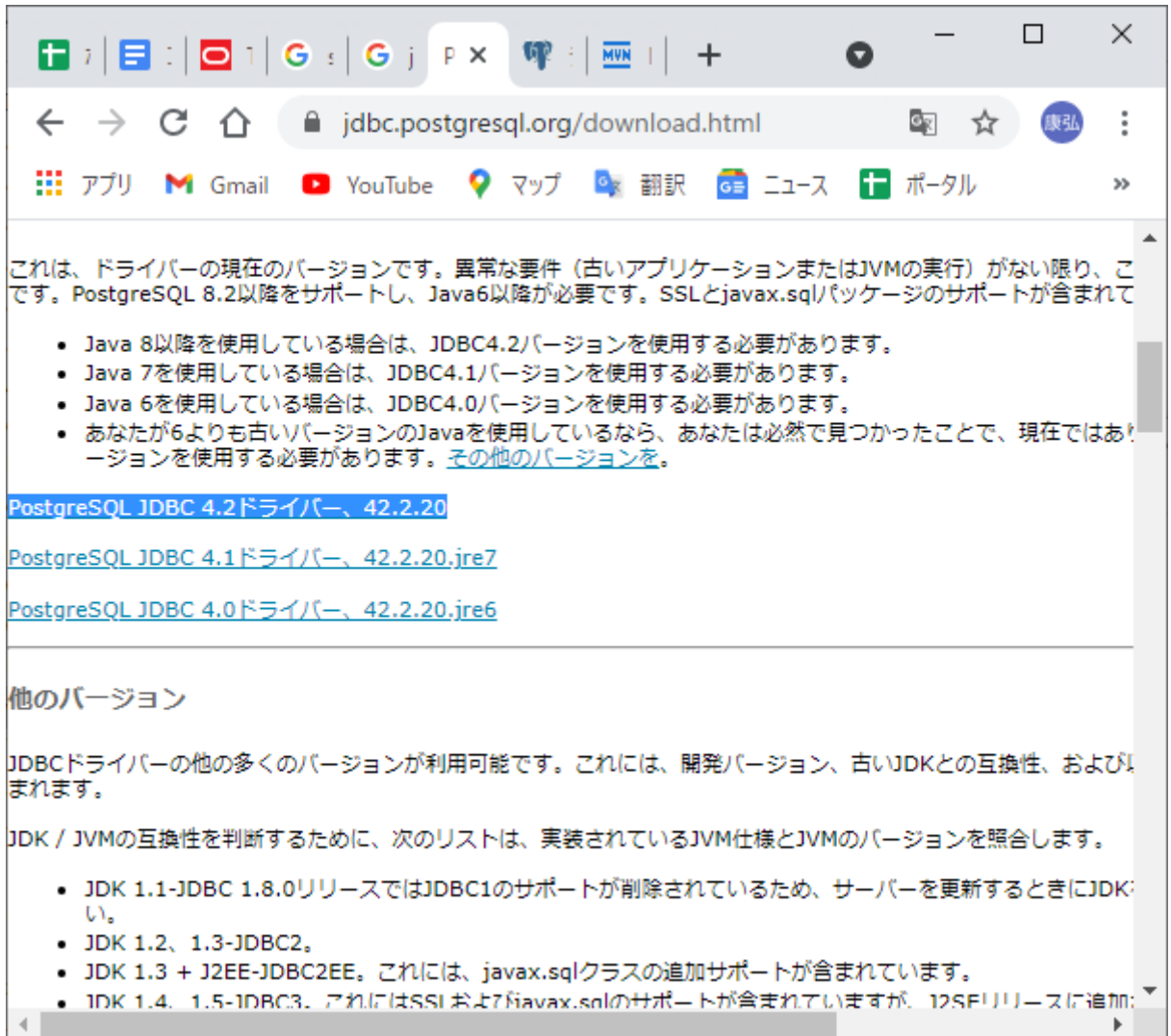


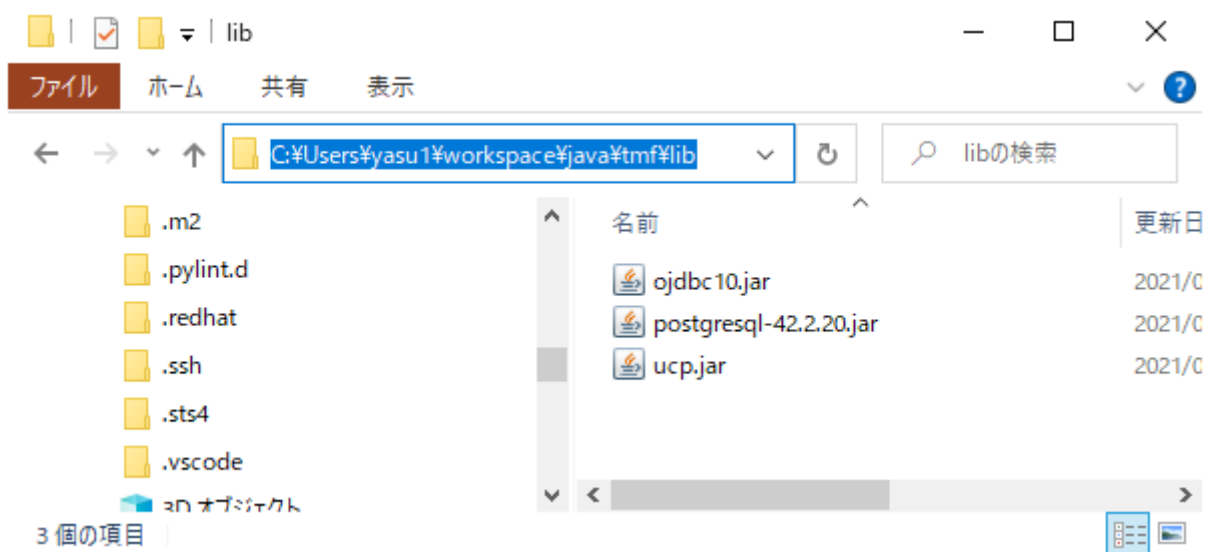
コンテナのJavaからコンテナのpostgresへの接続

1. postgresのjarをダウンロード

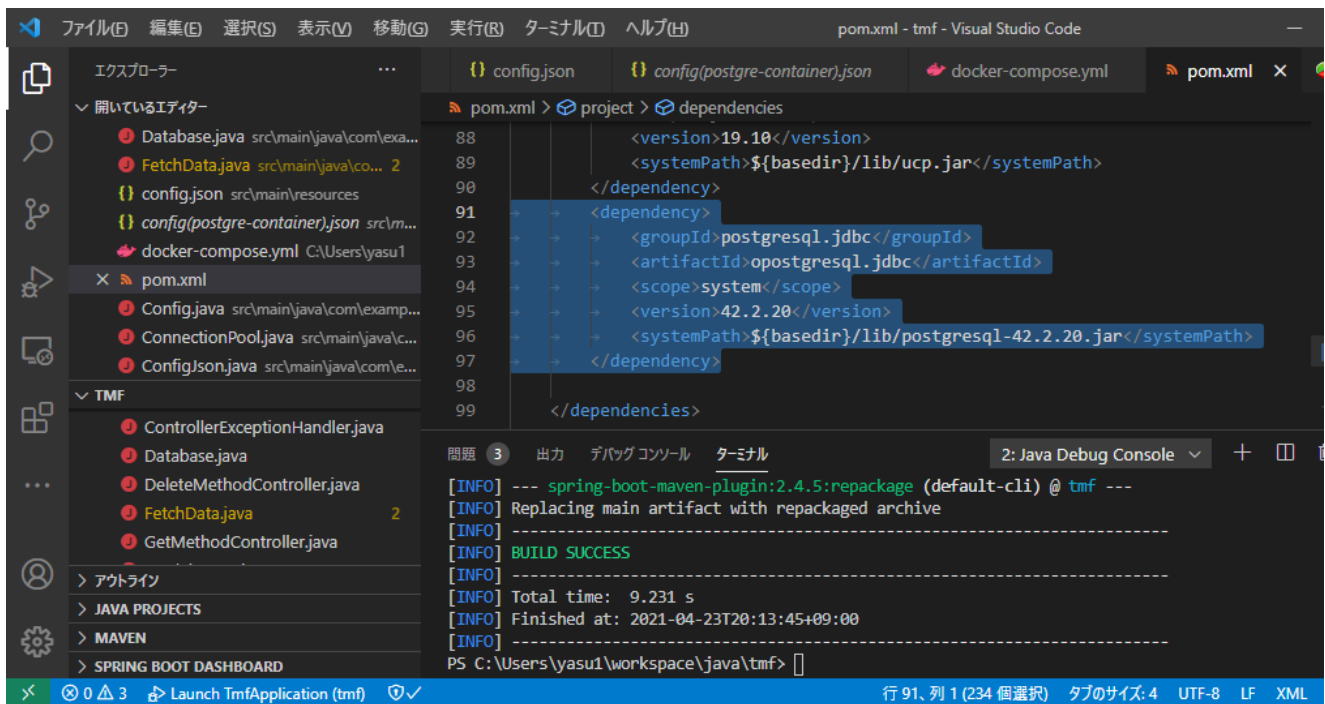
- a. 「<https://jdbc.postgresql.org/download.html>」にアクセス。「PostgreSQL JDBC 4.2ドライバー、42.2.20」をダウンロード。



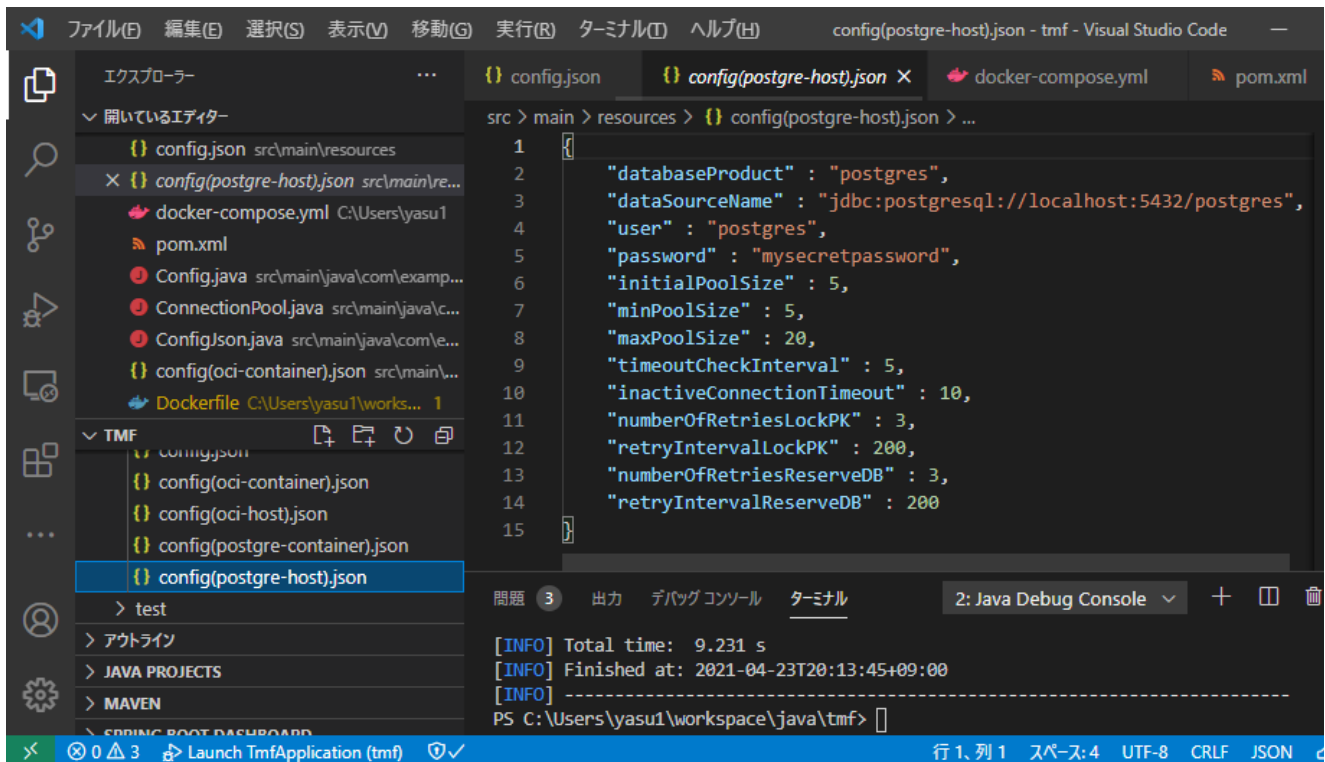
- b. 「C:\Users\yasu1\workspace\java\tmf\lib」に「postgresql-42.2.20.jar」をコピー。



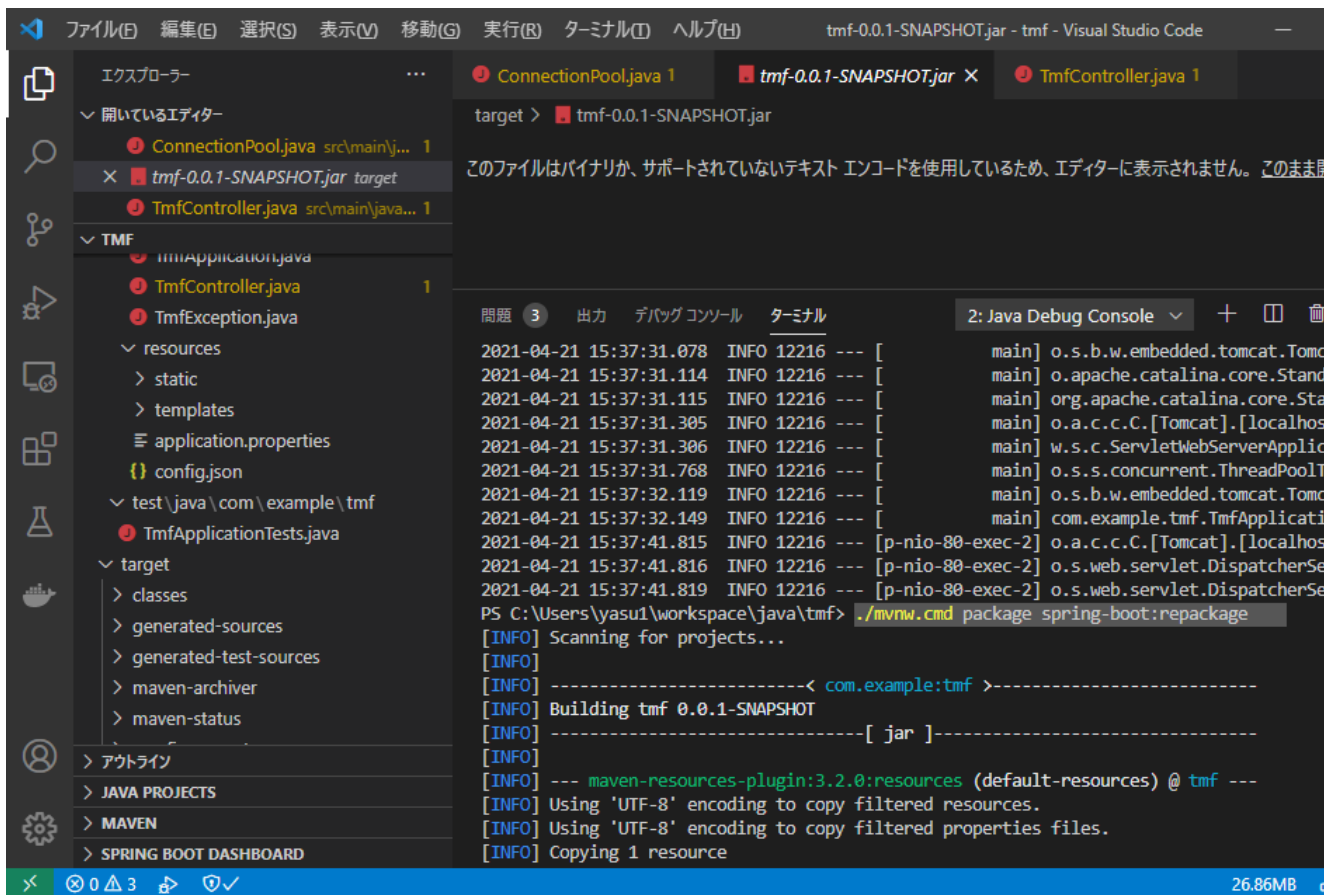
- c. 「pom.xml」に「<dependency>～</dependency>」を追記。



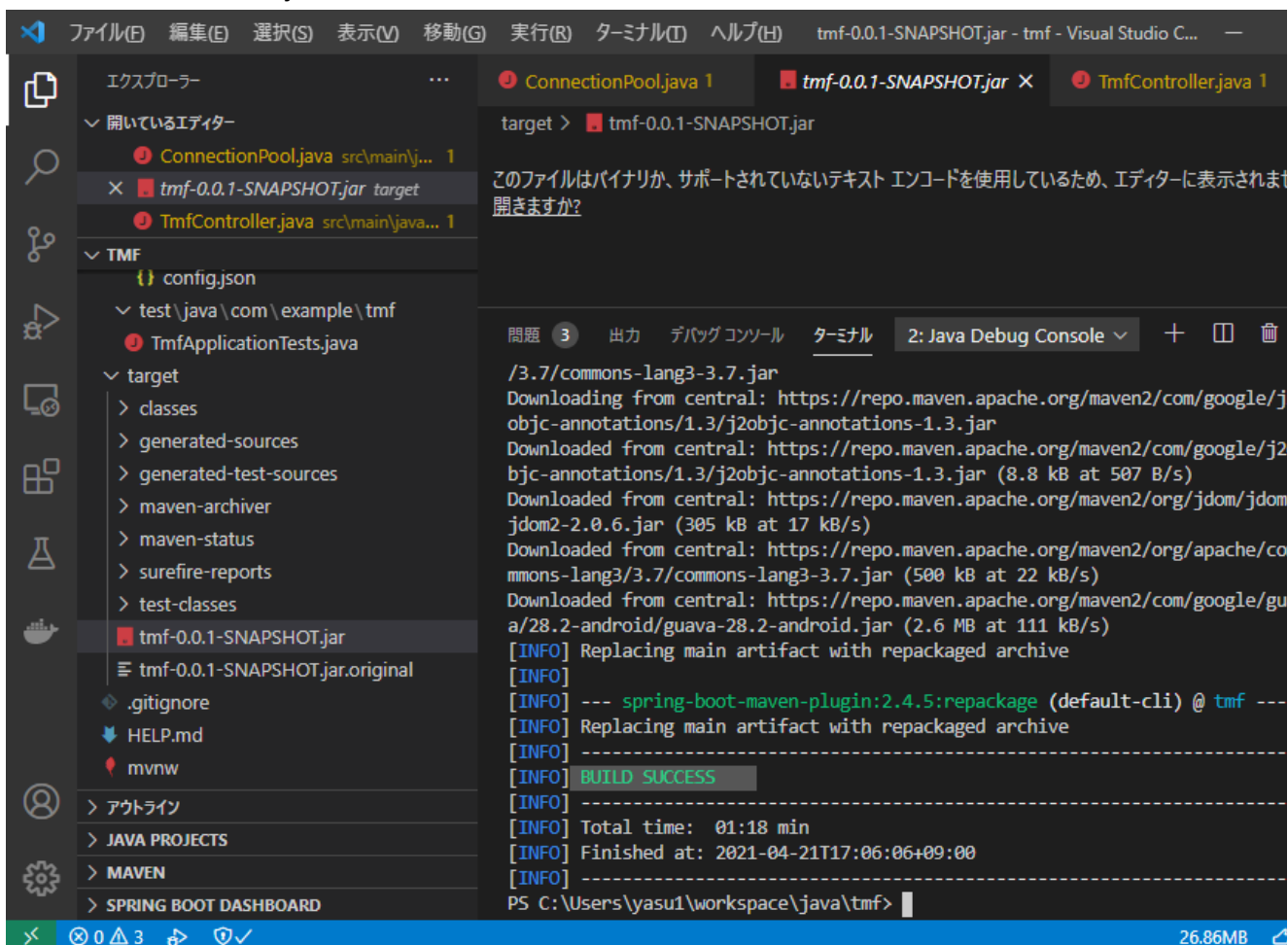
d. config.jsonのを以下のように修正。



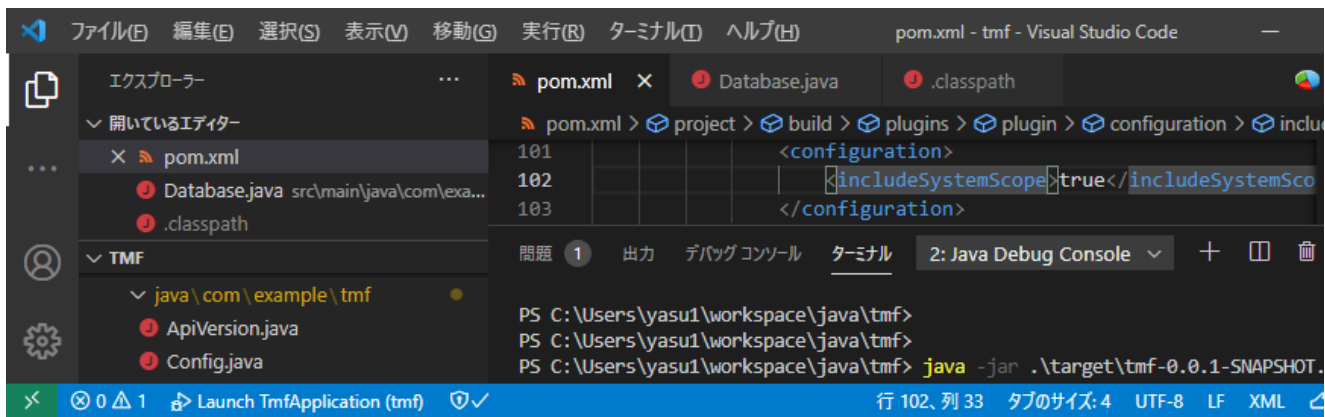
e. ターミナルに「./mvnw.cmd package spring-boot:repackage」と入力(MACの場合は「./mvnw package spring-boot:repackage」)。



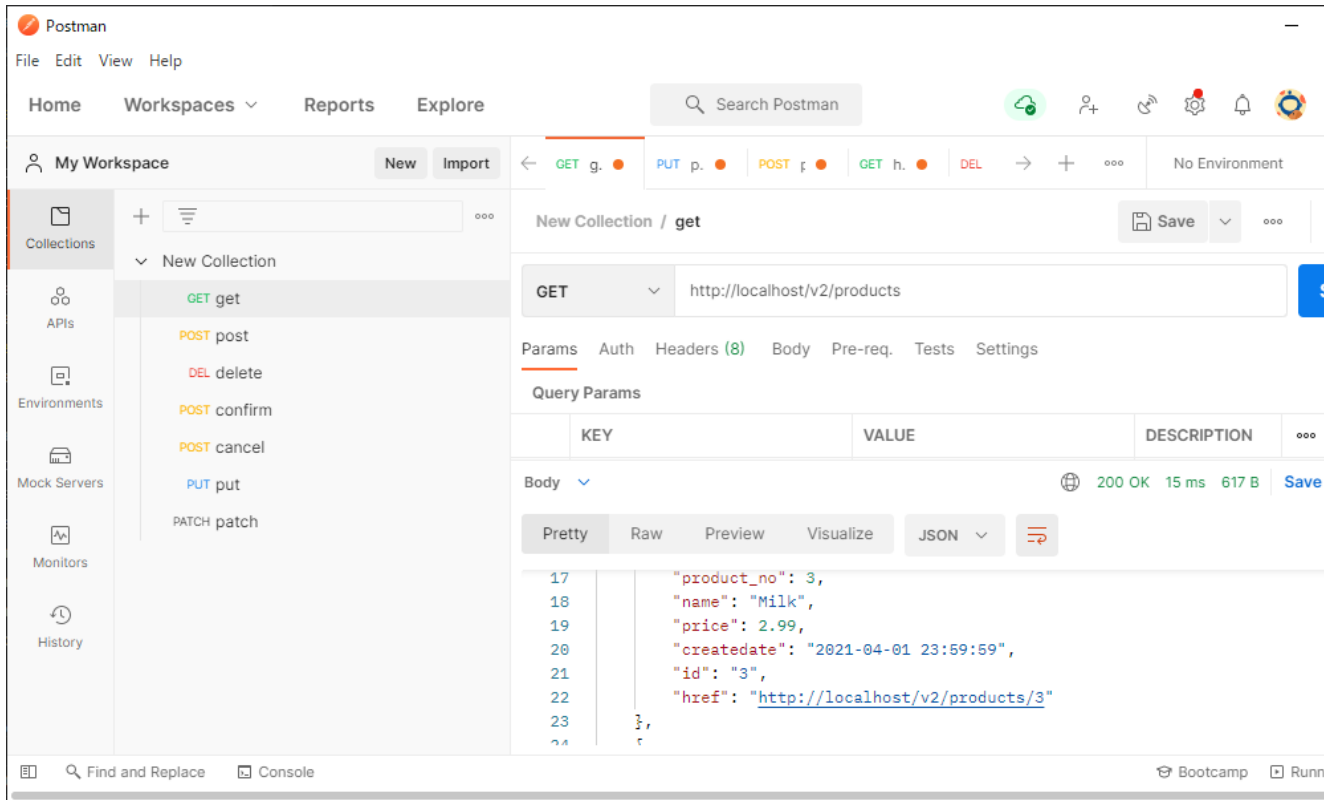
- f. 以下のように「BUILD SUCCESS」と表示されればOK。targetフォルダ直下に「tmf-0.0.1-SNAPSHOT.jar」ファイルが作成される。



- g. 「java -jar .\target\tmf-0.0.1-SNAPSHOT.jar」を入力し、起動し導通試験して確認。

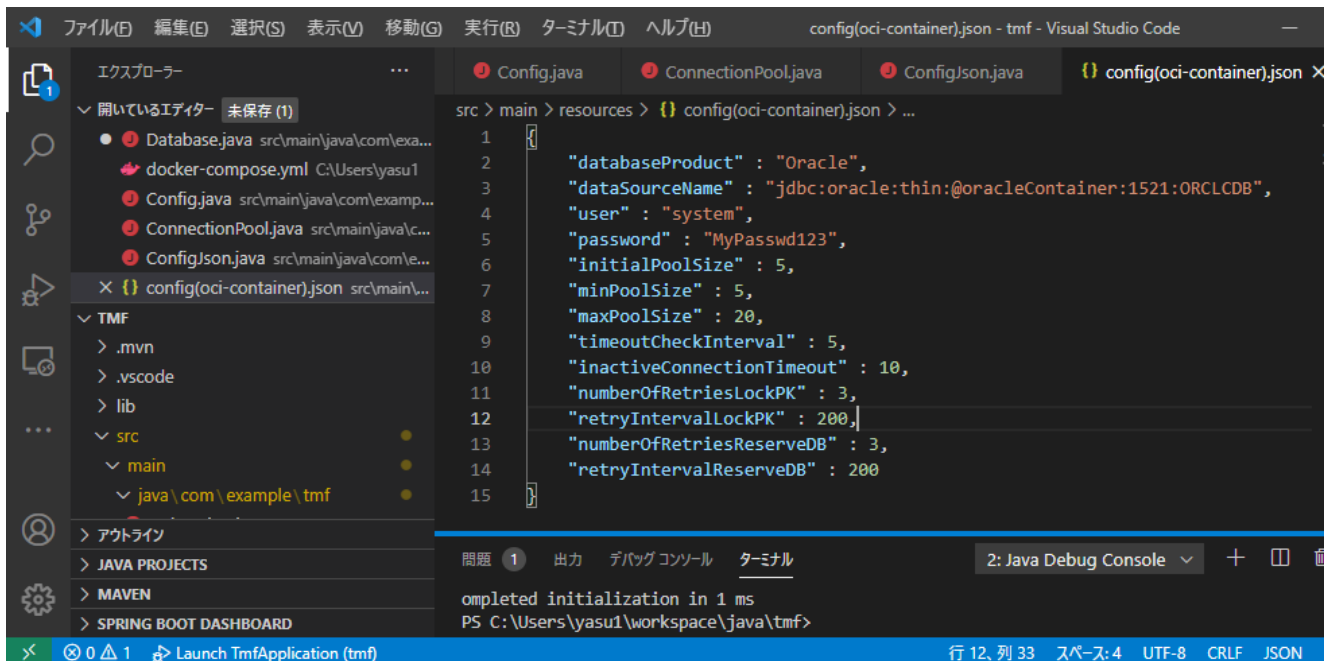


h. 導通試験して確認

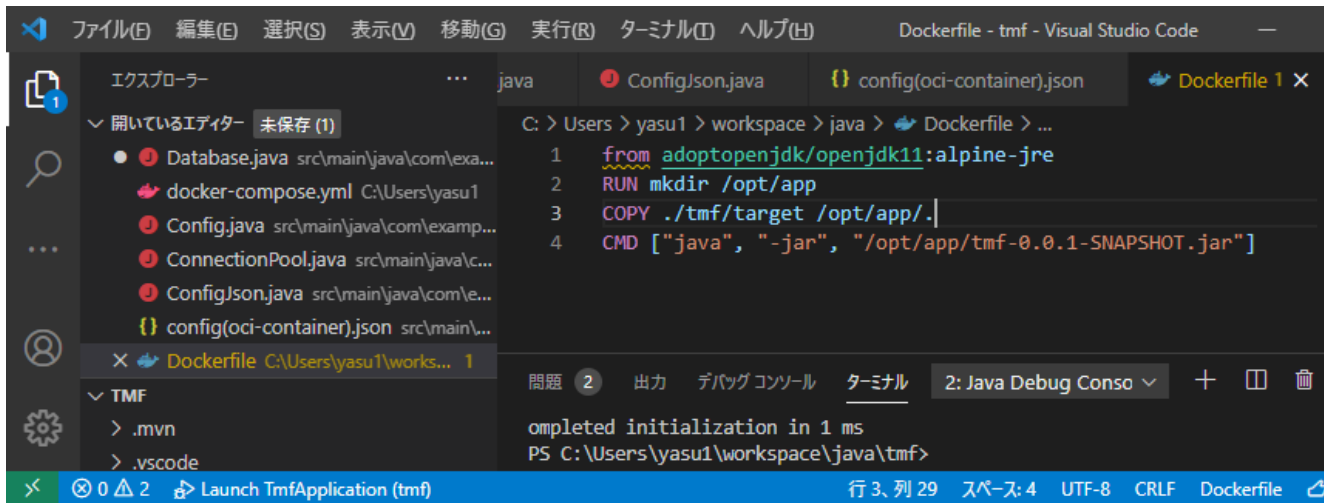


2. 導通試験

a. 以下のconfig.jsonに修正。ホスト名がコンテナ名となっていることに注意。



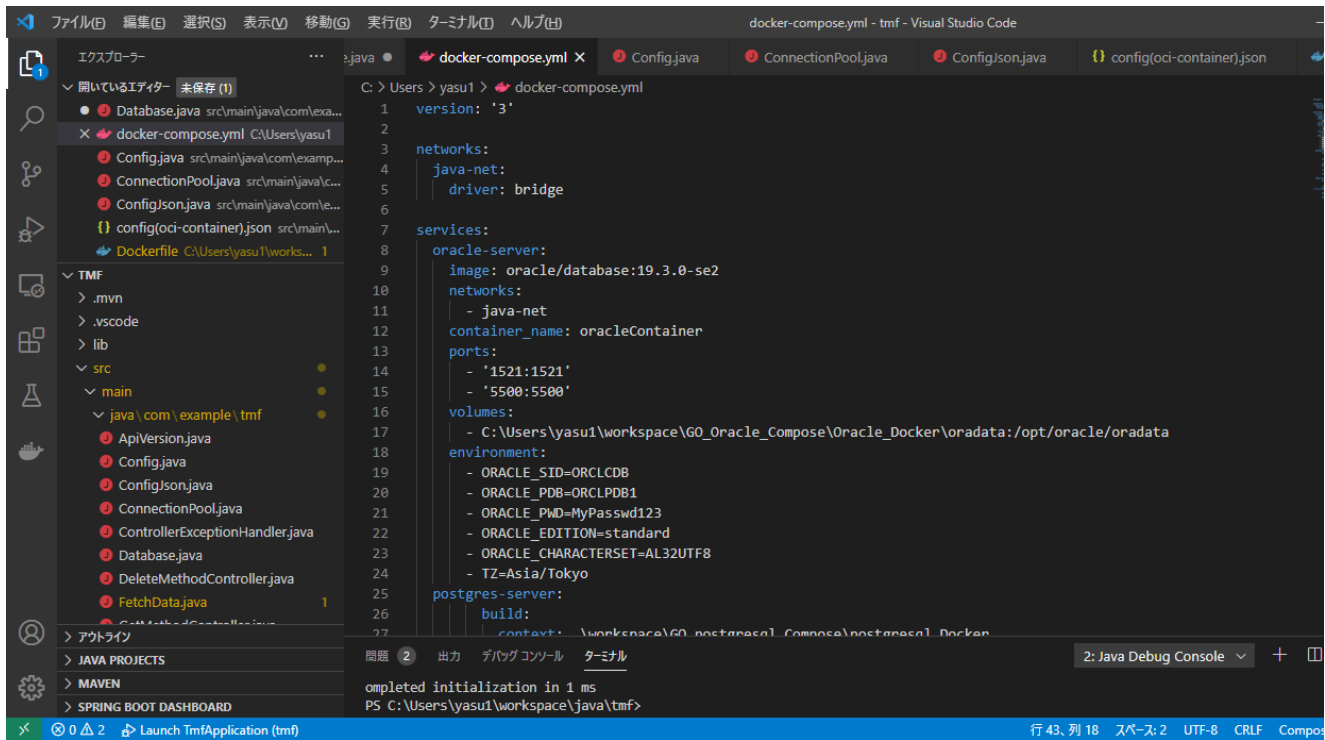
- b. 以下のDockerfileを作成。



```
C: > Users > yasu1 > workspace > java > Dockerfile > ...
1 from adoptopenjdk/openjdk11:alpine-jre
2 RUN mkdir /opt/app
3 COPY ./tmf/target /opt/app/.|
4 CMD ["java", "-jar", "/opt/app/tmf-0.0.1-SNAPSHOT.jar"]
```

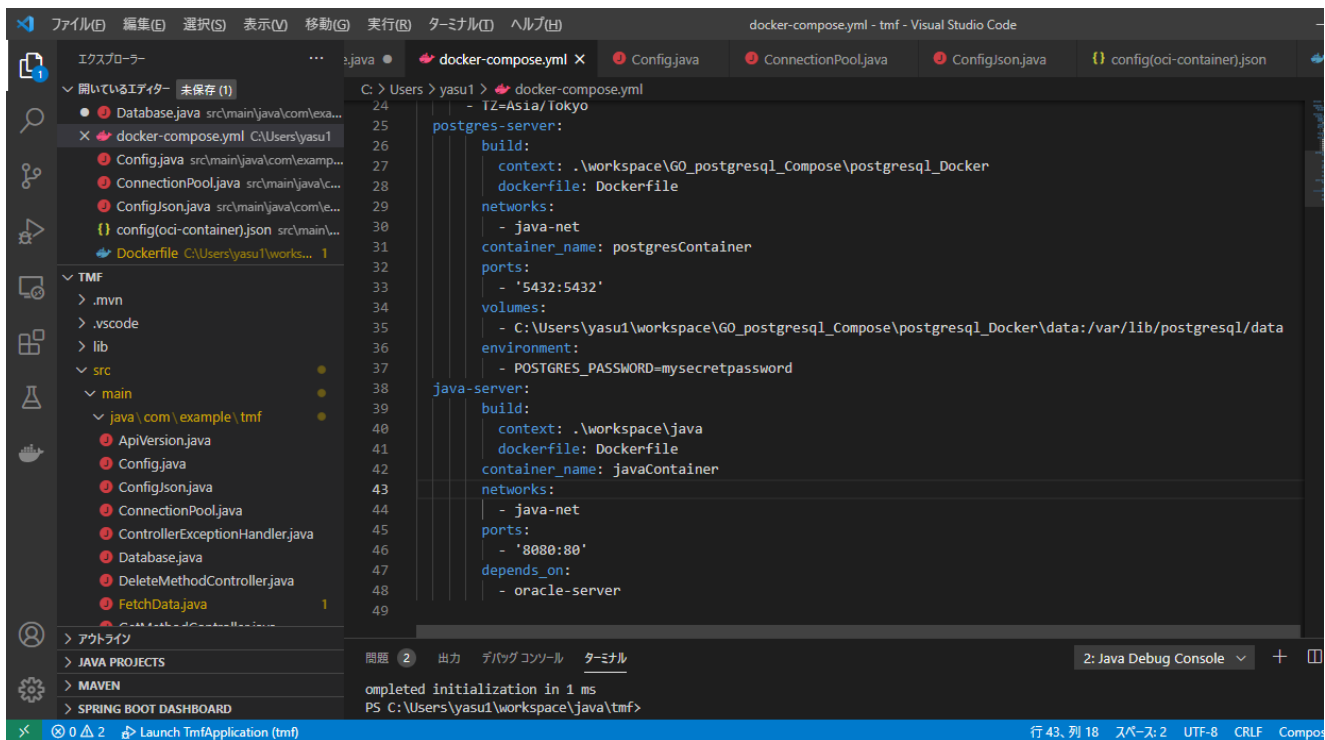
ompleted initialization in 1 ms
PS C:\Users\yasu1\workspace\java\tmf>

- c. 以下のdocker-compose.ymlを作成

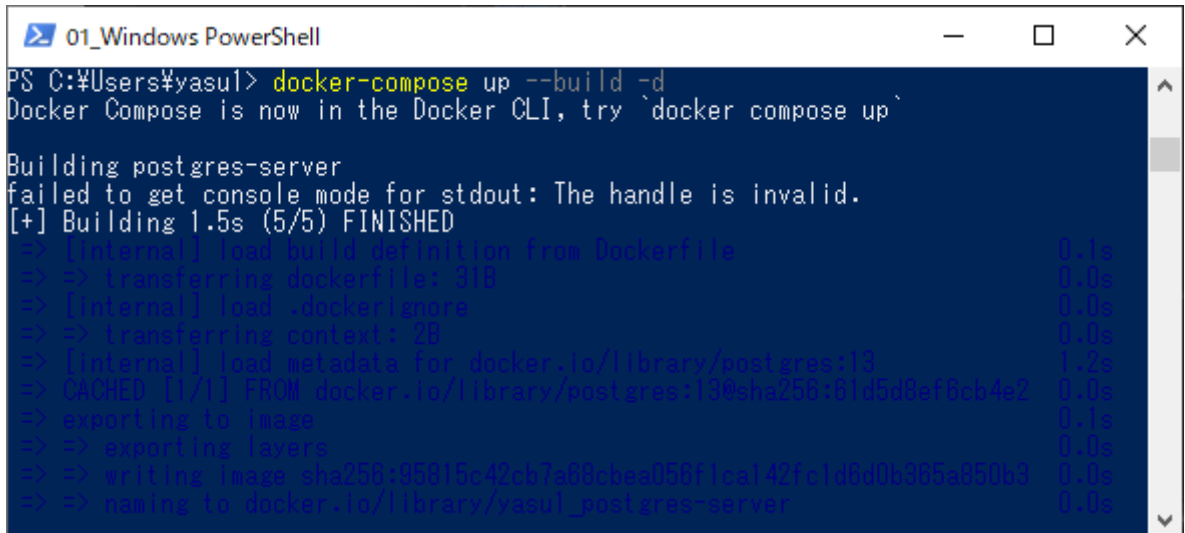


```
C: > Users > yasu1 > docker-compose.yml
1 version: '3'
2
3 networks:
4   java-net:
5     driver: bridge
6
7 services:
8   oracle-server:
9     image: oracle/database:19.3.0-se2
10    networks:
11      - java-net
12    container_name: oracleContainer
13    ports:
14      - '1521:1521'
15      - '5500:5500'
16    volumes:
17      - C:\Users\yasu1\workspace\G0_Oracle_Compose\Oracle_Docker\oradata:/opt/oracle/oradata
18    environment:
19      - ORACLE_SID=ORCLCDB
20      - ORACLE_PDB=ORCLPDB1
21      - ORACLE_PWD=MyPasswd123
22      - ORACLE_EDITION=standard
23      - ORACLE_CHARACTERSET=AL32UTF8
24      - TZ=Asia/Tokyo
25    postgres-server:
26      build:
27        context: \workspace\G0_postgresql_Compose\postgresql_Docker
```

ompleted initialization in 1 ms
PS C:\Users\yasu1\workspace\java\tmf>



d. 「docker-compose up --build -d」を実行



e. 導通試験して確認(ポート8080になっている事に注意)

Postman interface showing a REST client setup for a collection named "get".

Workspace: New Import

New Collection: GET get, POST post, DEL delete, POST confirm, POST cancel, PUT put, PATCH patch

GET g. PUT p. POST f. GET h. DEL → + ... No Environment

New Collection / get Save

GET http://localhost:8080/v2/products/

Params Auth Headers (8) Body Pre-req. Tests Settings

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body 200 OK 19 ms 548 B

Pretty Raw Preview Visualize JSON

```
15  },
16  {
17    "product_no": 3,
18    "name": "Milk",
19    "price": 2.99,
20    "id": "3",
21    "href": "http://localhost:8080/v2/products/3"
22  },
```

Find and Replace Console Bootcamp