Data Ingestion Platform

Prerequisites and Conditions

You can use Java/Scala/Kotlin and ecosystem for this coding challenge. We recommend you to use Quarkus or Reactive Spring for Java or Kotlin implementations.

- Use only Kafka and Cassandra
- The ingested data has to be integrated in a dedicated domain model
- The ingested data has to be buffered before enrichment and persistence
- The resulting artifacts have to be deployable as Docker container
  - Either Docker-compose or deployment on Kubernetes expected
- The API has to be available for the assessment team for testing
- There have to be unit tests available for the specified user stories
- Data Source: http://api.metro.net/agencies/lametro/vehicles/
- Map projection approach: Bing Maps Tile System (https://msdn.microsoft.com/en-us/library/bb259689.aspx)

Use Cases to be implemented

1. As a User, I want to have data from the given data source to be available as hot and cold data.
2. As a User, I want to be able to aggregate the collected data and create higher level events from it. The following aggregation has to be implemented:
   a. Amount of vehicles per tile
3. As a User, I want to request a list of available vehicles from the service API.
   a. Endpoint: http://<endpoint>/api/vehicles/list
   b. Request Type: GET
   c. Responses:
      i. Response: 200 – List[Vehicle]
      ii. Response: 204 – Empty response
      iii. Response: 500 - Error case with error message
   d. Response Content-Type:
      i. application/json
4. As a User, I want to request the last position of a vehicle from the service API.
   a. Endpoint: http://<endpoint>/api/vehicles/vehicle/<vehicleId>/lastPosition
   b. Request Type: GET
   c. URL Parameter:
      i. Vehicle ID
   d. Responses:
      i. Response: 200 - Trajectory of last position
      ii. Response: 404
      iii. Response: 500 - Error case with error message
   e. Response Content-Type:
      i. application/json
5. As a User, I want to request map tiles containing hot and aggregated data from the service API. Following requests have to be available:
   a. Request to determine which tiles are filled with aggregated data / vehicles
      i. Endpoint: http://<endpoint>/api/tiles/filled
      ii. Request Type: GET
      iii. Responses:
         1. Response: 200 - List[unique tile identifier]

        2. Response: 204 – Empty response
        3. Response: 500 - Error case with error message
    iv. Response Content-Type:
        1. application/json

b. Request to query a specific tile and get the vehicles currently located in the tile area
    i. Endpoint: http://<endpoint>/api/tiles/tile/<tile_id>/availableVehicles
    ii. Request Type: GET
    iii. URL Parameter:
        1. Tile identifier
    iv. Responses:
        1. Response: 200 - List[Vehicles]
        2. Response: 204 – Empty response
        3. Response: 500 - Error case with error message
    v. Response Content-Type:
        1. application/json

c. Request to query a specific bounding box of tiles to get the data from use case nr. 2
    i. Endpoint: http://<endpoint>/api/tiles/usecase/vehicleCount
    ii. Request Type: GET
    iii. Request Parameters:
        1. List of tiles
    iv. Responses:
        1. Response: 200 - Map[unique tile identifier, Int]
        2. Response: 401 - Bad request
        3. Response: 500 - Error case with error message
    v. Response Content-Type:
        1. application/json