IBM Challenge - Devsu Code Jam 2019

Table of Contents

1. Alcance

En un pasado cercano en que los servidores tardaban semanas en aprovisionarse y minutos en iniciarse, estaba de moda alardear de cuánto tiempo podría mantener sus servidores funcionando sin fallas. El hardware era costoso y cuantas más aplicaciones pudieras empacar en un servidor, menores serían los costos de funcionamiento. La alta disponibilidad (HA) se manejó mediante el uso de pares de servidores, y el escalamiento se realizaba de forma vertical, agregando más núcleos a una máquina. Cada servidor era único, precioso.

Los tiempos han cambiado. El hardware está virtualizado. Además, con tecnologías de contenerización, como Containerd/Docker, puede reducir al mínimo el sistema operativo circundante para que pueda iniciar un proceso aislado en segundos, como máximo. En esta nueva infraestructura, típicamente basada en la nube, el escalado puede ser horizontal, agregando y eliminando servidores o contenedores a voluntad, y pagando solo por lo que usa. Con esa libertad, ahora puede implementar "capas delgadas" de lógica de aplicaciones en tiempos de ejecución minimalistas en contenedores independientes y livianos. Ejecutar mucho más que solo un par de contenedores es común y limita los efectos de la caída de un contenedor. Mediante el uso de marcos de orquestación de contenedores, como Kubernetes, se puede introducir o eliminar contenedores rápidamente para escalar cargas de trabajo hacia arriba y hacia abajo. Hoy es posible tenerlo TODO definido por código, haciendo posible nuevos niveles de agilidad y resiliencia, aplicando DevSecOps de punta a punta.

Con el reto IBM queremos incentivar a los concursantes a adquirir los conocimientos que les permitan dominar esta nueva realidad, mediante la construcción de una prueba de concepto sencilla en IBM Cloud.

Quienes se inscriban, recibirán un código promocional que les permitirá contar con la capacidad necesaria para desarrollar, probar y poner en producción su prueba de concepto. **También vamos a proveer links y documentación que sirva de guía en este desarrollo.**

2. Requerimientos Funcionales:

El concursante deberá crear dos microservicios, cada uno con dos versiones distintas desplegadas en producción:

- **Microservicio 1**: Recibe un arreglo no ordenado de valores escalares, numéricos y alfanuméricos, ejecuta un algoritmo de ordenamiento y devuelve el resultado.
    - La versión 1.0 del microservicio realiza su ordenamiento considerando a todos los elementos del arreglo como cadenas tipo string y retorna un JSON con los valores ordenados (ver la especificación del microservicio)

    - La versión 1.1 divide el arreglo en dos partes: 1) numéricos y 2) alfanuméricos, los ordena por separado y retorna un JSON con la siguiente lógica (ver la especificación del microservicio):
        - En un campo retorna los valores numéricos ordenandos.
        - En otro campo retorna los alfanuméricos descartados.

- **Microservicio 2**: Recibe un arreglo no ordenado de valores escalares numéricos y devuelve, un objeto JSON con los campos (ignora cualquier valor no-numérico recibido):
    - count (cuenta total de elementos del arreglo)
    - average (promedio de los valores entregados)
    - mean (media aritmética de los valores)
    - mode (moda estadística)
    - max (valor máximo del arreglo)
    - min (valor mínimo del arreglo)
    - stdev (desviación estándar)

Los microservicios deberán desplegarse en contenedores independientes y deben requerir autenticación básica para ser llamados.

Los usuarios y contraseñas que se usarán son:

- **Para los robots**:

User: ibm_test_robot_1
Password: ibm_test_1.123#

User: ibm_test_robot_2
Password: ibm_test_2.123#

User: ibm_test_robot_3
Password: ibm_test_3.123#

El despliegue de la solución deberá incluir reglas de ruteo para dirigir los requerimientos recibidos de acuerdo con las siguientes reglas:

- **Microservicio 1**: el 80% de los requerimientos los atenderá la versión 1.0 y el 20% la versión 1.1
- **Microservicio 2**: los requerimientos del usuario `ibm_test_robot_1` serán atendidos por la versión 1.0. Los requerimientos de los demás usuarios serán atendidos por la versión 1.1

3. Especificación Técnica de los Servicios

Modelo de Evaluación

Los concursantes deberán proveer el endpoint para los servicios construidos. Este endpoint deberá estar hosteado en la nube de IBM (*.appdomain.cloud). Una vez que el concursante haya completado el reto deberá subir su respuesta en el formulario provisto en devsucodejam.com/ibm. Al ingresar sus datos, se ejecutará pruebas automatizadas y evaluará los resultados, confirmado que cumplen con los esquemas descritos en la especificación técnica y que la distribución de carga no tuvo una desviación mayor al 3% en ninguno de los casos.

En caso de que no cumpla con las especificaciones, se mostrará un mensaje de error, y el usuario podrá hacer las correcciones necesarias y volver a subir su información.

El usuario concursante que complete exitosamente la prueba recibirá un correo electrónico con un número de boleto para participar en el sorteo. El sorteo se realizará una vez que haya culminado la final del Devsu Code Jam, el 30 de noviembre de 2019, en la UDLA, Campus Queri, en Quito.

Documentación Microservicio 1 version 1.0:

Version: 1.0.0
BasePath:/ibmchallengemic1

Access

1. Basic Auth

Methods:

- POST /element_sorter

POST /element_sorter

Receives a set of values and returns them back sorted

Consumes

This API call consumes the following media types via the Content-Type request header:

- application/json

Request body

Example data

Content-Type: application/json

```
{
   "elements" : [1, "a", 12.4, "text"]
}
```

Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

Responses:

200

200 OK

When request is success:
status: “success”
message: “ok”

400

400 Bad Request

When request has wrong values or generate any kind of error:
status: “error”
message: error message

Example ok data

Content-Type: application/json

```
{
  "status": "success",
  "message": "ok",
  "data": {
    "sorted": ["1", "12.4", "a", "text"]
  }
}
```

Example data error message:

Content-Type: application/json

```
{
  "status": "error",
   "message": "The method add(Object, String) in the type ArrayList<String> is not applicable for the arguments (Object)",
}
```

Documentación Microservicio 1 version 1.1:

Version: 1.1.0
BasePath:/ibmchallengemic1

Access

1. Basic Auth

Methods

- POST /element_sorter

POST /element_sorter

Receives a set of values and returns them back sorted

Consumes

This API call consumes the following media types via the Content-Type request header:

- application/json

Request body

Example data
Content-Type: application/json

```
{
   "elements" : [25,4,”","Banana", "Orange", "Apple", "Mango",1]
}
```

Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

Responses:

200
200 OK
When request is success:
status: “success”
message: “ok”

400
400 Bad Request
When request has wrong values or generate any kind of error:

status: “error”
message: error message

Example data
Content-Type: application/json

```
{
 "status": "success",
 "message": "41",
 "data": {
  "numbers": [1, 4, 25],
  “others”:["Apple", "Banana", "Mango", "Orange"]
 }
}
```

Example data error message:
Content-Type: application/json

```
{
 "status": "error",
  "message": "The method add(Object, String) in the type ArrayList<String> is not applicable for the arguments (Object)",
}
```

Documentación Microservicio 2:

Version: 1.0.0
BasePath:/ibmchallengemic2

Access

1. Basic Auth

Methods

- POST /statistics

POST /statistics

Receives a set of values and returns statistics data

Consumes

This API call consumes the following media types via the Content-Type request header:

- application/json

Request body

Example data

Content-Type: application/json

```
{
   "elements" : [1,4,5,7,8,4,3,0]
}
```

Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

Responses:

200

200 OK

When request is success:
status: “success”
message: “ok”

400

400 Bad Request

When request has wrong values or generate any kind of error:
status: “error”
message: error message

Example data

Content-Type: application/json

```
{
  "status": "success",
  "message": "ok",
  "data" : {
    "mode" : 4.0,
    "average" : 4.0,
    "min" : 0.0,
    "max" : 8.0,
    "mean" : 4.0,
    "count" : 8,
    "stdev" : 2.725540575
  },
}
```

Example data error message:

Content-Type: application/json

```
{
```

```
  "status": "error",
  "message": "The method add(Object, String) in the type ArrayList<String> is not applicable for the arguments (Object)",
}
```