

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

The original site



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 ...

AWS services

Find a service by name or feature (for example, EC2, S3 or VM, storage).



Recently visited services



RDS



API Gateway



DynamoDB



S3



Billing

Database options

Database name

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.

DB parameter group [Info](#)



Option group [Info](#)



Configurations

ARN

arn:aws:rds:us-east-2:646163223753:db:shaferrds

Engine

MariaDB 10.1.31

License Model

General Public License

DB Name

shafervote4moviesdb

Username

shaferrdsadmin

Option Group

default:mariadb-10-1

Parameter group

default.mariadb10.1 (in-sync)

Resource ID

db-KCUQOITR667HNCU7DTWP6MCINY

Security and network

Availability zone

us-east-2b

VPC

vpc-64a5e60c

Subnet group

default

Subnets

subnet-69621f01

subnet-6d4a5420

subnet-96d57fec

Security groups

rds-launch-wizard (sg-044638dce1a1dff3e)

(active)

Publicly accessible

No

Certificate authority

rds-ca-2015 (Mar 5, 2020)

Instance and IOPS

Instance Class

db.t2.micro

Storage Type

General Purpose (SSD)

Storage

20 GiB

Availability and durability

DB instance status

creating

Multi AZ

No

Backup and Restore

Automated backups

Enabled (7 Days)

Backup window

08:40-09:10 UTC (GMT)

Copy tags to snapshots

Yes

Maintenance details

Auto minor version upgrade

Yes

Maintenance window

sat:06:34-sat:07:04 UTC (GMT)

Pending Modifications

Master User Password: ****

Pending maintenance

none

Encryption details

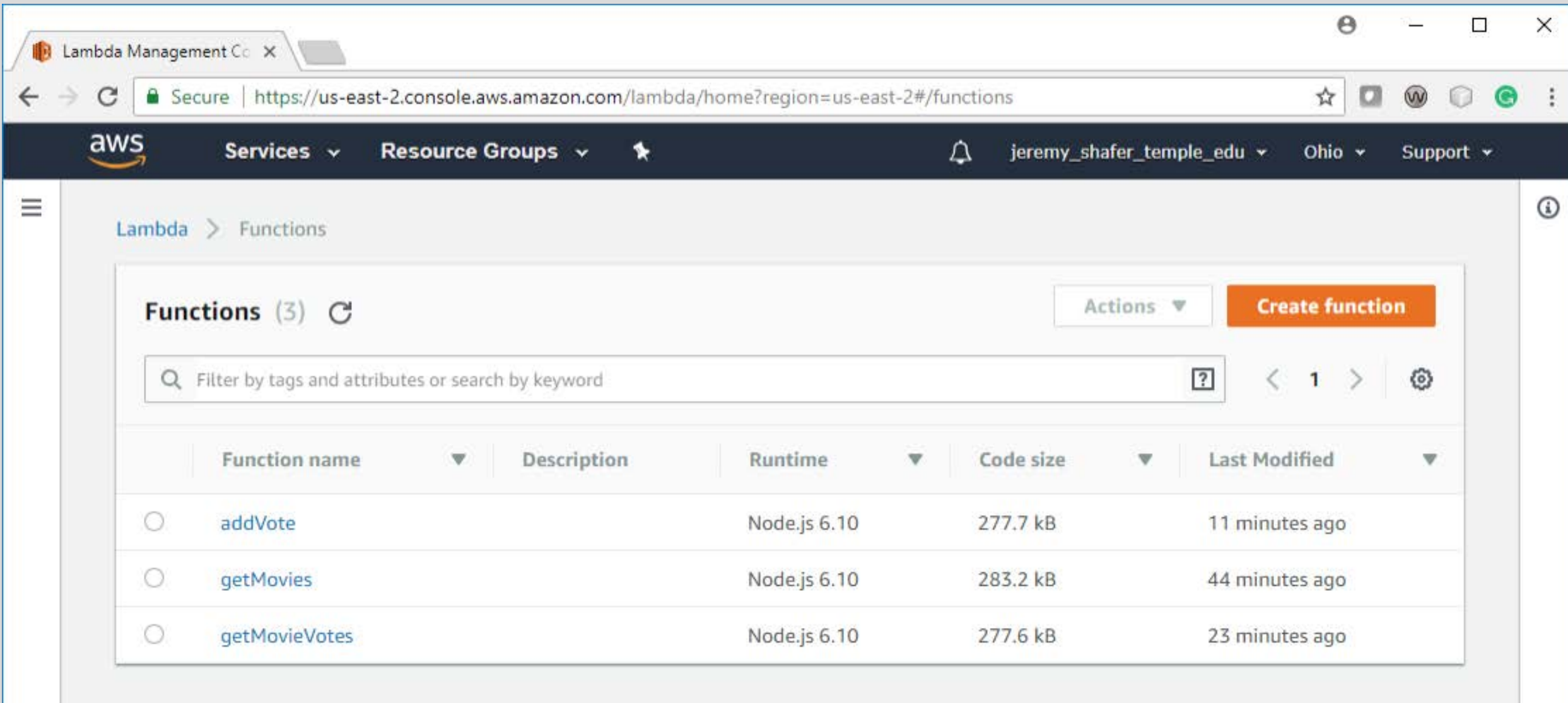
Encryption enabled

No

Define the AWS Lambda functions

Our simple application needed *four* functions.

So the following steps were repeated four times with appropriate variation.



The screenshot shows the AWS Lambda console interface. The browser address bar displays the URL `https://us-east-2.console.aws.amazon.com/lambda/home?region=us-east-2#/functions`. The AWS navigation bar includes the logo, 'Services', 'Resource Groups', a notification bell, the user 'jeremy_shafer_temple_edu', the region 'Ohio', and a 'Support' link. The main content area is titled 'Lambda > Functions' and shows a list of three functions. A search bar is present above the table, and a 'Create function' button is visible in the top right of the list area.

	Function name	Description	Runtime	Code size	Last Modified
<input type="radio"/>	addVote		Node.js 6.10	277.7 kB	11 minutes ago
<input type="radio"/>	getMovies		Node.js 6.10	283.2 kB	44 minutes ago
<input type="radio"/>	getMovieVotes		Node.js 6.10	277.6 kB	23 minutes ago

Author from scratch [Info](#)

Name

addMovie

Runtime

Node.js 6.10

Role

Defines the permissions of your function. Note that new roles may not be available for a few minutes after creation. [Learn more](#) about Lambda execution roles.

Create new role from template(s)

Lambda will automatically create a role with permissions from the selected policy templates. Note that basic Lambda permissions (logging to CloudWatch) will automatically be added. If your function accesses a VPC, the required permissions will also be added.

Role name

Enter a name for your new role.

role_addMovie

i This new role will be scoped to the current function. To use it with other functions, you can modify it in the IAM console.

Policy templates

Choose one or more policy templates. A role will be generated for you before your function is created. [Learn more](#) about the permissions that each policy template will add to your role.

Simple Microservice permissions X

Cancel

Create function

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.

Lambda > Functions > addMovie ARN - arn:aws:lambda:us-east-2:646163223753:function:addMovie

addMovie

Throttle Qualifiers ▼ Actions ▼ Select a test event.. ▼ Test Save

✔ Congratulations! Your Lambda function "addMovie" has been successfully created. You can now change its code and configuration. Click on the "Test" button to input a test event when you are ready to test your function. ✕

Code to be edited here. Each Lambda function needs to be tested.

The screenshot displays the AWS Lambda console interface for a function named 'addMovie'. The browser's address bar shows the URL: `https://us-east-2.console.aws.amazon.com/lambda/home?region=us-east-2#/functions/addMovie?tab=graph`. The console header includes the AWS logo, navigation menus for 'Services' and 'Resource Groups', and user information for 'jeremy_shafer_temple_edu' in the 'Ohio' region. The function configuration section shows 'Code entry type' set to 'Edit code inline', 'Runtime' set to 'Node.js 6.10', and 'Handler' set to 'index.handler'. Below this is a code editor window with a menu bar (File, Edit, Find, View, Goto, Tools, Window) and a file explorer on the left showing the 'addMovie' directory. The code in 'index.js' is as follows:

```
1 var mysql = require('mysql');
2 exports.handler = function(event, context) {
3   var connection = mysql.createConnection({
4     host: 'shaferrds.cn2trdxoswpr.us-east-2.rds.amazonaws.com',
5     user: '...',
6     password: '...',
7     database: 'shafervote4moviesdb'
8   });
9   sql = "INSERT INTO movies (movie_name, movie_year_released)";
10  var sql = sql + " VALUES('" + event.moviename + "','" + event.movieyear + "')";
11  connection.query(sql, function(error, results, fields) {
12    var response = {};
13    response['movieid'] = results.insertId;
14    context.succeed(response);
15  });
16 }
17
```

Define the Amazon API Gateway

RDS · AWS Console | API Gateway | Lambda Management Co

Secure | https://us-east-2.console.aws.amazon.com/apigateway/home?region=us-east-2#/apis/5f110wps35/resources/bxdkfq

aws Services Resource Groups

Amazon API Gateway APIs > shafer_vote4movies_api (5f110wps35) > Resources > /movies (bxdkfq) Show all hints ?

APIs

- shafer_dynamodb_api
- shafer_vote4movies_api
 - Resources
 - /movies**
 - GET
 - OPTIONS
 - /moviev...
 - GET
 - OPTIONS
 - /vote...
 - OPTIONS
 - POST
 - Stages
 - Authorizers
 - Gateway Responses
 - Models
 - Resource Policy
 - Documentation
 - Dashboard
 - Settings
- Usage Plans
- API Keys
- Custom Domain Names
- Client Certificates
- VPC Links
- Settings

Resources

Actions

- RESOURCE ACTIONS
 - Create Method
 - Create Resource
 - Enable CORS
 - Edit Resource Documentation
 - Delete Resource
- API ACTIONS
 - Deploy API
 - Import API
 - Edit API Documentation
 - Delete API

/movies Methods

Method	Function
GET	us-east-2:646163223753:functi...
Authorization	None
API Key	Not required

OPTIONS

Mock Endpoint

Authorization	None
API Key	Not required

© 2008 - 2018, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

- APIs
 - shafer_dynamodb_api
 - shafer_vote4movies_api
 - Resources**
 - Stages
 - Authorizers
 - Gateway Responses
 - Models
 - Resource Policy
 - Documentation
 - Dashboard
 - Settings
 - Usage Plans
 - API Keys
 - Custom Domain Names
 - Client Certificates
 - VPC Links
 - Settings

- Resources Actions
- /
 - /movies**
 - GET
 - OPTIONS
 - POST** ✓ ✕
 - /movievotes
 - GET
 - OPTIONS
 - /vote
 - OPTIONS
 - POST

/movies Methods

GET

arn:aws:lambda:us-east-2:646163223753:functi...

Authorization **None**

API Key **Not required**

OPTIONS

Mock Endpoint

Authorization **None**

API Key **Not required**

Now we integrate our new Lambda function with a web API.

The screenshot shows the AWS API Gateway console interface. The browser tabs include 'RDS · AWS Console', 'API Gateway', and 'Lambda Management Co'. The address bar shows the URL: `https://us-east-2.console.aws.amazon.com/apigateway/home?region=us-east-2#/apis/5f110wps35/resources/bxdkfq/methods/POST`. The navigation bar includes the AWS logo, 'Services', 'Resource Groups', and user information for 'jeremy_shafer_temple_edu' in 'Ohio'. The breadcrumb trail is: 'APIs > shafer_vote4movies_api (5f110wps35) > Resources > /movies (bxdkfq) > POST'. The left sidebar lists various API Gateway components like 'APIs', 'Resources', 'Stages', etc. The main content area is titled '/movies - POST - Setup' and contains the following configuration options:

- Integration type:** Radio buttons for 'Lambda Function' (selected), 'HTTP', 'Mock', 'AWS Service', and 'VPC Link'.
- Use Lambda Proxy integration:** A checkbox that is currently unchecked.
- Lambda Region:** A dropdown menu set to 'us-east-2'.
- Lambda Function:** An empty text input field.
- Use Default Timeout:** A checked checkbox.

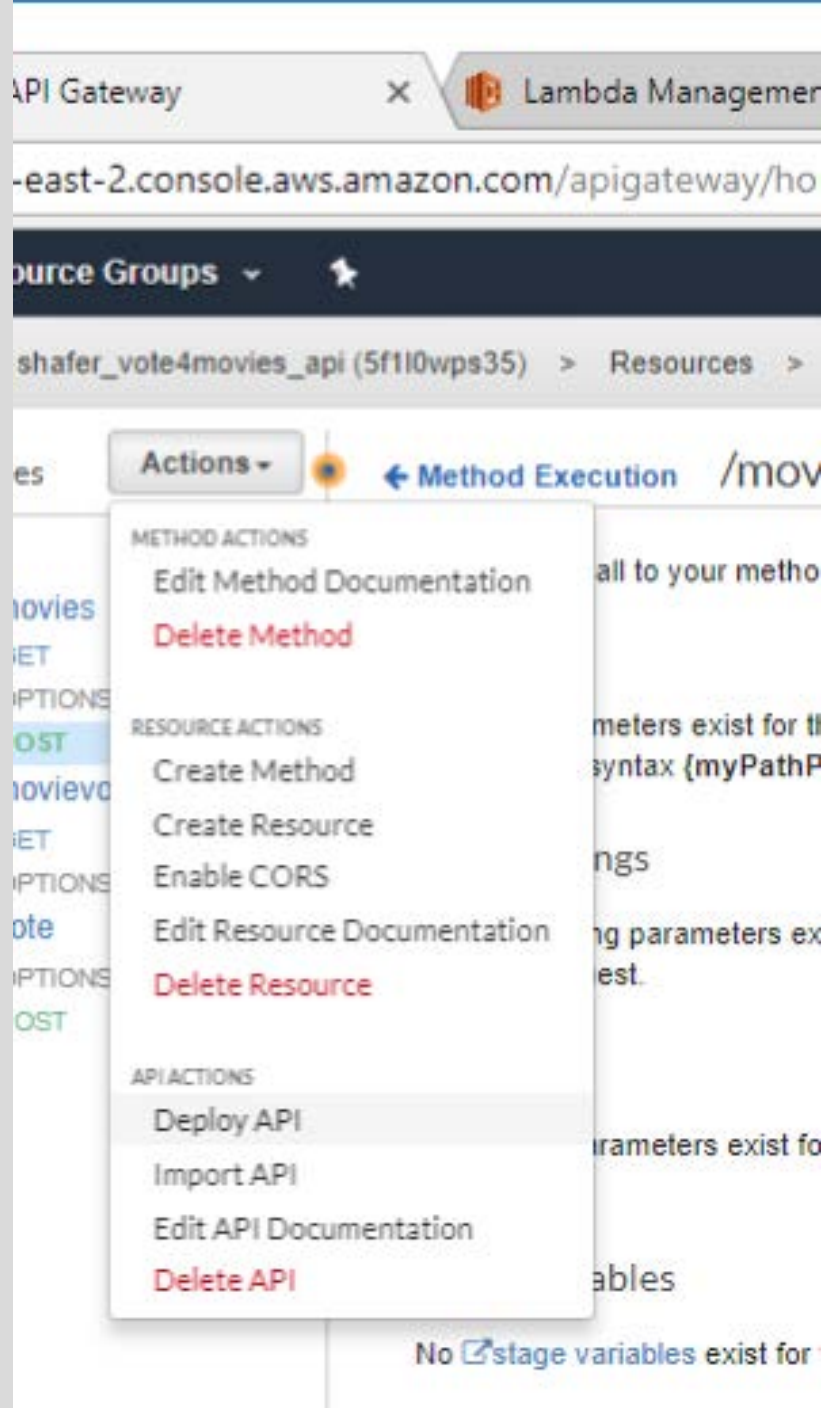
A 'Save' button is located in the bottom right corner of the configuration area.

The Amazon API Gateway gives a visual representation of the web call and response.

The screenshot displays the Amazon API Gateway console interface. The browser address bar shows the URL: `https://us-east-2.console.aws.amazon.com/apigateway/home?region=us-east-2#/apis/5f110wps35/resources/bxdkfq/methods/POST`. The console breadcrumb navigation is: `APIs > shafer_vote4movies_api (5f110wps35) > Resources > /movies (bxdkfq) > POST`. The main content area is titled `/movies - POST - Method Execution`. It features a vertical sidebar on the left with a tree view of resources: `/`, `/movies` (with sub-items `GET`, `OPTIONS`, `POST`), `/movievotes` (with `GET`, `OPTIONS`), and `/vote` (with `OPTIONS`, `POST`). The `POST` method under `/movies` is selected. The main area shows a flow diagram of the request and response process:

- Client**: A vertical box on the left representing the client.
- Method Request**: A box containing `Auth: NONE` and `ARN: arn:aws:execute-api:us-east-2:646163223753:5f110wps35:*/POST/movie`.
- Integration Request**: A box containing `Type: LAMBDA` and `Region: us-east-2`.
- Integration Response**: A box containing `HTTP status pattern: -` and `Output passthrough: Yes`.
- Method Response**: A box containing `HTTP Status: 200` and `Models: application/json => Empty`.
- Lambda addMovie**: A vertical box on the right representing the Lambda function.

Arrows indicate the flow: Client to Method Request, Method Request to Integration Request, Integration Request to Lambda addMovie, Lambda addMovie to Integration Response, Integration Response to Method Response, and Method Response to Client.



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.



Jeremy Shafer

 Vote

 Voting Report

 Add a movie

Vote for your favorite movie

[Home](#) / [Add a Movie](#)

Add a movie

Movie name:

Year released:

Submit