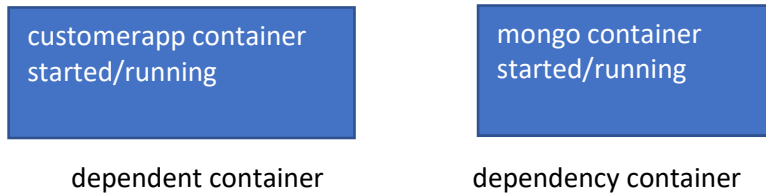


in previous sprint
mongo container started manually
after that
customerapp container started manually



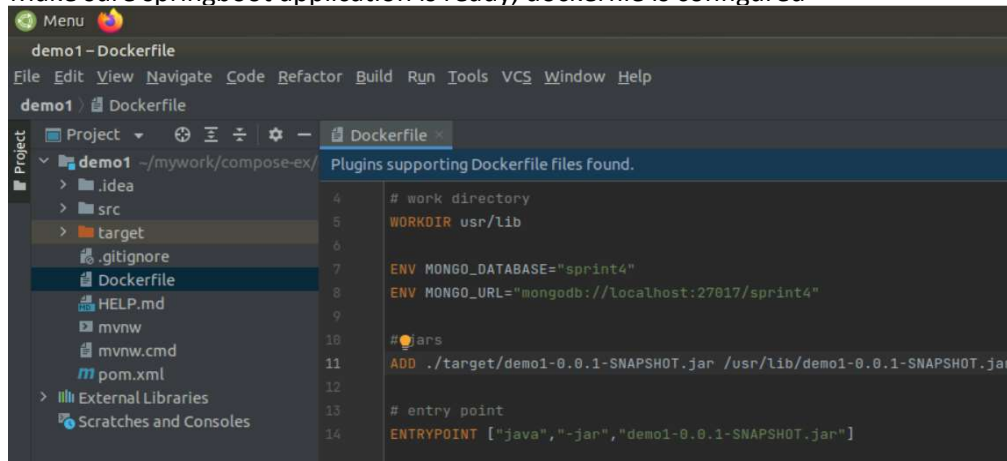
In sprint3, both containers to be started manually
if dependency container not started, dependent container wont work smooth

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



steps to add docker compose to existing application

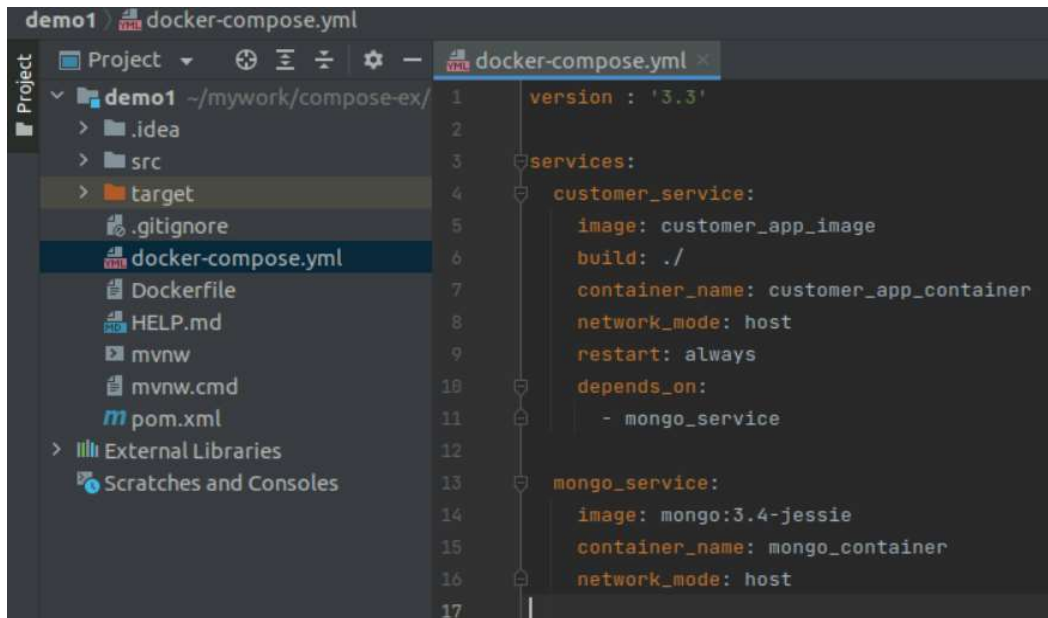
Step 1 Make sure springboot application is ready, dockerfile is configured



The screenshot shows an IDE window titled 'demo1 - Dockerfile'. The left sidebar displays a project structure with folders like '.idea', 'src', 'target', and files like '.gitignore', 'HELP.md', 'mvnw', 'mvnw.cmd', and 'pom.xml'. The 'Dockerfile' is selected. The main editor shows the following content:

```
Menu
demo1 - Dockerfile
File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
demo1 Dockerfile
Project
  demo1 ~/mywork/compose-ex/
    .idea
    src
    target
    .gitignore
    Dockerfile
    HELP.md
    mvnw
    mvnw.cmd
    pom.xml
  External Libraries
  Scratches and Consoles
Dockerfile
  Plugins supporting Dockerfile files found.
  4 # work directory
  5 WORKDIR usr/lib
  6
  7 ENV MONGO_DATABASE="sprint4"
  8 ENV MONGO_URL="mongodb://localhost:27017/sprint4"
  9
  10 # jars
  11 ADD ./target/demo1-0.0.1-SNAPSHOT.jar /usr/lib/demo1-0.0.1-SNAPSHOT.jar
  12
  13 # entry point
  14 ENTRYPOINT ["java", "-jar", "demo1-0.0.1-SNAPSHOT.jar"]
```

Step 2 create docker-compose file in root folder



The screenshot shows an IDE window titled 'demo1 - docker-compose.yml'. The left sidebar displays a project structure with folders like '.idea', 'src', 'target', and files like '.gitignore', 'docker-compose.yml', 'Dockerfile', 'HELP.md', 'mvnw', 'mvnw.cmd', and 'pom.xml'. The 'docker-compose.yml' is selected. The main editor shows the following content:

```
demo1 - docker-compose.yml
Project
  demo1 ~/mywork/compose-ex/
    .idea
    src
    target
    .gitignore
    docker-compose.yml
    Dockerfile
    HELP.md
    mvnw
    mvnw.cmd
    pom.xml
  External Libraries
  Scratches and Consoles
docker-compose.yml
  1 version : '3.3'
  2
  3 services:
  4   customer_service:
  5     image: customer_app_image
  6     build: ./
  7     container_name: customer_app_container
  8     network_mode: host
  9     restart: always
 10     depends_on:
 11       - mongo_service
 12
 13   mongo_service:
 14     image: mongo:3.4-jessie
 15     container_name: mongo_container
 16     network_mode: host
 17
```

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

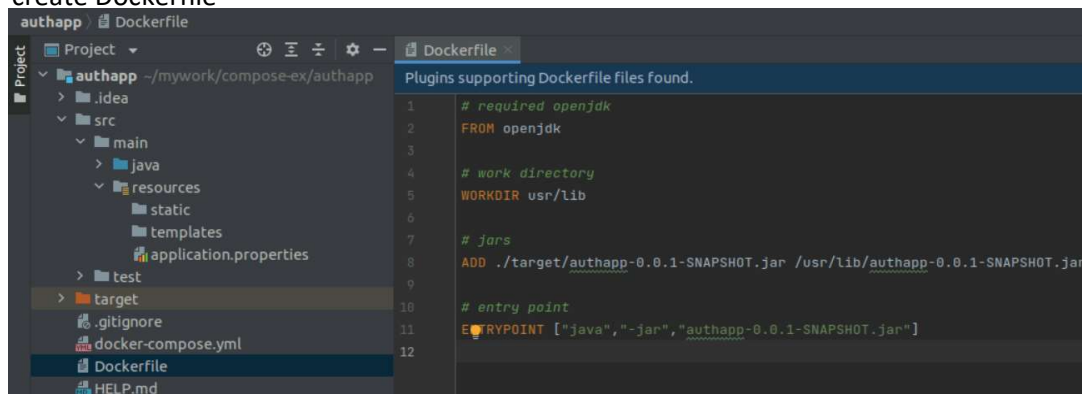
Step 1 Make sure authentication application is copied to linux machine
open the same in intellij

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



The screenshot shows an IDE with a project named 'authapp' and a Dockerfile. The Dockerfile content is as follows:

```
1 # required openjdk
2 FROM openjdk
3
4 # work directory
5 WORKDIR usr/lib
6
7 # jars
8 ADD ./target/authapp-0.0.1-SNAPSHOT.jar /usr/lib/authapp-0.0.1-SNAPSHOT.jar
9
10 # entry point
11 ENTRYPOINT ["java", "-jar", "authapp-0.0.1-SNAPSHOT.jar"]
12
```

Step 6 create docker-compose.yml

```
1  version : '3.3'
2
3  services:
4    auth_service:
5      image: auth_app_image
6      build: ./
7      container_name: auth_app_container
8      network_mode: host
9      restart: always
10     depends_on:
11       - mysql_service
12
13     mysql_service:
14       image: mysql:5.5
15       container_name: mysql_container
16       network_mode: host
17
18       environment:
19         MYSQL_ROOT_PASSWORD: root
20         MYSQL_USERNAME: user
21         MYSQL_PASSWORD: root
22         MYSQL_ALLOW_EMPTY_PASSWORD: "yes"
```

Step 7 mvn clean
mvn install

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