Test Environment

1. Dataset

The dataset chosen for this test was taken from the Kaggle website. The kaggle.com website launched in 2010. Kaggle is becoming a popular website amongst data scientists and machine learning engineers. It allows users to find and publish data sets, explore and build models. As well as to take part in enter competitions to solve data science challenges and problem.

The dataset used for test consists of over 3 million tweets and replies from the biggest brands. The dataset file contains a total of 93 instances and 7 attributes related to the customers. The data was updated one year ago and file size is 167MB. Until now the dataset views are 333000 with 3802 downloads and it is available for free online.

2. Azure setup

Trigger: HTTP
HTTP: version 2.0
Platform: 64 bits
Hosting plan: Consumption plan
Subscription: Pay as you go
Runtime: JavaScript
Operating system: Windows

To access MSSQL Database from an Azure Function using node.js function:

Create a function in the Azure Portal using the "Webhook + API" and choose JavaScript as runtime language. While creating function, choose consumption plan as a hosting plan, which defines how resources are allocated to function app. In this plan, resources are added dynamically as function required. Once the MSSQL database has been created, then configure firewall settings by adding IP address to access the database remotely. After that, create a Node.js application and add dependencies package such as Tedious and MSSQL to run function successfully.

On the other hand, to manage the database, download SQL Server Management Studio 2018. Then, create a table named “twcs” with an identifier field and a name field. In addition, upload csv file. To connect to the database, use the server name (e.g. server.database.windows.net) with SQL Server Authentication using the username and password for access remotely productively.

3. AWS setup

Platform: 64 bits
Trigger: HTTP
HTTP: version 2.0
Hosting plan: pay as you go
Runtime: Node.js 8.10
Operating system: Linux

To access MSSQL Database from an AWS using node.js function:

Use the Amazon Relational Database Service (RDS) to create a Microsoft SQL Server DB Instance in AWS console. For Amazon Lambda, specify the code source in a ZIP file and upload it in the Lambda function. When creating the function, the function needs minimal IAM roles to operate as it is not calling any AWS Services directly. In case of the Node.js application the dependencies are managed with NPM and MSSQL. Using NPM the required libraries for interacting with the services in Amazon are included in each function while uploading zip file into function.

4. Host Environment for performance testing

In this experiment, the performance tests were run with 15MB internet connection on JMeter version 5.1.1. The basic setup for using Apache Jmeter in our scenario is MSSQL database and two Java drivers. The Microsoft JDBC Driver 7.2 for SQL Server and JTDS Driver for SQL Server 1.3.1. They required Java version 8.

Important:

- Make sure the database is available for remote or local access
- Install and choose the right compatibility for JMeter and the Java library and driver versions

Note: Sometimes there are cases when the connection to the database is temporarily unavailable; the IP address needs to be assigned.