Finish the Ice Cream Ratings API

Background

Best For You Organics Company (BFYOC) has begun creation of a mobile application and public facing website, both of which will be used by customers to submit ratings and feedback of their ice cream. Both the website and application will be calling a set of APIs. BFYOC would like your team to create APIs to allow customers to submit ratings, and for business users to retrieve the ratings for reporting purposes.

It's difficult to predict the load generated by user reviews which can be impacted by several factors. Introduction or cancellation of a flavor will cause a spike in reviews, as will local events, like concerts or an unexpected heatwave. As a result, the solution you implement must be able to scale dynamically, and minimize costs whenever possible.

BFYOC would like to ensure that the proper engineering fundamentals are implemented from the beginning of the application development and want to ensure the team establishes a pipeline where updates are integrated (https://docs.microsoft.com/azure/devops/learn/what-is-continuous-integration/) and deployed (https://docs.microsoft.com/azure/devops/learn/what-is-continuous-delivery/) continuously. For compliance reasons, they also need to be able to track changes, and rollback any deployments as necessary.

Challenge

Setup

Your team has been asked to implement a solution that will automatically deploy the Ratings API. The solution needs to include a process to revert to a previous (code) commit and redeploy, as well as tracking all changes made to the code.

Your solution should handle automatically building the code every time it is checked/merged in to the `master` branch and generate an artifact that can be deployed. Assuming the build succeeds and applicable unit tests pass, the artifact should then be automatically deployed to an instance of your Ratings API.

Take this opportunity to discuss as a team how you will be working as team. Implement the appropriate branching strategies that will enable collaboration.

API Development

This challenge will require information about Users and Products. Three APIs have already been created for getting info about Users and Products:

- Get Products (https://serverlessohapi.azurewebsites.net/api/GetProducts)
- Get Product (https://serverlessohapi.azurewebsites.net/api/GetProduct?productId=75542e38-563f-436f-adeb-f426f1dabb5c) (expects a `productId` query parameter in the URL)
- Get Users (https://serverlessohapi.azurewebsites.net/api/GetUsers)
- Get User (https://serverlessohapi.azurewebsites.net/api/GetUser?userId=cc20a6fb-a91f-4192-874d-132493685376) (expects a `userId` query parameter in the URL)

You now should work together as a team to create an Azure Function App in your Azure subscription that will implement the Ice Cream Ratings part of the API.

Your challenge is to define, create and deploy three functions in this function app:

- **Input payload example**:

```
{
  "userId": "cc20a6fb-a91f-4192-874d-132493685376",
  "productId": "4c25613a-a3c2-4ef3-8e02-9c335eb23204",
  "locationName": "Sample ice cream shop",
  "rating": 5,
  "userNotes": "I love the subtle notes of orange in this ice cream!"
}
```

- **Requirements**
  - Validate both `userId` and `productId` by calling the existing API endpoints. You can find a user id to test with from the sample payload above
  - Add a property called `id` with a GUID value
  - Add a property called `timestamp` with the current UTC date time
  - Validate that the `rating` field is an integer from 0 to 5
  - Use a data service to store the ratings information to the backend
  - Return the entire review JSON payload with the newly created `id` and `timestamp`
- **Output payload example**:

```
{
  "id": "79c2779e-dd2e-43e8-803d-ecbebed8972c",
  "userId": "cc20a6fb-a91f-4192-874d-132493685376",
  "productId": "4c25613a-a3c2-4ef3-8e02-9c335eb23204",
  "timestamp": "2018-05-21 21:27:47Z",
  "locationName": "Sample ice cream shop",
  "rating": 5,
  "userNotes": "I love the subtle notes of orange in this ice cream!"
}
```

GetRating

- **Verb**: GET
- **Query string or route parameter**: `ratingId`
- **Requirements**
  - Get the rating from your database and return the entire JSON payload for the review identified by the id
  - Additional route parameters or query string values may be used if necessary.
- **Output payload example**:

```
{
  "id": "79c2779e-dd2e-43e8-803d-ecbebed8972c",
  "userId": "cc20a6fb-a91f-4192-874d-132493685376",
  "productId": "4c25613a-a3c2-4ef3-8e02-9c335eb23204",
  "timestamp": "2018-05-21 21:27:47Z",
  "locationName": "Sample ice cream shop",
  "rating": 5,
  "userNotes": "I love the subtle notes of orange in this ice cream!"
}
```

GetRatings

- **Verb**: GET
- **Query string or route parameter**: `userId`
- **Requirements**
  - Get the ratings for the user from your database and return the entire JSON payload for the reviews for the user identified by the id.
  - Additional route parameters or query string values may be used if necessary.
- **Output payload example**:

```
[
  {
    "id": "79c2779e-dd2e-43e8-803d-ecbebed8972c",
    "userId": "cc20a6fb-a91f-4192-874d-132493685376",
    "productId": "4c25613a-a3c2-4ef3-8e02-9c335eb23204",
    "timestamp": "2018-05-21 21:27:47Z",
    "locationName": "Sample ice cream shop",
    "rating": 5,
    "userNotes": "I love the subtle notes of orange in this ice cream!"
  },
  {
    "id": "8947f7cc-6f4c-49ed-a7aa-62892eac8f31",
    "userId": "cc20a6fb-a91f-4192-874d-132493685376",
    "productId": "e4e7068e-500e-4a00-8be4-630d4594735b",
    "timestamp": "2018-05-20 09:02:30Z",
    "locationName": "Another Sample Shop",
    "rating": 4,
    "userNotes": "I really enjoy this grape ice cream!"
  }
]
```

Besides tools like Curl and Postman, you can use the Rating Test Website (https://softserverless-rating.trafficmanager.net/) to test your CreateRating method.

Hint: If using the Rating Test Website make sure to add the https://softserverless-rating.trafficmanager.net/ (https://softserverless-rating.trafficmanager.net/) website url to your function's CORS rules!

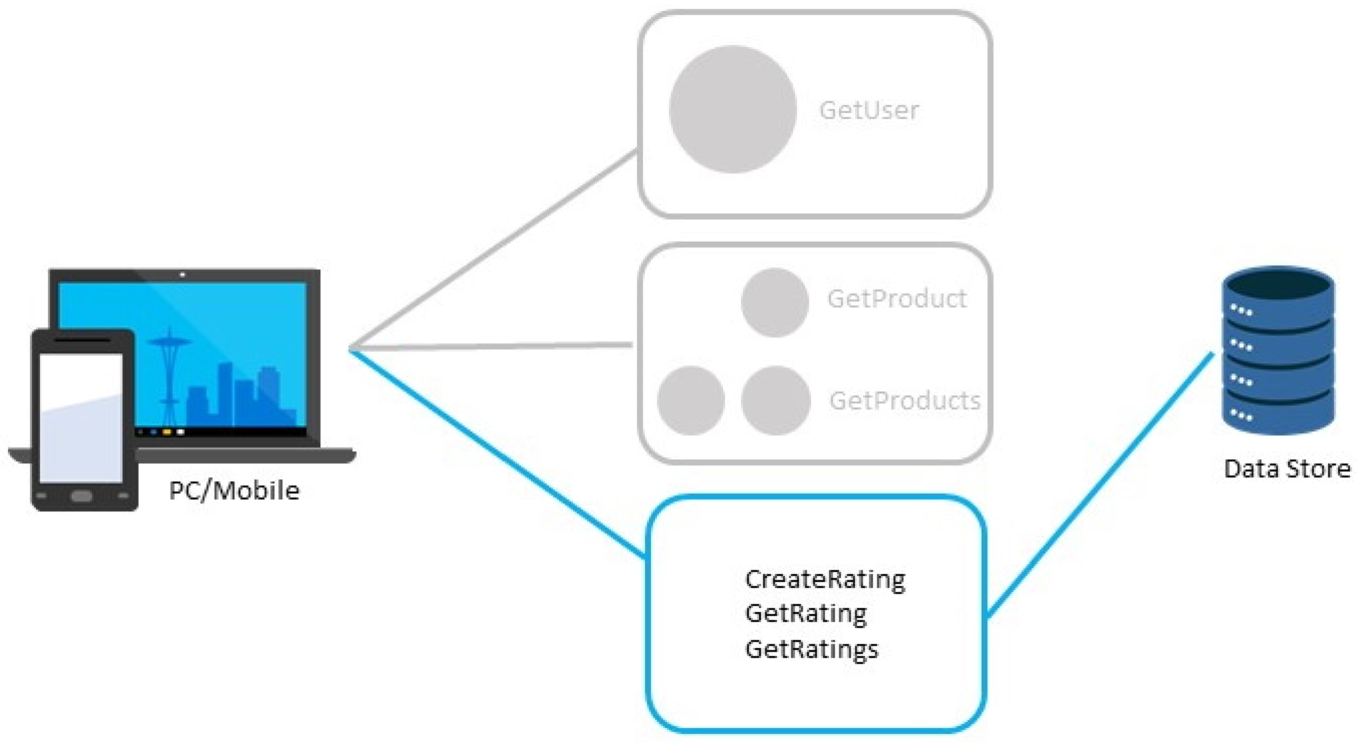
Success Criteria

- Have the rating API functions' code in source control
- Demonstrate a working CI/CD pipeline for your team's coaches
- Provide your coach the three endpoint URLs. The coach should be able to make a call to each of the endpoints and get successful results
- The coach should be able to use `GetRating` and `GetRatings` to query for ratings that the coach creates with `CreateRating`
- The endpoints should return standard HTTP status codes. For example, 404 when items are not found.

References

- What is Continuous Integration? (https://docs.microsoft.com/devops/develop/what-is-continuous-integration)
- What is Continuous Delivery? (https://docs.microsoft.com/devops/deliver/what-is-continuous-delivery)
- Continuous deployment for Azure Functions (https://docs.microsoft.com/azure/azure-functions/functions-continuous-deployment)
- Azure DevOps Pipelines (https://docs.microsoft.com/azure/devops/pipelines/get-started/?view=azure-devops)
- Git Actions Azure Functions Action (https://github.com/marketplace/actions/azure-functions-action)
- Deploy Azure Functions with Jenkins (https://docs.microsoft.com/azure/jenkins/jenkins-azure-functions-deploy)
- Deploy Azure Functions with Octopus Deploy (https://octopus.com/blog/azure-functions)
- Review the Supported Languages in Azure Functions (https://docs.microsoft.com/azure/azure-functions/supported-languages) guide and go into the specific guide for the language you are using
- Azure Functions triggers and bindings concepts (https://docs.microsoft.com/azure/azure-functions/functions-triggers-bindings)
- Introduction to Cosmos DB (https://docs.microsoft.com/azure/cosmos-db/introduction)
- Azure Cosmos DB bindings for Azure Functions (https://docs.microsoft.com/azure/azure-functions/functions-bindings-cosmosdb-v2)
- Azure Functions HTTP and webhook bindings (https://docs.microsoft.com/azure/azure-functions/functions-bindings-http-webhook)
- Store unstructured data using Azure Functions and Azure Cosmos DB (https://docs.microsoft.com/azure/azure-functions/functions-integrate-store-unstructured-data-cosmosdb)
- Getting started with JSON features in Azure SQL Database (https://docs.microsoft.com/azure/sql-database/sql-database-json-features)
- Azure Table storage overview (https://docs.microsoft.com/azure/cosmos-db/table-storage-overview)

Progress Diagram



Ratings API progress diagram