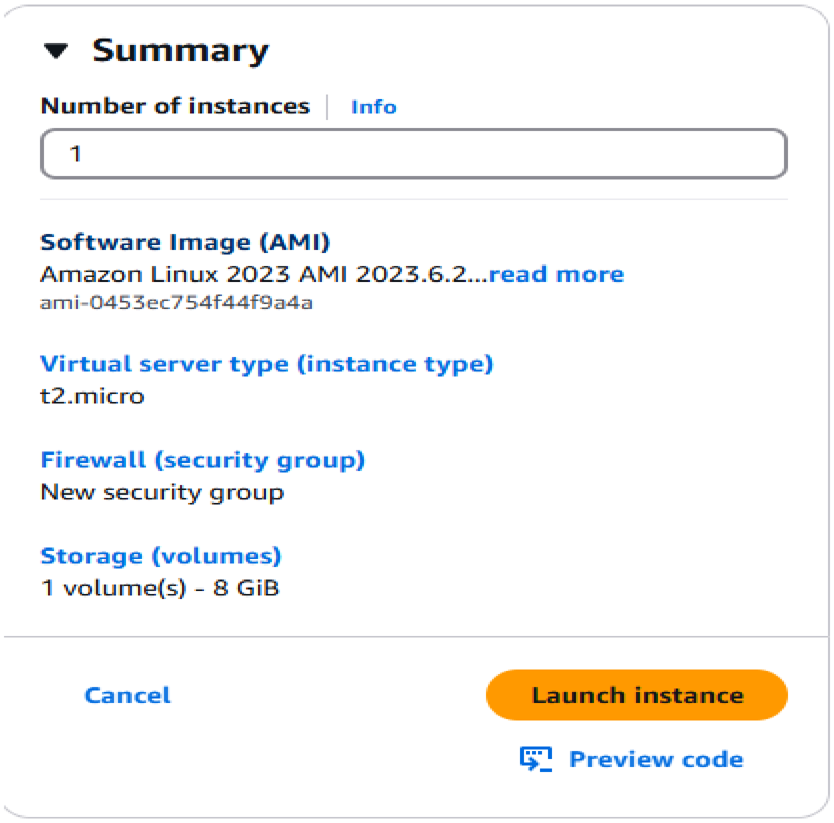
[✓ Create security group] Select existing security group

We'll create a new security group called '**launch-wizard-2**' with the following rules:

- ✓ Allow SSH traffic from — My IP 110.173.160.6/32
  Helps you connect to your instance
- ✓ Allow HTTPS traffic from the internet
  To set up an endpoint, for example when creating a web server
- ✓ Allow HTTP traffic from the internet
  To set up an endpoint, for example when creating a web server

⚠ Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

- Leave the default values for the remaining sections.
- Review the **Summary panel** of your instance configuration



▼ Summary

Number of instances Info

1

Software Image (AMI)
Amazon Linux 2023 AMI 2023.6.2...read more
ami-0453ec754f44f9a4a

Virtual server type (instance type)
t2.micro

Firewall (security group)
New security group

Storage (volumes)
1 volume(s) - 8 GiB

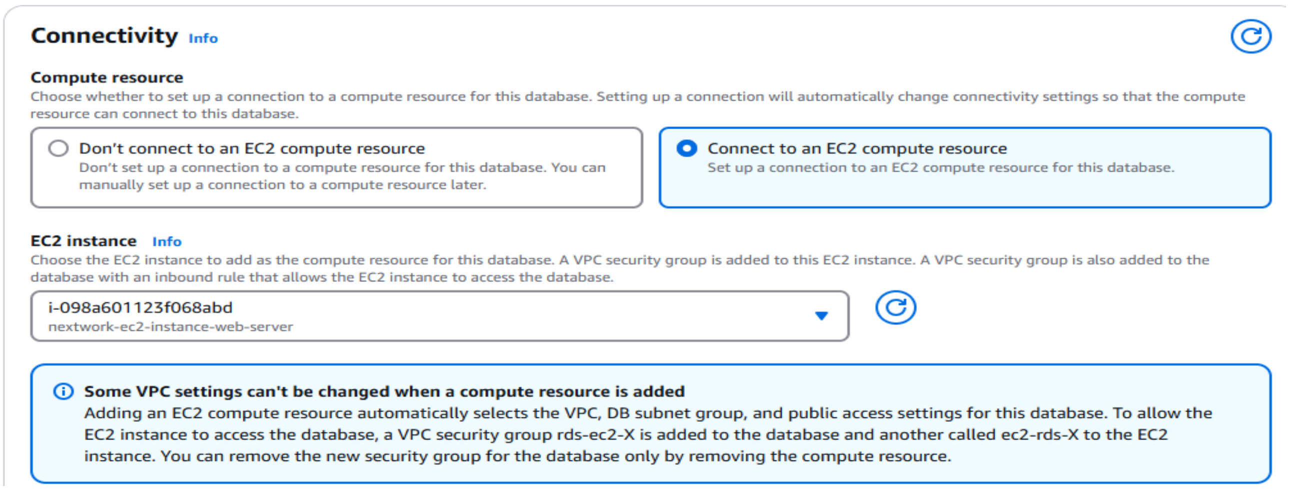
What does this Summary panel tell us?
Number of instances - the number of machines you're running.
Software Image (AMI) - the operating system of your virtual machine.
Virtual server type - the CPU, memory, and network performance package you selected.
Firewall - firewalls control the inbound and outbound traffic for your instances. In EC2, we use security groups for this.
Storage - the storage options for your instance.

- When you're ready, choose **Launch instance**.
- Navigate back to your list of EC2 instances, and then select the checkbox next to your new instance.
- In the **Details** tab, note the following important details:
  - In **Instance summary**, note the **Public IPv4 DNS**.
  - Note the value for **Key pair name**.

  Think about the **Public IPv4 DNS** as the location of a house, and then the **Key pair name** like the keys that let you in. It's no good just having the location without any keys, and you can't have keys without knowing where to go. Both are critical for accessing our EC2 instance.
- Wait until **Instance state** for your instance is **Running** before continuing.

➢ FINISH CREATING YOUR AURORA DATABASE

- Navigate back to your open tab that you were creating our Aurora database in.
- We were up to the **Connectivity** section. But this time we have one big advantage, we have an EC2 instance
  - For **Compute resource**, choose **Connect to an EC2 compute resource**.
  - Now select the drop down for **EC2 instance**, and choose **nextwork-ec2-instance-web-server**.
  - **NOTE:** You may need to select the refresh button to the right of the EC2 instance drop-down.


Connectivity Info

Compute resource
Choose whether to set up a connection to a compute resource for this database. Setting up a connection will automatically change connectivity settings so that the compute resource can connect to this database.

○ Don't connect to an EC2 compute resource
Don't set up a connection to a compute resource for this database. You can manually set up a connection to a compute resource later.

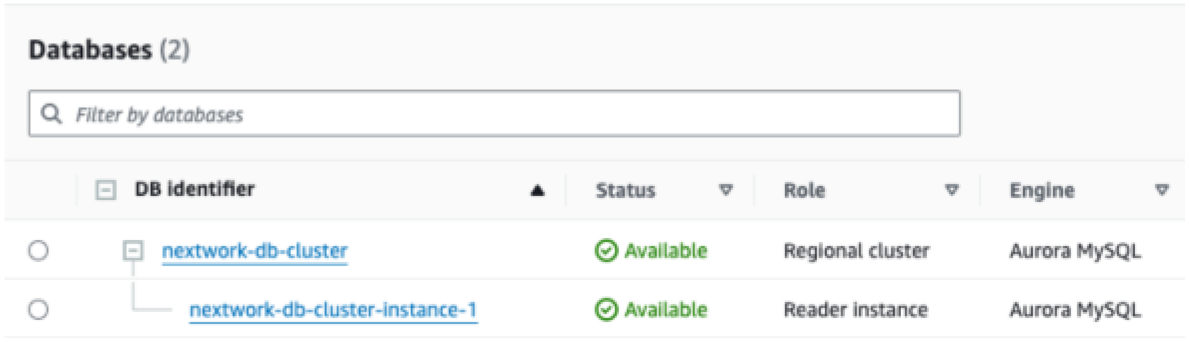
● Connect to an EC2 compute resource
Set up a connection to an EC2 compute resource for this database.

EC2 instance Info
Choose the EC2 instance to add as the compute resource for this database. A VPC security group is added to this EC2 instance. A VPC security group is also added to the database with an inbound rule that allows the EC2 instance to access the database.

i-098a601123f068abd
nextwork-ec2-instance-web-server

Some VPC settings can't be changed when a compute resource is added
Adding an EC2 compute resource automatically selects the VPC, DB subnet group, and public access settings for this database. To allow the EC2 instance to access the database, a VPC security group rds-ec2-X is added to the database and another called ec2-rds-X to the EC2 instance. You can remove the new security group for the database only by removing the compute resource.

- Scroll down and open the **Additional configuration** section.
- Enter sample for **Initial database name**.
- Keep the default settings for the other options.
- Select **Create database**.
- Close any pop-ups that appear.
- Your new DB cluster will show in the **Databases** list with the status **Creating**



Databases (2)

Filter by databases

DB identifier	Status	Role	Engine
nextwork-db-cluster	✓ Available	Regional cluster	Aurora MySQL
nextwork-db-cluster-instance-1	✓ Available	Reader instance	Aurora MySQL

A database cluster in Aurora is a group of database copies that work together so your data is always available.
Each cluster consists of a primary instance (where all write operations occur) and multiple read replicas as back-ups. If your database's primary instance fails, one of the replicas can be promoted to primary automatically.

- Wait for the **Status** of your new DB cluster to show as **Available**.
- Select the **DB identifier** of your top database to take a look at the details.
- Notice that there are **two** Endpoints in our Database. This is our **cluster** in action.

Our **writer** database instance is our primary instance that handles all the "write" operations like INSERT, UPDATE, and DELETE.
Our **reader** database instance is our backup instance that can do very basic operations like SELECT. This is used to get data, but not to add or change data.
Endpoints are like contact points where data flows in and out. A super popular example is a website URL! When you enter a website's address , your browser uses that endpoint to receive data and load the page. For databases, an endpoint is how your app finds a database to ask for data or update it.

➢ CREATE YOUR WEB APP

❖ Open your local terminal

On a **Windows/Linux**:

- Press **Windows + R** to open the Run dialog.
- Type cmd or powershell and press **Enter**.

❖ Connect to your EC2 Instance

- You need to access your **.pem** file in order to login successfully to your EC2 instance - remember, the **.pem** file is like your keys to your EC2 instance!

- Find your **.pem** file on your local computer (it's probably in your downloads folder) and put it in a new folder on your **Desktop** labelled nextwork
- Now we need to navigate to that folder from your terminal, so we can use it.
- Run the command **ls** in your terminal. This shows you all the folders that you can see from your current terminal position.
- If you can see the **desktop** folder when you run ls then you're in the right place.
- If you can't see the **desktop** folder, then use the following commands to navigate up and down your folders until you can see **desktop**...
- To go back one folder run the command **cd ../.**
- To go into a folder, run the command **cd folder-name** and replace **folder-name** with the name of the folder you'd like to enter.
- For example, to navigate to my **desktop** folder, I took the following steps:
- Once you can see your **Desktop** folder, navigate into that folder by running cd Desktop.
- Then navigate into your **nextwork** folder by running cd nextwork.
- Run **ls** to make sure your **.pem** file is there
- Back in your AWS console, navigate to the details page of your **nextwork-ec2-instance-web-server** EC2 instance.
- Copy your **Public IPv4 DNS** address.
- Back in your terminal run the following command:
  “**ssh -i YOUR_PEM_FILE_NAME ec2-user@YOUR_EC2_ADDRESS**”

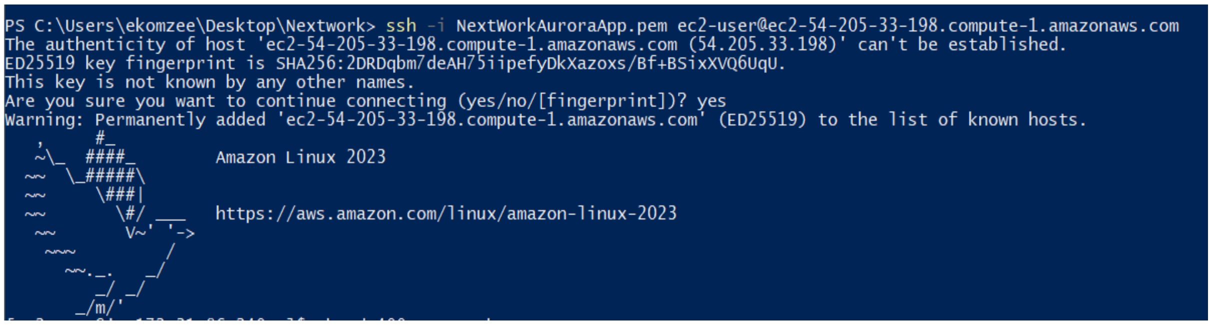
Make sure you replace '**YOUR_PEM_FILE_NAME**' with the name of your .pem file, and '**YOUR_EC2_ADDRESS**' with the copied EC2 address.

- Did you get an error telling you permissions denied?
- That's because you need the right permissions to access your **.pem** file.
- Try running this command if you're on **Mac** or **Linux**:
  “**chmod 400 YOUR_PEM_FILE_NAME**”

Make sure you replace '**YOUR_PEM_FILE_NAME**' with the name of your .pem file

1. chmod stands for **Change Mode**. It’s a command used to change the permissions of a file or directory in Linux systems.
2. The 400 has a lot of meaning in it
   4 **means** "read-only" permission. The owner of the file can read the file but cannot write to it or execute it.
   0 **means** "no permission." Neither the group nor others have any permissions (cannot read, write, or execute the file).
   The order of these numbers also has meaning**.** The first digit represents the permission for the file owner, the second and third digits (0 and 0) represent the permissions for the group and others (everyone else).
   So, 400 means the owner of the file can read it, but no one else (including the group or other users) can access it.

- Once you've given your computer access to your **.pem** file, run the command to connect to our EC2 instance again:
  “**ssh -i YOUR_PEM_FILE_NAME ec2-user@YOUR_EC2_ADDRESS**”



```
PS C:\Users\ekomzee\Desktop\Nextwork> ssh -i NextWorkAuroraApp.pem ec2-user@ec2-54-205-33-198.compute-1.amazonaws.com
The authenticity of host 'ec2-54-205-33-198.compute-1.amazonaws.com (54.205.33.198)' can't be established.
ED25519 key fingerprint is SHA256:2DRDqbm7deAH75iipefyDkXazoxs/Bf+BSixXVQ6UqU.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-54-205-33-198.compute-1.amazonaws.com' (ED25519) to the list of known hosts.
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
```

- Now remember that our EC2 instance is just another computer. First thing we need to do is make sure all the software on it is up to date.
- Run the following command:

"**sudo dnf update –y**"

sudo means 'superuser do' and is like saying you have admin or root user rights.
dnf is what is used to manage all the software on our EC2 instance.
update tells 'dnf' to update all the software on the instance.
-y means automatically answer yes to any prompts that appear during the update process.
After the updates complete, we're going to install a whole bunch of stuff needed to run your web app.

- an **Apache web server** - the most widely used web server in the world. A web server is a software that gets your content (e.g. web pages) to users via the internet.
- **PHP** - a programming language used for writing beautiful app pages.
- **php-mysqli** - a PHP library (i.e. a collection of pre-written code to help you save time) for establishing a MySQL connection to your database.
- **MariaDB** - a relational database management system. Your Aurora database knows how to manage its data (e.g. how to add a new row or retrieve data), but your web server doesn't know how to send instructions to and understand the responses from your Aurora database. Your web server would need MySQL-compatible client libraries i.e. software designed for servers to communicate with a MySQL database. Installing MariaDB in your EC2 instance is one of the many ways to install these client libraries so it can interact with Aurora.
- Run the following command:

"**sudo dnf install -y httpd php php-mysqli mariadb105**"

sudo means 'superuser do' and is like saying you have admin or root user rights.
dnf is what is used to manage all the software on our EC2 instance.
install tells 'dnf' that you want to install software on the instance.
-y means automatically answer yes to any prompts that appear during the update process.
httpd: installs the Apache HTTP Server package.
php: installs PHP.
php-mysqli: installs MySQL extension for PHP. **mariadb105**: installs MariaDB, a version of the MySQL database management system.

Let's start the most basic version of our web app with the following command:
"**sudo systemctl start httpd**"

Test that your web app is running by entering your **IPv4 DNS** address into your browser.

- E.g. http://ec2-42-8-168-21.us-west-1.compute.amazonaws.com - just make sure it's actually your one
- Make sure you change **https** to **http** - no 's'!
- Now we have our app set up and we can access it via our terminal, we can connect it to our Aurora database.

➢ CONNECT THE WEB APP TO A DATABASE

❖ Connect your EC2 Instance to Database

While still connected to your EC2 instance, navigate to the **www** folder (this is where we store all our files for our web app!)
"**cd /var/www**"

❖ Create a new sub-folder named **inc**.

"**mkdir inc**"

- Did you get another **permission denied** error?
- Let's find out what's going on here. Who actually has permissions in this folder? Run the following command to find out:

"**ls –ld**"
The command **ls** means list, and **-ld** means "list the details"
When you access an EC2 instance using a key pair, you're by default logging in as an admin user (called ec2-user) which is **not** the root user!
Let's change the owner of this folder so you (ec2-user) can edit it.

- Let's navigate back to where we were at the start:

"**cd ../../**"
Run the following command:
"**sudo chown ec2-user:ec2-user /var/www**"
This changes the owner of the folder to the **ec2-user**....us!

chown means "change the owner permissions".
ec2-user:ec2-user means we change the individual and group user from root to ec2-user.

- Now let's see the details of that folder again...

"**ls -ld var/www**"
Let's try it again.

- Navigate to your **www** folder

"**cd /var/www**"

- Create a new sub-folder named **inc**.

"**mkdir inc**"

- No errors! Navigate into your new **inc** folder

"**cd inc**"

- Create a new file in the **inc** directory named **dbinfo.inc**

“>dbinfo.inc”

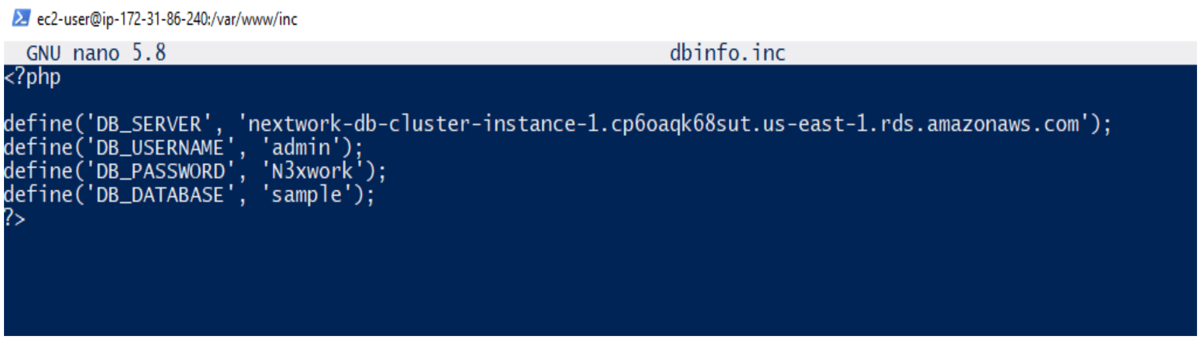
- Now we have a blank file. We can edit our new file by calling **nano**.
  “**nano dbinfo.inc**”

Nano is a way we can edit files in the terminal. It looks a bit different, but it's nothing to be scared of!

We've just opened our blank **dbinfo.inc** file, and now we're going to add some code that connects it to our AWS database.

- **dbinfo.inc** is a settings file that stores the connection details our EC2 instance will need to connect to our Aurora database. To complete this file, we need our Aurora database **endpoint**.
- Navigate to your Aurora database details in AWS and copy the **Endpoint** of your **Writer** instance.
- Copy and paste the following code to connect our EC2 instance to our Aurora database to the **dbinfo.inc** file.
- Make sure you replace **YOUR_ENDPOINT** with your actual Aurora Endpoint!
- If you've set up your user with a password other than n3xtw0rk, make sure to update DB_PASSWORD too.

```
“<?php



define('DB_SERVER', 'YOUR_ENDPOINT');

define('DB_USERNAME', 'admin');

define('DB_PASSWORD', 'n3xtw0rk');

define('DB_DATABASE', 'sample');

?>”
```



```
ec2-user@ip-172-31-86-240:/var/www/inc
  GNU nano 5.8                                    dbinfo.inc
<?php

define('DB_SERVER', 'nextwork-db-cluster-instance-1.cp6oaqk68sut.us-east-1.rds.amazonaws.com');
define('DB_USERNAME', 'admin');
define('DB_PASSWORD', 'N3xwork');
define('DB_DATABASE', 'sample');
?>
```

Save and close the **dbinfo.inc** file by using **Ctrl+S** to save and then **Ctrl+X** to exit.

Now that our database is all connected

- **UPGRADE YOUR WEB APP**
  - Navigate to your **html** folder:

“**cd /var/www/html**”

  - Create a new file in the **html** directory named **SamplePage.php**, and then edit the **file by calling nano**

“>SamplePage.php
nano SamplePage.php”

  - Copy and paste the following script into your SamplePage.php file:

```
<?php include "../inc/dbinfo.inc"; ?>
<html>
<body>
<h1>Sample page</h1>
<?php

  /* Connect to MySQL and select the database. */
  $connection = mysqli_connect(DB_SERVER, DB_USERNAME, DB_PASSWORD);

  if (mysqli_connect_errno()) echo "Failed to connect to MySQL: " . mysqli_connect_error();

  $database = mysqli_select_db($connection, DB_DATABASE);

  /* Ensure that the EMPLOYEES table exists. */
  VerifyEmployeesTable($connection, DB_DATABASE);

  /* If input fields are populated, add a row to the EMPLOYEES table. */
  $employee_name = htmlentities($_POST['NAME']);
  $employee_address = htmlentities($_POST['ADDRESS']);

  if (strlen($employee_name) || strlen($employee_address)) {
    AddEmployee($connection, $employee_name, $employee_address);
  }
?>

<!-- Input form -->
<form action="<?PHP echo $_SERVER['SCRIPT_NAME'] ?>" method="POST">
  <table border="0">
    <tr>
      <td>NAME</td>
      <td>ADDRESS</td>
    </tr>
    <tr>
      <td>
        <input type="text" name="NAME" maxlength="45" size="30" />
      </td>
      <td>
        <input type="text" name="ADDRESS" maxlength="90" size="60" />
      </td>
      <td>
        <input type="submit" value="Add Data" />
      </td>
    </tr>
  </table>
</form>

<!-- Display table data. -->
<table border="1" cellpadding="2" cellspacing="2">
  <tr>
    <td>ID</td>
    <td>NAME</td>
    <td>ADDRESS</td>
  </tr>

<?php

$result = mysqli_query($connection, "SELECT * FROM EMPLOYEES");

while($query_data = mysqli_fetch_row($result)) {
  echo "<tr>";
  echo "<td>",$query_data[0], "</td>",
       "<td>",$query_data[1], "</td>",
       "<td>",$query_data[2], "</td>";
  echo "</tr>";
}
?>

</table>

<!-- Clean up. -->
<?php

  mysqli_free_result($result);
  mysqli_close($connection);

?>
```

```
</body>
</html>


<?php

/* Add an employee to the table. */
function AddEmployee($connection, $name, $address) {
   $n = mysqli_real_escape_string($connection, $name);
   $a = mysqli_real_escape_string($connection, $address);

   $query = "INSERT INTO EMPLOYEES (NAME, ADDRESS) VALUES ('$n', '$a');";

   if(!mysqli_query($connection, $query)) echo("<p>Error adding employee data.</p>");
}

/* Check whether the table exists and, if not, create it. */
function VerifyEmployeesTable($connection, $dbName) {
  if(!TableExists("EMPLOYEES", $connection, $dbName))
  {
     $query = "CREATE TABLE EMPLOYEES (
         ID int(11) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
         NAME VARCHAR(45),
         ADDRESS VARCHAR(90)
       )";

     if(!mysqli_query($connection, $query)) echo("<p>Error creating table.</p>");
  }
}

/* Check for the existence of a table. */
function TableExists($tableName, $connection, $dbName) {
  $t = mysqli_real_escape_string($connection, $tableName);
  $d = mysqli_real_escape_string($connection, $dbName);

  $checktable = mysqli_query($connection,
      "SELECT TABLE_NAME FROM information_schema.TABLES WHERE TABLE_NAME = '$t'
AND TABLE_SCHEMA = '$d'");

  if(mysqli_num_rows($checktable) > 0) return true;

  return false;
}
```

?>

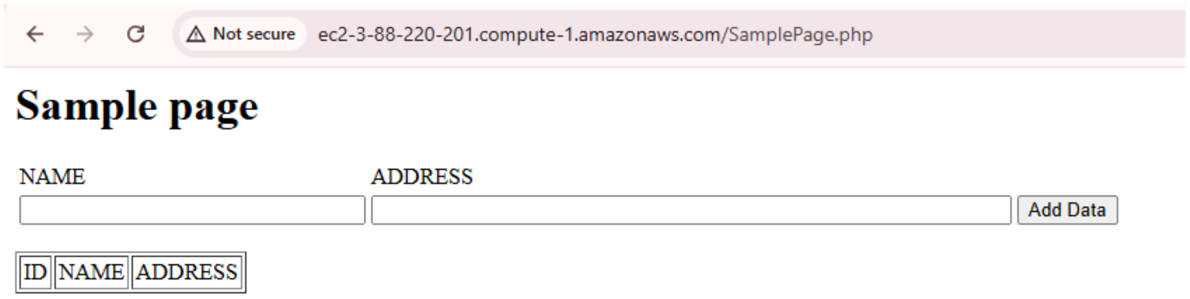
This big block of code is what makes our new web app look so cool! It's pulling in the details from our **dbinfo.inc** file we created earlier and using it to display up-to-date changes directly from our website.

- ❖ Save and close the **SamplePage.php** file by using **Ctrl+S** to save and then **Ctrl+X** to exit.

If you get another Permission Denied error, you know what to do!

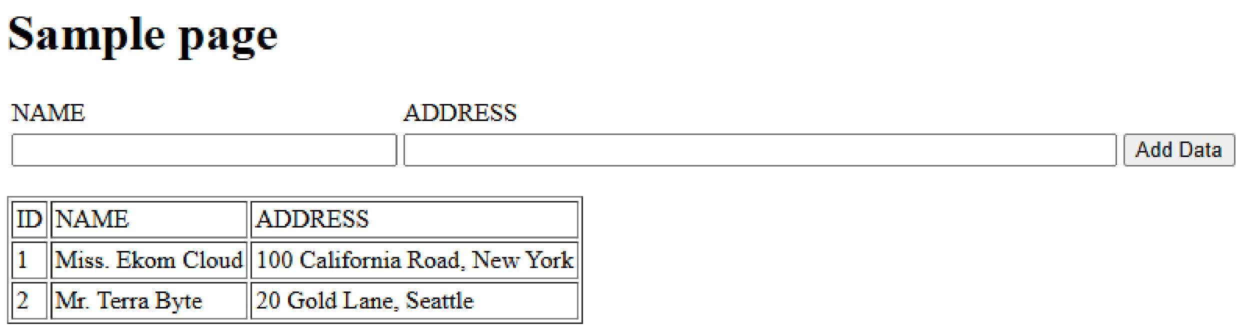
1. Exit nano by choosing **Ctrl+X**.
2. Run ls -ld to check who is the owner of this folder
3. Update the owner to you by running sudo chown ec2-user:ec2-user .
4. Try pasting and saving your **SamplePage.php** file again.

- ❖ Verify that your web server successfully connects to your DB cluster by opening a web browser and browsing to http://EC2-instance-endpoint/SamplePage.php



ID	NAME	ADDRESS

- ➢ **TEST YOUR NEW APP TO SEE IF IT WORKED**
- In your browser, where your new web app is running, add some new data.
- Notice how your results show on the page? If you're wondering how that happens, go back and have a look at your **SamplePage.php** file.



ID	NAME	ADDRESS
1	Miss. Ekom Cloud	100 California Road, New York
2	Mr. Terra Byte	20 Gold Lane, Seattle

- ❖ Connect to your Database using MySQL CLI

- To access your database, you're going to use **MySQL**.

MySQL CLI is a powerful, secure, and efficient tool for interacting with MySQL databases. It provides direct access to all MySQL features, supports automation, and is a preferred method for managing databases on remote servers, such as EC2 instances.

- ❖ Download the MySQL repository into your EC2 instance:

"**sudo yum install https://dev.mysql.com/get/mysql80-community-release-el7-3.noarch.rpm -y**"

- ❖ Install MySQL

"**sudo yum install mysql-community-client –y**"

- ❖ Connect to your Aurora MySQL Database:

"**mysql -h YOUR_ENDPOINT -P YOUR_PORT -u YOUR_AURORA_USERNAME –p**"

- Make sure you replace:
  - **YOUR_ENDPOINT** with the Endpoint from your Aurora Writer instance
  - **YOUR_PORT** with the Port from your Aurora Writer instance; 3306
  - **YOUR_AURORA_USERNAME** with your Aurora username; admin
- Enter in your Aurora password when prompted; Technology

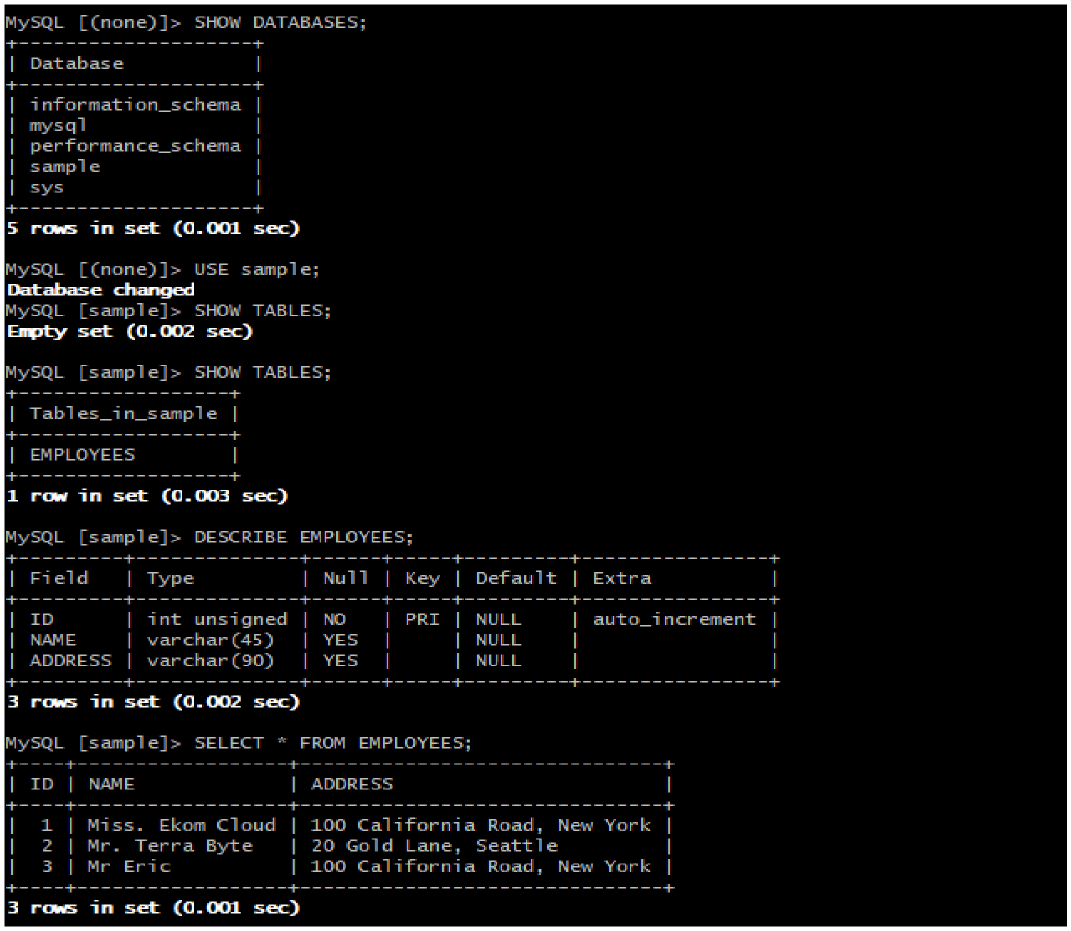
- Run **SHOW DATABASES**;
  - This shows you the databases that are in your MYSQL server.

- Run **USE sample**;
  - **sample** is the name you gave your first database schema.

- Run **SHOW TABLES**;
  - This shows you the tables in your **sample** schema.
  - If you're wondering where **Employees** came from, check your **SamplePage.php** file. It's all there! (Hint: this is also where you can change it if you're feeling creative)

Run **DESCRIBE EMPLOYEES**;
- This tells you how the **Employees** table is structured

Run **SELECT * FROM EMPLOYEES**;
- This shows you the actual data in your table!



```
MySQL [(none)]> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sample             |
| sys                |
+--------------------+
5 rows in set (0.001 sec)

MySQL [(none)]> USE sample;
Database changed
MySQL [sample]> SHOW TABLES;
Empty set (0.002 sec)

MySQL [sample]> SHOW TABLES;
+------------------+
| Tables_in_sample |
+------------------+
| EMPLOYEES        |
+------------------+
1 row in set (0.003 sec)

MySQL [sample]> DESCRIBE EMPLOYEES;
+---------+--------------+------+-----+---------+----------------+
| Field   | Type         | Null | Key | Default | Extra          |
+---------+--------------+------+-----+---------+----------------+
| ID      | int unsigned | NO   | PRI | NULL    | auto_increment |
| NAME    | varchar(45)  | YES  |     | NULL    |                |
| ADDRESS | varchar(90)  | YES  |     | NULL    |                |
+---------+--------------+------+-----+---------+----------------+
3 rows in set (0.002 sec)

MySQL [sample]> SELECT * FROM EMPLOYEES;
+----+------------------+----------------------------+
| ID | NAME             | ADDRESS                    |
+----+------------------+----------------------------+
|  1 | Miss. Ekom Cloud | 100 California Road, New York |
|  2 | Mr. Terra Byte   | 20 Gold Lane, Seattle      |
|  3 | Mr Eric          | 100 California Road, New York |
+----+------------------+----------------------------+
3 rows in set (0.001 sec)
```

➢ DELETE YOUR RESOURCES

- Delete your database cluster: Open the Amazon RDS console, select **Databases**, and choose nextwork-db-cluster. Then, from the Actions drop-down menu, choose **Delete**.
  - When deleting your database cluster, make sure to *uncheck* **Create final snapshot** and *uncheck* **Retain automated backups** so you don't get charged.
  - Delete your EC2 instance: Open the Amazon EC2 console, select **Instances**, and choose the instance you created. Click **Actions** at the top, then **Terminate instance**.
  - Delete your key pair: Open the Amazon EC2 console, select **Key Pairs**, and choose the key pair you created. Click **Actions** at the top, then **Delete Key Pair**.