in previous sprint
mongo container started manually
after that
customerapp container started manually

customerapp container started/running

mongo container started/running

dependent container

dependency container

In sprint3, both containers to be started manually
if dependency container not started, dependent container wont work smooth

in Sprint 4, these both containers can be grouped
Note: JWT demo applications are used for below demos, filtering disabled in customer app temporarly

Demo1

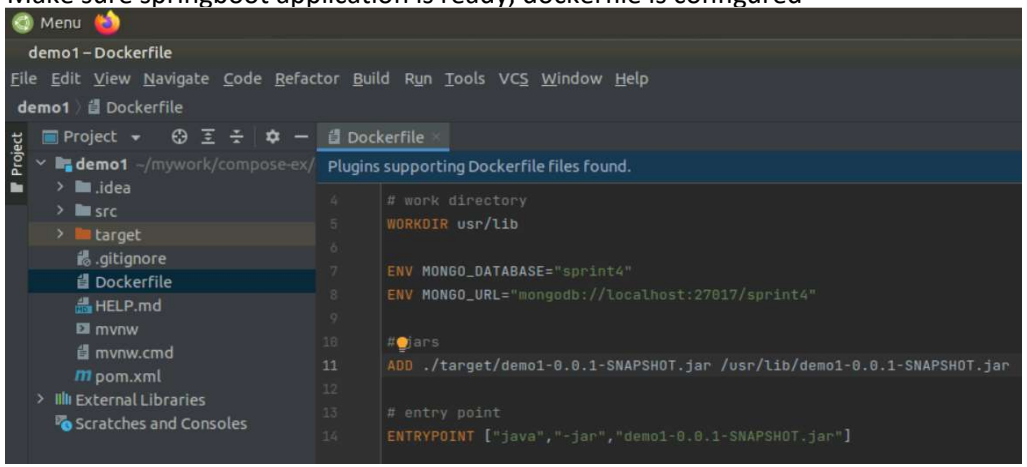
Sprintboot customer application container
MongoDB container

customerapp container started/running

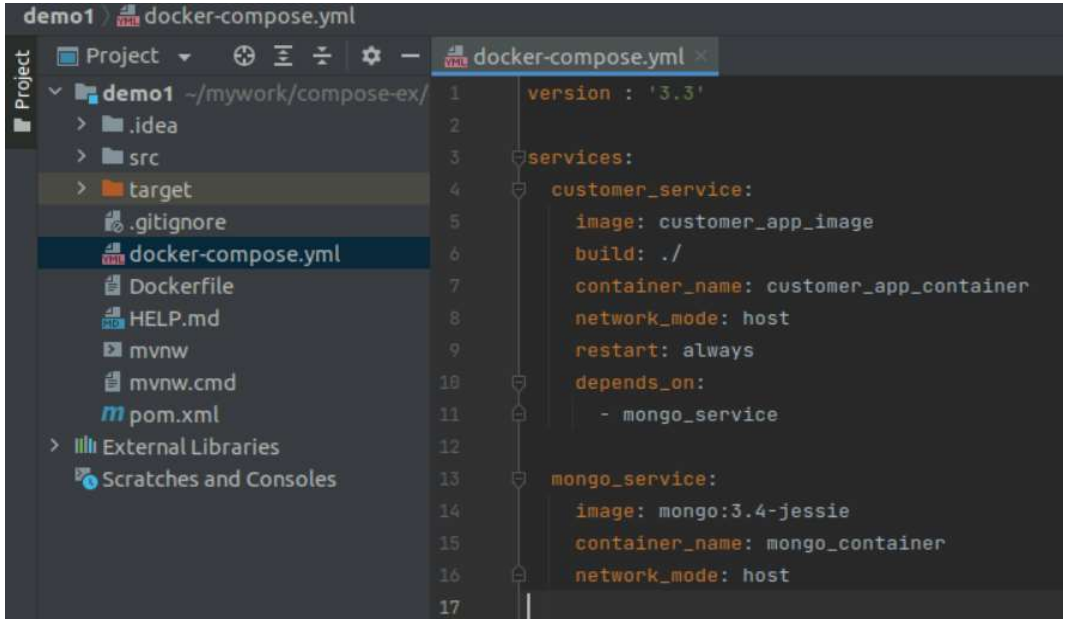
mongo container started/running

steps to add docker compose to existing application

Step 1 Make sure springboot application is ready, dockerfile is configured



Step 2 create docker-compose file in root folder



Step 3 run docker compose command

sudo docker-compose up --build

```
ubuntu@ip-172-31-36-21:~/mywork/compose-ex/demo1$ sudo docker-compose up --build
Pulling mongo_service (mongo:3.4-jessie)...
3.4-jessie: Pulling from library/mongo
2a639da97f77: Pull complete
073b4f52defe: Pull complete
bce37d0f5c17: Pull complete
379dc19f9963: Pull complete
e44806c61e63: Pull complete
b76faf91d209: Pull complete
dd1d9be5b26b: Pull complete
9420e1982a2f: Pull complete
9ad2432e6a03: Pull complete
a80971ca2409: Pull complete
3a0937743e17: Pull complete
```

```
ubuntu@ip-172-31-36-21:~/mywork/compose-ex/demo1$ sudo docker-compose down
Stopping customer_app_container ... done
Stopping mongo_container        ... done
Removing customer_app_container ... done
Removing mongo_container        ... done
```

Demo2

Sprintboot authentication application container
MySql container

authentication container started/running

mysql container started/running

Step 1 Make sure authentication application is copied to linux machine
open the same in intellij

Step 2 Make sure mysql service stopped in linux machine

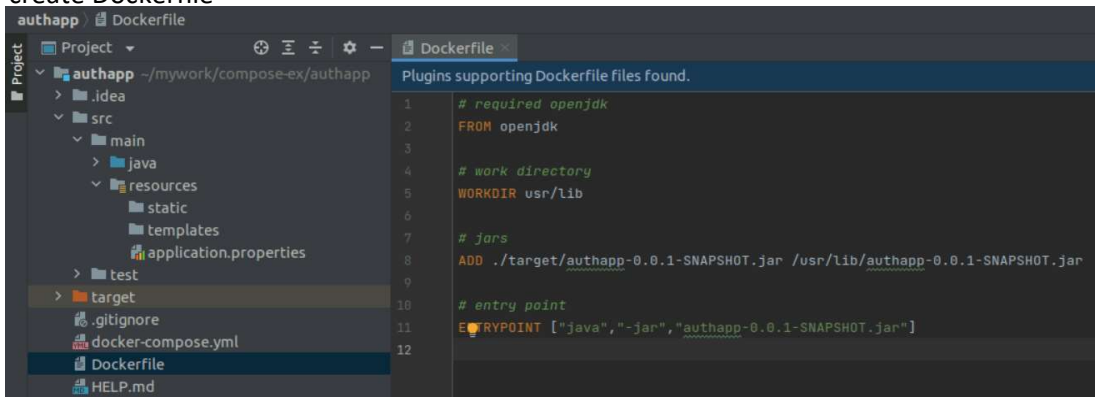
sudo service mysql stop

sudo service mysql status

Step 3 modify application.properties

```
server.port=8888
driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/wave31db?useSSL=false&createDatabaseIfNotExist=true&allowPublicKeyRetrieval=true
spring.datasource.username=root
spring.datasource.password=root
spring.jpa.hibernate.ddl_auto=update
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL57Dialect
```

Step 4 create Dockerfile



Step 6 create docker-compose.yml

```yaml
version : '3.3'

services:
  auth_service:
    image: auth_app_image
    build: ./
    container_name: auth_app_container
    network_mode: host
    restart: always
    depends_on:
      - mysql_service

  mysql_service:
    image: mysql:5.5
    container_name: mysql_container
    network_mode: host

    environment:
      MYSQL_ROOT_PASSWORD: root
      MYSQL_USERNAME: user
      MYSQL_PASSWORD: root
      MYSQL_ALLOW_EMPTY_PASSWORD: "yes"
```

Step 7 mvn clean
mvn install

Step 8 run compose up command
sudo docker-compose up --build