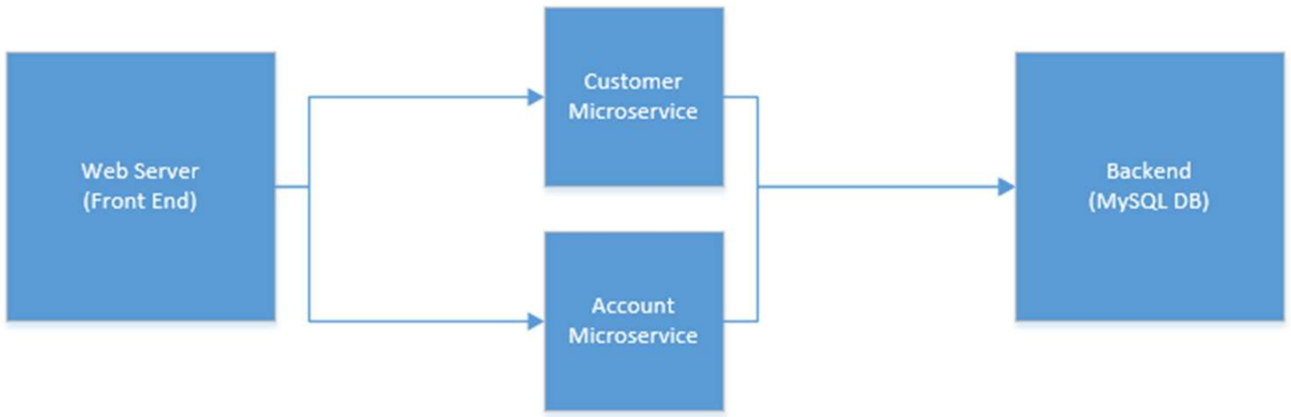
DOCKER & KUBERNETES TRAINING CASE STUDY

Contributors :

Aditya Kaushik, Vinay C Mahajan,Ram Singh Yadav

Case Study

- **Below is the visualization of the Case Study of deployment methodology for a modern web application**

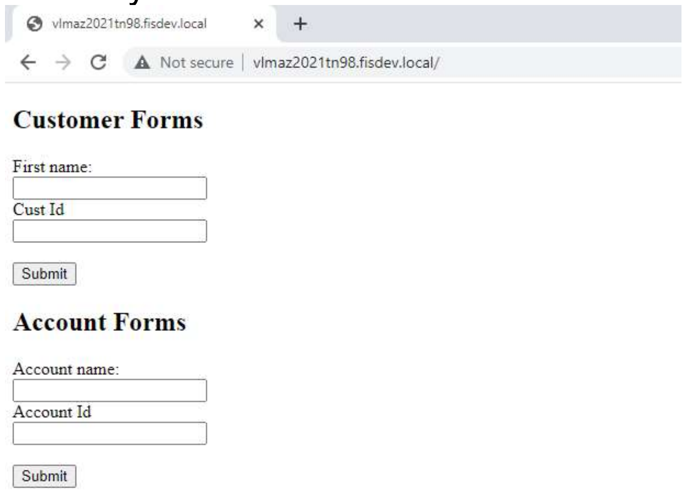


Stages covered in Case Study

- 1. Created Web page on nginx web server – Customer.html [Frontend Application]
- 2. Created Micro Services – Customer and Account REST API [Backend Service]
- 3. Created Database in MySQL – springbootdb [Database Service]
- 4. Designed Docker-Compose – docker-compose.yml - To build and deploy all above layers
- 5. Launch Application and Make Transactions
- 6. Case Study Project Features

Created Web page on nginx web server – Customer.html

- Created Frontend page as customer.html using html and JavaScript for hitting the REST micro services with hosted the same on web server nginx. – via accessible on port: 80
- Here is Dockerfile : nginx-Dockerfile.zip
- UI Layout :



vlmaz2021tn98.fisdev.local

Not secure | vlmaz2021tn98.fisdev.local/

Customer Forms

First name:

Cust Id

Submit

Account Forms

Account name:

Account Id

Submit

Created Micro Services – Customer and Account REST API

- Created two REST micro services in Java spring boot technology :
  - Customer Micro Service- accessible on port : 9080
  - Account Micro Service- accessible on port : 9081
  - Both services serve as middle tier between Frontend and Database layers
  - Here are Dockerfiles :

customer-DockerFile.zip

account-DockerFile.zip

Created Database in MySQL – springbootdb

- Created MySQL database as a Database layer to save the customer/account service data.
  - Runs on the default mysql port : 3306
- Here is Dockerfile :

mysql-Dockerfile.zip

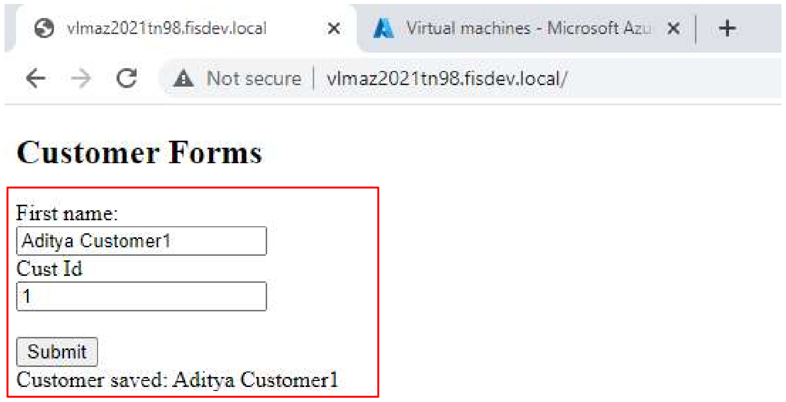
Designed Docker-Compose – docker-compose.yml

- Designed Docker-Compose – docker-compose.yml - to build and deploy all desing layers.
- In the docker-compose.yml- we have declared 4 services :
  - ✓ 1 for building frontend using nginx-dockerfile.
  - ✓ 2 backend Services (account and customer) for building the springboot microservices using account-DockerFile and customer-DockerFile.
  - ✓ 1 Database Service – for builing mysql database using mysql-dockerfile.
- Run command : **docker-compose up –build –d :-** To run the docker-compose.yml file to build and deploy all the services mentioned above.
- Here is docker-compose.yml

docker-compose.zip

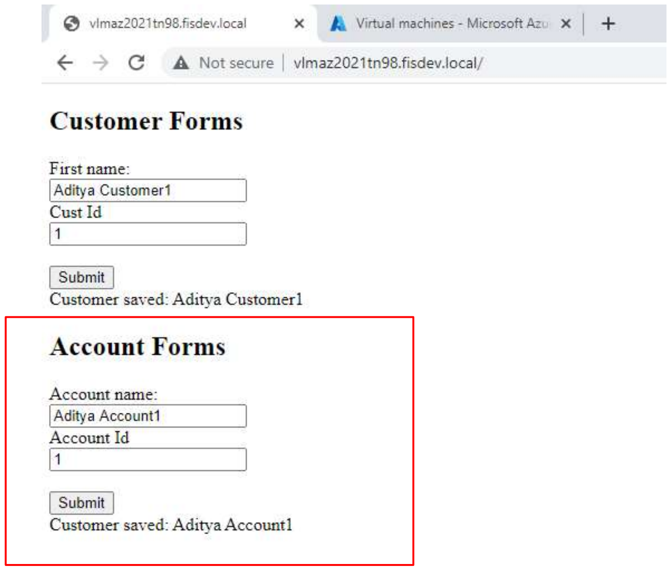
Launch Application and Make Transactions

- Open below URL on local machine and did the registration for a customer.
  - URL : http://vlmaz2021tn89.fisdev.local/
- Enter Customer & Account details and Submit to save data in database


vlmaz2021tn98.fisdev.local
Virtual machines - Microsoft Azu
Not secure
vlmaz2021tn98.fisdev.local/

Customer Forms

First name:
Aditya Customer1
Cust Id
1
Submit
Customer saved: Aditya Customer1


vlmaz2021tn98.fisdev.local
Virtual machines - Microsoft Azu
Not secure
vlmaz2021tn98.fisdev.local/

Customer Forms

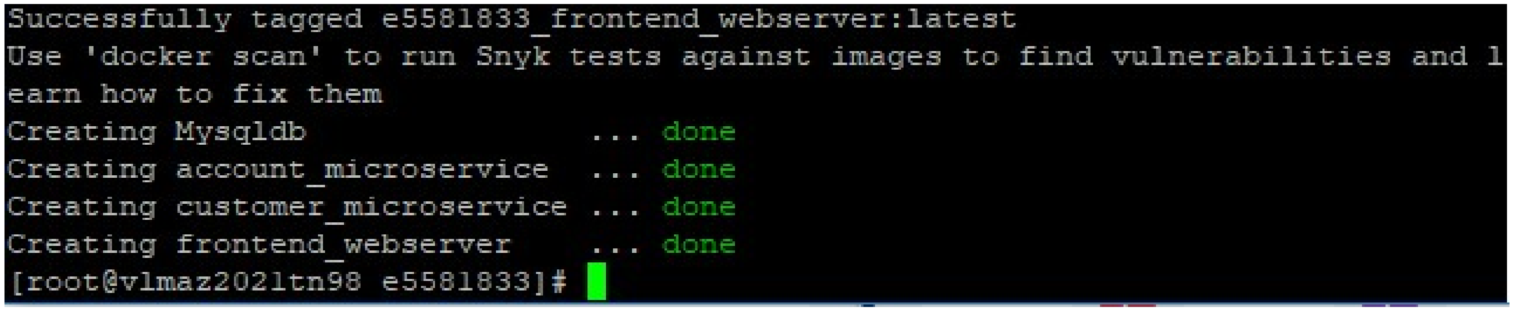
First name:
Aditya Customer1
Cust Id
1
Submit
Customer saved: Aditya Customer1

Account Forms

Account name:
Aditya Account1
Account Id
1
Submit
Customer saved: Aditya Account1

Case Study Project Features #1

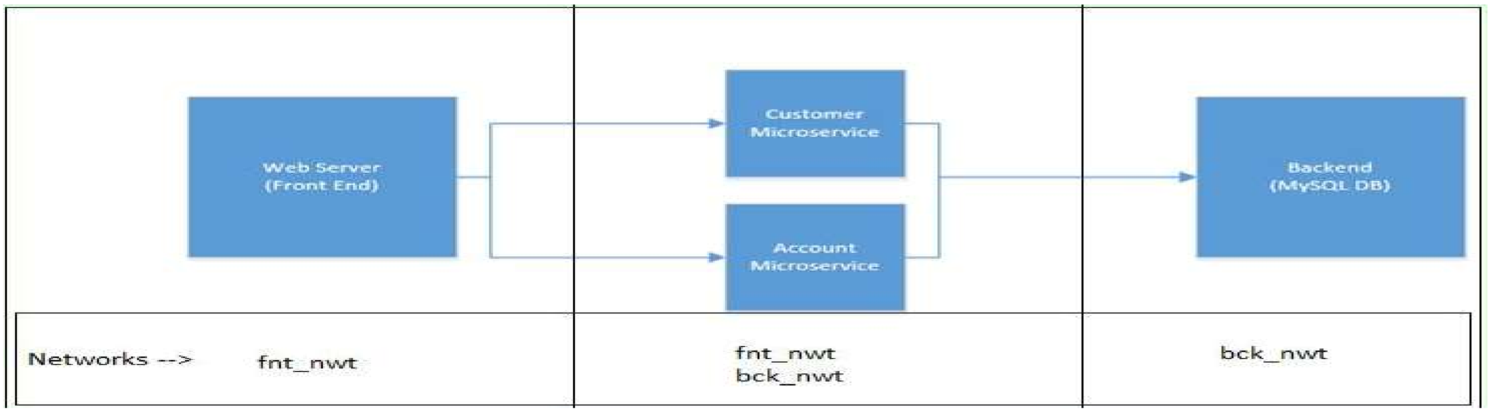
- **Multi Tier Application Architecture**
- Each of the tiers i.e., The web server, microservices and backend run on containers.
  - ✓ Front Layer- html and =javscript –nginx server
  - ✓ Middle Layer- Java Springboot services – inbuilt tomcat server
  - ✓ Database – mysql server



```
Successfully tagged e5581833_frontend_webserver:latest
Use 'docker scan' to run Snyk tests against images to find vulnerabilities and l
earn how to fix them
Creating Mysqldb                  ... done
Creating account_microservice    ... done
Creating customer_microservice ... done
Creating frontend_webserver      ... done
[root@vlmaz2021tn98 e5581833]#
```

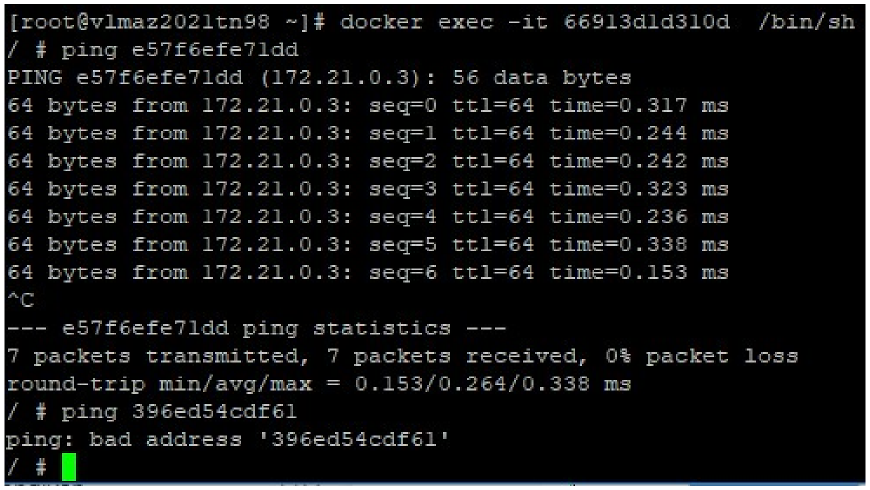
Case Study Project Features #2

- **Network isolation**
  - Web Server cannot directly communicate to backend.
  - Microservice can talk to web server and database.
  - Backend can talk to microservice but not web server directly

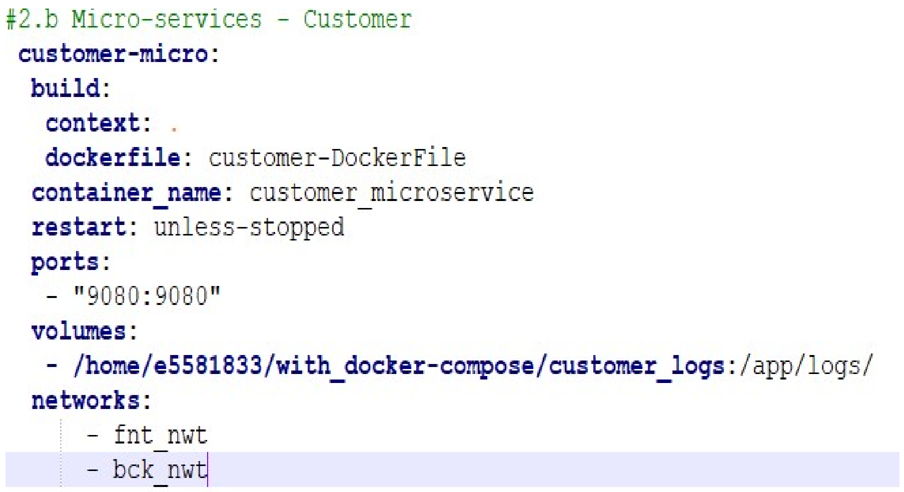


Case Study Project Features #2…Cont.

- **Network isolation**
  - Ping logs from Frontend Server(66913d1d310d) to Database Server [Database] (396ed54cdf61) -ping failing as Front-end network(**fnt_nwt**) can not establish direct contact with database network(**bck_nwt**)- refer docker-compose.yml
  - Ping logs from Frontend Server(66913d1d310d) (**fnt_nwt**) to– Customer Service Container Id Networks(**fnt_nwt**), (**bck_nwt**)- : (e57f6efe71dd)-passing as configured in docker-compose.yml-refer screenshots below



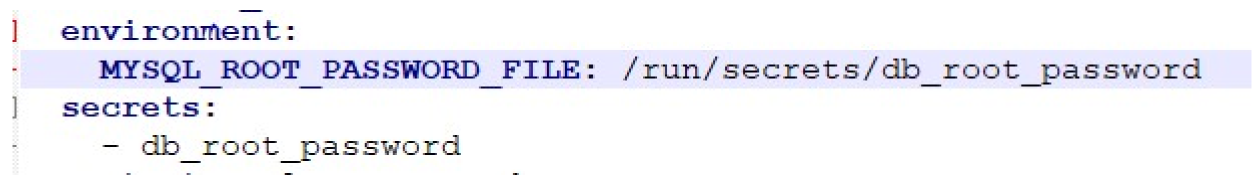
```
[root@vlmaz2021tn98 ~]# docker exec -it 66913d1d310d  /bin/sh
/ # ping e57f6efe71dd
PING e57f6efe71dd (172.21.0.3): 56 data bytes
64 bytes from 172.21.0.3: seq=0 ttl=64 time=0.317 ms
64 bytes from 172.21.0.3: seq=1 ttl=64 time=0.244 ms
64 bytes from 172.21.0.3: seq=2 ttl=64 time=0.242 ms
64 bytes from 172.21.0.3: seq=3 ttl=64 time=0.323 ms
64 bytes from 172.21.0.3: seq=4 ttl=64 time=0.236 ms
64 bytes from 172.21.0.3: seq=5 ttl=64 time=0.338 ms
64 bytes from 172.21.0.3: seq=6 ttl=64 time=0.153 ms
^C
--- e57f6efe71dd ping statistics ---
7 packets transmitted, 7 packets received, 0% packet loss
round-trip min/avg/max = 0.153/0.264/0.338 ms
/ # ping 396ed54cdf61
ping: bad address '396ed54cdf61'
/ #
```



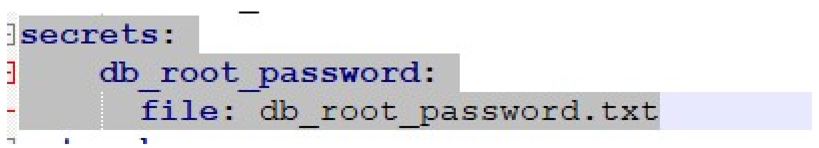
```
#2.b Micro-services - Customer
 customer-micro:
  build:
   context: .
   dockerfile: customer-DockerFile
  container_name: customer_microservice
  restart: unless-stopped
  ports:
   - "9080:9080"
  volumes:
   - /home/e5581833/with_docker-compose/customer_logs:/app/logs/
  networks:
     - fnt_nwt
     - bck_nwt
```

Case Study Project Features #3

- **Security**
  - The credentials used by microservices to access database used from external files saved securely by secrets and which is being called during mySQL image build from docker-compose file through environment variables.



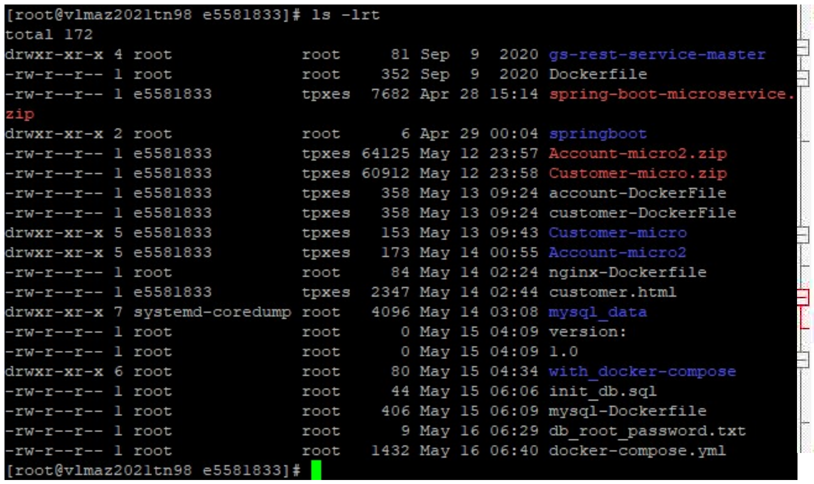
```
environment:
  MYSQL_ROOT_PASSWORD_FILE: /run/secrets/db_root_password
secrets:
  - db_root_password
```



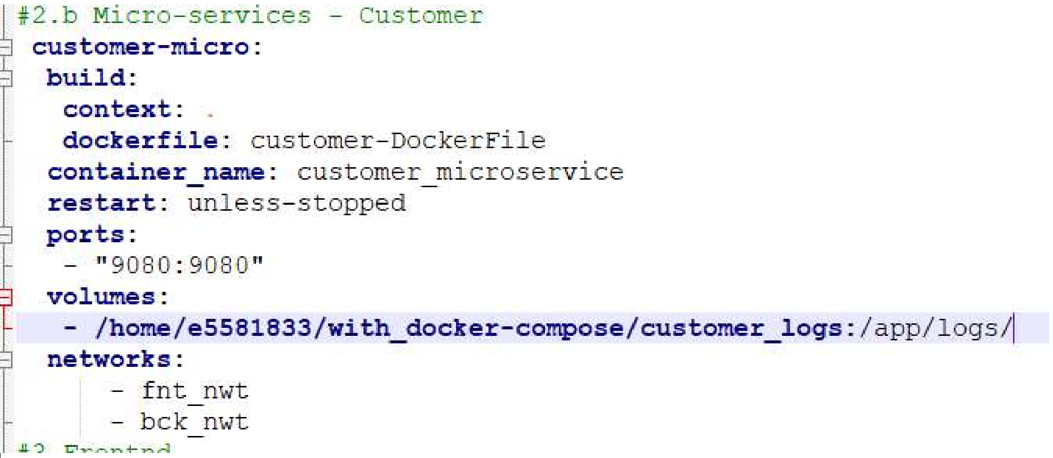
```
secrets:
    db_root_password:
      file: db_root_password.txt
```

Case Study Project Features #4

- **Configuration**
  - We have made the logs of each micro service application as well as frontend application as persistent data by declaring the Volumes in the docker-compose.yml



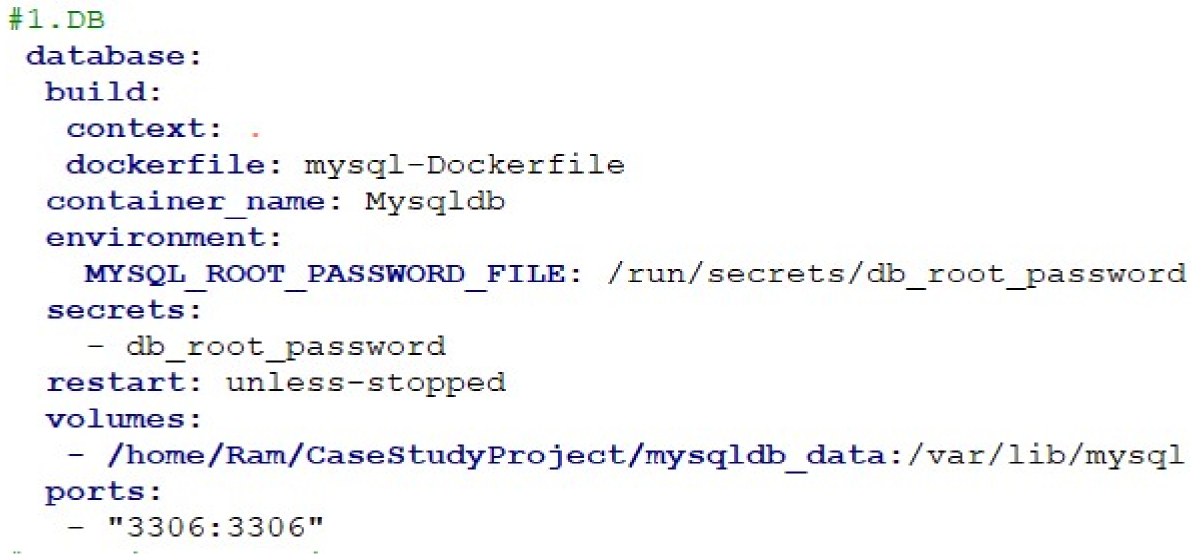
```
[root@vlmaz2021tn98 e5581833]# ls -lrt
total 172
drwxr-xr-x 4 root            root     81 Sep  9  2020 gs-rest-service-master
-rw-r--r-- 1 root            root    352 Sep  9  2020 Dockerfile
-rw-r--r-- 1 e5581833        tpxes  7682 Apr 28 15:14 spring-boot-microservice.
zip
drwxr-xr-x 2 root            root      6 Apr 29 00:04 springboot
-rw-r--r-- 1 e5581833        tpxes 64125 May 12 23:57 Account-micro2.zip
-rw-r--r-- 1 e5581833        tpxes 60912 May 12 23:58 Customer-micro.zip
-rw-r--r-- 1 e5581833        tpxes   358 May 13 09:24 account-DockerFile
-rw-r--r-- 1 e5581833        tpxes   358 May 13 09:24 customer-DockerFile
drwxr-xr-x 5 e5581833        tpxes   153 May 13 09:43 Customer-micro
drwxr-xr-x 5 e5581833        tpxes   173 May 14 00:55 Account-micro2
-rw-r--r-- 1 root            root     84 May 14 02:24 nginx-Dockerfile
-rw-r--r-- 1 e5581833        tpxes  2347 May 14 02:44 customer.html
drwxr-xr-x 7 systemd-coredump root  4096 May 14 03:08 mysql_data
-rw-r--r-- 1 root            root      0 May 15 04:09 version:
-rw-r--r-- 1 root            root      0 May 15 04:09 1.0
drwxr-xr-x 6 root            root     80 May 15 04:34 with_docker-compose
-rw-r--r-- 1 root            root     44 May 15 06:06 init_db.sql
-rw-r--r-- 1 root            root    406 May 15 06:09 mysql-Dockerfile
-rw-r--r-- 1 root            root      9 May 16 06:29 db_root_password.txt
-rw-r--r-- 1 root            root   1432 May 16 06:40 docker-compose.yml
[root@vlmaz2021tn98 e5581833]#
```



```
#2.b Micro-services - Customer
 customer-micro:
  build:
   context: .
   dockerfile: customer-DockerFile
  container_name: customer_microservice
  restart: unless-stopped
  ports:
   - "9080:9080"
  volumes:
   - /home/e5581833/with_docker-compose/customer_logs:/app/logs/
  networks:
      - fnt_nwt
      - bck_nwt
#3 Frontnd
```

Case Study Project Features #5

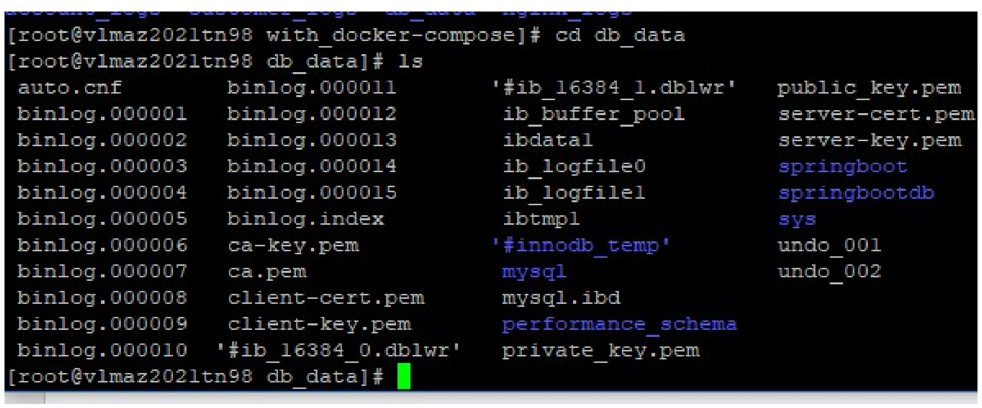
- **Data**
  - We have used Volumes for persistent the data of mySQL server also defined folders for storage path in the docker-compose.yml



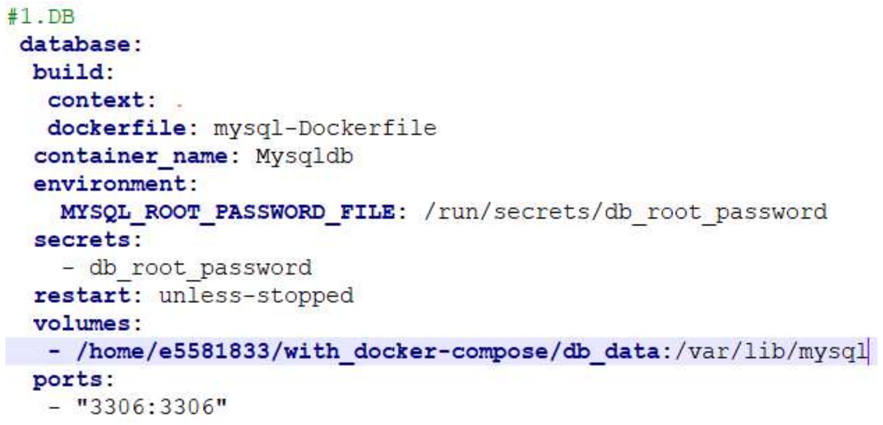
```
#1.DB
 database:
  build:
   context: .
   dockerfile: mysql-Dockerfile
  container_name: Mysqldb
  environment:
    MYSQL_ROOT_PASSWORD_FILE: /run/secrets/db_root_password
  secrets:
    - db_root_password
  restart: unless-stopped
  volumes:
   - /home/Ram/CaseStudyProject/mysqldb_data:/var/lib/mysql
  ports:
   - "3306:3306"
```

Case Study Project Features #5 …Cont.

- **Database : mysqldb_data**
  - Below is snapshot of application database : mysqldb_data



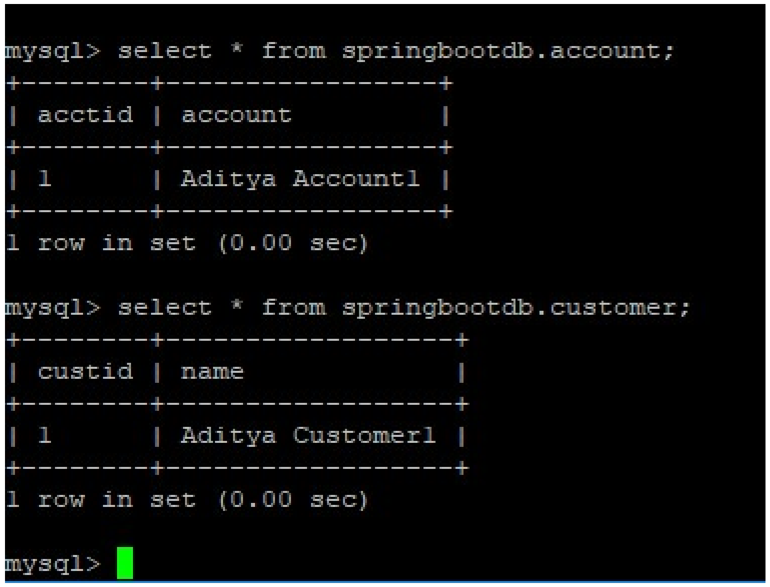
```
[root@vlmaz2021tn98 with_docker-compose]# cd db_data
[root@vlmaz2021tn98 db_data]# ls
 auto.cnf          binlog.000011        '#ib_16384_1.dblwr'   public_key.pem
 binlog.000001     binlog.000012         ib_buffer_pool       server-cert.pem
 binlog.000002     binlog.000013         ibdata1              server-key.pem
 binlog.000003     binlog.000014         ib_logfile0          springboot
 binlog.000004     binlog.000015         ib_logfile1          springbootdb
 binlog.000005     binlog.index          ibtmp1               sys
 binlog.000006     ca-key.pem           '#innodb_temp'        undo_001
 binlog.000007     ca.pem                mysql                undo_002
 binlog.000008     client-cert.pem       mysql.ibd
 binlog.000009     client-key.pem        performance_schema
 binlog.000010    '#ib_16384_0.dblwr'    private_key.pem
[root@vlmaz2021tn98 db_data]#
```



```
#1.DB
 database:
  build:
   context: .
   dockerfile: mysql-Dockerfile
  container_name: Mysqldb
  environment:
    MYSQL_ROOT_PASSWORD_FILE: /run/secrets/db_root_password
  secrets:
    - db_root_password
  restart: unless-stopped
  volumes:
   - /home/e5581833/with_docker-compose/db_data:/var/lib/mysql
  ports:
   - "3306:3306"
```

Case Study Project Features #5 …Cont.

- **Application Saved Data**
  - Below is snapshot of saved data via Frontend into database.



```
mysql> select * from springbootdb.account;
+--------+-----------------+
| acctid | account         |
+--------+-----------------+
| 1      | Aditya Account1 |
+--------+-----------------+
1 row in set (0.00 sec)

mysql> select * from springbootdb.customer;
+--------+------------------+
| custid | name             |
+--------+------------------+
| 1      | Aditya Customer1 |
+--------+------------------+
1 row in set (0.00 sec)

mysql>
```

Thank You