

Create PHP application on EC2 instance with Amazon RDS/MySQL as backend.

Objectives:

1. Learn to configure RDS with MySQL engine.
2. Learn to create a PHP application on EC2 instance with Amazon RDS/MySQL as backend.

Step 1: In **EC2** service console, go to **Security groups** in side panel. Click on **Create security groups**.



Create a Security Group for Linux Server with following configuration:

Click on **Create security group**.

Configure it as follows:

Security group name: **MyWebServerSG**

Description: **Security Group for EC2 Webserver in custom VPC**

Add three rules under **Inbound Rules**:

1. **Type:** **HTTP**
Source: **0.0.0.0/0**
2. **Type:** **HTTP**
Source: **::/0**
3. **Type:** **SSH**
Source: **0.0.0.0/0**

Inbound rules			
Type	Protocol	Port range	Source
HTTP	TCP	80	0.0.0.0/0
HTTP	TCP	80	::/0
SSH	TCP	22	0.0.0.0/0

Click on **Create security group** button in bottom right corner. Confirm that it is created.

Cloud Plus Plus Services



Go back to **EC2** service console, go to **Security groups** in side panel. Click on **Create security groups**.

Provide **Security group name** as **RDS-SG**.

Provide **Description** as **Security Group for Database**.

Provide following Inbound rules:

1. **Type:** **SSH**
Source: **0.0.0.0/0**
2. **Type:** **MySQL/Aurora**
Source: **0.0.0.0/0**
3. **Type:** **MySQL/Aurora**
Source: **MyWebServerSG** (security group that would be used to create the EC2 instance)

Inbound rules			
Type	Protocol	Port range	Source
SSH	TCP	22	0.0.0.0/0
MYSQL/Aurora	TCP	3306	0.0.0.0/0
MYSQL/Aurora	TCP	3306	sg-0ef5565514aef0217 (WebServerSG)

Click on **Create security group** button in bottom right corner. Confirm that it is created.

Step 2: Go to **RDS** service console. Click on **Create database**.

Choose a **Standard create** database creation method.

RDS > Create database

Create database

Choose a database creation method [Info](#)

☒ **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.

Cloud Plus Plus Services



Select the **MySQL** radio button in **Engine options**. Confirm **MySQL 8.0.20**

Engine options

Engine type [Info](#)

☐ Amazon Aurora

☒ MySQL

☐ MariaDB

☐ PostgreSQL

☐ Oracle

☐ Microsoft SQL Server

Edition

☒ MySQL Community

Known Issues/Limitations
Review the [Known Issues/Limitations](#) to learn about potential compatibility issues with specific database versions.

Version

MySQL 8.0.20

Select the **Dev/Test** in **Templates**.

Templates

Choose a sample template to meet your use case.

☐ Production
Use defaults for high availability and fast, consistent performance.

☒ Dev/Test
This instance is intended for development use outside of a production environment.

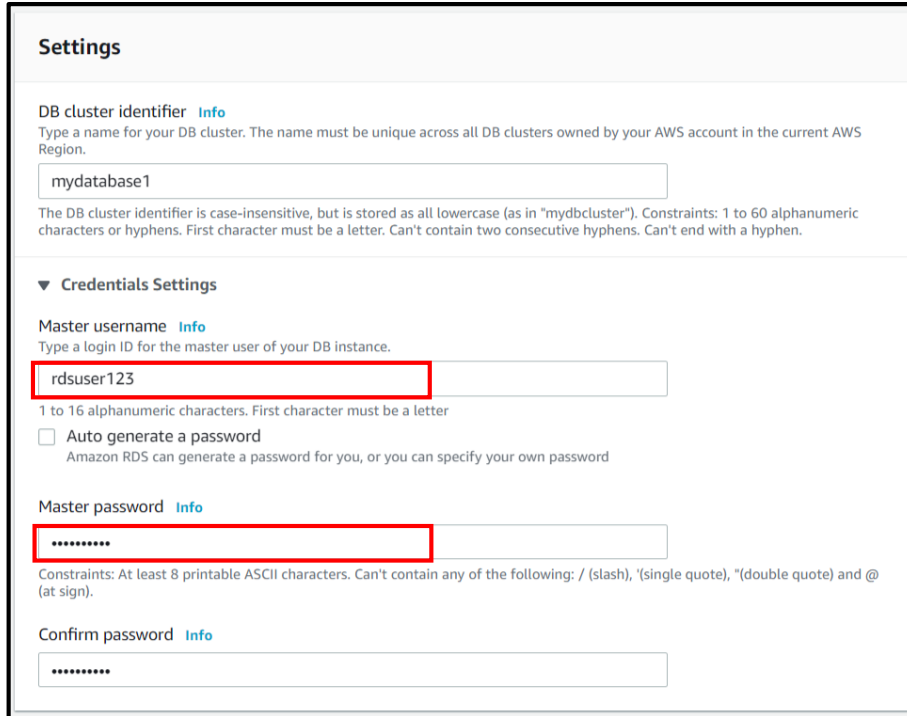
☐ Free tier
Use RDS Free Tier to develop new applications, test existing applications, or gain hands-on experience with Amazon RDS. [Info](#)

Under **Settings**:

Give **DB instance identifier** as **mydatabase1**.

Provide **Credentials Settings** as per your choice and store it in a secure place. We refer to following values for this document:

- **Master Username:** **rdsuser123**
- **Master password:** **rdspass123**



The screenshot shows the 'Settings' section of the AWS RDS 'Create DB Instance' wizard. It includes fields for 'DB cluster identifier' (mydatabase1), 'Master username' (rdsuser123), 'Master password' (masked with dots), and 'Confirm password' (masked with dots). The 'Master username' and 'Master password' fields are highlighted with red boxes. The 'Credentials Settings' section is expanded, showing the 'Master username' and 'Master password' fields. The 'Auto generate a password' checkbox is unchecked.

Let the **DB instance class** be **Standard Classes** and **size** be **db.m5.xlarge**.

Confirm the default **Storage** settings as **Storage type**: **General type- General Purpose (SSD)** and **Allocated storage** **20 GiB**.

Uncheck the **Enable storage autoscaling**.

Under **Availability & durability** select **Do not create a standby instance** radio button.

Connectivity section will have Default VPC selected.

Click on **Additional connectivity configuration** for drop down.

- Select **default-vpc** Subnet group.
- Select **Yes** radio button under **Public access**.
- Select **Choose Existing** radio button in **VPC security group**.
- In the **Existing VPC security groups** default will be selected. Remove this security group by clicking on the **cross** sign. Select the **RDS-SG** created in previous step.
- In **Availability Zone** select **1a** which should be **same as the AZ of Linux Instance**.
- The **Database port** will be default **3306**.

Cloud Plus Plus Services



Public access [Info](#)

☒ **Yes**
Amazon EC2 instances and devices outside the VPC can connect to your database. Choose one or more VPC security groups that specify which EC2 instances and devices inside the VPC can connect to the database.

☐ **No**
RDS will not assign a public IP address to the database. Only Amazon EC2 instances and devices inside the VPC can connect to your database.

VPC security group
Choose a VPC security group to allow access to your database. Ensure that the security group rules allow the appropriate incoming traffic.

☒ **Choose existing**
Choose existing VPC security groups

☐ **Create new**
Create new VPC security group

Existing VPC security groups

Choose VPC security groups

RDS-SG X

Availability Zone [Info](#)

ap-south-1a

Database authentication is set to **Password authentication**.

Click on **Additional configuration** drop down.

Provide **Initial database name** as **cloudPlusPlusLab**.

DB parameter group and **Option group** will be **default .mysql8.0**.

▼ **Additional configuration**
Database options, encryption disabled, backup enabled, backtrack disabled, Performance Insights disabled, Enhanced Monitoring disabled, maintenance, CloudWatch Logs, delete protection disabled

Database options

Initial database name [Info](#)

cloudPlusPlusLab

If you do not specify a database name, Amazon RDS does not create a database.

DB parameter group [Info](#)

default.mysql8.0

Option group [Info](#)

default:mysql-8-0

In **Backup** the **Enable automatic backups** will be checked.

Backup retention period will be **0 days**.

Backup window will have **No preference** selected.

Uncheck **Copy tags to snapshots**.

Uncheck **Enable Encryption**.

Cloud Plus Plus Services



Uncheck Enable Performance Insights.

Uncheck Enable Enhanced monitoring.

Uncheck all options under Log Exports.

Uncheck Enable auto minor version upgrade.

Select No preference for Maintenance window.

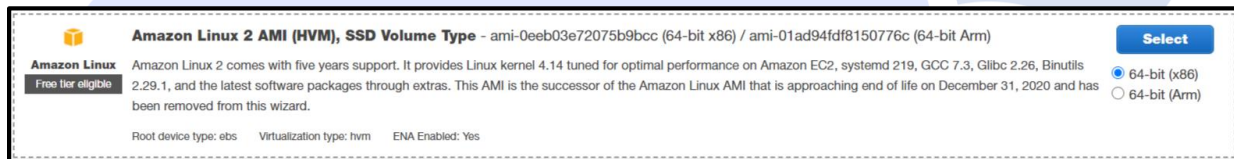
Uncheck Enable deletion protection.

Ensure no additional cost is being incurred in the Estimated monthly cost section.

Click on Create database button in bottom right corner.

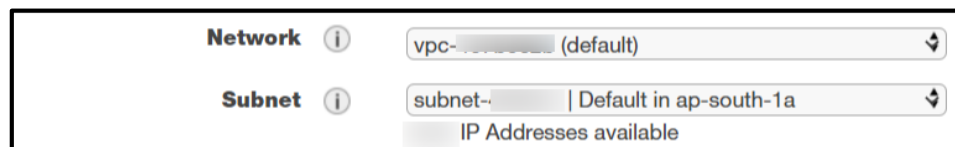
Confirm that the database is created and Available.

Step 3: Go to EC2 in AWS Console. Click on Launch Instance. Select The Linux2 AMI.



In **Step 2: Choose an Instance Type** keep the default **t2.micro** and go to next step.

In **Step 3**, select a subnet same as the one in which RDS is created. In our case we select ap-south-1a.



Keep the defaults for **Step 4: Add Storage**. And go to Step 5. Provide the **Key** as **Name** and **Value** as **LinuxWebServerforRDS**.

In **Step 6** select the **existing** Linux Web Server Security Group **MyWebServerSG** created earlier in this exercise. For more information refer to our blog in [Linux Server Configuration here](#). **Review, acknowledge the Key-pair** and **Launch** the Instance. Make sure it is running.

Step 4: SSH into the instance.

Run the following command. This will get the latest bug fixes and security updates by updating the software on your EC2 instance:

```
sudo yum update
```

Now install the PHP software using following command. This command installs multiple software packages and related dependencies:

```
sudo amazon-linux-extras install -y lamp-mariadb10.2-php7.2 php7.2
```

Now we Install Apache web server:

```
sudo yum install -y httpd
```

Start the Apache web server using following command:

```
sudo systemctl start httpd
```

Configure the web server to start with each system boot using following command:

```
sudo systemctl enable httpd
```

```
[ec2-user@ip-172-31-32-14 ~]$ sudo systemctl enable httpd
Created symlink from /etc/systemd/system/multi-user.target.wants/httpd.service
to /usr/lib/systemd/system/httpd.service.
```

Test your web server by entering the DNS name of your EC2 instance in the address bar of the web browser. You should see the Apache test page.

To allow ec2-user to manage files in the default root directory for your Apache web server, modify the ownership and permissions of the /var/www directory.

We will add a group named www to your EC2 instance. Then we give that group ownership of the /var/www directory and add write permissions for the group. Any members of that group can then add, delete, and modify files for the web server.

Add the www group to your EC2 instance with the following command:

```
sudo groupadd www
```

Add the ec2-user user to the www group:

```
sudo usermod -a -G www ec2-user
```

Log out to refresh your permissions and include the new www group:

Exit

Log back in again and verify that the `www` group exists with the next command. You should see output as below:

`groups`

```
[ec2-user@ip-172-31-32-14 ~]$ groups  
ec2-user adm wheel systemd-journal www
```

Change the group ownership of the `/var/www` directory and its contents to the `www` group:

`sudo chgrp -R www /var/www`

Change the directory permissions of `/var/www` and its subdirectories to add group write permissions and set the group ID on subdirectories created in the future:

`sudo chmod 2775 /var/www`

`find /var/www -type d -exec sudo chmod 2775 {} +`

Recursively change the permissions for files in the `/var/www` directory and its subdirectories to add group write permissions:

`find /var/www -type f -exec sudo chmod 0664 {} +`

Connect your Apache web server to your DB instance. Change the directory to `/var/www` and create a new subdirectory named `inc`:

`cd /var/www`

```
[ec2-user@ip-172-31-32-14 ~]$ cd /var/www  
[ec2-user@ip-172-31-32-14 www]$
```

Create a directory **inc** and enter the directory. Inside the directory we create a file `dbinfo.inc`:

`mkdir inc`

`cd inc`

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

`>dbinfo.inc`

Go into the nano editor and add the following code into the `dbinfo.inc` file:

`nano dbinfo.inc`

The Blue part in following code is to be replaced by **your database endpoint**, username, password and database name that was created in the above steps.

```
<?php  
define('DB_SERVER', 'my-database-1. .ap-south-1.rds.amazonaws.com');  
define('DB_USERNAME', 'rdsuser123');  
define('DB_PASSWORD', 'rdspass123');  
define('DB_DATABASE', 'cloudPlusPlusLab');  
?>
```

Press **Ctrl+X** to exit, further **Y** to save changes and **Enter** to keep the name unchanged.

Change the directory to html:

```
cd /var/www/html
```

Here create a SamplePage.php document. Go to Nano editor to edit the document:

```
>SamplePage.php
```

```
nano SamplePage.php
```

Add the following code:

```
<?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;
}
?>
```

Press **Ctrl+X** to exit, further **Y** to save changes and **Enter** to keep the name unchanged.

Step 5: Verify that your web server successfully connects to your DB instance by opening a web browser and browsing to

EC2 instance endpoint/**SamplePage.php**.

The Blue part will be your DNS name:

ec2-172-31-32-14.ap-south-1.compute.amazonaws.com/**SamplePage.php**

Sample page

NAME

ADDRESS

Add Data

ID

NAME

ADDRESS

Cloud Plus Plus Services



As you enter the data, it will be stored on the created database server.

Sample page

NAME ADDRESS

ID	NAME	ADDRESS
1	A	a123
2	B	B123
3	C	c123

Step 6: Thus we have created a Web Server on EC2 Instance and connected to MySQL database successfully.

Note: Delete RDS instance, Security groups and terminate the instances if you no longer need them.

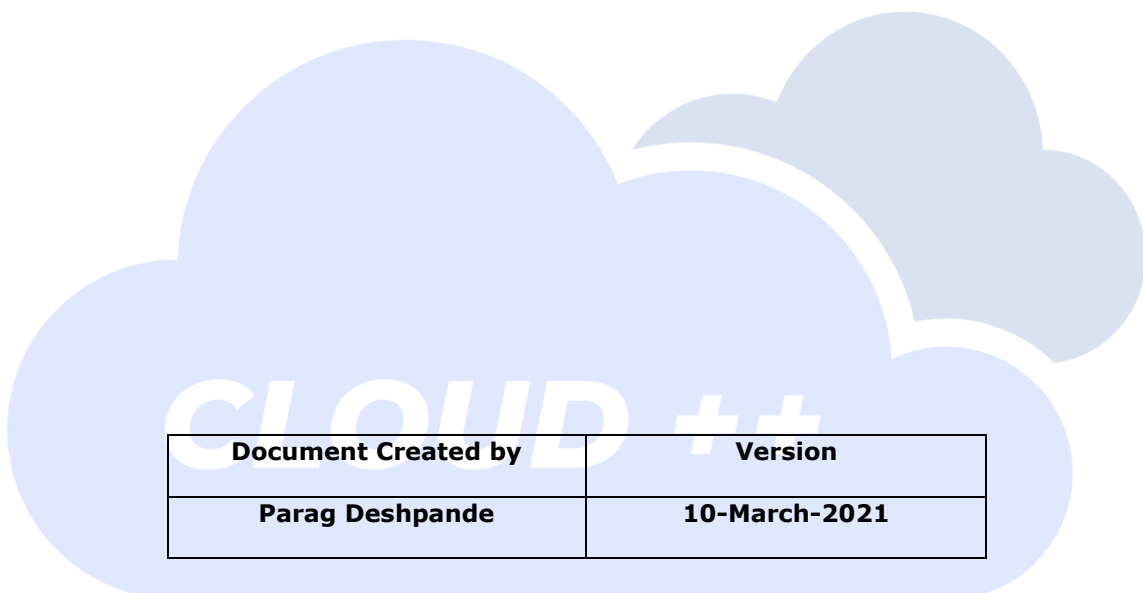
CLOUD ++

Your trusted partner for
cloud enablement

Cloud Plus Plus Services



Was this document helpful? YES / NO



Document Created by	Version
Parag Deshpande	10-March-2021

Your trusted partner for
cloud enablement