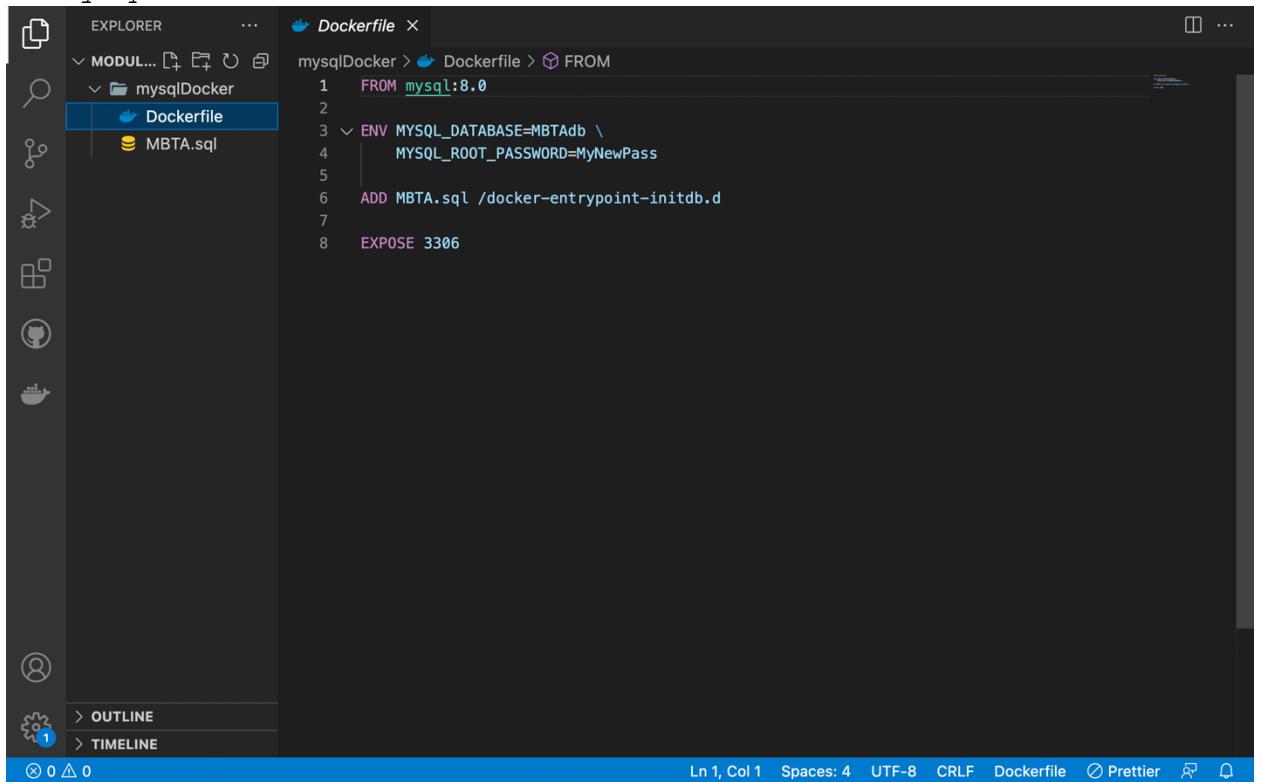


1. Provide a screenshot to show that you have successfully created the MBTANetwork network.

```
~/Documents/dev
> docker network create MBTANetwork
6546d11a1f115051f4630df567e35e0dff327c5880f723ccb3905a47a431e259

~/Documents/dev
>
```

2. Provide a screenshot to show that you have successfully opened the mysqlDocker folder.

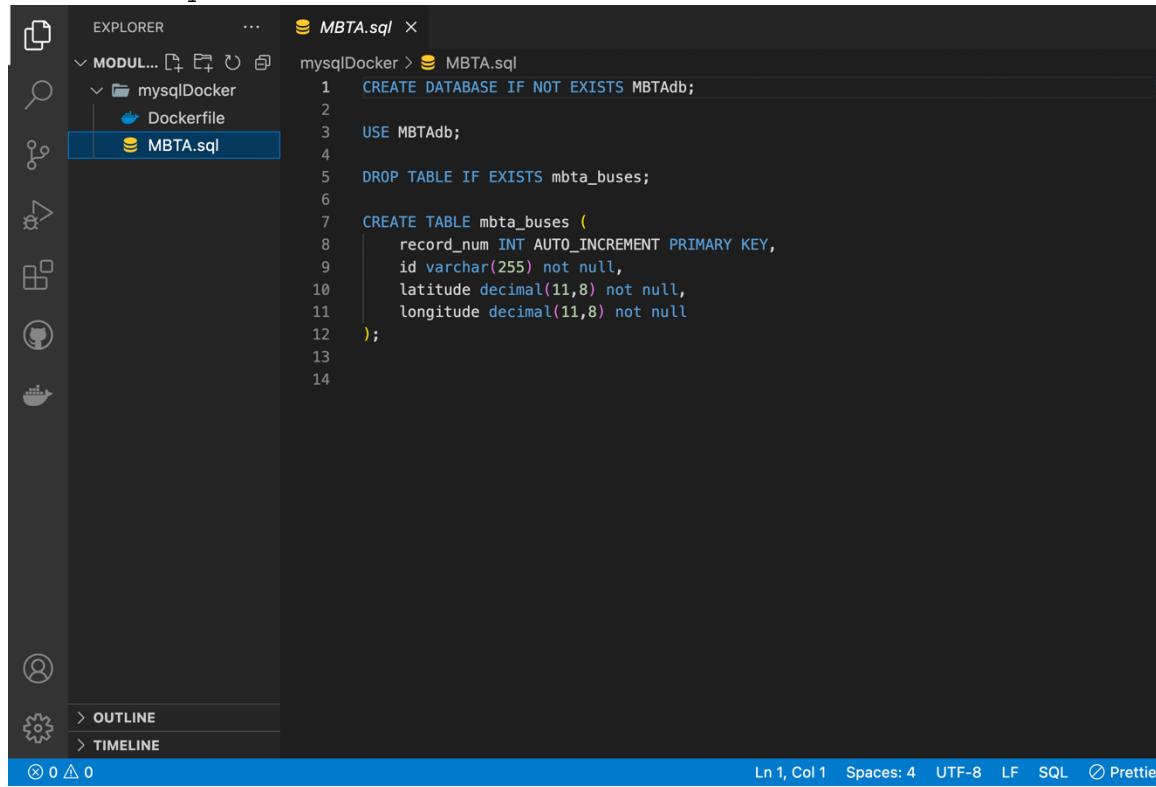


The screenshot shows the Visual Studio Code interface with the following details:

- EXPLORER** sidebar: Shows a folder named "MODUL..." containing a "mysqlDocker" folder. Inside "mysqlDocker", there is a "Dockerfile" which is currently selected and highlighted with a blue background.
- Dockerfile** editor tab: Displays the contents of the Dockerfile:

```
mysqlDocker > Dockerfile > FROM
1  FROM mysql:8.0
2
3  ENV MYSQL_DATABASE=MBTAdb \
4      MYSQL_ROOT_PASSWORD=MyNewPass
5
6  ADD MBTA.sql /docker-entrypoint-initdb.d
7
8  EXPOSE 3306
```
- Bottom Status Bar:** Shows "Ln 1, Col 1" and "Spaces: 4" and "CRLF". There are also icons for "Dockerfile", "Prettier", and "Format" (represented by a double arrow).

1. Provide a screenshot to show that you have successfully opened the MBTA.sql file.



The screenshot shows a code editor interface with a dark theme. On the left is a sidebar with icons for Explorer, Search, Issues, Pull Requests, and Docker. The main area shows a file named 'MBTA.sql' with the following SQL code:

```
1 CREATE DATABASE IF NOT EXISTS MBTAb;
2
3 USE MBTAb;
4
5 DROP TABLE IF EXISTS mbta_buses;
6
7 CREATE TABLE mbta_buses (
8     record_num INT AUTO_INCREMENT PRIMARY KEY,
9     id varchar(255) not null,
10    latitude decimal(11,8) not null,
11    longitude decimal(11,8) not null
12 );
13
14
```

At the bottom right, there are status indicators: Ln 1, Col 1, Spaces: 4, UTF-8, LF, SQL, and a Prettier icon.

2. For this step, you will provide two screenshots. The first screenshot should show that you have successfully run the provided code in a Jupyter Notebook. The second screenshot should show that you have successfully added at least five additional fields to the mbta_buses table.

```

mbtaURL = "https://api-v3.mbta.com/vehicles?filter[route]=1&include=trip"

import urllib.request, json
with urllib.request.urlopen(mbtaURL) as url:
    data = json.loads(url.read().decode())

    with open('data.json', 'w') as outfile:
        json.dump(data, outfile)

    with open('data.txt', 'w') as outfile:
        json.dump(json.dumps(data, indent=4, sort_keys=True), outfile)

print(json.dumps(data, indent=4, sort_keys=True))

```

[1] ✓ 0.4s

... Output exceeds the `size_limit`. Open the full output data [in a text editor](#)

{

"data": [

{

"attributes": {

"bearing": 139,

"current_status": "IN_TRANSIT_TO",

"current_stop_sequence": 17,

"direction_id": 1,

"label": "1898",

"latitude": 42.33664506,

"longitude": -71.077211649,

"occupancy_status": "MANY_SEATS_AVAILABLE",

"speed": null,

"updated_at": "2022-08-07T18:30:14-04:00"

},

}

EXPLORER

MODULE 16 - PROJECT

- .ipynb_checkpoints
- DebeziumCDC
- java-quick-start
- Module16ProjectFlask
 - templates
 - client.py
 - MBTAApiClient.py
 - mysqlDb.py
 - server.py
 - timer.py
 - mysqlDocker
 - Dockerfile
 - MBTA.sql
 - screenshots
 - data.json
 - data.txt
 - mbta.csv
- Module16 - Template.ipynb
- module16.ipynb

Jupyter Server: Local Cell 1 of 1 Pre

EXPLORER

MODULE 16 - PROJECT

- .ipynb_checkpoints
- DebeziumCDC
- java-quick-start
- Module16ProjectFlask
 - templates
 - client.py
 - MBTAApiClient.py
 - mysqlDb.py
 - server.py
 - timer.py
 - mysqlDocker
 - Dockerfile
 - MBTA.sql
 - screenshots
 - data.json
 - data.txt
 - mbta.csv
- Module16 - Template.ipynb
- module16.ipynb

Module16 - Template.ipynb

MySQLDocker > MBTA.sql

```

1 CREATE DATABASE IF NOT EXISTS MBTAdb;
2
3 USE MBTAdb;
4
5 DROP TABLE IF EXISTS mbta_buses;
6
7 CREATE TABLE mbta_buses (
8     record_num INT AUTO_INCREMENT PRIMARY KEY,
9     route_number INT default 1,
10    id varchar(255) not null,
11    bearing INT not null,
12    current_status varchar(255) not null,
13    current_stop_sequence INT null,
14    direction_id INT not null,
15    label varchar(255) default null,
16    latitude decimal(11,8) not null,
17    longitude decimal(11,8) not null,
18    occupancy_status varchar(255) null,
19    speed int default null,
20    updated_at datetime null
21 );
22
23

```

- Provide a screenshot to show that you have successfully created the mysqlmbtamastering Docker image.

```

~/Documents/dev/data_engineering_bootcamp/Module 16 - Project/mysqlDocker
> docker build -t mysqlmbtmasterimg .
[+] Building 12.3s (8/8) FINISHED
=> [internal] load build definition from Dockerfile
=> transferring dockerfile: 266B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/mysql:8.0
=> [auth] library/mysql:pull token for registry-1.docker.io
=> [1/2] FROM docker.io/library/mysql:8.0@sha256:ce2ae3bd3e9f001435c4671cf073d1d5ae55d138b16927268474fc54ba09ed79
=> => resolving docker.io/library/mysql:8.0@sha256:ce2ae3bd3e9f001435c4671cf073d1d5ae55d138b16927268474fc54ba09ed79
=> sha256:fdb9779d4678f8577c74505cc31bd42788a9ff1eba563340b8ef29946b3e55 857.15kB / 857.15kB
=> sha256:ce2ae3bd3e9f001435c4671cf073d1d5ae55d138b16927268474fc54ba09ed79 549B / 549B
=> sha256:d5063370fd166aa88c9fc832d4401b10b47d7bd097d973d1d5ae55d138b16927268474fc54ba09ed79 7.48kB / 7.48kB
=> sha256:ecf56004177eb6ca162c88de29e84bc4ba3d2e7efd3703df9ece51b89003d52 38.02MB / 38.02MB
=> sha256:914ec0d396561e9ca92d8fc1b5f618f3c261a3c202822c94e041ceeeda4bed7a 2.62kB / 2.62kB
=> sha256:1091d9f012e1e44cf00d872494ff008feb11a824a940f5ccaa0cd0f043e951 888B / 888B
=> sha256:d6278448e2e194f0e28e7d2f29495296d594f7bd2e63d0306037e5b4f5125eb 4.40MB / 4.40MB
=> sha256:6050832741b8d8e4dfba20e0d31c88f56b9b645c97752c9bbec708740c3e6e15 2.61kB / 2.61kB
=> sha256:a0be9d1181fe5f7d6c39819a56c680e5e010a7b808b96e00eabdeacd66ee6c1 335B / 335B
=> sha256:cbda0fd3eb50f6880d4e27c2ba02a47193ba4e1e1959b4647e5e794b0872e981c5b 53.78MB / 53.78MB
=> sha256:a74c1ad0f09fc99fb019bcfb0b1b57f4f7fcf7e968e6ece353a06d583cab8ed 315B / 315B
=> sha256:ad2b7db10f4ea0cccef72231d395d9920b81dd4bba86be2a8994b8ee9161ec92 44.12MB / 44.12MB
=> sha256:e27624d90cadf384c8f6b5480da51cf29d2e80e488fb5a8f8336a5fd831fbfd 5.16kB / 5.16kB
=> extracting sha256:ecf56004177eb6ca162c88de29e84bc4ba3d2e7efd3703df9ece51b89003d52
=> sha256:cel07dc947b5a031fd79995e9506e109c3148037b67fe726101e08e122895e 121B / 121B
=> extracting sha256:1091d9f012e1e44cf00d872494ff008feb11a824a940f5ccaa0cd0f043e951
=> extracting sha256:fabd9779d4678f8577c74505cc31bd42788a9ff1eba563340b8ef29946b3e55
=> extracting sha256:d6278448e2e194f0e28e7d2f29495296d594f7bd2e63d0306037e5b4f5125eb
=> extracting sha256:6050832741b8d8e4dfba20e0d31c88f56b9b645c97752c9bbec708740c3e6e15
=> extracting sha256:a0be9d1181fe5f7d6c39819a56c680e5e010a7b808b96e00eabdeacd66ee6c1
=> extracting sha256:cbda0fd3eb50f6880d4e27c2ba02a47193ba4e1e1959b4647e5e794b0872e981c5b
=> extracting sha256:a74c1ad0f09fc99fb019bcfb0b1b57f4f7fcf7e968e6ece353a06d583cab8ed
=> extracting sha256:ad2b7db10f4ea0cccef72231d395d9920b81dd4bba86be2a8994b8ee9161ec92
=> extracting sha256:e27624d90cadf384c8f6b5480da51cf29d2e80e488fb5a8f8336a5fd831fbfd

```

Docker Desktop Upgrade plan

Containers Images Volumes Dev Environments LOCAL REMOTE REPOSITORIES

Search In use only

NAME	TAG	IMAGE ID	CREATED	SIZE
mysqlmbtmasterimg	latest	fc9e2b13176a	less than a minute ago	490.32 MB

- Provide a screenshot to show that you have successfully created the mysqlserver Docker container.

```

~/Documents/dev/data_engineering_bootcamp/Module 16 - Project/mysqlDocker 13s
> docker run --rm --name mysqlserver -p 3306:3306 --network MBTANetwork -d mysqlmbtmasterimg
b27116fecc98a88a1973a9b9f4fa78e27de953bcc88dd21b95f92a3faa2bde6

~/Documents/dev/data_engineering_bootcamp/Module 16 - Project/mysqlDocker

```

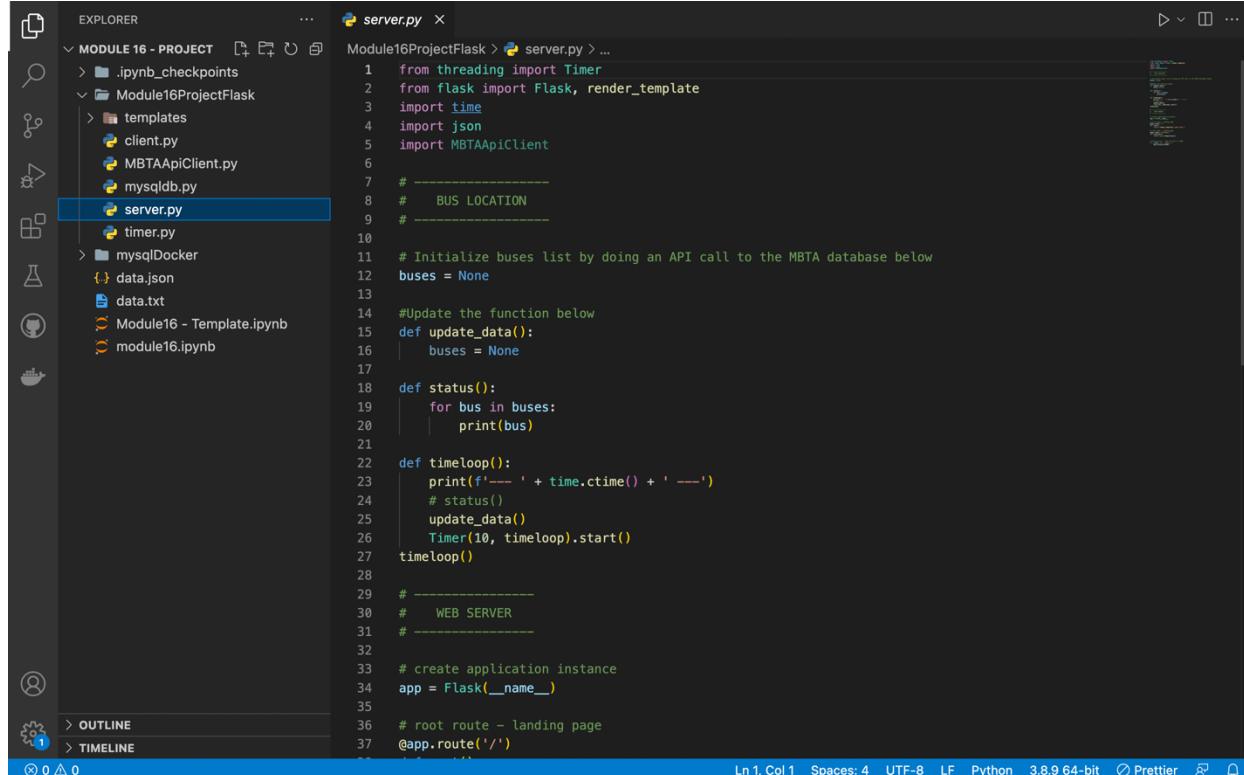
3. Provide a screenshot to show that you have successfully created the `some-mongo` Docker container.

```
~/Documents/dev/data_engineering_bootcamp/Module 16 - Project/mysqlDocker
> docker run --name some-mongo --network MBTANetwork -p 27017:27017 -d mongo
Unable to find image 'mongo:latest' locally
latest: Pulling from library/mongo
a749a280e3e9: Pull complete
70f92e752512: Pull complete
5fdda1347325: Pull complete
c5e02463a0f4: Pull complete
ff9bc5b0968e: Pull complete
33f0d07ad9ca: Pull complete
e5c25f5ea983: Pull complete
53f9cf44c1e9: Pull complete
fbcd4a874cd21: Pull complete
Digest: sha256:4200c3073389d5b303070e53ff8f5e4472efb534340d28599458ccc24f378025
Status: Downloaded newer image for mongo:latest
6cd8a288fe292d4a596e6164e068478c2d471ad049f8e2c6f79a547fb63e8d12

~/Documents/dev/data_engineering_bootcamp/Module 16 - Project/mysqlDocker 18s
```

- 4.

- a. Provide a screenshot to show that you have successfully opened the `Module16ProjectFlask.zip` folder in VS Code.



- b. Provide a screenshot to show that you have successfully modified the `mysqlDb.py` file.

```

import mysql.connector
mydb = mysql.connector.connect(
  host="localhost",
  user="root",
  password="MyNewPass",
  database="MBTAdb"
)
mycursor = mydb.cursor()
sql = "insert into mbta_buses (route_number, id, bearing, current_status, current_stop_sequence, direction_id, label, latitude, longitude, occupancy_status, speed, updated_at) values (%s, %s, %s)"
for mbtaDict in mbtaList:
    val = (mbtaDict['route_number'], mbtaDict['id'], mbtaDict['bearing'], mbtaDict['current_status'], mbtaDict['current_stop_sequence'], mbtaDict['direction_id'], mbtaDict['label'], mbtaDict['latitude'], mbtaDict['longitude'], mbtaDict['occupancy_status'], mbtaDict['speed'], mbtaDict['updated_at'])
    mycursor.execute(sql, val)
mydb.commit()

```

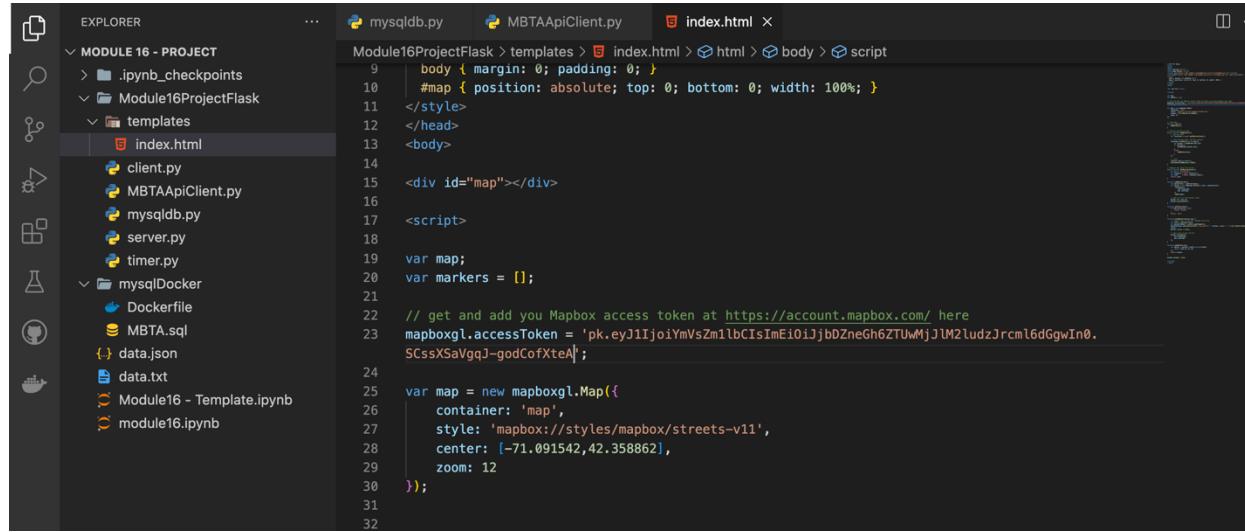
- c. Provide a screenshot to show that you have successfully modified the `MBTAApiClient.py` file.

```

def callMBTAApi():
    mbtaDictList = []
    mbtaUrl = 'https://api-v3.mbta.com/vehicles?filter[route]=1&include=trip'
    with urllib.request.urlopen(mbtaUrl) as url:
        data = json.loads(url.read().decode())
    for bus in data['data']:
        busDict = dict()
        busDict['route_number'] = 1
        busDict['id'] = bus['id']
        busDict['bearing'] = bus['attributes']['bearing']
        busDict['current_status'] = bus['attributes']['current_status']
        busDict['current_stop_sequence'] = bus['attributes']['current_stop_sequence']
        busDict['direction_id'] = bus['attributes']['direction_id']
        busDict['label'] = bus['attributes']['label']
        busDict['latitude'] = bus['attributes']['latitude']
        busDict['longitude'] = bus['attributes']['longitude']
        busDict['occupancy_status'] = bus['attributes']['occupancy_status']
        busDict['speed'] = bus['attributes']['speed']
        busDict['updated_at'] = bus['attributes']['updated_at']
        mbtaDictList.append(busDict)
    mysqlDb.insertMBTAREcord(mbtaDictList)
    return mbtaDictList

```

- d. Provide a screenshot to show that you have successfully added your Mapbox access token in the index.html file.



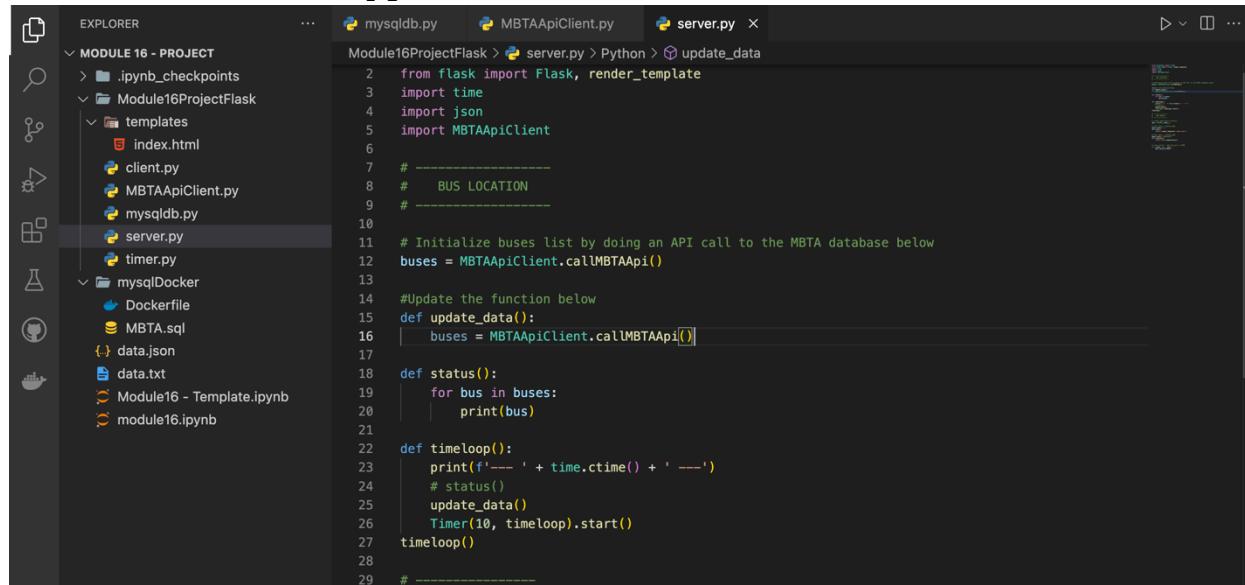
```

EXPLORER          ...      mysqlDb.py   MBTAApiClient.py   index.html x
MODULE 16 - PROJECT
> .ipynb_checkpoints
< Module16ProjectFlask
  < templates
    < index.html
      client.py
      MBTAApiClient.py
      mysqlDb.py
      server.py
      timer.py
    < mysqlDocker
      Dockerfile
      MBTA.sql
    { data.json
    < Module16 - Template.ipynb
    < module16.ipynb

index.html
Module16ProjectFlask > templates > index.html > html > body > script
9 |   body { margin: 0; padding: 0; }
10|   #map { position: absolute; top: 0; bottom: 0; width: 100%; }
11|   </style>
12|   </head>
13|   <body>
14|
15|   <div id="map"></div>
16|
17|   <script>
18|
19|     var map;
20|     var markers = [];
21|
22|     // get and add your Mapbox access token at https://account.mapbox.com/ here
23|     mapboxgl.accessToken = 'pk.eyJ1IjoiYmVsZmlbCIsImEiOiJjbDZneGh6ZTUwMjJM2ludzJrcml6dGgwIn0.
24|     SCssXSaVgqJ-godCofXteA';
25|
26|     var map = new mapboxgl.Map({
27|       container: 'map',
28|       style: 'mapbox://styles/mapbox/streets-v11',
29|       center: [-71.091542, 42.358862],
30|       zoom: 12
31|     });
32|

```

- e. Provide a screenshot to show that you have successfully initialized the buses list in the server.py file.



```

EXPLORER          ...      mysqlDb.py   MBTAApiClient.py   server.py x
MODULE 16 - PROJECT
> .ipynb_checkpoints
< Module16ProjectFlask
  < templates
    < index.html
    client.py
    MBTAApiClient.py
    mysqlDb.py
    server.py
    timer.py
  < mysqlDocker
    Dockerfile
    MBTA.sql
  { data.json
  < Module16 - Template.ipynb
  < module16.ipynb

server.py
Module16ProjectFlask > server.py > Python > update_data
2 from flask import Flask, render_template
3 import time
4 import json
5 import MBTAApiClient
6
7 # -----
8 #   BUS LOCATION
9 # -----
10
11 # Initialize buses list by doing an API call to the MBTA database below
12 buses = MBTAApiClient.callMBTAApi()
13
14 #Update the function below
15 def update_data():
16     buses = MBTAApiClient.callMBTAApi()
17
18 def status():
19     for bus in buses:
20         print(bus)
21
22 def timeloop():
23     print('--- ' + time.ctime() + ' ---')
24     # status()
25     update_data()
26     Timer(10, timeloop).start()
27 timeloop()
28
29 # -----

```

- f. Provide a screenshot to show that you have successfully run the server.py file in VS Code.

The screenshot shows the VS Code interface with the server.py file open in the editor. The terminal tab is active, displaying the output of the application's startup. The output shows the application is running on port 3000, serving a Flask app named 'server'. It also shows several log entries indicating requests for favicon.ico and /location, and a warning about using it in production.

```

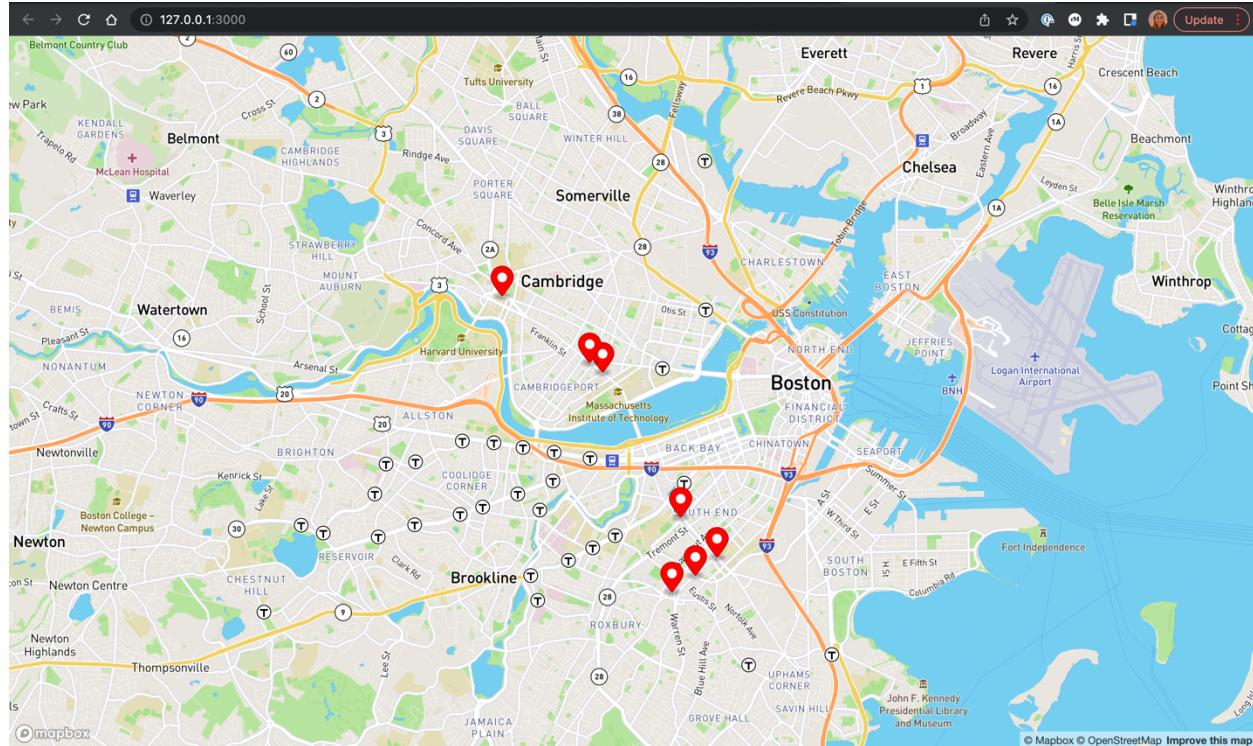
Module16ProjectFlask > 🏷 server.py > ...
1  from flask import Flask, render_template
2  import time
3  import json
4  import MBTAApiClient
5
6  # -----
7  #   BUS LOCATION
8  #
9  #
10
11 # Initialize buses list by doing an API call to the MBTA database below
12 buses = MBTAApiClient.callMBTAApi()
13
14 #Update the function below
15 def update_data():
16     buses = MBTAApiClient.callMBTAApi()
17
18 def status():
19     for bus in buses:
20         print(bus)
21
22 def timeloop():
23     print(f'--- {time.ctime()} ---')
24     # status()
25     update_data()

er.py
--- Sun Aug  7 16:12:55 2022 ---
* Serving Flask app 'server' (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:3000 (Press CTRL+C to quit)
127.0.0.1 - - [07/Aug/2022 16:13:04] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [07/Aug/2022 16:13:05] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [07/Aug/2022 16:13:05] "GET /location HTTP/1.1" 200 -
--- Sun Aug  7 16:13:06 2022 ---

Ln 4, Col 12  Spaces: 4  UTF-8  LF  Python  3.8.9 64-bit  ⚙ Prettier  🔍

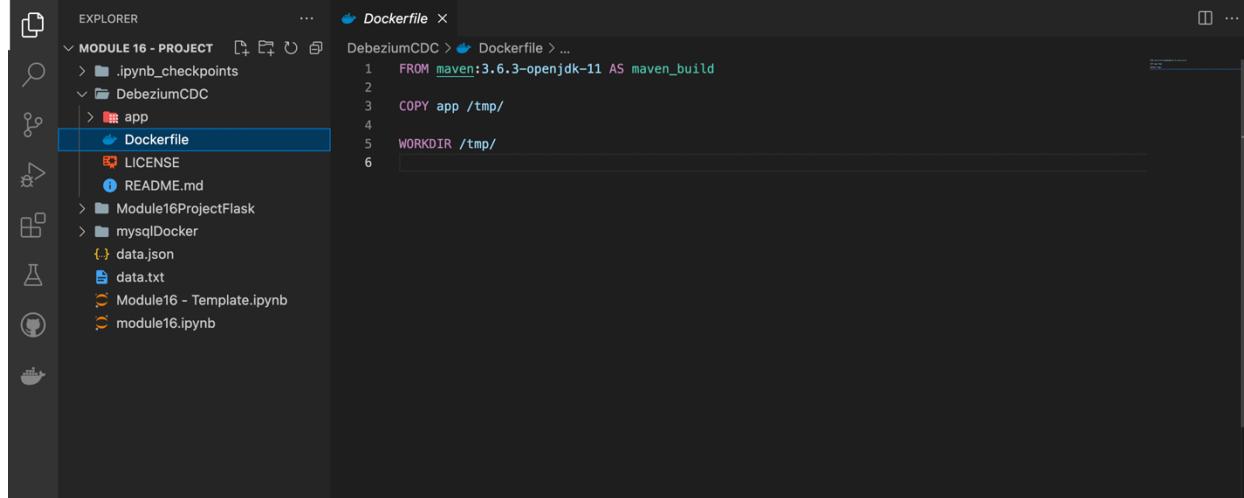
```

- g. Provide a screenshot to show that you have successfully navigated to localhost:3000.



5.

- a. Provide a screenshot to show that you have successfully opened the DebeziumCDC.zip folder in VS Code.



- b. Provide a screenshot to show that you have successfully created the debeziummodule16 Docker image.

```

~/Documents/dev/data_engineering_bootcamp/Module 16 - Project/DebeziumCDC
> docker build -t debeziummodule16 .
[+] Building 19.1s (9/9) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 200B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/maven:3.6.3-openjdk-11
=> [auth] library/maven:pull token for registry-1.docker.io
=> [internal] load build context
=> => transferring context: 28.04kB
=> [1/3] FROM docker.io/library/maven:3.6.3-openjdk-11@sha256:1d29ccf46ef2a5e64f7de3d79a63f9bcff4dc56be0ae3daed5ca5542b38aa2d
=> => resolve docker.io/library/maven:3.6.3-openjdk-11@sha256:1d29ccf46ef2a5e64f7de3d79a63f9bcff4dc56be0ae3daed5ca5542b38aa2d
=> sha256:6d2132ee8438b67166591ec919cf2f5fd90180a3975ec663938e5ecc6441eb26 2.42kB
=> sha256:de7948a941defda38829ce0d898e315dbc6226a3b9f5b39103c85deab3fe5fd0 8.92kB
=> sha256:299f3631f6b52be065a7342d0a046978d55cb0d15c57fae22f4ca24effc295a 9.98MB / 9.98MB
=> sha256:1d29ccf46ef2a5e64f7de3d79a63f9bcff4dc56be0ae3daed5ca5542b38aa2d 549B / 549B
=> sha256:ef28e7e77ecbd3b3b426832bc12e8f5e629959683767466e9bac149c3286e126 49.23MB / 49.23MB
=> sha256:344dd2d9a9cf41c137b0db41df255f95fb812a3771a10ee2ab5a805047c62c4 7.69MB / 7.69MB
=> sha256:ba70c372ae296f23e908bf1e0d9f4c0c81a8a6d7fc48c0e2db16035bb9b7a54 52.17MB / 52.17MB
=> sha256:0702beadc5c899e2ff6387365daa85f4bd9d0b3bc20b0a88ac2688e033334c2 5.28MB / 5.28MB
=> sha256:6a2cea38573a215a9e8b119f049b0477440126e110b1b0c7f3a03d01e65cab 211B / 211B
=> sha256:636c44b951395e5fa7b30bde4e93a103baac16d63bb355d7623266827fd027 200.40MB / 200.40MB
=> sha256:48a10246539ac47e95b1673d7dad9c58b091e7bf312dca1b16f576b1d795 9.58MB / 9.58MB
=> extracting sha256:ef28e7e77ecbd3b3b426832bc12e8f5e629959683767466e9bac149c3286e126
=> sha256:e30dbc7f25bbf70a0c6909a7846e581e9d3fb1b2b5ca2b6ca78fb95ecf38 855B / 855B
=> sha256:9d50241f26f28ad254cd9ae8b3c6f0085aa95a6d26a4d69e0123d5c0e72d4b352 359B / 359B
=> extracting sha256:344dd2d9a9cf41c137b0db41df255f95fb812a3771a10ee2ab5a805047c62c4
=> extracting sha256:299f3631f6b52be065a7342d0a046978d55cb0d15c57fae22f4ca24effc295a
=> extracting sha256:ba70c372ae296f23e908bf1e0d9f4c0c81a8a6d7fc48c0e2db16035bb9b7a54
=> extracting sha256:0702beadc5c899e2ff6387365daa85f4bd9d0b3bc20b0a88ac2688e033334c2
=> extracting sha256:6a2cea38573a215a9e8b119f049b0477440126e110b1b0c7f3a03d01e65cab
=> extracting sha256:636c44b951395e5fa7b30bde4e93a103baac16d63bb355d7623266827fd027
=> extracting sha256:48a10246539ac47e95b1673d7dad9c58b091e7bf312dca1b16f576b1d795

```

- c. Provide a screenshot to show that you have successfully created the Docker container and associated it with the MBTANetwork network.

```

~/Documents/dev/data_engineering_bootcamp/Module 16 - Project/DebeziumCDC
> docker run -it --rm --name debeziumserver --network MBTANetwork debeziummodule16 bash
root@cea7f132d3ed:/tmp#

```

- d. Provide a screenshot to show that you have successfully installed the nano text editor in your shell.

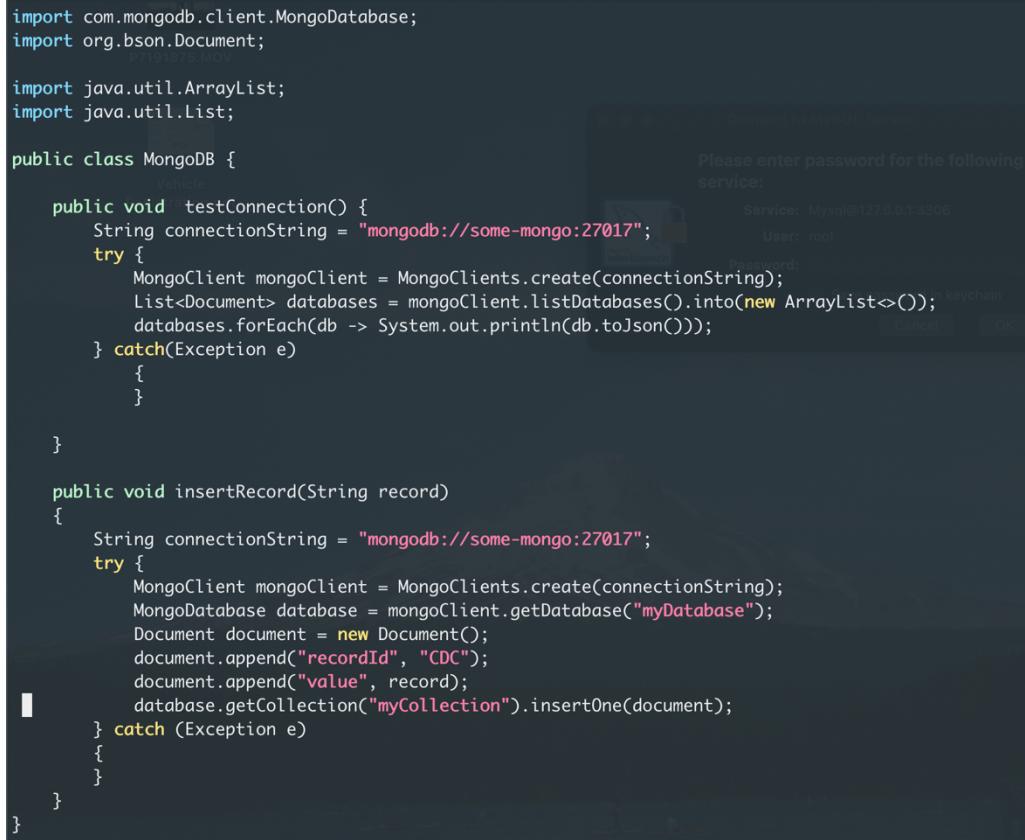
```
E: Invalid operation nano
root@cea7f132d3ed:/tmp# apt-get install nano
Reading package lists... Done
Building dependency tree
Reading state information... Done
Suggested packages:
  spell
The following NEW packages will be installed:
  nano
0 upgraded, 1 newly installed, 0 to remove and 47 not upgraded.
Need to get 541 kB of archives.
After this operation, 2290 kB of additional disk space will be used.
Get:1 http://deb.debian.org/debian buster/main arm64 nano arm64 3.2-3 [541 kB]
Fetched 541 kB in 0s (4199 kB/s)
debconf: delaying package configuration, since apt-utils is not installed
Selecting previously unselected package nano.
(Reading database ... 12567 files and directories currently installed.)
Preparing to unpack .../archives/nano_3.2-3_arm64.deb ...
Unpacking nano (3.2-3) ...
Setting up nano (3.2-3) ...
update-alternatives: using /bin/nano to provide /usr/bin/editor (editor) in auto mode
update-alternatives: using /bin/nano to provide /usr/bin/pico (pico) in auto mode
root@cea7f132d3ed:/tmp#
```

- e. Provide a screenshot to show that you have successfully modified the MongoDB.java class.

```
import com.mongodb.client.MongoDatabase;
import org.bson.Document;
import java.util.ArrayList;
import java.util.List;

public class MongoDB {
    public void testConnection() {
        String connectionString = "mongodb://some-mongo:27017";
        try {
            MongoClient mongoClient = MongoClients.create(connectionString);
            List<Document> databases = mongoClient.listDatabases().into(new ArrayList<>());
            databases.forEach(db -> System.out.println(db.toJson()));
        } catch(Exception e) {
        }
    }

    public void insertRecord(String record) {
        String connectionString = "mongodb://some-mongo:27017";
        try {
            MongoClient mongoClient = MongoClients.create(connectionString);
            MongoDatabase database = mongoClient.getDatabase("myDatabase");
            Document document = new Document();
            document.append("recordId", "CDC");
            document.append("value", record);
            database.getCollection("myCollection").insertOne(document);
        } catch (Exception e) {
        }
    }
}
```



The screenshot shows a terminal window with Java code for interacting with a MongoDB database. The code includes methods for testing a connection and inserting a record. An overlay dialog box titled 'Connect to MySQL Server' is visible, asking for a password for the service 'Mysql@127.0.0.1:3306' and user 'root'. The Java code uses the MongoDB Java Driver (MongoClients) and the Document API to interact with the database.

- f. Provide a screenshot to show that you have successfully modified the handleChangeEvent method.

```

@Component
public class DebeziumListener {

    private final Executor executor = Executors.newSingleThreadExecutor();
    private final DebeziumEngine<RecordChangeEvent<SourceRecord>> debeziumEngine;

    public DebeziumListener(Configuration customerConnectorConfiguration) {
        registration.set
        this.debeziumEngine = DebeziumEngine.create(ChangeEventFormat.of(Connect.class))
            .using(customerConnectorConfiguration.asProperties())
            .notifying(this::handleChangeEvent)
            .build();

        // this.customerService = customerService;
    }

    private void handleChangeEvent(RecordChangeEvent<SourceRecord> sourceRecordRecordChangeEvent) {
        SourceRecord sourceRecord = sourceRecordRecordChangeEvent.record();

        MongoDB mongoDB = new MongoDB();
        mongoDB.testConnection();
        mongoDB.insertRecord(sourceRecord.value().toString());

        System.out.println("Key = " + sourceRecord.key() + " value = " + sourceRecord.value() + "+");
    }

    @PostConstruct
    private void start() {
        this.executor.execute(debeziumEngine);
    }

    @PreDestroy
    private void stop() throws IOException {
}

```

- g. Provide a screenshot to show that you have successfully run the Maven SpringBoot application.

```

DebeziumCDC: docker run -it --rm --name debeziumserver --network MBTANetwork bash
ANSIT_TO,current_stop_sequence=1,direction_id=0,occupancy_status=MANY_SEATS_AVAILABLE},source=Struct{version=1.5.2.Final,connector=mysql,name=localhost_MBTAdb,ts_ms=1659930816000,db=MBTAdb,table=mbta_buses,server_id=1,file=binlog_000002,pos=19551,row=0},op=c,ts_ms=1659930816742' : Cluster created with settings {hosts=[some-mongo:27017], mode=SINGLE, requiredClusterType=UNKNOWN, serverSelectionTimeout='30000 ms'}
2022-08-08 03:53:36.869 INFO 501 --- [pool-1-thread-1] org.mongodb.driver.cluster : Cluster description not yet available. Waiting for 30000 ms before timing out
2022-08-08 03:53:36.869 INFO 501 --- [ome-mongo:27017] org.mongodb.driver.connection for the following: Opened connection [connectionId{localValue:535, serverValue:535}] to some-mongo:27017
2022-08-08 03:53:36.870 INFO 501 --- [ome-mongo:27017] org.mongodb.driver.cluster : Monitor thread successfully connected to server with description ServerDescription{address=some-mongo:27017, type=STANDALONE, state=CONNECTED, ok=true, minWireVersion=0, maxWireVersion=13, maxDocumentSize=16777216, logicalSessionTimeoutMinutes=30, roundTripTimeNanos=74708}
2022-08-08 03:53:36.870 INFO 501 --- [ome-mongo:27017] org.mongodb.driver.connection : Opened connection [connectionId{localValue:536, serverValue:536}] to some-mongo:27017
2022-08-08 03:53:36.870 INFO 501 --- [pool-1-thread-1] org.mongodb.driver.connection : Opened connection [connectionId{localValue:537, serverValue:537}] to some-mongo:27017
2022-08-08 03:53:36.870 INFO 501 --- [ome-mongo:27017] org.mongodb.driver.cluster : Cluster created with settings {hosts=[some-mongo:27017], mode=SINGLE, requiredClusterType=UNKNOWN, serverSelectionTimeout='30000 ms'}
2022-08-08 03:53:36.872 INFO 501 --- [pool-1-thread-1] org.mongodb.driver.cluster : Cluster description not yet available. Waiting for 30000 ms before timing out
2022-08-08 03:53:36.873 INFO 501 --- [ome-mongo:27017] org.mongodb.driver.connection : Opened connection [connectionId{localValue:538, serverValue:538}] to some-mongo:27017
2022-08-08 03:53:36.873 INFO 501 --- [ome-mongo:27017] org.mongodb.driver.cluster : Monitor thread successfully connected to server with description ServerDescription{address=some-mongo:27017, type=STANDALONE, state=CONNECTED, ok=true, minWireVersion=0, maxWireVersion=13, maxDocumentSize=16777216, logicalSessionTimeoutMinutes=30, roundTripTimeNanos=450833}
2022-08-08 03:53:36.874 INFO 501 --- [ome-mongo:27017] org.mongodb.driver.connection : Opened connection [connectionId{localValue:539, serverValue:539}] to some-mongo:27017
2022-08-08 03:53:36.874 INFO 501 --- [pool-1-thread-1] org.mongodb.driver.connection : Opened connection [connectionId{localValue:540, serverValue:540}] to some-mongo:27017
Key = 'Struct{record_num=90}' value = 'Struct{after=Struct{record_num=90,id=y1786,latitude=42.32981000,longitude=-71.08353000,bearing=90,current_status=IN_TRANSIT_TO,current_stop_sequence=1,direction_id=1},source=Struct{version=1.5.2.Final,connector=mysql,name=localhost_MBTAdb,ts_ms=1659930816000,db=MBTAdb,table=mbta_buses,server_id=1,file=binlog_000002,pos=19743,row=0},op=c,ts_ms=1659930816742}'
```

6.

- a) Provide a screenshot of your Docker desktop to show the `javadmaven` container running.

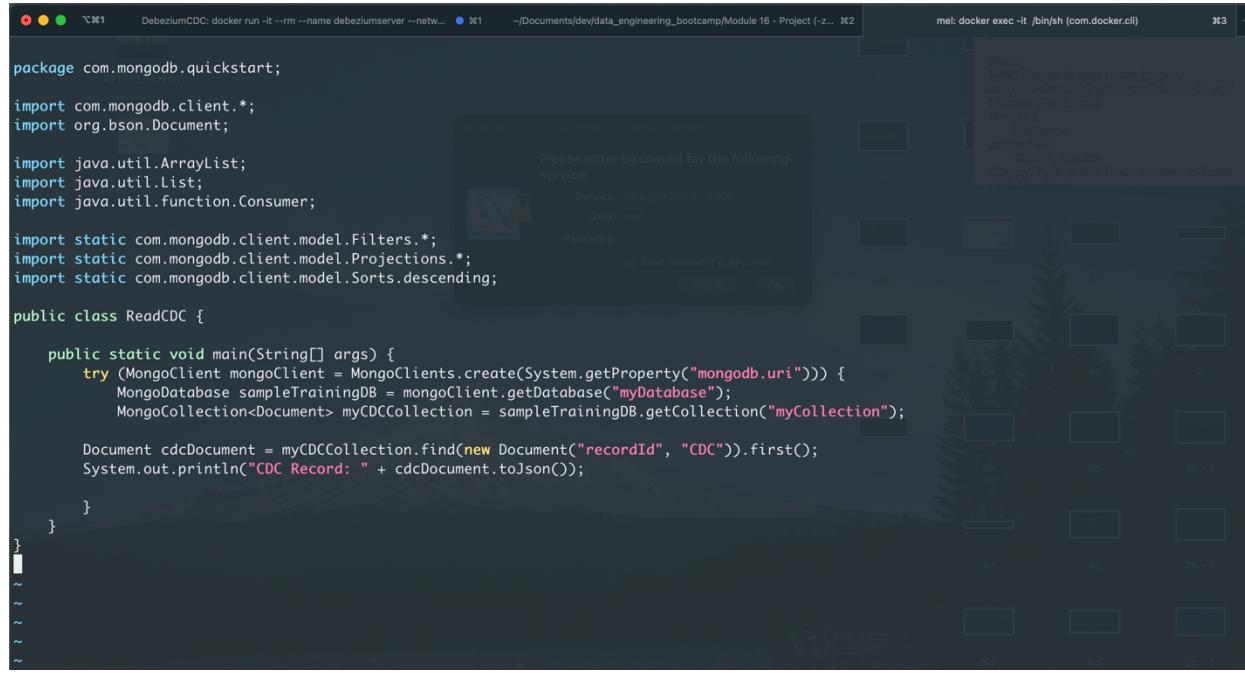
The screenshot shows the Docker Desktop interface with the 'Containers' tab selected. On the left sidebar, there are links for 'Containers', 'Images', 'Volumes', and 'Dev Environments'. Under 'Extensions', it says 'BETA' and has a 'Add Extensions' button. The main area displays a table titled 'Containers' with the following data:

	NAME	IMAGE	STATUS	PORT(S)	STARTED	
mysqlserver	b27116fec98	mysqlmbatamasterimg:latest	Running	3306	5 hours ago	[actions]
some-mongo	6cd8a288fe29	mongo:latest	Running	27017	5 hours ago	[actions]
debeziumserver	cea7f132d3ed	debeziummodule16:latest	Running	-	39 minutes ago	[actions]
javadmaven	fc78f61118ef	maven:3.6.3-openjdk-11	Running	8080	1 minute ago	[actions]

- b) Provide a screenshot to show that you successfully navigated to the `directory` and listed the files.

```
# cd /java-quick-start/src/main/java/com/mongodb/quickstart
# ls
AggregationFramework.java  Connection.java  Delete.java      MappingPOJO.java  Update.java  models
ChangeStreams.java         Create.java     HelloMongoDB.java  Read.java       csfle
# ls -la
total 56
drwxr-xr-x  4 501 dialout 4096 Aug  8 04:05 .
drwxr-xr-x  3 501 dialout 4096 Aug  8 04:05 ..
-rw-r--r--  1 501 dialout 3131 Aug  8 04:05 AggregationFramework.java
-rw-r--r--  1 501 dialout 4991 Aug  8 04:05 ChangeStreams.java
-rw-r--r--  1 501 dialout 592 Aug  8 04:05 Connection.java
-rw-r--r--  1 501 dialout 2396 Aug  8 04:05 Create.java
-rw-r--r--  1 501 dialout 1585 Aug  8 04:05 Delete.java
-rw-r--r--  1 501 dialout 161 Aug  8 04:05 HelloMongoDB.java
-rw-r--r--  1 501 dialout 3170 Aug  8 04:05 MappingPOJO.java
-rw-r--r--  1 501 dialout 3026 Aug  8 04:05 Read.java
-rw-r--r--  1 501 dialout 4060 Aug  8 04:05 Update.java
drwxr-xr-x  2 501 dialout 4096 Aug  8 04:05 csfle
drwxr-xr-x  2 501 dialout 4096 Aug  8 04:05 models
#
```

- c) Provide a screenshot to show that you successfully created the ReadCDC.java file and copied the code.



```

package com.mongodb.quickstart;

import com.mongodb.client.*;
import org.bson.Document;

import java.util.ArrayList;
import java.util.List;
import java.util.function.Consumer;

import static com.mongodb.client.model.Filters.*;
import static com.mongodb.client.model.Projections.*;
import static com.mongodb.client.model.Sorts.descending;

public class ReadCDC {

    public static void main(String[] args) {
        try (MongoClient mongoClient = MongoClients.create(System.getProperty("mongodb.uri"))) {
            MongoDatabase sampleTrainingDB = mongoClient.getDatabase("myDatabase");
            MongoCollection<Document> myCDCCollection = sampleTrainingDB.getCollection("myCollection");

            Document cdcDocument = myCDCCollection.find(new Document("recordId", "CDC")).first();
            System.out.println("CDC Record: " + cdcDocument.toJson());
        }
    }
}

```

A modal dialog box titled "Please enter password for the following service:" is displayed. It shows "Service: MySQL@127.0.0.1:3306", "User: root", and a "Password:" field. There are checkboxes for "Save password in keychain" and "Cancel". A "OK" button is partially visible.

- d) Provide a screenshot to show the results of the bash command to execute the ReadCDC.java class.

```

at org.codehaus.plexus.classworlds.launcher.Launcher.mainWithExitCode (Launcher.java:406)
at org.codehaus.plexus.classworlds.launcher.Launcher.main (Launcher.java:347)
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 29.714 s
[INFO] Finished at: 2022-08-08T04:09:19Z
[INFO] -----
# 

```