

IST769 Homework Submission

Name: **Sathish Kumar Rajendiran**

SUID: **666555028**

Email: **srajendi@syr.edu**

Date Due: **07/13/2021**

Homework #:1

Exercise(s):

1. What would be the command to bring up the redis environment? How is the command different from the mssql environment? How is it the same?

The command used to bring up the redis environment was “**docker-compose up -d**”. I was able to start the Redis instance on Docker container following the same. However, it has to be executed having the directory under **/srajendi/adv-db-labs/redis**. This command created a network for “redis_default” and it was running on **port 6379**. It used the image from docker compose.

on MSSQL, command is also the same but the directory under the execution was on **/srajendi/adv-db-labs/mssql**. In addition, mssql image was pulled from the yaml file and built the image from that repository. This reference is updated on the yml file. Later, this command created a network for “mssql_default” and it was running on default **port 1433**.

I was also able to run both services on the docker container.

Evidence:

The image shows two separate windows of the TextPad text editor. Both windows have the title bar "TextPad - [C:\Users\LocalAdmin\srajendi\adv-db-labs\mssql\docker-compose.yml]" and "TextPad - [C:\Users\LocalAdmin\srajendi\adv-db-labs\redis\docker-compose.yml]".

The left window (mssql) contains the following YAML configuration:

```
version: '3'
services:
  mssql:
    image: "mcr.microsoft.com/mssql/server:2017-latest"
    #build: mssql/
    container_name: mssql
    environment:
      - "ACCEPT_EULA=Y"
      - "SA_PASSWORD=SU20range!"
    ports:
      - "1433:1433"
    volumes:
      - mssql-data:/var/opt/mssql
      - C:/root
  volumes:
    mssql-data:
```

The right window (redis) contains the following YAML configuration:

```
version: "3"
services:
  redis:
    image: redis
    restart: always
    hostname: redis
    ports:
      - 6379:6379
```

769-Win10Docker-srajendi

Administrator: Command Prompt

```
C:\Users\LocalAdmin\srajendi\adv-db-labs\redis>docker-compose ps
Name      Command     State    Ports
-----
C:\Users\LocalAdmin\srajendi\adv-db-labs\redis>docker-compose up -d
Creating network "redis_default" with the default driver
Creating redis_redis_1 ... done
C:\Users\LocalAdmin\srajendi\adv-db-labs\redis>
```

769-Win10Docker-srajendi

The screenshot shows the Docker desktop application interface. At the top, there's a blue header bar with the Docker logo. Below it, the main window has a sidebar on the left labeled 'Containers / Apps' with options for 'Images', 'Volumes', and 'Dev Environments'. The 'Dev Environments' section includes a 'PREVIEW' button. The main pane displays a list of containers. One container, named 'redis', is shown as 'RUNNING'. It has a green icon and the name 'redis' next to it. To the right of the container list is a search bar with the placeholder 'Search...'. At the bottom of the main pane, there's a terminal window showing the logs for the 'redis' container. The logs include Redis startup messages such as 'Redis is starting', 'Redis version=6.2.0', and 'Ready to accept connections'.

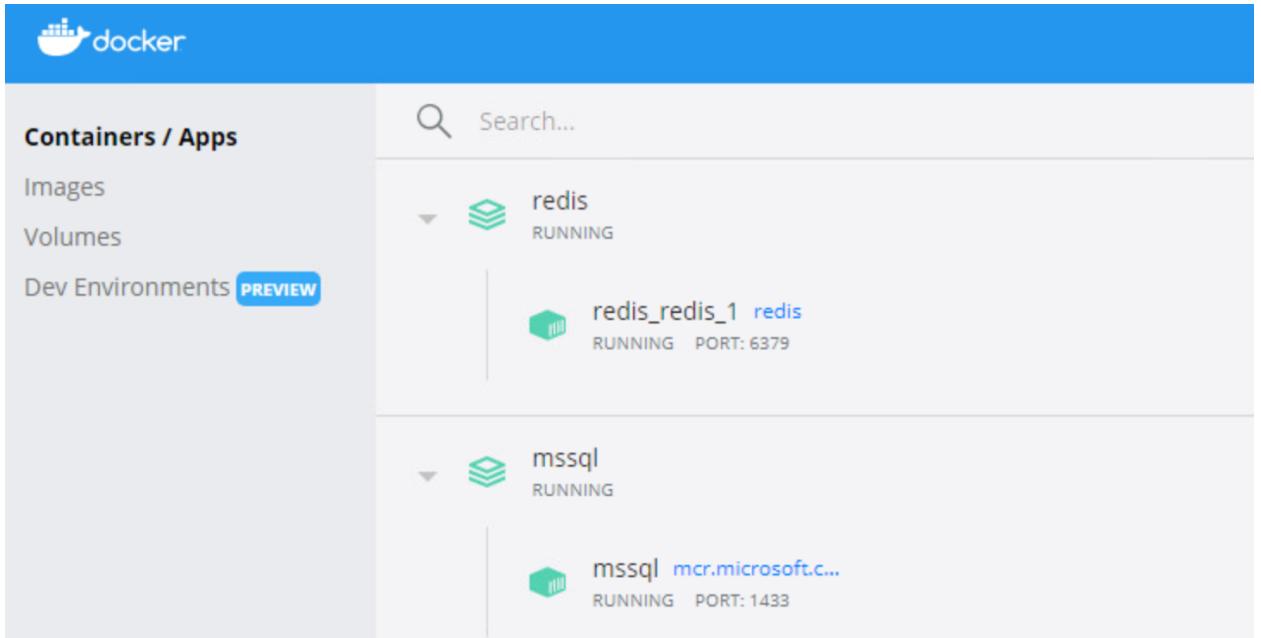
```
Attaching to redis_redis_1
redis_1 | 1:C 13 Jul 2021 02:18:17.052 # o000e000e000 Redis is starting o000e000e000
redis_1 | 1:C 13 Jul 2021 02:18:17.052 # Redis version=6.2.0, bits=64, commit=00000000,
redis_1 | modified=0, pid=1, just started
redis_1 | 1:C 13 Jul 2021 02:18:17.052 # Warning: no config file specified, using the
redis_1 | default config. In order to specify a config file use redis-server /path/to/redis.conf
redis_1 | 1:M 13 Jul 2021 02:18:17.054 * monotonic clock: POSIX clock_gettime
redis_1 | 1:M 13 Jul 2021 02:18:17.055 * Running mode=standalone, port=6379.
redis_1 | 1:M 13 Jul 2021 02:18:17.055 # Server initialized
redis_1 | 1:M 13 Jul 2021 02:18:17.055 * Ready to accept connections
```

```
C:\Users\LocalAdmin\srajendi\adv-db-labs\mssql>docker-compose ps
Name      Command     State    Ports
-----
C:\Users\LocalAdmin\srajendi\adv-db-labs\mssql>docker-compose up -d
Pulling mssql (mcr.microsoft.com/mssql/server:2017-latest)...
2017-latest: Pulling from mssql/server
61e03ba1d414: Pull complete
4afb39f216bd: Pull complete
e489abdc9f90: Pull complete
999ffff7bcc24: Pull complete
ef0b62888e99: Pull complete
9464681f0202: Pull complete
884f6b75bc0d: Pull complete
aea6c44dc7a9: Pull complete
Digest: sha256:046fdb8988b92a9da973d2c4a237a769afee9010d79d76dd807b930a3d0e56e9
Status: Downloaded newer image for mcr.microsoft.com/mssql/server:2017-latest
Creating mssql ... done

C:\Users\LocalAdmin\srajendi\adv-db-labs\mssql>
C:\Users\LocalAdmin\srajendi\adv-db-labs\redis>cd..
C:\Users\LocalAdmin\srajendi\adv-db-labs>cd mssql
C:\Users\LocalAdmin\srajendi\adv-db-labs\mssql>docker-compose ps
Name      Command     State    Ports
-----
C:\Users\LocalAdmin\srajendi\adv-db-labs\mssql>docker-compose up -d
Creating network "mssql_default" with the default driver
Creating mssql ... done

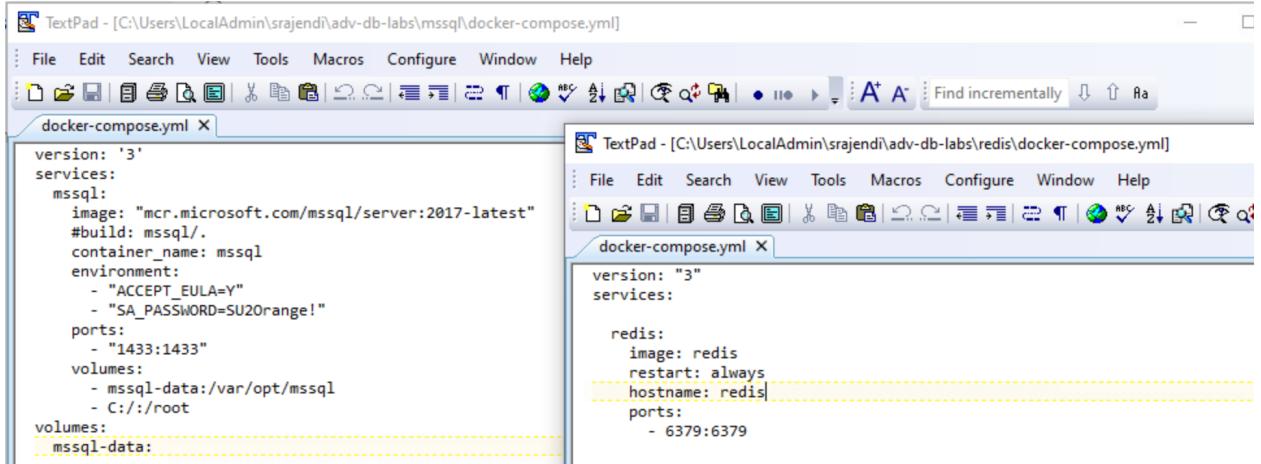
C:\Users\LocalAdmin\srajendi\adv-db-labs\mssql>docker-compose ps
Name          Command           State           Ports
mssql        /opt/mssql/bin/nonroot_msg ...   Up      0.0.0.0:1433->1433/tcp,:::1433->1433/tcp

C:\Users\LocalAdmin\srajendi\adv-db-labs\mssql>
```



2. Where is the specific configuration information about each environment stored?

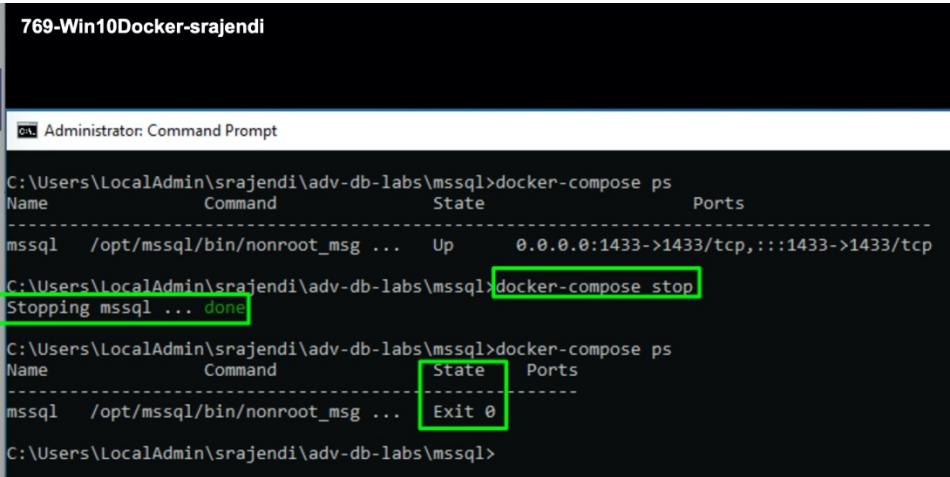
Configuration information about each environment is stored in “**docker-compose.yml**” file under respective components/services folder.



3. Explain the difference between stopping an environment and bringing it down. Elaborate with use-cases for each.

- Stopping an environment:
 - Docker command: **docker-compose stop**; Whenever the services are **up**, executing a **stop** command would bring the services to halt state and the resources allocated would

be released immediately. However, the data collected, and components created are still associated to that image and can be resumed later. This would also update the current state of the service to “Exit 0”. i.e., graciously stopping the service.



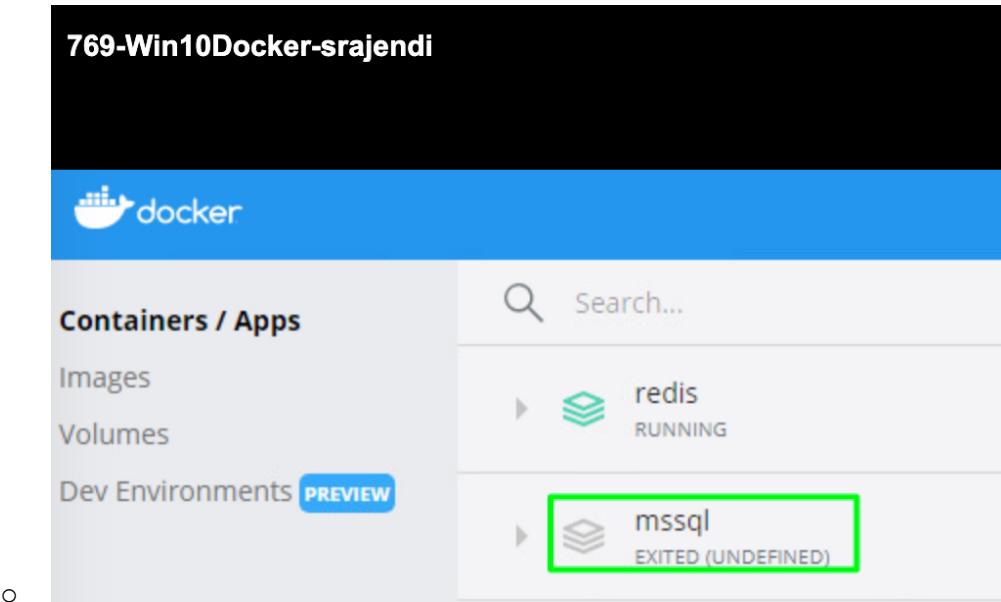
```

769-Win10Docker-srajendi

Administrator: Command Prompt
C:\Users\LocalAdmin\srajendi\adv-db-labs\mssql>docker-compose ps
Name          Command           State          Ports
mssql        /opt/mssql/bin/nonroot_msg ...   Up      0.0.0.0:1433->1433/tcp,:::1433->1433/tcp
C:\Users\LocalAdmin\srajendi\adv-db-labs\mssql>docker-compose stop
Stopping mssql ... done

C:\Users\LocalAdmin\srajendi\adv-db-labs\mssql>docker-compose ps
Name          Command           State          Ports
mssql        /opt/mssql/bin/nonroot_msg ...   Exit 0
C:\Users\LocalAdmin\srajendi\adv-db-labs\mssql>

```



- bringing it down: **docker-compose down**
 - Whenever the services are **stopped**, executing a “**down**” command would remove the image and default network created on that port immediately. It’s a hard delete of the docker image and all the data would be lost with a successful execution. There won’t be any information on the current status if we execute the “**ps**” command.
 - On another hand, if we execute “**down**” command while having the service in **running** mode, it does 3 steps
 - stopping the service
 - removing the image
 - removing the network created.

769-Win10Docker-srajendi

```
Administrator: Command Prompt
C:\Users\LocalAdmin\srajendi\adv-db-labs\mssql>docker-compose ps
Name          Command           State    Ports
-----
mssql        /opt/mssql/bin/nonroot_msg ...   Exit 0

C:\Users\LocalAdmin\srajendi\adv-db-labs\mssql>docker-compose down
Removing mssql ... done
Removing network mssql_default

C:\Users\LocalAdmin\srajendi\adv-db-labs\mssql>docker-compose ps
Name      Command     State    Ports
-----
```

○ 769-Win10Docker-srajendi



Containers / Apps

Images

Volumes

Dev Environments PREVIEW



Search...

▶ redis
RUNNING

769-Win10Docker-srajendi

```
Administrator: Command Prompt

C:\Users\LocalAdmin\srajendi\adv-db-labs\mssql>cd ..

C:\Users\LocalAdmin\srajendi\adv-db-labs>cd redis

C:\Users\LocalAdmin\srajendi\adv-db-labs\redis>docker-compose ps
  Name          Command           State      Ports
-----  -----
redis_redis_1   docker-entrypoint.sh redis ...   Up        0.0.0.0:6379->6379/tcp,:::6379->6379/tcp

C:\Users\LocalAdmin\srajendi\adv-db-labs\redis>docker-compose stop
Stopping redis_redis_1 ... done

C:\Users\LocalAdmin\srajendi\adv-db-labs\redis>docker-compose ps
  Name          Command           State      Ports
-----  -----
redis_redis_1   docker-entrypoint.sh redis ...   Exit 0

C:\Users\LocalAdmin\srajendi\adv-db-labs\redis>docker-compose up -d
Starting redis_redis_1 ... done

C:\Users\LocalAdmin\srajendi\adv-db-labs\redis>docker-compose ps
  Name          Command           State      Ports
-----  -----
redis_redis_1   docker-entrypoint.sh redis ...   Up        0.0.0.0:6379->6379/tcp,:::6379->6379/tcp

C:\Users\LocalAdmin\srajendi\adv-db-labs\redis>docker-compose down
Stopping redis_redis_1 ... done
Removing redis_redis_1 ... done
Removing network redis_default

C:\Users\LocalAdmin\srajendi\adv-db-labs\redis>docker-compose ps
Name  Command  State  Ports
-----
```

○ C:\Users\LocalAdmin\srajendi\adv-db-labs\redis>

4. What happens when you bring up an environment that is already up?

- start the service: **docker-compose up -d**; When the service is running, executing the “up -d” command would retain the same session. It doesn’t really cause any changes to the execution state. However, it may look for any upgrades available on the image between the time it was created vs current time and might update the image. I also believe there won’t be any data/network loss as all other would remain the same. In our case, it didn’t make any impact as it returned with a status as “mssql is up-to-date”

769-Win10Docker-srajendi

```
C:\Users\LocalAdmin\srajendi\adv-db-labs\mssql>docker-compose ps
Name      Command     State    Ports
-----
C:\Users\LocalAdmin\srajendi\adv-db-labs\mssql>docker-compose up -d
Creating network "mssql_default" with the default driver
Creating mssql ... done
C:\Users\LocalAdmin\srajendi\adv-db-labs\mssql>docker-compose ps
Name      Command     State    Ports
-----
```

Name	Command	State	Ports
mssql	/opt/mssql/bin/nonroot_msg ...	Up	0.0.0.0:1433->1433/tcp,:::1433->1433/tcp

```
C:\Users\LocalAdmin\srajendi\adv-db-labs\mssql>docker-compose up -d
mssql is up-to-date
C:\Users\LocalAdmin\srajendi\adv-db-labs\mssql>
```

○ 769-Win10Docker-srajendi

The screenshot shows the Docker desktop application window. At the top, there's a blue header bar with the Docker logo. Below it, the main interface has a sidebar on the left with options: 'Containers / Apps', 'Images', and 'Volumes'. On the right, there's a search bar labeled 'Search...'. The main area displays two running containers:

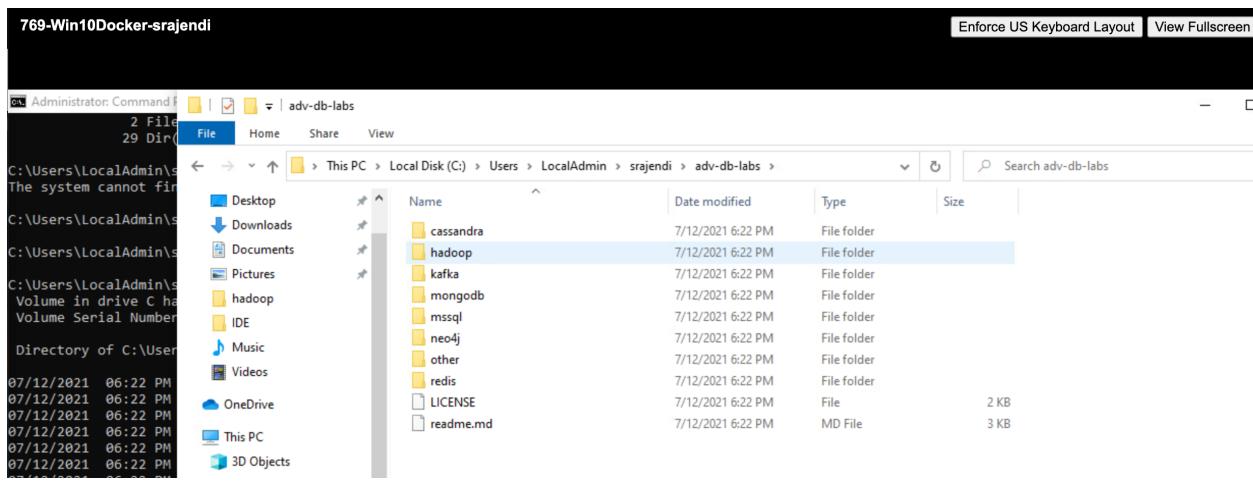
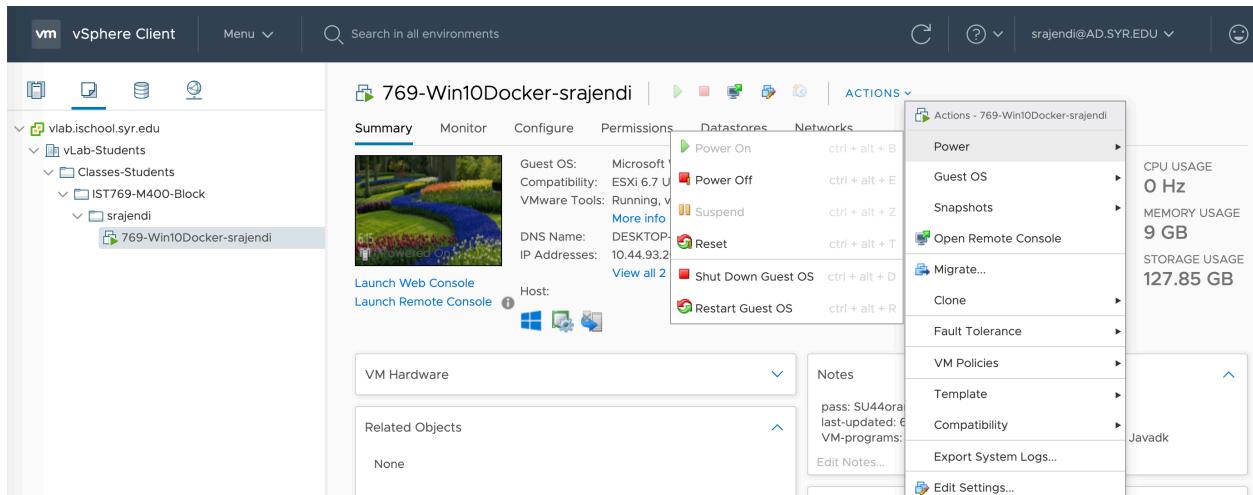
- redis**: Status is RUNNING, indicated by a green icon.
- mssql**: Status is RUNNING, indicated by a blue icon.

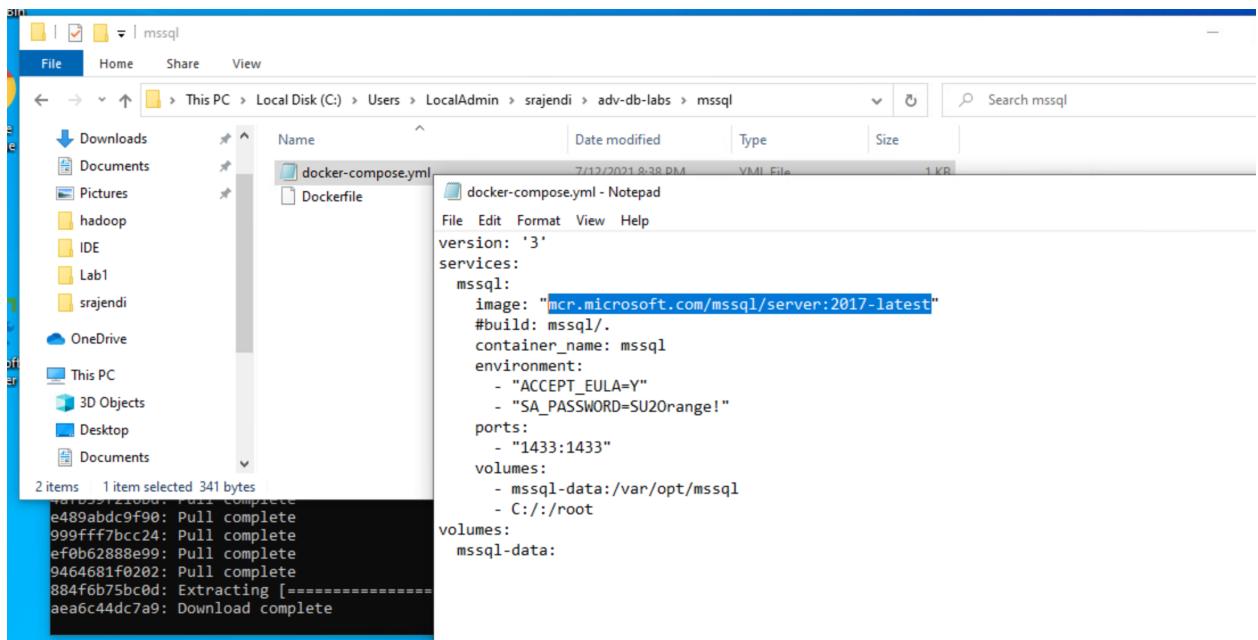
5. What was the most difficult aspect of this lab? What changes could be made to make it less difficult?

I don't think it was challenging. Maybe it would have been better if we had few hands on for Redis. I personally haven't used Redis database in the past. Only other challenge was with mssql image version. But one of the students had mentioned about the workaround to fix it. So, it was quite straightforward.

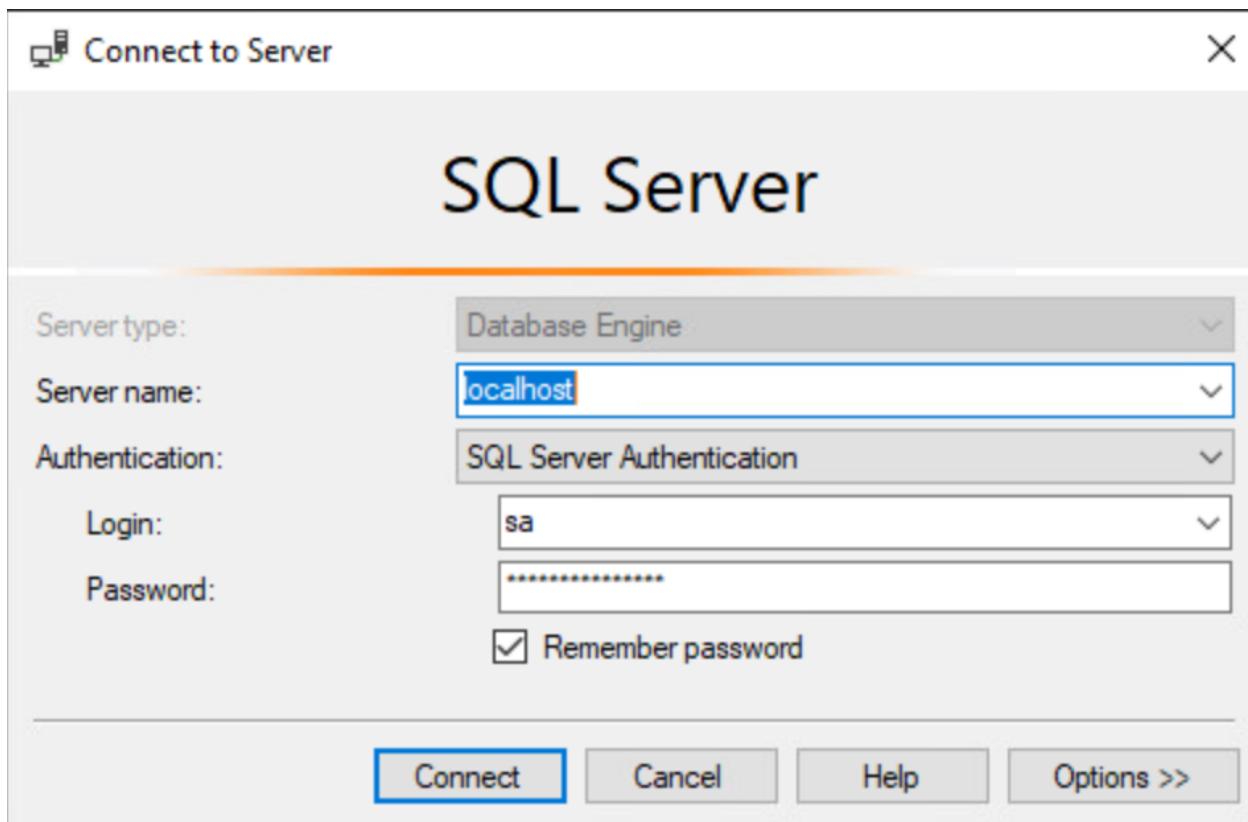
As long as vlab instance on iSchool – I am quite comfortable with this exercise. However, I would like to try to the same on my mac machine. I am not a huge fan of SQL Workbench. So, preferred MSSQL on the vlab.

Appendix:





```
C:\Users\LocalAdmin\srajendi\adv-db-labs\mssql>docker-compose ps
Name      Command     State    Ports
-----
C:\Users\LocalAdmin\srajendi\adv-db-labs\mssql>docker-compose up -d
Pulling mssql (mcr.microsoft.com/mssql/server:2017-latest)...
2017-latest: Pulling from mssql/server
61e03ba1d414: Pull complete
4afb39f216bd: Pull complete
e489abdc9f90: Pull complete
99ffff7bcc24: Pull complete
ef0b62888e99: Pull complete
9464681f0202: Pull complete
884f6b75bc0d: Pull complete
aea6c44dc7a9: Pull complete
Digest: sha256:046fdb8988b92a9da973d2c4a237a769afee9010d79d76dd807b930a3d0e56e9
Status: Downloaded newer image for mcr.microsoft.com/mssql/server:2017-latest
Creating mssql ... done
C:\Users\LocalAdmin\srajendi\adv-db-labs\mssql>
```



The screenshot shows the Microsoft SQL Server Management Studio interface. The title bar reads 'LabHW1_srajendi.sql - localhost.master (sa (52)) - Microsoft SQL Server Management Studio'. The menu bar includes File, Edit, View, Query, Project, Tools, Window, Help. The toolbar has various icons for database management. The Object Explorer sidebar shows 'master'. The main window displays a query results grid with the following data:

	Value
Title	: Lab Homework1: The Lab Environment
Author	: Sathish Kumar Rajendiran
Course	: IST769 M400
Term	: July, 2021

Below the grid, the text '*****' is displayed. The next few rows show the following text:

```
1. Create the fudgemart and demo databases
```

```
32 | IF NOT EXISTS (SELECT * FROM sys.databases WHERE name='fudgemart_v3')
33 | BEGIN
34 |     CREATE DATABASE fudgemart_v3
35 | END
36 | GO
37 |
38 | IF NOT EXISTS (SELECT * FROM sys.databases WHERE name='demo')
39 | BEGIN
40 |     CREATE DATABASE demo
41 | END
42 | GO
43 | SELECT name FROM sys.databases
```

Results

	name
1	master
2	tempdb
3	model
4	msdb
5	fudgemart_v3
6	demo

Query executed successfully.

fudgemart_v3_Scripts.sql - localhost.fudgemart_v3 (sa (54)) - Microsoft SQL Server Management Studio

```
1 USE [fudgemart_v3]
2 GO
3 /****** Object: Table [dbo].[fudgemart_creditcards]      Script Date: 5/31/2016 10:52:4
4 SET ANSI_NULLS ON
5 GO
6 SET QUOTED_IDENTIFIER ON
7 GO
```

Messages

(1 row affected)

(1 row affected)

(1 row affected)

Query executed successfully.

fudgemart_v3_Scripts.sql - localhost.fudgemart_v3 (sa (54)) - M

File Edit View Project Tools Window Help

New Query MDX D

fudgemart_v3 Execute

Object Explorer

Connect

- localhost (SQL Server 14.0.3048.4 - sa)
 - Databases
 - System Databases
 - Database Snapshots
 - + demo
 - + fudgemart_v3
 - Database Diagrams
 - Tables
 - System Tables
 - FileTables
 - External Tables
 - Graph Tables
 - dbo.fudgemart_creditcards
 - dbo.fudgemart_customer_creditcards
 - dbo.fudgemart_customers
 - dbo.fudgemart_departments_lookup
 - dbo.fudgemart_employee_timesheets
 - dbo.fudgemart_employees
 - dbo.fudgemart_jobtitles_lookup
 - dbo.fudgemart_order_details
 - dbo.fudgemart_orders
 - dbo.fudgemart_products
 - dbo.fudgemart_shipvia_lookup
 - dbo.fudgemart_vendors
 - Views

```
C:\Users\LocalAdmin\srajendi\adv-db-labs\mssql>docker-compose ps
Name          Command           State           Ports
-----
mssql        /opt/mssql/bin/sqlservr   Up      0.0.0.0:1433->1433/tcp,:::1433->1433/tcp

C:\Users\LocalAdmin\srajendi\adv-db-labs\mssql>docker-compose down -v --rmi all
Stopping mssql ... done
Removing mssql ... done
Removing network mssql_default
Removing volume mssql_mssql-data
Removing image microsoft/mssql-server-linux:2017-latest

C:\Users\LocalAdmin\srajendi\adv-db-labs\mssql>
```

The screenshot shows the Docker desktop application interface. On the left, there's a sidebar with icons for recycle bin, Google Chrome, Microsoft SQL Server, and Docker Desktop. The main area is titled "Containers / Apps" and contains a search bar. Below it, two containers are listed: "redis" and "mssql", both marked as "EXITED (UNDEFINED)".

The screenshot shows the Docker desktop application interface. On the left, there's a sidebar with icons for recycle bin, Google Chrome, Microsoft SQL Server, and Docker Desktop. The main area is titled "Containers / Apps" and contains a search bar. In the center, there's a large blue cube icon with the text "No containers running". Below it, a message says "Try running a container: Copy and paste this command into your terminal and then come back" followed by a command line input field containing "docker run -d -p 80:80 docker/getting-started".

