

Task B: The Platform

Chris Dusyk - 200268705

Amazon Aurora

Aurora is AWS' custom relational database engine. It uses the MySQL structure, attached to AWS' new database engine. The database engine makes use of Amazon's distributed cloud services platform as opposed to a traditional database server. The engine itself is sharded across many virtual processors, allowing for very high performance and throughput.

Structure

Aurora uses the MySQL database structure and language on top, making it easy to port an existing MySQL database over to it. Aurora follows all standard MySQL syntax and design, so any new database is handled like a MySQL database.

Underlying Structure

Under-the-hood, Aurora is very different than any current database engine. It's designed around the fact that it's hosted in large datacentres where the database workload can be distributed across many virtual processors. Unfortunately, specific details are hard to come by as this is Amazon's proprietary engine, and is directly competing with other products from Microsoft Azure and Google Cloud.

Getting Set Up

Create an AWS Account

AWS uses Amazon accounts, so if you already have an Amazon account, AWS will use that. To create an account, or login, go to <http://aws.amazon.com>.


AWS Console


Once logged into the AWS Management Console, go to Services > RDS. This will take you to the RDS Dashboard. From here, you can launch a DB Instance. AWS has instance support for Aurora, MySQL, MariaDB, PostgreSQL, Oracle, and Microsoft SQL Server. For this assignment, we will use Amazon Aurora:


Step 1: **Select Engine**


Select Engine


To get started, choose a DB Engine below and click Select.


**Amazon Aurora**

**MySQL**

**MariaDB**

**PostgreSQL**

**ORACLE**

**Microsoft SQL Server**

Aurora

High performance MySQL-compatible, enterprise-class database at 1/10th the cost of commercial databases

- Up to 5x the throughput and 10x the capacity of RDS MySQL
- Up to 15 promotable read replicas with less than 10 ms lag
- Up to 64TB of auto-scaling storage with 6-way replication

Select

Cancel

Once Amazon Aurora is selected, you will be taken to the next setup step. Here you will choose the Instance Class, Multi-Availability Zone

Deployment, the instance identifier, and the master user. The Instance Class is the size and power of the instance. For a small database like this example I used db.r3.large (2 vCPU, 15 GB RAM) as that was more than sufficient for the database I needed. Multi-AZ Deployment allows your database to be replicated across 3 availability zones such that if anything happens to the health of your primary instance, there will be immediate failover to one of the replicated instances, so that service is not interrupted. The DB Instance Identifier is the name of your database instance, such as 'cs401-db'. The Master Username and Password are the master credentials for the database:

Specify DB Details

Instance Specifications

DB Engine Aurora - compatible with MySQL 5.6.10a

DB Instance Class db.r3.large — 2 vCPU, 15 GiB RAM ▼

Multi-AZ Deployment No ▼

Settings

DB Instance Identifier* cs401-db

Master Username* dbmaster

Master Password*

Confirm Password*

Retype the value you specified for Master Password.

* Required

Cancel

Previous

Next Step

Once Next Step is clicked, you will be taken to the Advanced Settings page. You can leave these at default and click Launch DB Instance.

Once the instance has finished setting up, you can go to the Instances tab in RDS. Here you will see all of the general information on your DB Instance.

Filter: All Instances ▼ Search DB Instances... X Viewing 1 of 1 DB Instances

Engine	DB Instance	Status	CPU	Current Activity	Maintenance	Class	VPC	Multi-AZ	Replication Role	Encrypted
Aurora	app401-db	available	5.17%	2 Selects/sec	None	db.r3.large	vpc-87678fe2	No	writer	No

Cluster Endpoint: app401-db-cluster.cluster-ctnsn4lhnv7t.us-west-2.rds.amazonaws.com:3306 (authorized) ⓘ

Configuration Details

Engine Aurora 5.6.10a

Created Time January 18, 2016 at 9:47:27 PM UTC-6

DB Name

Username dbmaster

Parameter Group default:aurora5.6 (in-sync)

DB Cluster Parameter Group default:aurora5.6 (in-sync)

Copy Tags To Snapshots No

Security and Network

Availability Zone us-west-2c

VPC vpc-87678fe2

Subnet Group default (Complete)

Subnets subnet-7bbea20f
subnet-cc1d228a
subnet-08ca356d

Security Groups rds-launch-wizard-1 (sg-5cc42d3b) (active)

Publicly Accessible Yes

Endpoint app401-db.cluster-ctnsn4lhnv7t.us-west-2.rds.amazonaws.com

Port 3306

Certificate Authority rds-ca-2015 (Mar 5, 2020)

Instance and IOPS

Instance Class db.r3.large ⓘ

Storage Type DB Cluster

Encryption Details

Encryption Enabled No

Availability and Durability

DB Instance Status available

Multi AZ No

Maintenance Details

Auto Minor Version Upgrade Yes

Maintenance Window sun:09:00-sun:09:30

Pending Maintenance None

Instance Actions ▼ Tags Logs

To connect to your database, you will want to use the Endpoint address. In the above example, the endpoint would be "app401-db.ctnsn4lhnv7t.us-west-2.rds.amazonaws.com".

Instance Security

AWS handles access to resources with Security Groups. In the DB Instance dashboard, click on the Security Group link for the instance. In this case, "rds-launch-wizard-1 (sg-5cc42d3b)". This will take you to the Security Group management. Add your IP as an Inbound rule for the

Security Group:

The screenshot shows the AWS Management Console interface for a Security Group. The main panel displays the 'Edit inbound rules' dialog for the Security Group 'sg-5cc42d3b'. The dialog has a table with four columns: Type, Protocol, Port Range, and Source. There are four rules listed, all for MySQL/Aurora on port 3306 using TCP. The sources are Custom IP addresses and My IP. The 'Add Rule' button is visible at the bottom left, and 'Cancel' and 'Save' buttons are at the bottom right.

Type	Protocol	Port Range	Source
MySQL/Aurora	TCP	3306	Custom IP 216.174.152.1
MySQL/Aurora	TCP	3306	Custom IP 64.201.202.18
MySQL/Aurora	TCP	3306	Custom IP 23.96.246.200
MySQL/Aurora	TCP	3306	My IP 216.174.152.1

Connecting

Now that all of this is set up, you can connect to the Aurora instance from any management studio that supports MySQL. Simply enter the endpoint, master username, and master password to the connection manager in your tool, and you will be connected to the database. Best practices recommends creating a separate application account in the database, as it makes granular security easier to manage inside the database.

.NET and MySQL

The code in this example uses an ASP.NET web application and the MySQL Connector to connect to Aurora. The MySQL Connector can be obtained by downloading the MySQL Installer from <http://dev.mysql.com/downloads/installer/>. You will also need to add the DLL's to the ASP.NET project to make use of it in the code:

- ■ `MySQL.Data`
- ■ `MySQL.Data.Entity.EF6`
- ■ `MySQL.Web`

The connection string will need to be added to the web.config file as well. Once this is done, you will be able to connect to the Aurora database with whichever method you prefer.