Creating Serverless Microservices on AWS
Objective: Reproduce a (very simple) monolithic MVC application found at: https://misdemo.temple.edu/vote4movies

Steps:
- Define the database tier
- Define the AWS Lambda functions
- Define the Amazon API Gateway
- Develop the application itself (HTML/CSS/JavaScript)
- Post to Amazon S3
Vote 4 Movies
Vote for your favorite movie

Menu
- Vote
- Voting report
- Add a movie

Add a movie
- Movie name:
- Year released:

Add

Created for the MIS Department at Temple University
Define the database tier
Setting up my relational database ...
Database options

Database name
shafervote4moviesdb

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

Database port
TCP/IP port the DB instance will use for application connections.
3306

DB parameter group
default.mariadb10.1

Option group
default:mariadb-10-1
<table>
<thead>
<tr>
<th>Configurations</th>
<th>Security and network</th>
<th>Instance and IOPS</th>
<th>Maintenance details</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARN</td>
<td>Availability zone</td>
<td>Instance Class</td>
<td>Auto minor version upgrade</td>
</tr>
<tr>
<td>arn:aws:rds:us-east-2:26461652257533:db:shaferds</td>
<td>us-east-2b</td>
<td>db.t2.micro</td>
<td>Yes</td>
</tr>
<tr>
<td>Engine</td>
<td>VPC</td>
<td>Storage Type</td>
<td>Maintenance window</td>
</tr>
<tr>
<td>MariaDB 10.1.31</td>
<td>vpc-64a5e60c</td>
<td>General Purpose (SSD)</td>
<td>sat 06:34 - sat 07:04 UTC (GMT)</td>
</tr>
<tr>
<td>License Model</td>
<td>Subnet group</td>
<td>Storage</td>
<td>Pending Modifications</td>
</tr>
<tr>
<td>General Public License</td>
<td>default</td>
<td>20 GB</td>
<td>Master User Password: ****</td>
</tr>
<tr>
<td>DB Name</td>
<td>Subnets</td>
<td>Availability and durability</td>
<td>Pending maintenance</td>
</tr>
<tr>
<td>shafervote4moviesdb</td>
<td>subnet-69621f01</td>
<td>DB instance status</td>
<td>none</td>
</tr>
<tr>
<td>Username</td>
<td>subnet-6d4a5420</td>
<td>creating</td>
<td>Encryption details</td>
</tr>
<tr>
<td>shaferdsadmin</td>
<td>subnet-96d57fe4</td>
<td>Multi AZ</td>
<td>Encryption enabled</td>
</tr>
<tr>
<td>Option Group</td>
<td>Security groups</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>defaultmariadb-10-1</td>
<td>rds-launch-wizard (sg-044638dce1a1d7ff3e)</td>
<td>Backup and Restore</td>
<td></td>
</tr>
<tr>
<td>Parameter group</td>
<td>(active)</td>
<td>Automated backups</td>
<td>Built-in backups not enabled (7 days)</td>
</tr>
<tr>
<td>defaultmariadb 10.1 (in-sync)</td>
<td></td>
<td>Backup window</td>
<td>06:40 - 09:10 UTC (GMT)</td>
</tr>
<tr>
<td>Resource ID</td>
<td>Publicly accessible</td>
<td>Copy tags to snapshots</td>
<td>Yes</td>
</tr>
<tr>
<td>db-KCUQOITR667HNU7DTWP6MCINY</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Certificate authority</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rds-ca-2015 (Mar 5, 2020)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Define the AWS Lambda functions
Our simple application needed **four** functions. So the following steps were repeated four times with appropriate variation.
It was necessary to author the Lambda functions from scratch for a simple example that did not use some existing AWS service.

The language options were C#, Node.js, Python, and Go.

It was also necessary to specify the security role for each function.
Getting started ... note the ARN, as it will be needed later.
Code to be edited here. Each Lambda function needs to be tested.
Define the Amazon API Gateway
Now we integrate our new Lambda function with a web API.
The Amazon API Gateway gives a visual representation of the web call and response.
Now deploy the API
The results
Four Endpoints

# Simple GET request: Gets all the movies - returns a JSON array
# Example: {"movie_id":22,"movie_name":"The Avengers","movie_year_released":"2012"}
https://5f1l0wps35.execute-api.us-east-2.amazonaws.com/shafer_vote4movies/movies

# Simple GET request: Gets movie names and the number of votes - returns a JSON array
# Example: {"movie_id":22,"movie_name":"The Avengers","movie_year_released":"2012"}

https://5f1l0wps35.execute-api.us-east-2.amazonaws.com/shafer_vote4movies/movievotes

# Vote for a movie.
# This is a POST. POST the following JSON {"movieid":"22"} (Here the 22 corresponds to "The Avengers")
# Returns the new vote_id and the original movie_id
https://5f1l0wps35.execute-api.us-east-2.amazonaws.com/shafer_vote4movies/vote

# Add a movie.
# This is a POST. POST the following JSON {"moviename":"E.T.", "movieyear":"1982"}
# The api returns the key of the movie just created.
https://5f1l0wps35.execute-api.us-east-2.amazonaws.com/shafer_vote4movies/movies
Now it’s time to build the web application.
Vote for your favorite movie

Add a movie

Movie name:

Year released:

Submit