

Secure & Cost-Effective Aurora MySQL RDS Setup for WordPress (Demo Project)

Step-by-step guide with screenshots — includes security and cost best practices.

Prepared: 06 Sep 2025

1) Goal & Scope

Run a small WordPress site whose database is hosted on Amazon RDS (Aurora MySQL). This guide shows how to create the Aurora cluster, lock down access using Security Groups, verify connectivity from an EC2 instance, and take a manual snapshot. It also lists security and cost-optimization tips suitable for a startup demo environment.

2) High-Level Architecture

- 1 VPC with subnets for EC2 and RDS.
- 2 Aurora MySQL DB cluster (writer endpoint used by WordPress).
- 3 Security Groups: RDS allows MySQL (3306) only from the EC2 security group.
- 4 Backups via automated backups + on-demand snapshots.

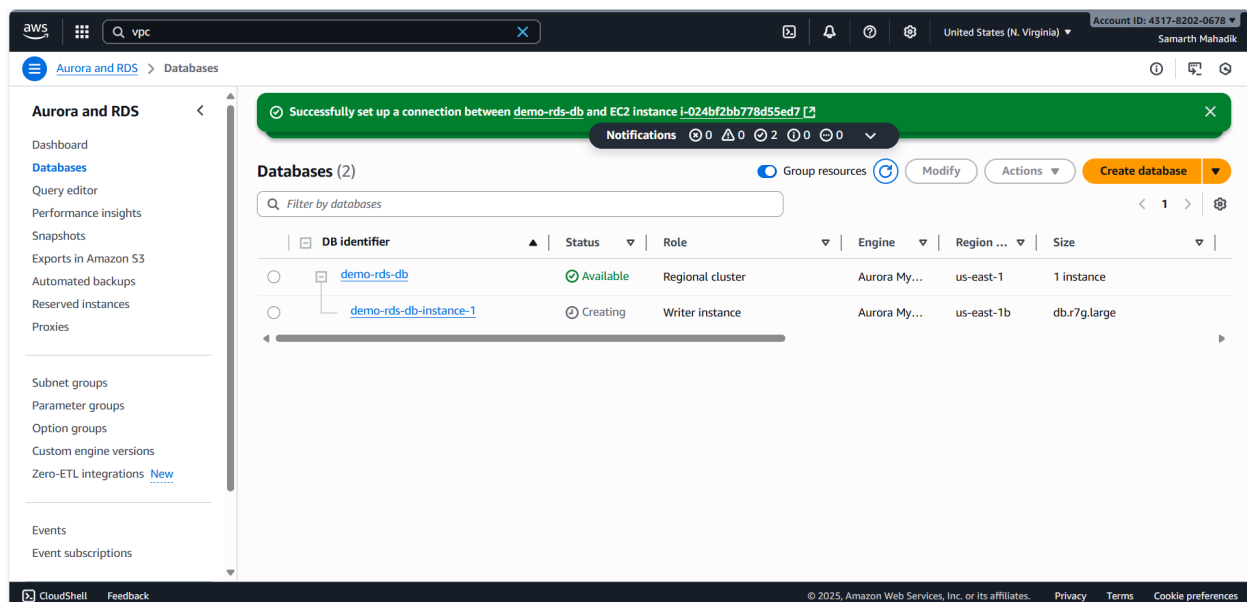
3) Prerequisites

- AWS account with access to RDS, EC2, VPC, and IAM.
- One Linux EC2 instance for testing (bastion/app server) with MySQL client installed.
- Existing VPC and subnets. (RDS should not be publicly accessible for production.)
- WordPress application host (EC2/Elastic Beanstalk/ECS) ready to consume the DB.

4) Step-by-Step

Step 1 — Create Aurora MySQL DB Cluster

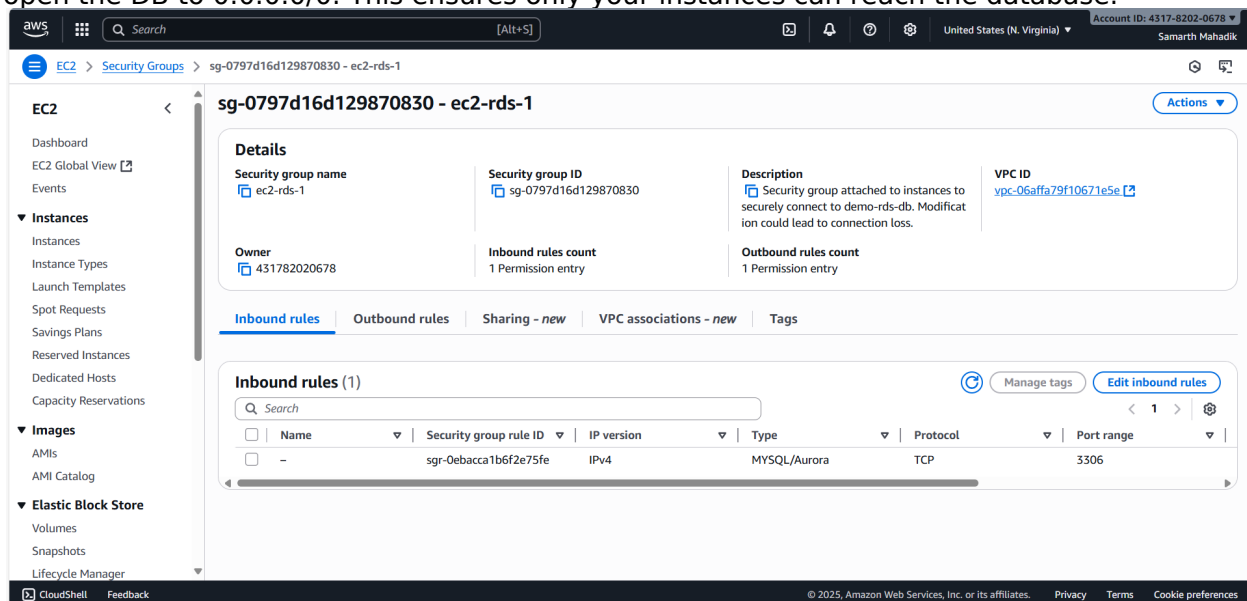
In the RDS console → Databases → Create database → Standard create → Engine: Aurora MySQL. Choose a small instance class (for demo), disable public access, and place the cluster in private subnets. Create a master user for initial setup. Save the writer endpoint — WordPress will use it as DB_HOST.



RDS: Aurora MySQL cluster created (demo-rds-db).

Step 2 — Lock down Security Groups

Create/identify an EC2 security group (app/bastion). On the RDS security group, add an inbound rule: Type = MYSQL/Aurora, Port = 3306, Source = the EC2 security group. Do NOT open the DB to 0.0.0.0/0. This ensures only your instances can reach the database.

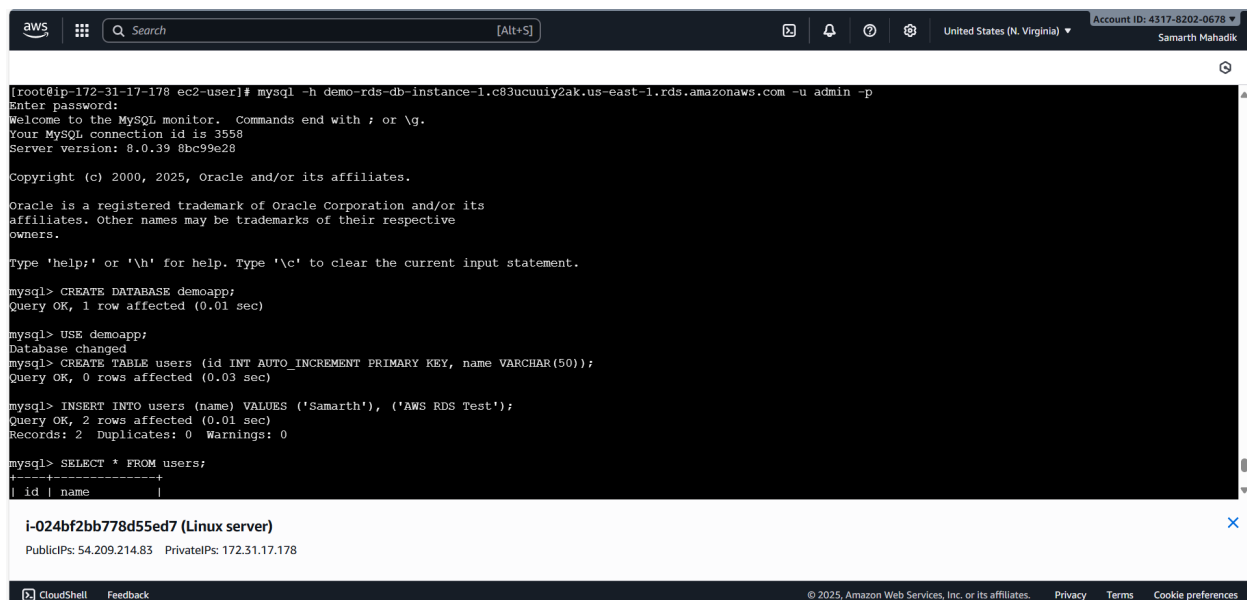


EC2 security group rule for RDS access on port 3306.

Step 3 — Verify connectivity from EC2

SSH to the EC2 instance and connect to the cluster endpoint using the MySQL client. Below we also show commands to prepare a dedicated WordPress database/user with least privilege.

```
# Connect (replace with your cluster writer endpoint) mysql -h -u -p -- Create a dedicated DB for
WordPress CREATE DATABASE wordpress DEFAULT CHARACTER SET utf8mb4 COLLATE
utf8mb4_unicode_ci; -- Create a least-privileged app user CREATE USER 'wp_user'@'%' IDENTIFIED BY
'ChangeMe_Strong#Password1'; -- Grant only what WordPress needs on its own DB GRANT ALL
PRIVILEGES ON wordpress.* TO 'wp_user'@'%; FLUSH PRIVILEGES; -- (Optional) Example connectivity
check USE wordpress; CREATE TABLE ping (id INT PRIMARY KEY AUTO_INCREMENT, note VARCHAR(50));
INSERT INTO ping (note) VALUES ('hello from EC2'); SELECT * FROM ping;
```



```
[root@ip-172-31-17-178 ec2-user]# mysql -h demo-rds-db-instance-1.c83ucuiy2ak.us-east-1.rds.amazonaws.com -u admin -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 3558
Server version: 8.0.39 8bc99e28

Copyright (c) 2000, 2025, 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> CREATE DATABASE demoapp;
Query OK, 1 row affected (0.01 sec)

mysql> USE demoapp;
Database changed
mysql> CREATE TABLE users (id INT AUTO_INCREMENT PRIMARY KEY, name VARCHAR(50));
Query OK, 0 rows affected (0.03 sec)

mysql> INSERT INTO users (name) VALUES ('Samarth'), ('AWS RDS Test');
Query OK, 2 rows affected (0.01 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM users;
+----+-----+
| id | name |
+----+-----+
| 1  | Samarth |
| 2  | AWS RDS Test |
+----+-----+
```

i-024bf2bb778d55ed7 (Linux server)
PublicIPs: 54.209.214.83 PrivateIPs: 172.31.17.178

Connectivity test from EC2: connected to Aurora and ran SQL.

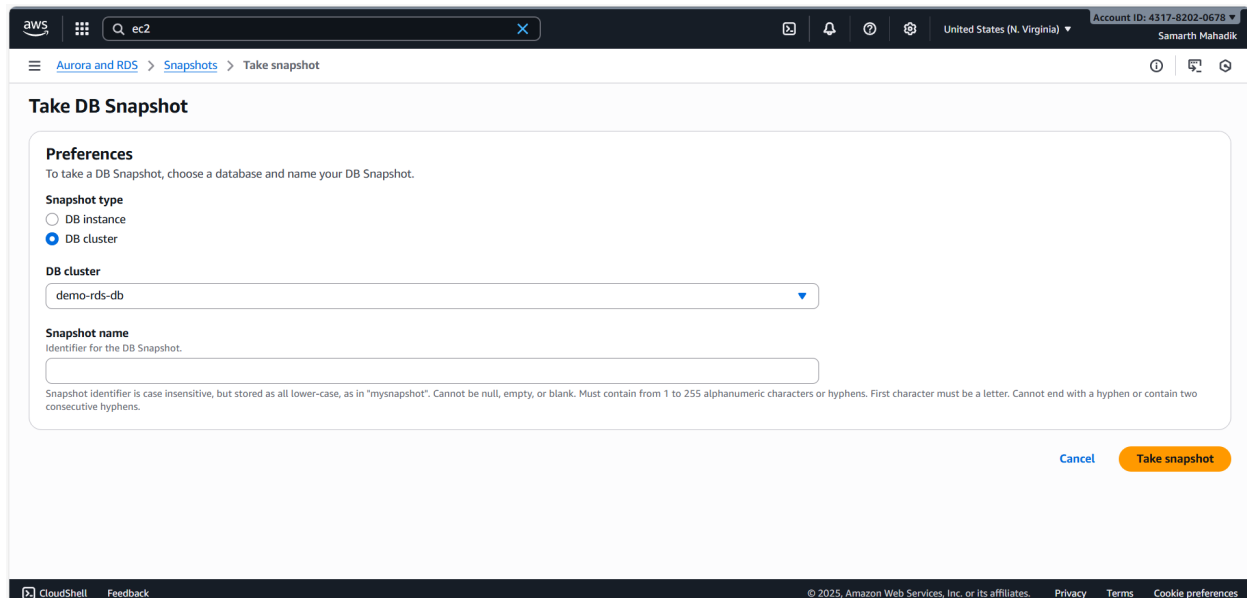
Step 4 — Point WordPress to RDS

Edit wp-config.php on the WordPress host as below:

```
// in wp-config.php define('DB_NAME', 'wordpress'); define('DB_USER', 'wp_user');
define('DB_PASSWORD', 'ChangeMe_Strong#Password1'); // Prefer AWS Secrets Manager for production
define('DB_HOST', ':3306'); define('DB_CHARSET', 'utf8mb4'); define('DB_COLLATE', '');
```

Step 5 — Take a manual snapshot

In the RDS console → Snapshots → Take snapshot → Select DB cluster → Provide a name → Take snapshot. Use snapshots before risky changes or upgrades; they are also useful to clone environments.



Taking a manual DB cluster snapshot.

5) Security Best Practices

- Keep RDS not publicly accessible; reach it from private subnets and from trusted security groups only.

- Use AWS Secrets Manager to store DB credentials; rotate regularly.
- Enable and enforce TLS. In the cluster parameter group, set `require_secure_transport=ON` (Aurora MySQL).
- Create a dedicated database user for WordPress and avoid using the master user.
- Apply least-privilege IAM policies to administrators and pipelines.
- Turn on automated backups; consider Multi-AZ for production (skip for low-cost demos).

6) Cost Optimization Checklist

- Choose a small burstable instance class or consider Aurora Serverless v2 with a low minimum capacity for low traffic.
- Use Single-AZ for demo/non-prod. Enable Multi-AZ only when you truly need HA.
- Right-size backup retention; delete old manual snapshots you no longer need.
- Turn off Performance Insights and enhanced monitoring if not required for the demo.
- Use Graviton-based instance classes where compatible to reduce cost.

7) Operations & DR Tips

- Enable automatic minor version upgrades during a maintenance window.
- Use point-in-time restore for accidental deletes; keep backup retention long enough to cover your risk.
- Before app releases, take a manual snapshot so you can roll back quickly.

8) Cleanup (to avoid charges)

- Delete the WordPress stack (or stop the app server) when done.
- Delete the RDS cluster after taking a final snapshot if you need a copy; otherwise delete snapshots as well.
- Remove security group rules that are no longer required.

Appendix — Quick Commands

Install MySQL client on Amazon Linux: `sudo yum install -y mysql` Connect: `mysql -h <aurora-writer-endpoint> -u <user> -p`

End of guide.