

ECE 528: Cloud Computing



Assignment #5

Midterm Tutorial Execution: Deploying an API with
AWS Lambda and AWS API Gateway

Michael Bowyer

March 31, 2021

1 Assignment Description

This assignment entails following the prescribed exercise steps created by another student in the ECE 528 course. The exercise steps I have selected for this assignment is that made by Jake Huneau, who describes what steps to take to utilize AWS Lambda and API Gateway for hosting an API.

The steps described in the exercise of his tutorial indicate the following:

1. Update your API code so it includes a POST, PUT, and DELETE HTTP method
2. Upload your code to AWS Lambda using AWS s3 instead of uploading a zip
3. Set up another AWS Lambda function for a production version of your code
4. Set up your code pipeline so it automatically uploads changes to your code to AWS Lambda and also updates the dev version of API Gateway

The items themselves are a little confusing, as it appears item number three is redundant of item one. Item one indicates the creation of a new API with multiple methods, while item three indicates the setup of a new lambda function. Therefore, I have combined items one and three into the creation of a single lambda function which is for a new API I have written. Item two is deploying it using AWS S3 rather than the inline editor which is described in the following sections. Item four, however, is unfortunately vague and lacking details in the tutorial itself, so it isn't clear how to complete this step due to the fact that it appears to rely on other AWS services not described in the tutorial such as AWS CodePipeline. Instead of conducting this step, proof that the API has been successfully deployed to AWS API Gateway is provided.

2 Description of work done

2.1 Update your API code so it includes a POST, PUT, and DELETE HTTP method

The first step I took to accomplishing the excersises was to make my own API which dif-
ferences from the tutorial so that I could gain knowledge of how to do so. The API I made
was a simple sports team finder/editor one. The general idea is that someone can query the
API to do the following:

1. Get all of the team names in the NFL, NBA, or EPL (GET)
2. Find if a team exists in any of those leagues (GET)
3. Find what league a team belongs to (GET)
4. Add a new team to a league (POST)
5. Update a team name (PUT)
6. Delete a team from a league (DELETE)

Each of these API calls are demonstrated in the below images locally using uvicorn and
fastapi as described in the tutorial.

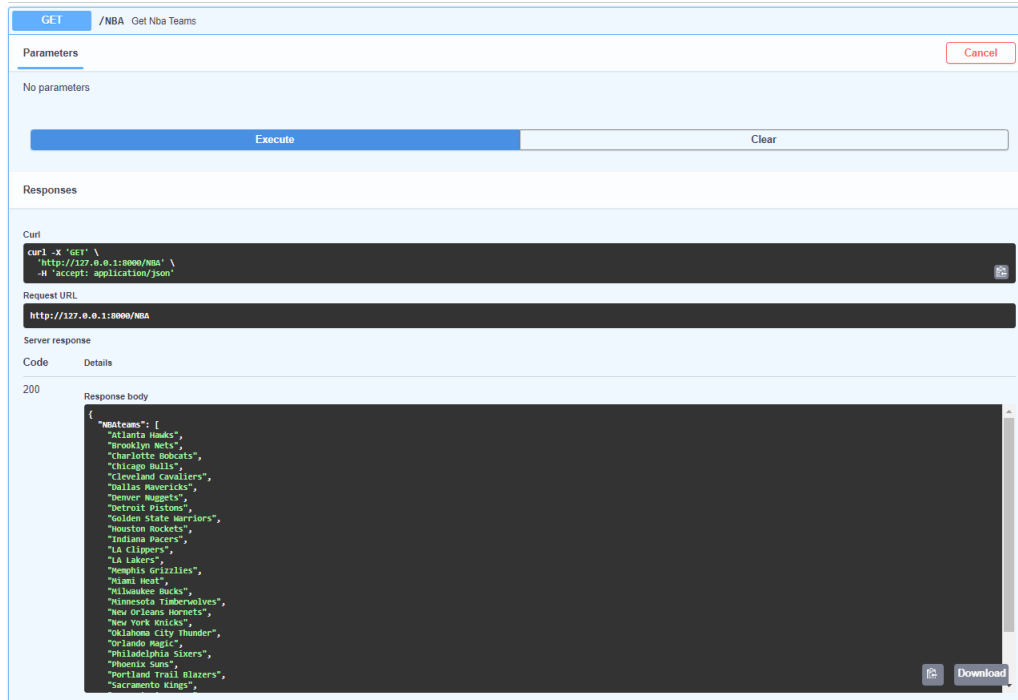


Figure 1: GET API function definition for obtaining all NBA team list

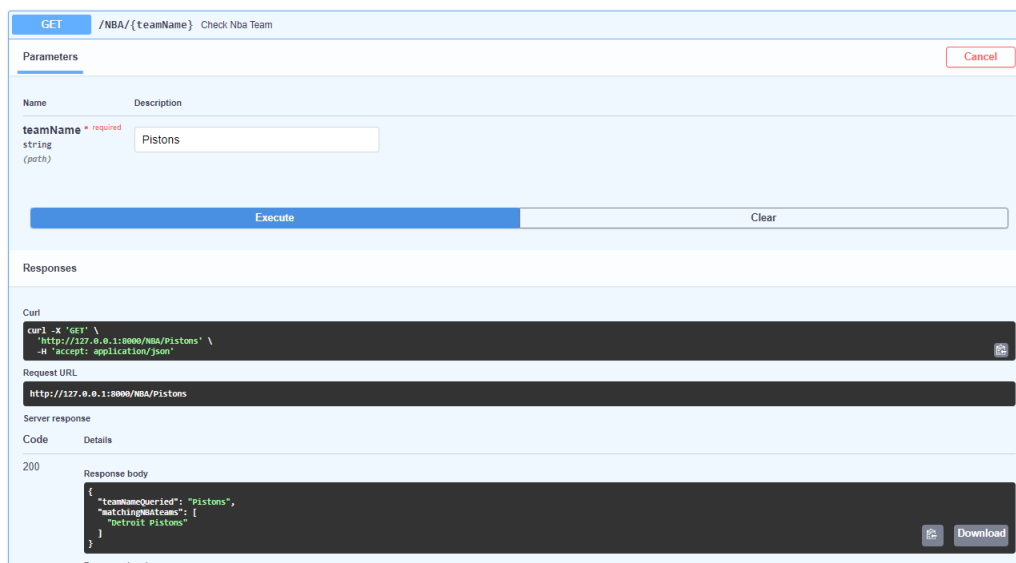


Figure 2: GET API function definition for getting NBA teams which meets a given name

GET /findTeam/{teamName} Find Team

Parameters

Name	Description
teamName * required string (path)	Arsenal

Execute **Clear**

Responses

200

Response body

```
{
  "teamNameQueried": "Arsenal",
  "matchingTeams": [
    {}
  ]
}
```

Figure 3: GET API function definition for finding any team in any league which meets a given name

POST /add/{league}/{teamName} Addteamtoleague

Parameters

Name	Description
league * required string (path)	NBA
teamName * required string (path)	MikesTeam

Execute **Clear**

Responses

200

Response body

```
{
  "league": "NBA",
  "teamName": "MikesTeam",
  "teams": [
    "Chicago Bulls",
    "Cleveland Cavaliers",
    "Dallas Mavericks",
    "Denver Nuggets",
    "Detroit Pistons",
    "Golden State Warriors",
    "Houston Rockets",
    "Indiana Pacers",
    "LA Clippers",
    "LA Lakers",
    "Memphis Grizzlies",
    "Miami Heat",
    "Milwaukee Bucks",
    "Minnesota Timberwolves",
    "New Orleans Pelicans",
    "New York Knicks",
    "Orlando Magic",
    "Philadelphia 76ers",
    "Phoenix Suns",
    "Portland Trail Blazers",
    "Sacramento Kings",
    "San Antonio Spurs",
    "Toronto Raptors",
    "Utah Jazz",
    "Washington Wizards",
    "MikesTeam"
  ]
}
```

Figure 4: POST API function definition for adding a new team to a given league

PUT

/[league]/[teamNameToUpdate]/[NewName] Update Teamname

Cancel

Parameters

Name	Description
league * required string (path)	EPL
teamNameToUpdate * required string (path)	Arsenal
NewName * required string (path)	Arsenal2

Execute Clear

Responses

Curl

```
curl -X 'PUT' \
  'http://127.0.0.1:8000/EPL/Arsenal/Arsenal2' \
  -H 'accept: application/json'
```

Request URL

```
http://127.0.0.1:8000/EPL/Arsenal/Arsenal2
```

Server response

Code	Details
200	<div>Response body</div> <pre>{ "updateLeagueTeams": ["Arsenal2", "A.F.C. Bournemouth", "Brighton & Hove Albion", "Burnley", "Chelsea", "Crystal Palace", "Everton", "Huddersfield Town", "Leicester City", "Liverpool", "Manchester City", "Manchester United", "Newcastle United", "Southampton", "Stoke City", "Sunderland", "Tottenham Hotspur", "Watford", "West Bromwich Albion", "West Ham United"] }</pre> <div>Download</div>

Figure 5: PUT API function definition for updating a team name in a given league

DELETE

/[league]/[teamNameToDelete] Delete Team

Cancel

Parameters

Name	Description
league * required string (path)	EPL
teamNameToDelete * required string (path)	Arsenal2

Execute Clear

Responses

Curl

```
curl -X 'DELETE' \
  'http://127.0.0.1:8000/EPL/Arsenal2' \
  -H 'accept: application/json'
```

Request URL

```
http://127.0.0.1:8000/EPL/Arsenal2
```

Server response

Code	Details
200	<div>Response body</div> <pre>{ "updateLeagueTeams": ["A.F.C. Bournemouth", "Brighton & Hove Albion", "Burnley", "Chelsea", "Crystal Palace", "Everton", "Huddersfield Town", "Leicester City", "Liverpool", "Manchester City", "Manchester United", "Newcastle United", "Southampton", "Stoke City", "Sunderland", "Tottenham Hotspur", "Watford", "West Bromwich Albion", "West Ham United"] }</pre> <div>Download</div>

Figure 6: DELETE API function definition for deleting a team name in a given league

2.2 Upload your code to AWS Lambda using AWS s3 instead of uploading a zip

It is a little difficult to prove that an AWS lambda function is actually using a zip file, but the following images are my best attempt to prove that it was completed.

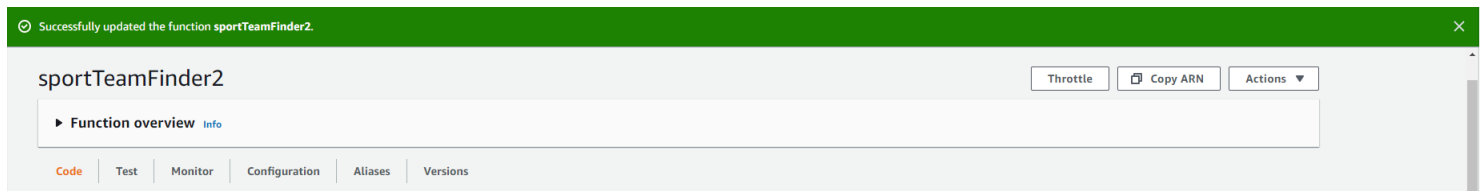


Figure 7: Proof of successful update of lambda function after connecting it to S3 storage

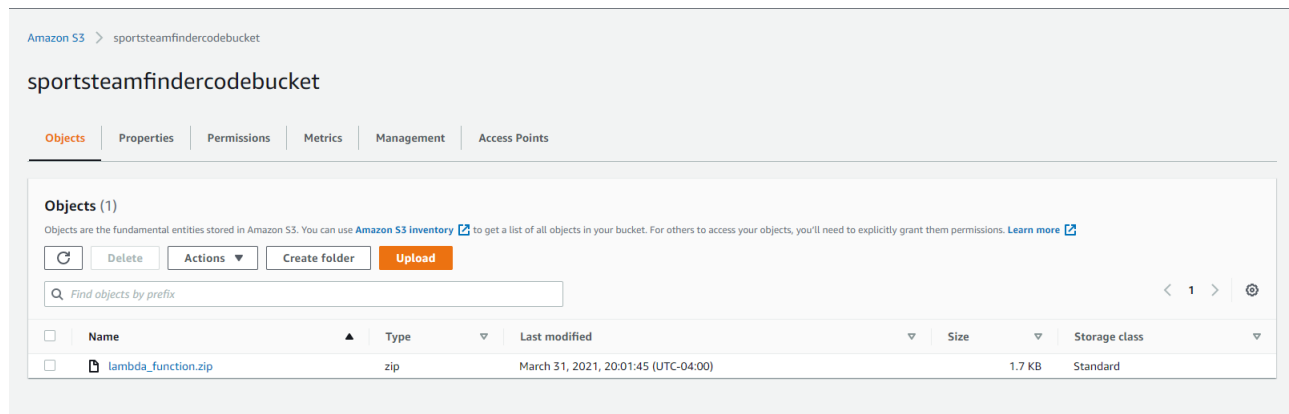


Figure 8: Proof of existing S3 bucket containing API code

2.3 Connect AWS Lambda Function to API Gateway

Finally, the API developed was connected to the API Gateway and the GET methods were tested. The below images show the result of the tests.

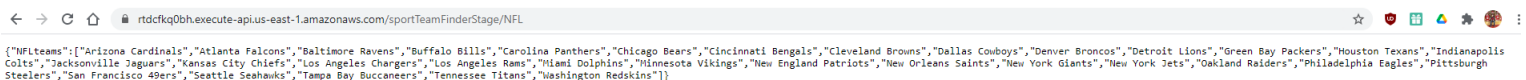
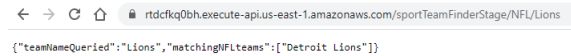


Figure 9: Proof of NFL GET method returns all NFL teams from AWS API Gateway URL



```
rtcdkq0bh.execute-api.us-east-1.amazonaws.com/sportTeamFinderStage/NFL/Lions
```

```
{ "teamNameQueried": "Lions", "matchingNFLteams": ["Detroit Lions"] }
```

Figure 10: Proof of NFL GET method returns Detroit Lions when queried for "Lions"

3 Code

All code generated within scope of this assignment can be found in the Assignment 5 folder in the GitHub repository: https://github.com/mikebowyer/ECE528_Assignments.