CA 3

Docker & Setting up Cassandra

Für:

Big Data Engineering & Analysis (SS22)

Prof. Dr. Oliver Hummel

Bei:

Milad Afshar Jahanshahi

2150426

Docker:

Docker is a set of platform as a service products that use OS-level virtualization to deliver software in packages called containers. The software that hosts the containers is called Docker Engine.

Instead of creating the environment for future use or duplication or any use such use, Docker can be used, where user can directly pull the image and run it, in spite of creating the whole environment and running the required software.

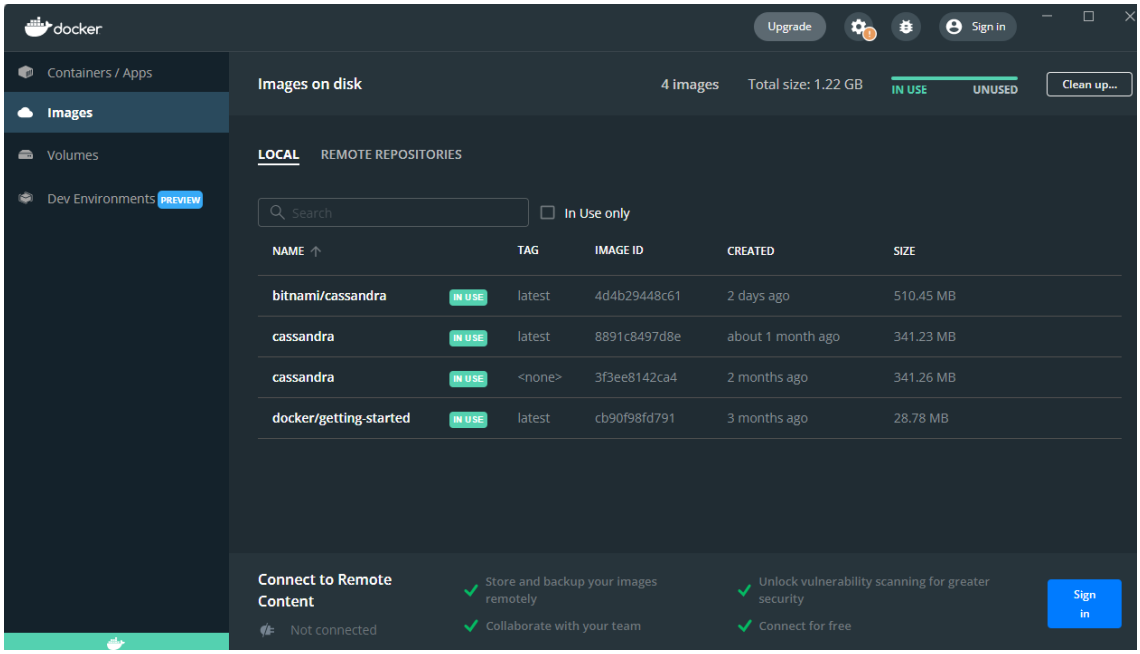
Getting Started with Docker

Once I've downloaded the Docker from their official website with default configuration, and started the Docker, it gave option to search for images, either locally or through remote repositories. It also gave edge to run the Docker commands with Docker GUI or through windows Command Prompt.

Then I've written the command

`docker pull Cassandra`

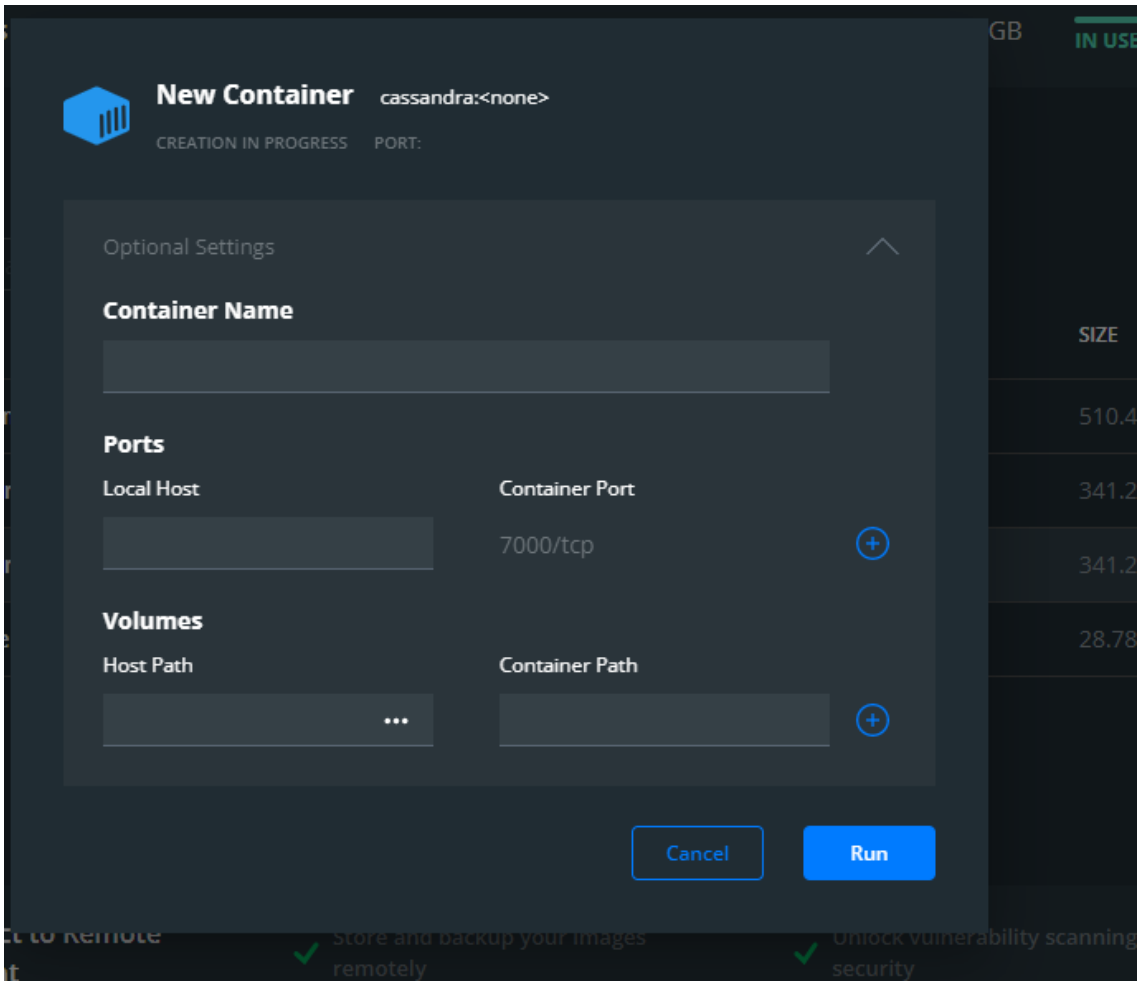
It fetched the latest Cassandra pre ready image from the Cassandra's official website. Once fetching is completed, it also started showing in the Docker GUI



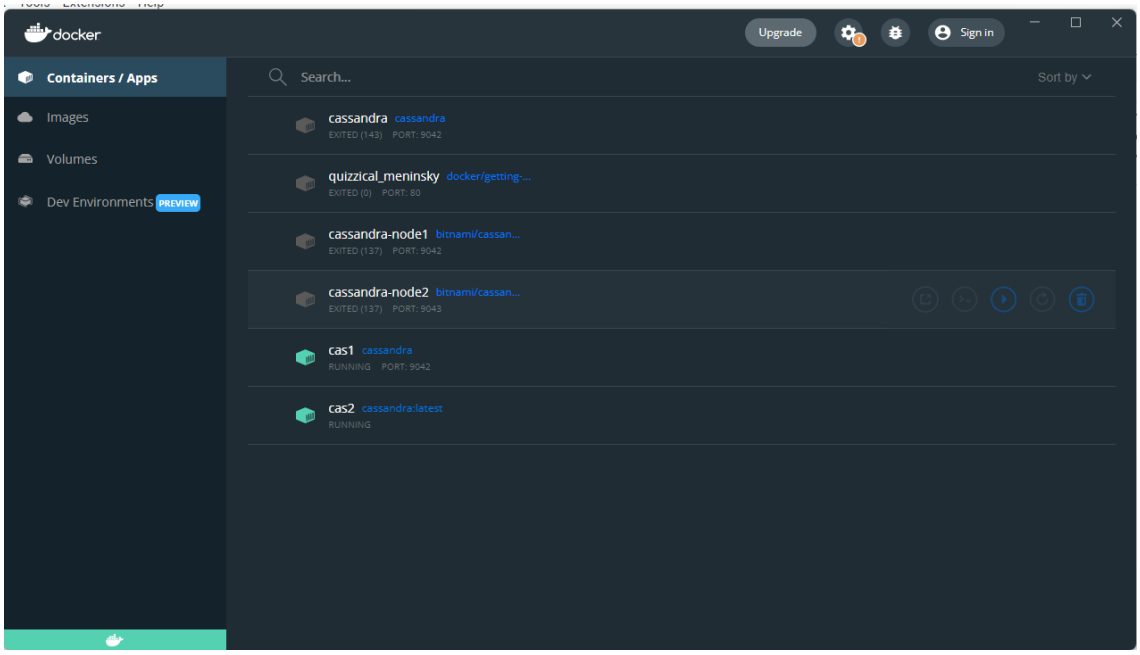
Running the Image

Once the image is fetched, it can be directly run through hovering the cursor on the image I want to start, like in the image below.

When I pressed the RUN button, it showed an other popup to configure the image if want to change any default setting.



From where user have to name the container for the running image, and network details (if network required), then the container will start running and also be seen through the Container Tab in Docker GUI



Here CAS1 and CAS2 , two containers are running in above image.

Creating 2 Node Cluster of Cassandra

In order to create the Cassandra 2 Node cluster, there can be several ways, but I've used the simple CLI based method to create it on windows Command Prompt

First of all I've downloaded the Docker, configured and installed it. After it I've pulled the official Cassandra Image and verified it by running its single instance.

Once the pre-reqs are done, I've created a directory named, data in the system holding the two sub directories node1 and node2.

```
→  data tree -a
.
├── node1
└── node2
2 directories, 1 file
```

After creating the directories, I ran the query below in CMD windows,

```
docker run --name cas1 -p 9042:9042 -v
/Development/PetProjects/CassandraTut/data/node1:/var/lib/cassandra/data -e
CASSANDRA_CLUSTER_NAME=MyCluster -e
CASSANDRA_ENDPOINT_SNITCH=GossipingPropertyFileSnitch -e
CASSANDRA_DC=datacenter1 -d cassandra
```

the command written above, created the docker container of Cassandra which I've downloaded in the previous step, which is listening to the 9042 port and is stored in C:/data/node1 directory of my system. The above command have also created the cluster with the name of MyCluster and and also a named datacenter1 in that cluster.

Once this command ran successfully,

I ran the command below which showed node status, showing that it is created and up working fine.

```
➜  data docker exec -it cas1  nodetool status
Datacenter: datacenter1
=======================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address     Load       Tokens       Owns (effective)  Host ID                               Rack
UN  172.17.0.2  103.67 KiB  256          100.0%            bcb57440-7303-4849-9afc-af0237587870  rack1
```

After this, I ran the second query written below,

```
docker run --name cas2 -v
C:/data/node2:/var/lib/cassandra/data -e
CASSANDRA_SEEDS="$(docker inspect --format='{{ .NetworkSettings.IPAddress }}' cas1)" -e
CASSANDRA_CLUSTER_NAME=MyCluster -e
CASSANDRA_ENDPOINT_SNITCH=GossipingPropertyFileSnitch -e
CASSANDRA_DC=datacenter1 -d cassandra:latest
```

This command created the second node, with the name of CAS2, in the directory C:/data/node2

With the network configuration used in the first node. The above query also fetched the other details required to create the cluster automatically from the first node, CAS1.

Once the above, command ran successfully, I've also verified the cluster by the command below

```
PS C:\data> docker exec -it cas1 nodetool status
Datacenter: datacenter1
=======================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address     Load        Tokens  Owns  Host ID                               Rack
UN  172.17.0.3  159.29 MiB  16      ?     52408e10-acab-42fd-8638-8838ae262f1e  rack1
UN  172.17.0.2  158.98 MiB  16      ?     e9c3ad5f-8388-474f-a7f8-07641b78c7d3  rack1
```

As the above image is showing that both nodes are up and synced together.

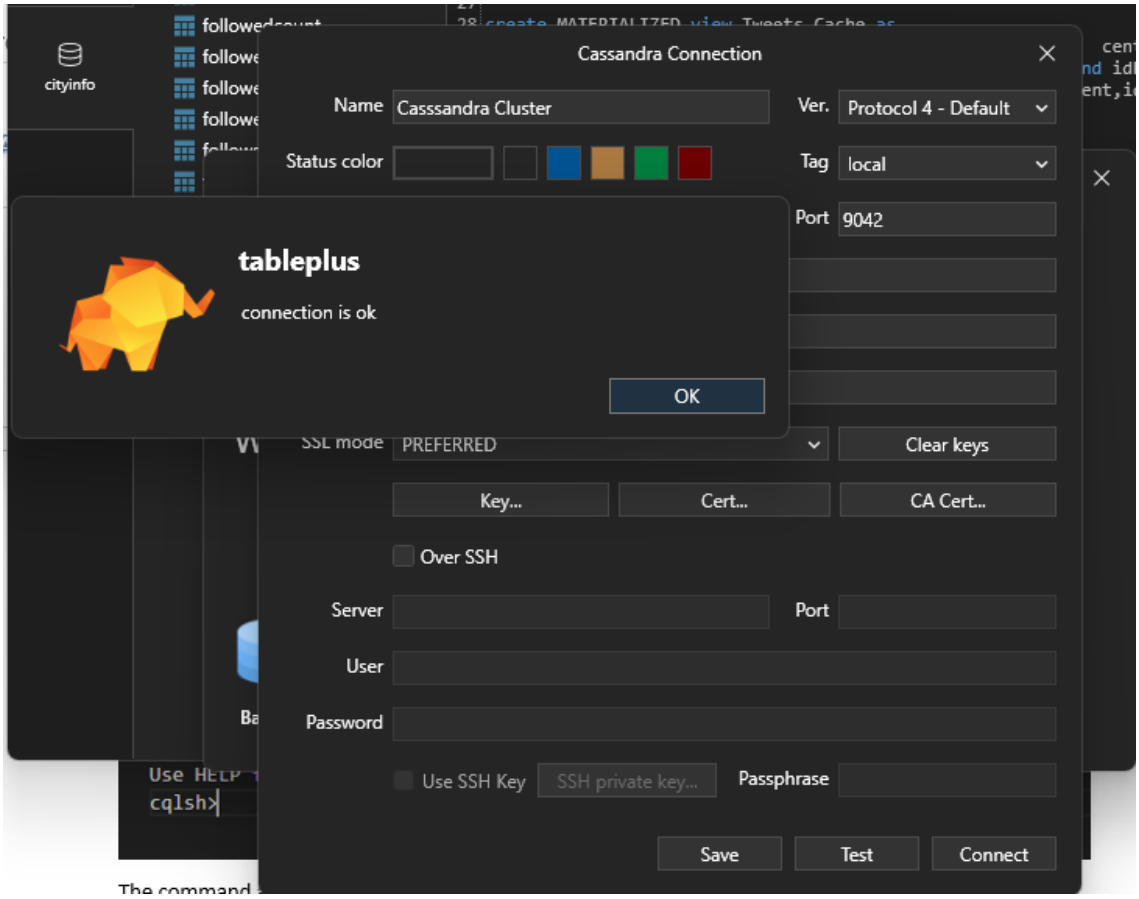
In order to connect to the CQL, the below command is used.

```
➜  CassandraTut docker exec -it cas2  cqlsh
Connected to MyCluster at 127.0.0.1:9042.
[cqlsh 5.0.1 | Cassandra 3.11.4 | CQL spec 3.4.4 | Native protocol v4]
Use HELP for help.
cqlsh>
```

The command above, we can also use the same command but with cas1 to connect to the cluster.

Cassandra Connectivity Test

Once the Cassandra cluster is up, I've also tested it with Tableup query tool, in order to verify its connectivity, it also worked fine.



Keyspace

In cassnadra, databases are reffered as key spaces with in the Cassandra, like in sql server or my sql Databases are called.

In order to create the new key space, I've used the command below,

```
[cqlsh 6.0.0 | Cassandra 4.0.4 | CQL spec 3.4.5 | Native protocol v5]
Use HELP for help.
cqlsh> create keyspace twitter_dataset with replication = {'class' : 'SimpleStrategy', 'replication_factor':2}
```

With the replication factor setted to two for two nodes.

Then I've verified that this key space is created or not with the command,

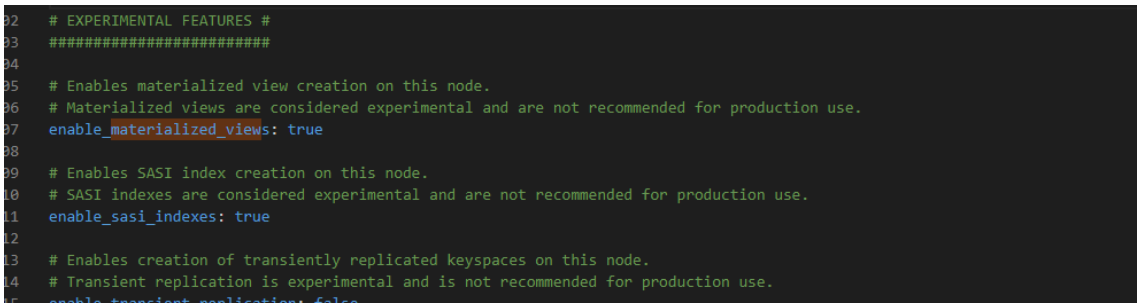
```
cqlsh> describe keyspaces;

cityinfo    system_auth         system_traces          twitter_dataset
mykeyspace  system_distributed  system_views
system      system_schema       system_virtual_schema
```

And in the above image, result can be seen and verified as twitter_dataset is created and shown in top right of the above image.

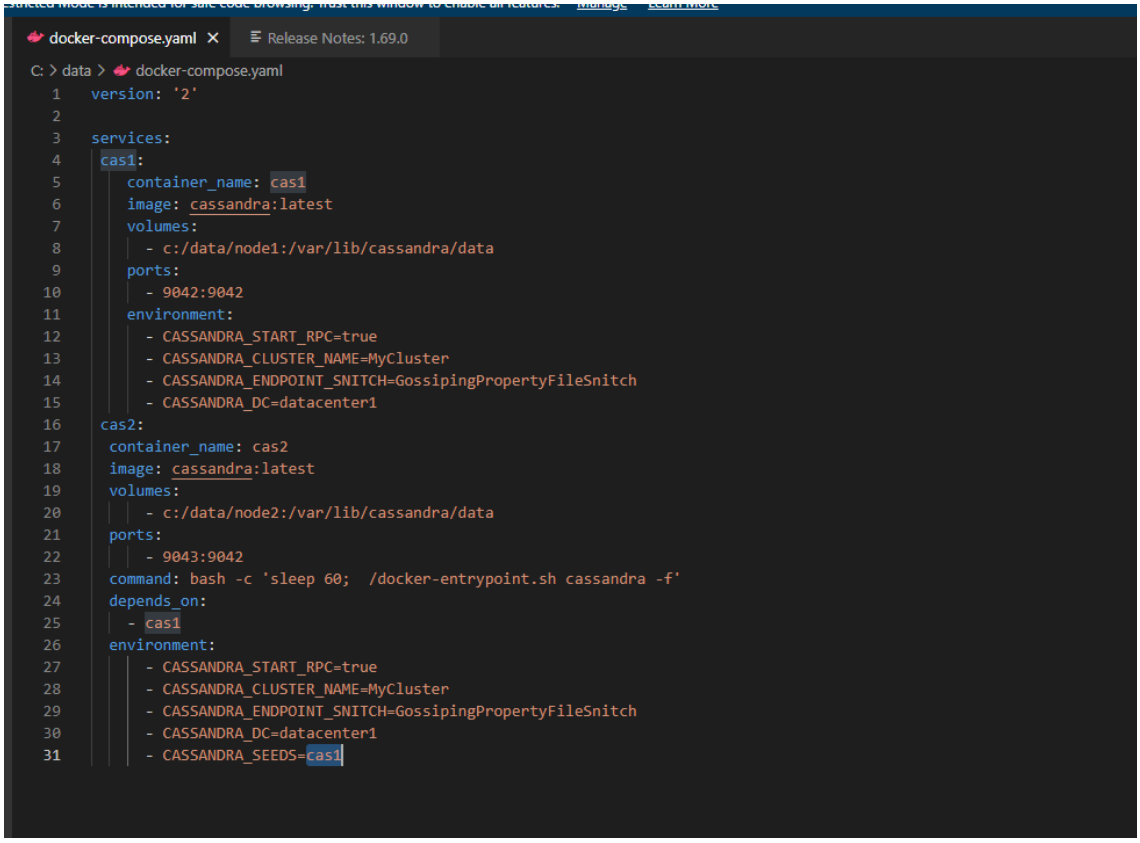
Updating Cassandra.yaml file

I've also updated the Cassandra.yaml file to activate the materialized view and SASI indexes, by setting their values to true in the file.



For second time

If was have to do the all in second time, I've created the docker-compose.yaml file to do the whole process. In that case, I only have to download and install the docker and pull the image. Then in the required directly, I'll place this file, and run the it.


docker-compose.yaml
Release Notes: 1.69.0
C: > data > docker-compose.yaml
version: '2'
services:
cas1:
container_name: cas1
image: cassandra:latest
volumes:
- c:/data/node1:/var/lib/cassandra/data
ports:
- 9042:9042
environment:
- CASSANDRA_START_RPC=true
- CASSANDRA_CLUSTER_NAME=MyCluster
- CASSANDRA_ENDPOINT_SNITCH=GossipingPropertyFileSnitch
- CASSANDRA_DC=datacenter1
cas2:
container_name: cas2
image: cassandra:latest
volumes:
- c:/data/node2:/var/lib/cassandra/data
ports:
- 9043:9042
command: bash -c 'sleep 60; /docker-entrypoint.sh cassandra -f'
depends_on:
- cas1
environment:
- CASSANDRA_START_RPC=true
- CASSANDRA_CLUSTER_NAME=MyCluster
- CASSANDRA_ENDPOINT_SNITCH=GossipingPropertyFileSnitch
- CASSANDRA_DC=datacenter1
- CASSANDRA_SEEDS=cas1