CS499 Cloud Computing and Big Data
Hackathon 2 - Bronco Express Live Map

Due Date

SHOW US IN CLASS: Monday 3:50pm, February 13, 2017
SAME DAY DELIVERY: Monday 11:59pm, February 13, 2017
IF YOU ARE BUSY: Tuesday 11:59pm, February 14, 2017
FINAL DUE: Wednesday 2:00pm, February 15, 2017

Note: Bonus/penalties will be considered based on these times/dates. Generally, the earlier you can submit, the better. The later you submit, the more likely you will lose points, and we will be picky on the functions and code quality. No submission will be allowed after Wed 2:00pm. Don't ask questions on late submissions.

"The critical ingredient is getting off your butt and doing something. It's as simple as that. A lot of people have ideas, but there are few who decide to do something about them now. Not tomorrow. Not next week. But today. The true entrepreneur is a doer, not a dreamer."

-- Nolan Bushnell

Score

50

Directions

Let's build a map to track Bronco Express in real-time!

In this project you will work with one partner (or just yourself) to develop the **backend** for a shuttle mapping serverless webservice. You will be working with AWS Lambda (ideally together with Serverless framework) and DynamoDB. Below are the functions you need to implement.

1. Create an API for storing bus information on DynamoDB

We have implemented an HTTP API to return a list of bus information in real-time: https://rqato4w151.execute-api.us-west-1.amazonaws.com/dev/info

Your first job is to call this API and storing all the returned bus information into a Dynamo table so we can access it later.

Similar to the waiting time example we demonstrated in class, you need to expose this function as an HTTP API. This API triggers fetching the bus information from the URL we provided and storing the bus information in a DynamoDB table. The DynamoDB table should use bus ID and timestamp as the primary key and sort key. For the values in this table, you should store all the information returned such as lat/lon, logo, etc.

Please expose this API via a GET HTTP method using AWS Lambda (Serverless framework).

2. Create an API for retrieving the most recent bus locations from DynamoDB

This API should return a JSON object with the most up to date bus locations on your Dynamo instance. The format should be exactly the same as the format from the URL above, but feel free to change logo (One idea here could be to link to bus routes to their icon so that all the buses on route A would be labeled as such, See 6.1) Expose this API via a GET HTTP method.

3. Configuring CORS

Before you testing your endpoints, you must configure your Lambda function HTTP API to allow CORS. In Serverless/Lambda, you can accomplish this by adding the following headers section to your response. See the example here.

```
const response = {
      statusCode: 200,
      headers: {
            "Access-Control-Allow-Origin" : "*"
      },
```

```
        body: JSON.stringify({
            message: 'OK!'
        }),
    };
```

4. Test your APIs with the Frontend

Step #1: Visit http://cs499-shuttle-service.s3-website-us-east-1.amazonaws.com/ and enter the http link to your retrival API (Step 2). If the map fills with buses then your API is correct.

Step #2: Visit http://cs499-shuttle-service.s3-website-us-east-1.amazonaws.com/ and use the orange button in the bottom right corner of the screen to force an update on the database. Enter the http link to your storing API (Step 1) and the location of the buses on the screen should be updated to reflect the new location in your database.

PLEASE FOCUS ON TASK 1-4 FOR YOUR SUBMISSION

5. Explore the BroncoShuttle API URL

The URL you used in this project is based on the current live map system (https://www.broncoshuttle.com). Like most other dynamic web services, BroncoShuttle (https://www.broncoshuttle.com) uses GET requests in order to retrieve the content it serves to its users. Using the developer window in your browser, navigate to the network tab. From here you can watch files being loaded from BroncoShuttle's server. You should be able to find a URL which is being used to serve BroncoShuttle data to the user. It could be a URL related to route details, a list of stops, or bus location information. If you are having trouble, try limiting your search to XML Http Requests (XHR). The following is an example of what you are looking for.

```
[
    {
      "ID": 3164,
      "ArrivalsEnabled": true,
      "DisplayName": "A Route A - INNOVATION AND IBW Daily Service",
    ....
    }
]
```

6. Extended Goals (Not required for submission, but will grant a bonus on completion)

6.1 Link Routes to Icons

Our frontend uses the image urls provided from your API to draw buses on the screen. Modify this URL on a per bus basis in order to label the buses on the different routes. This will require some research to determine which routeID corresponds to each route.

6.2 Bus Location Estimation API

Create an API which generates a 'best guess' for where the bus might be at a particular time and date. You can generate this in many ways but the simplest way might be to find the entry that is closest to the time requested. Expose this API via a GET HTTP method with query elements for the date and time (ex: http://yourapi/?date=searchTime&time=searchDate).

To test this service use the white button appears when you hover over the orange button in the bottom right hand corner of the screen.

Submission

All the code must be checked in to GitHub, but DO NOT check in your AWS credentials.

Please focus on the Task 1-4. You need to submission the following:

1. Team member names (only one member needs to submit it)
2. The URL to your GitHub repo that contains the web service source code
3. The URL to Step 1
4. The URL to Step 2
5. Please include the URLs to any other Endpoints that you finished

Please use Google Form (https://goo.gl/forms/PeBudNsP0d7EWtj93) to submit your work.

After Submission

Include this project in your resume. The keywords should be “Cloud Computing In-class Hackathon”, “Bronco Express Live Tracking Map”, “Web Service”, “AWS DynamoDB”, “AWS Lambda”, “Serverless Framework”, etc.