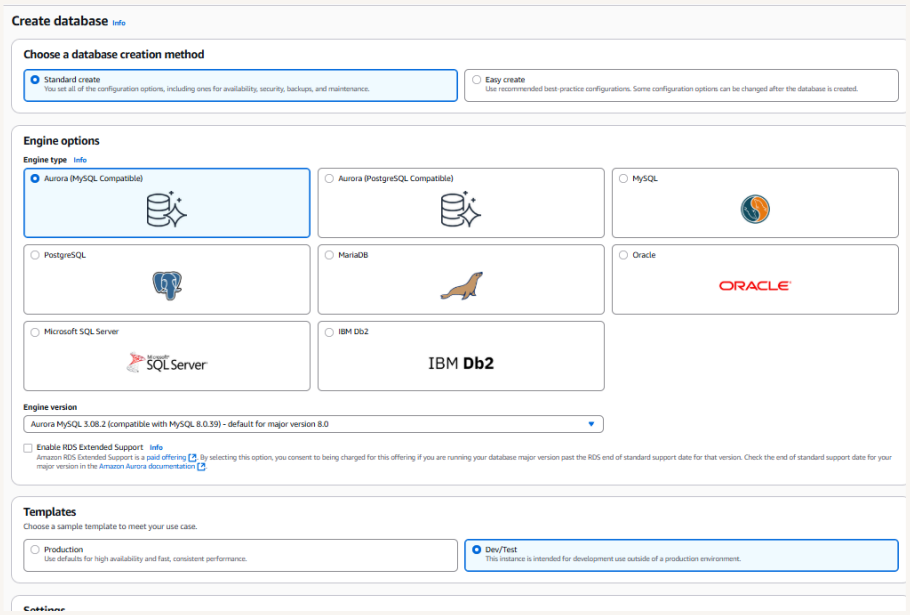
nextwork.org

Connect a Web App to Amazon Aurora

Joba Amunigun
https://www.linkedin.com/in/dvoice



Joba Amunigun
NextWork Student

Introducing Today's Project!

What is Amazon Aurora?

Amazon Aurora is a fully managed relational database engine from AWS that is compatible with MySQL and PostgreSQL. It combines the performance and availability of high-end commercial databases with the simplicity and cost-effectiveness of open-source databases. It is useful because it offers: High performance (up to 5x faster than standard MySQL) Automatic backups and replication Seamless scalability High availability with failover support Built-in security features These benefits make Aurora ideal for modern cloud applications that need a reliable, fast, and secure database solution — without the complexity of managing infrastructure.

How I used Amazon Aurora in this project

✍️ In today's project, I used Amazon Aurora to create a MySQL-compatible relational database and connect it to an Amazon EC2 instance. I configured Aurora with the right settings, created a secure environment using security groups, and ensured my EC2 instance could communicate with the database. This setup lays the groundwork for hosting a web application that relies on a powerful, scalable, and highly available backend database — all while gaining hands-on experience with cloud-based relational database architecture.

One thing I didn't expect in this project was...

One thing I didn't expect in this project was how seamless and automated the cluster creation process was in Amazon Aurora. I anticipated having to manually configure replication or failover, but Aurora handled the setup of both writer and reader instances with built-in high availability and failover support — all with just a few clicks.

It really highlighted how cloud services can simplify what used to be complex infrastructure tasks.

This project took me...

This project took me approximately 1–2 hours to complete. I spent time carefully setting up the Aurora database, launching and configuring the EC2 instance, adjusting security groups, and understanding how database clusters work. It was a productive session that deepened my hands-on AWS skills — and yes, I rewarded myself with chicken afterward! 😄

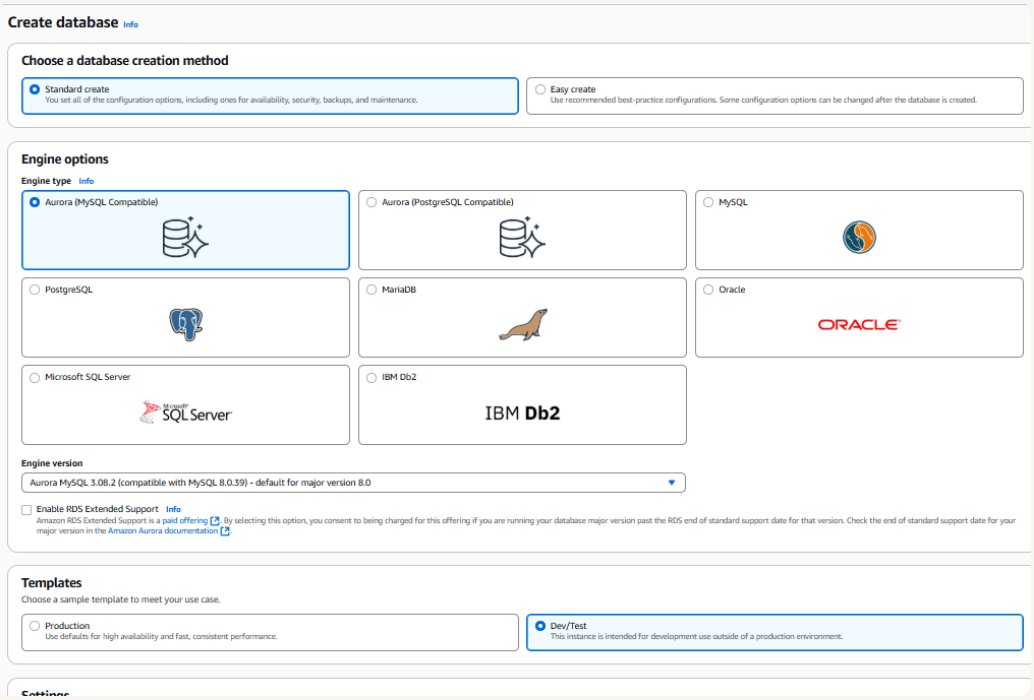
Joba Amunigun
NextWork Student

In the first part of my project...

Creating an Aurora Cluster

A relational database is a type of database that organizes data into tables, which are collections of rows and columns. Kind of like a spreadhsheet! We call it "relational" because the rows relate to the columns and vice-versa.

Aurora is a good choice when dealing with something large-scale , with peak performance and uptime. This is because Aurora databases use clusters (more on that later!). Ordinary relational databases, like MySQL and Oracle are more generic and cost-effective. They suit smaller databases and less demanding workloads. In summary, Aurora is for the big jobs. Fun fact: Coca-Cola uses Amazon Aurora to store their consumer data globally!


Create database Info
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
ORACLE
Microsoft SQL Server
SQL Server
IBM Db2
IBM Db2
Engine version
Aurora MySQL 3.08.2 (compatible with MySQL 8.0.39) - default for major version 8.0
Enable RDS Extended Support Info
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.

Joba Amunigun
NextWork Student

Halfway through I stopped!

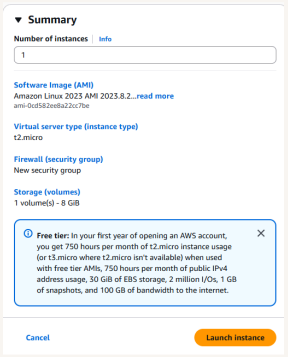
I stopped creating my Aurora database because i needed to create my EC2 Instance - I will be connecting a web app server to my Aurora database which will be connected Via EC2 Instance

Features of my EC2 instance

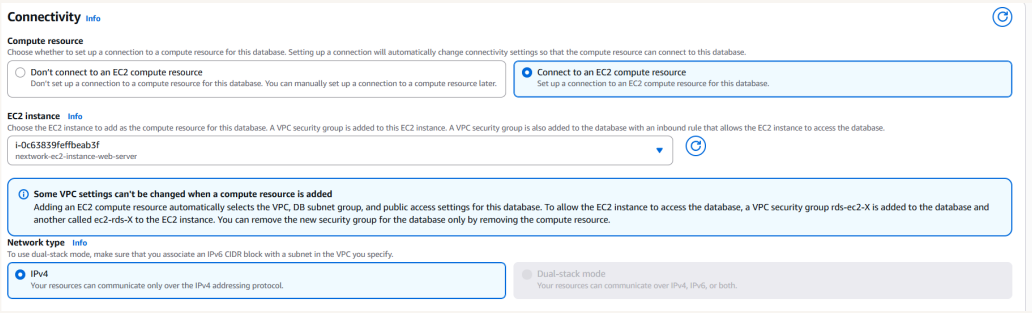
I created a new key pair for my EC2 instance because a key pair in EC2 is needed to access the EC2 instance. Keys are needed for EC2 instance in order to add, change, or update how the EC2 instance is running. Since a EC2 instance is a virtual machine, a good way to think about key pairs is like the login credentials to your own computer...except it's all virtual!

When I created my EC2 instance, in the "Details Tab" I took particular note of the "Public IPv4 DNS" in the Instance Summary and the value for the Key pair name because: The Public IPv4 DNS is the address I will use to connect to my EC2 instance remotely — it’s essentially the “location” of the virtual machine on the internet. The Key pair name is critical for authentication; it ensures that only someone with the corresponding private key can access the instance securely. Without the DNS, I wouldn’t know where to connect, and without the key pair, I wouldn’t be able to gain access — both are essential for managing the EC2 instance.

Joba Amunigun
NextWork Student



Then I could finish setting up my database



Aurora Database uses clusters because clusters provide high availability, scalability, and fault tolerance. By having a writer instance for all write operations and one or more reader instances for read operations, Aurora can distribute the load efficiently and handle large volumes of traffic. If the primary (writer) instance fails, a reader can be automatically promoted to take over — ensuring minimal downtime. This design makes Aurora a powerful and resilient choice for production-grade relational databases in the cloud.

The place to learn & showcase your skills

Check out nextwork.org for more projects