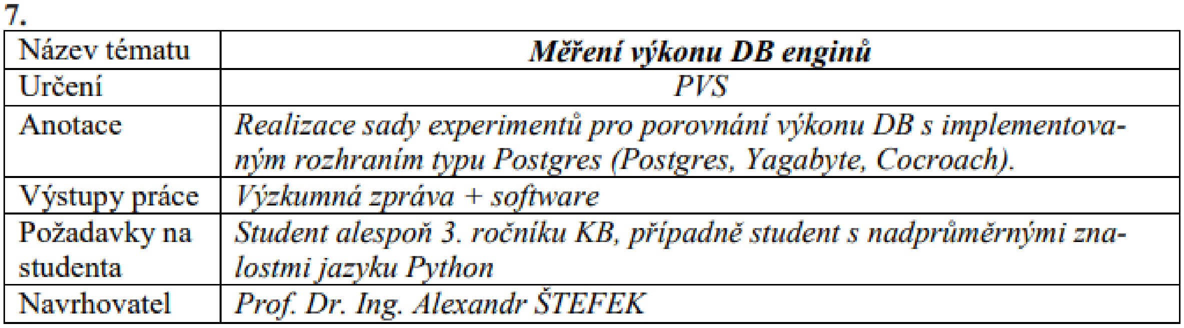
Report 2: DBMS - Postgresql, Cockroach, Yugabyte - Basic

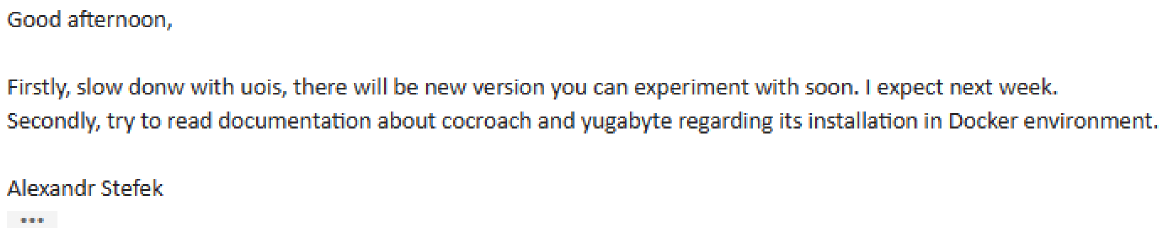


7.

Název tématu	***Měření výkonu DB enginů***
Určení	*PVS*
Anotace	*Realizace sady experimentů pro porovnání výkonu DB s implementovaným rozhranim typu Postgres (Postgres, Yagabyte, Cocroach).*
Výstupy práce	*Výzkumná zpráva + software*
Požadavky na studenta	*Student alespoň 3. ročniku KB, případně student s nadprůměrnými znalostmi jazyku Python*
Navrhovatel	*Prof. Dr. Ing. Alexandr ŠTEFEK*

How I understand what I have to do:

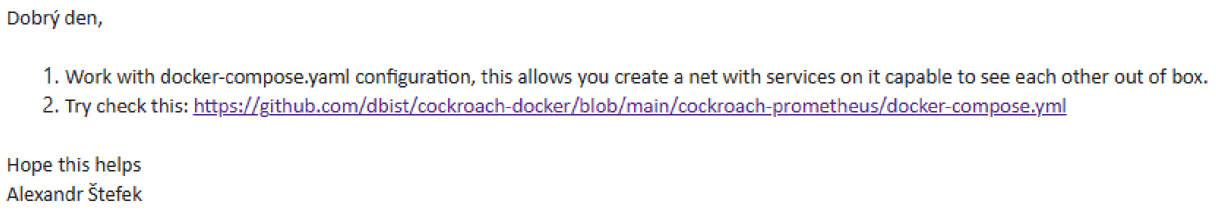
Topic Title:	Measuring the **Performance** of **DB Engines**
Application:	PVS
Annotation	Implementation of a **set of experiments to compare** the **performance of DB** with the implemented interface of the Postgres type (Postgres, Yagabyte, Cocroach).
Outputs	Research Report + Software
Student Requirements	Student of at least the 3rd year of KB, or a student with above-average knowledge of Python
Applicant:	Prof. Dr. Ing. Alexandr ŠTEFEK



Good afternoon,

Firstly, slow donw with uois, there will be new version you can experiment with soon. I expect next week.
Secondly, try to read documentation about cocroach and yugabyte regarding its installation in Docker environment.

Alexandr Stefek



Dobrý den,

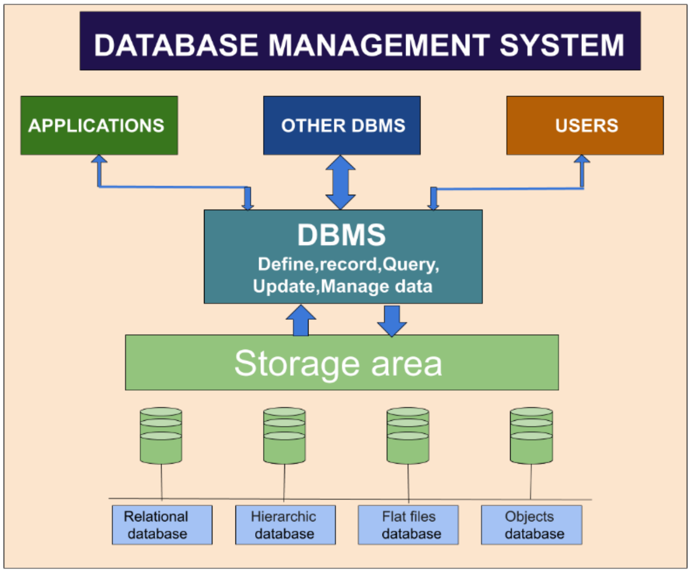
1. Work with docker-compose.yaml configuration, this allows you create a net with services on it capable to see each other out of box.
2. Try check this: https://github.com/dbist/cockroach-docker/blob/main/cockroach-prometheus/docker-compose.yml

Hope this helps
Alexandr Štefek

1. Definitions

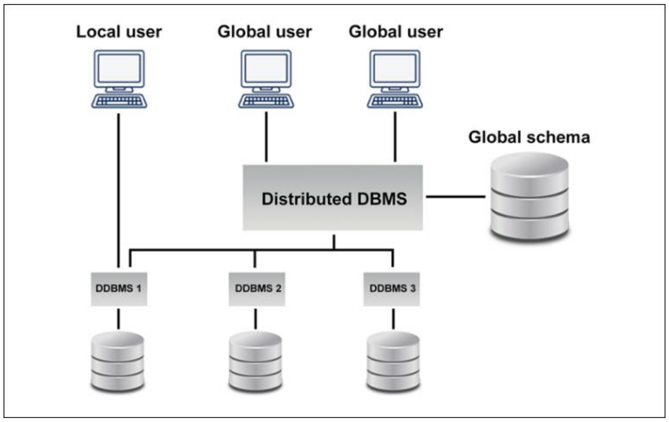
What is DBMS?

Database Management Systems (DBMS) are software systems used to store, retrieve, and run queries on data. A DBMS serves as an interface between an end-user and a database, allowing users to create, read, update, and delete data in the database.



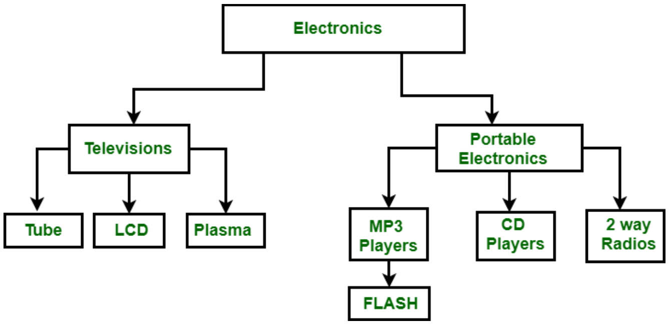
Distributed database management system

A distributed DBMS is a set of logically interrelated databases distributed over a network that is managed by a centralized database application. This type of DBMS synchronizes data periodically and ensures that any change to data is universally updated in the database.

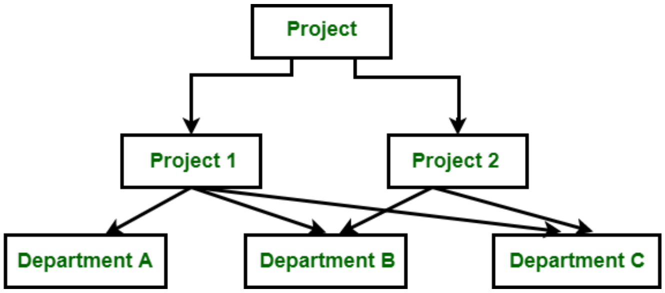


Hierarchical database management system

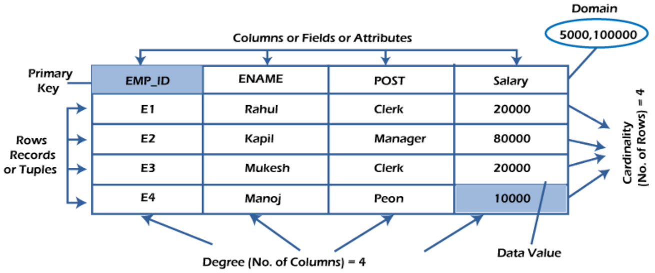
Hierarchical databases organize model data in a tree-like structure. Data storage is either a top-down or bottom-up format and is represented using a parent-child relationship.



Network database management system
The network database model addresses the need for more complex relationships by allowing each child to have multiple parents. Entities are organized in a graph that can be accessed through several paths.



Relational database management system
Relational database management systems (RDBMS) are the most popular data model because of its user-friendly interface. It is based on normalizing data in the rows and columns of the tables. This is a viable option when you need a data storage system that is scalable, flexible, and able to manage lots of information.


Domain
5000,100000
Columns or Fields or Attributes
Primary Key
Rows Records or Tuples
Cardinality (No. of Rows) = 4
Degree (No. of Columns) = 4
Data Value

EMP_ID	ENAME	POST	Salary
E1	Rahul	Clerk	20000
E2	Kapil	Manager	80000
E3	Mukesh	Clerk	20000
E4	Manoj	Peon	10000

1.2 DB-Engines

A database engine is a component of software that facilitates working with databases. It controls how information in a database is saved, retrieved, and altered. Database engines act as a bridge between the database's data and the user, facilitating interaction with the information stored there

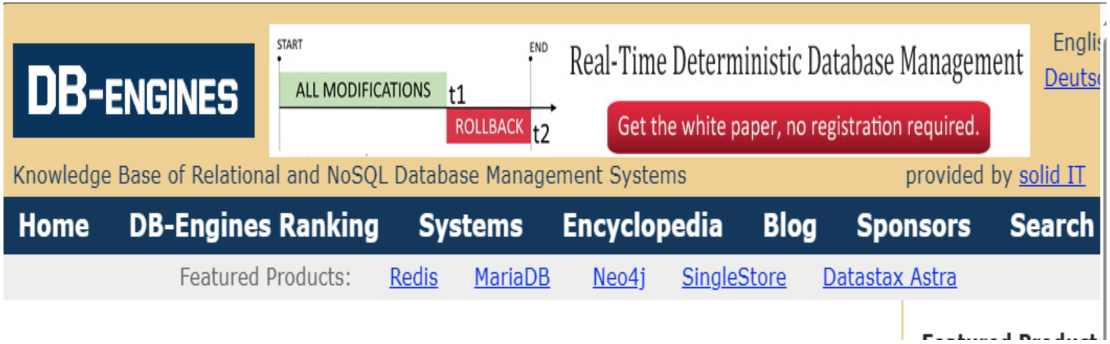
A database engine might handle multiple users, transactions, throughput, ….. **As I mentioned in previous report** , buffers and caches, ACID (atomicity, consistency, isolation, durability), as well as different isolation levels.

- A read may pull data from memory, remote databases, and multiple tables on disk processing it using SQL through multiple explicit and/or implicit code paths in order to present it to the requesting application.
- A create may allocate storage, provision structures, assign values, and do it's own processing before storing data. Etc.

DB-Engines website

DB-Engines is an initiative to collect and present information on database management systems (DBMS)..
The DB-Engines Ranking is a list of DBMS ranked by their current popularity. The list is updated monthly.

DB-Engines has been created and is maintained by solid IT - an Austrian IT consulting company with a special focus on software development, consulting and training for database-centric applications.



Method of calculating the scores of the DB-Engines Ranking

DB-Engines Ranking - Trend of CockroachDB vs. PostgreSQL vs. YugabyteDB Popularity

CockroachDB vs. PostgreSQL vs. YugabyteDB Comparison (db-engines.com)

Relational Database Engines

Relational database engines are the most widely used type of database engines.

Relational database engines store data in tables, with each table representing a different entity or object. Each table has a unique primary key that is used to identify the rows in the table. Tables can be related to each other using foreign keys, which are used to establish relationships between tables.

To retrieve data from a relational database, users use SQL queries. SQL is a standard language used to manipulate relational databases. SQL queries are used to select, insert, update, and delete data from the database.

Performance.

Relational databases perform well with intensive read/write operations on small to medium datasets. They also offer improved speed of data retrieval by adding indexes to data fields to query and join tables. However, the performance may suffer when the amount of data and user requests grows.

2. Ideas from case studies:

Mainly, studies use the same criteria to determine performance: the execution time of the different operations

The operations choosed in the studies are **INSERT, DELETE, UPDATE,** and **SELECT**. These four operations are the primary operations of the DBMSs. Therefore, the performance of different DBMSs is based on these operations.

While another research studies the performance in terms of throughput and latency. The performance is measured by measuring throughput and latency for four queries.

- **S1**: testing these operations with **increasing number of records**. This study is based on 1, 100, 500, 1 000, 5 000, 10 000, 25 000, and 50 000 records. The study shows that the execution time differs in databases when executing with different numbers of records.

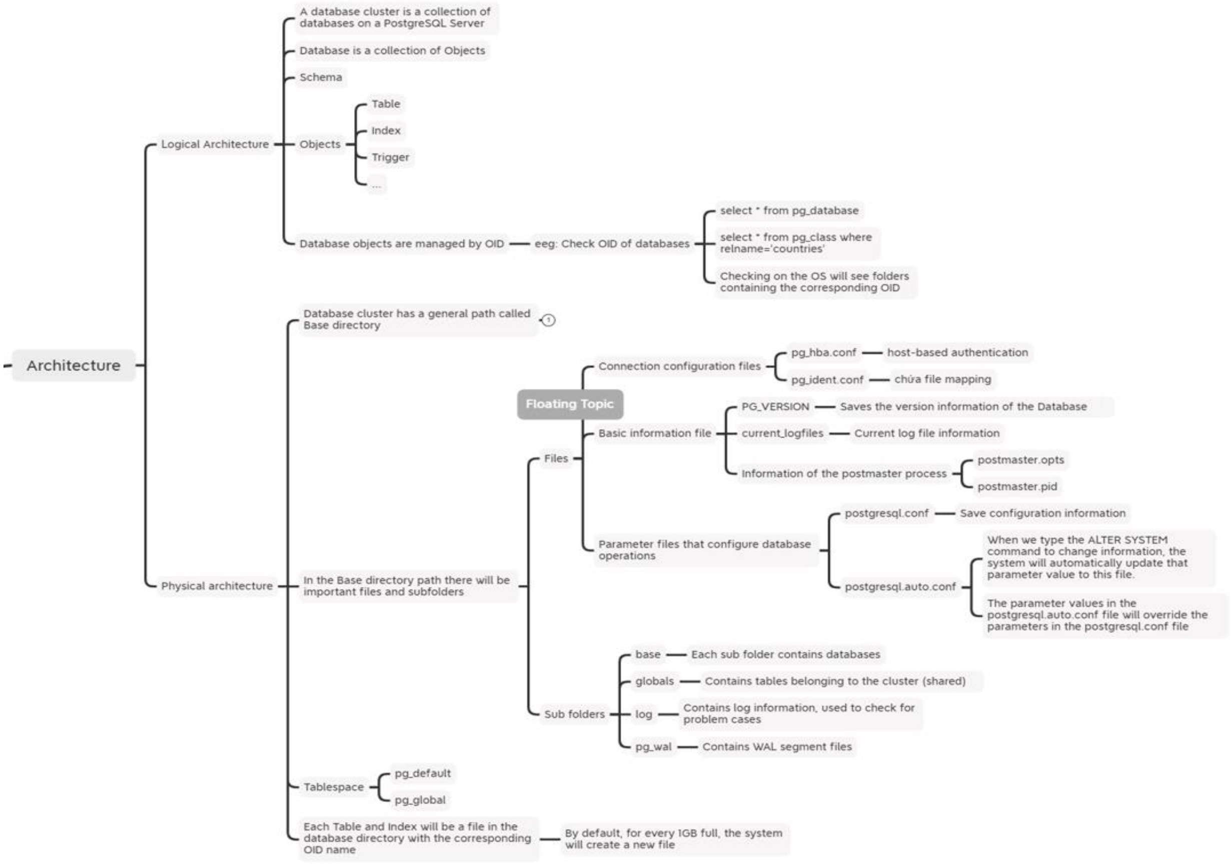
- **S2**: the number of records varies from 1 000 to 100 000. The study examined the databases by conducting tests on the CRUD operations, creating, reading, updating, and deleting. **Each test has been conducted five times on the same operation and the same number of records.** Finally, the result is entirely based on the **average execution times of the tests**. Furthermore, the operations were conducted to change and update this information.
- **S3**: Examines the performance by conducting tests on operations of **insertion and search.**
- **S4**: databases contain more than 100 000 records The execution time of these 30 sets for the four operations has been presented. The differences are slight in terms of the execution time for these sets. Therefore the average execution time for each operation has been presented. This execution average time shows the differences between the operations.

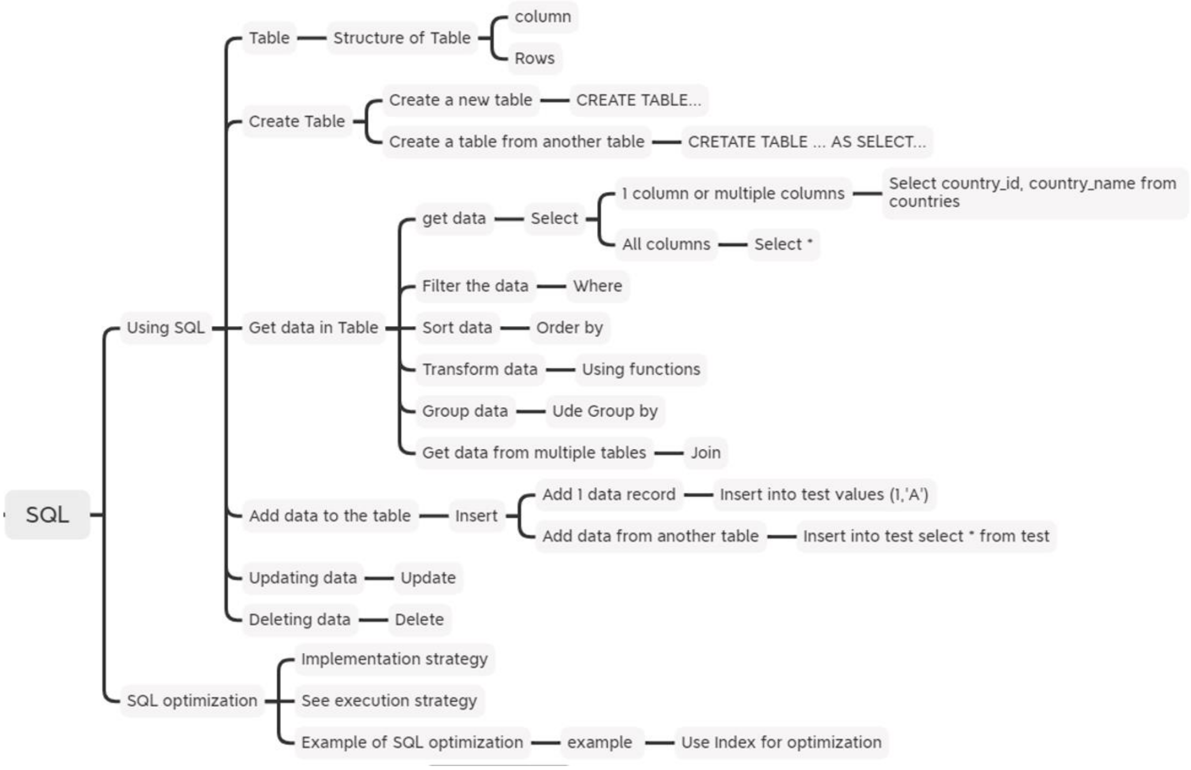
3. Jump in Postgresql, Cockroach, Yugabyte - Basic Introduce + Installation- preparing for the tests

3.1 Postgresql

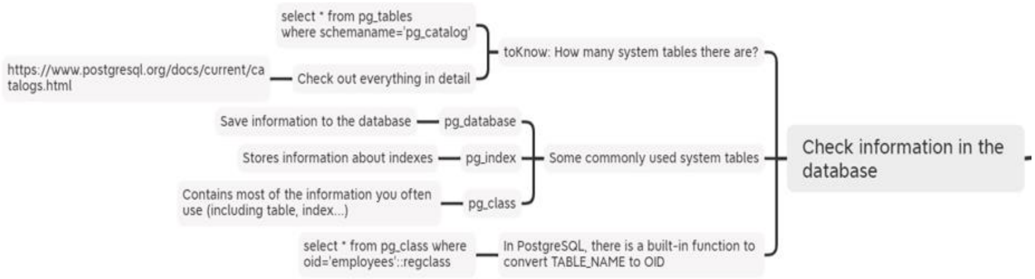
PostgreSQL is an advanced, enterprise-class, and open-source relational database system.
PostgreSQL supports both SQL (relational) and JSON (non-relational) querying.
PostgreSQL is a highly stable database backed by more than 20 years of development by the open-source community.

Postgresql mindmap

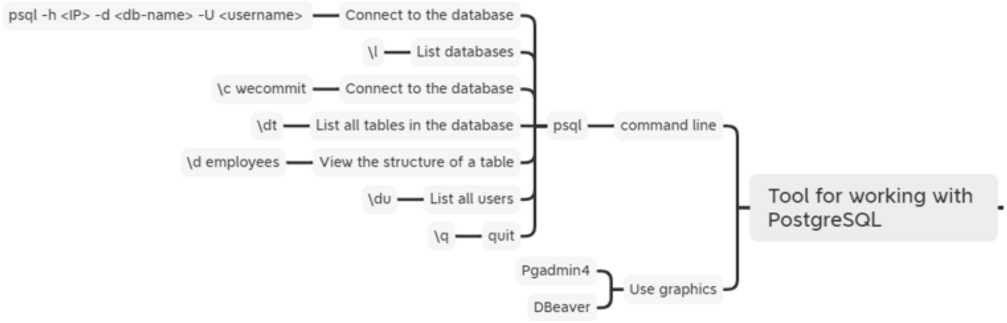

Architecture
Logical Architecture
A database cluster is a collection of databases on a PostgreSQL Server
Database is a collection of Objects
Schema
Objects
Table
Index
Trigger
...
Database objects are managed by OID
eeg: Check OID of databases
select * from pg_database
select * from pg_class where relname='countries'
Checking on the OS will see folders containing the corresponding OID
Physical architecture
Database cluster has a general path called Base directory
1
Floating Topic
Connection configuration files
pg_hba.conf
host-based authentication
pg_ident.conf
chứa file mapping
Basic information file
PG_VERSION
Saves the version information of the Database
current_logfiles
Current log file information
Information of the postmaster process
postmaster.opts
postmaster.pid
Files
Parameter files that configure database operations
postgresql.conf
Save configuration information
postgresql.auto.conf
When we type the ALTER SYSTEM command to change information, the system will automatically update that parameter value to this file.
The parameter values in the postgresql.auto.conf file will override the parameters in the postgresql.conf file
In the Base directory path there will be important files and subfolders
Sub folders
base
Each sub folder contains databases
globals
Contains tables belonging to the cluster (shared)
log
Contains log information, used to check for problem cases
pg_wal
Contains WAL segment files
Tablespace
pg_default
pg_global
Each Table and Index will be a file in the database directory with the corresponding OID name
By default, for every 1GB full, the system will create a new file



- SQL
  - Using SQL
    - Table
      - Structure of Table
        - column
        - Rows
    - Create Table
      - Create a new table
        - CREATE TABLE...
      - Create a table from another table
        - CRETATE TABLE ... AS SELECT...
    - Get data in Table
      - get data
        - Select
          - 1 column or multiple columns
            - Select country_id, country_name from countries
          - All columns
            - Select *
      - Filter the data
        - Where
      - Sort data
        - Order by
      - Transform data
        - Using functions
      - Group data
        - Ude Group by
      - Get data from multiple tables
        - Join
    - Add data to the table
      - Insert
        - Add 1 data record
          - Insert into test values (1,'A')
        - Add data from another table
          - Insert into test select * from test
    - Updating data
      - Update
    - Deleting data
      - Delete
  - SQL optimization
    - Implementation strategy
    - See execution strategy
    - Example of SQL optimization
      - example
        - Use Index for optimization



- Check information in the database
  - toKnow: How many system tables there are?
    - select * from pg_tables where schemaname='pg_catalog'
    - Check out everything in detail
      - https://www.postgresql.org/docs/current/catalogs.html
  - Some commonly used system tables
    - pg_database
      - Save information to the database
    - pg_index
      - Stores information about indexes
    - pg_class
      - Contains most of the information you often use (including table, index...)
  - In PostgreSQL, there is a built-in function to convert TABLE_NAME to OID
    - select * from pg_class where oid='employees'::regclass

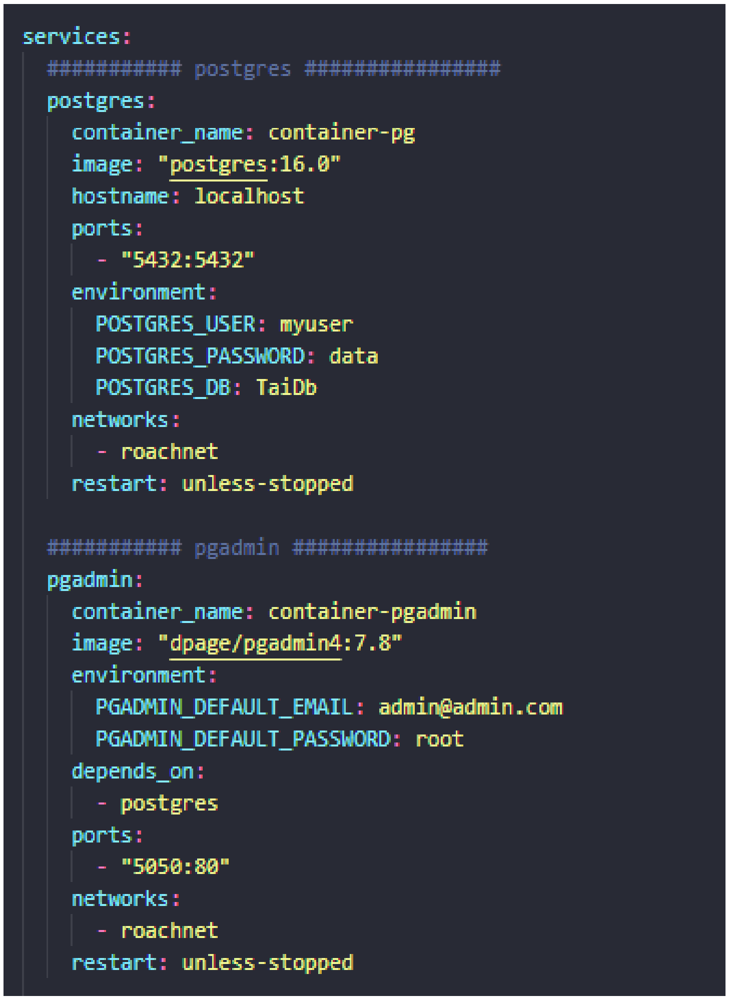


- Tool for working with PostgreSQL
  - command line
    - psql
      - Connect to the database
        - psql -h <IP> -d <db-name> -U <username>
      - List databases
        - \l
      - Connect to the database
        - \c wecommit
      - List all tables in the database
        - \dt
      - View the structure of a table
        - \d employees
      - List all users
        - \du
      - quit
        - \q
  - Use graphics
    - Pgadmin4
    - DBeaver

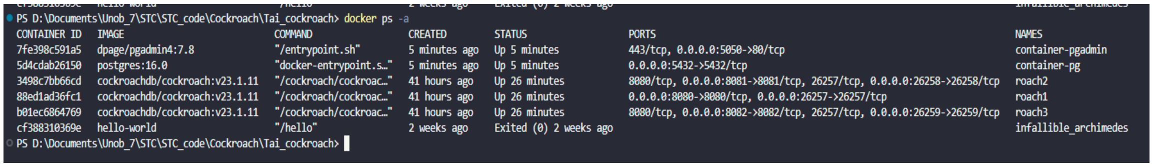
Docker

Main reasons for using docker database image for your local development and why I think you should do it too:

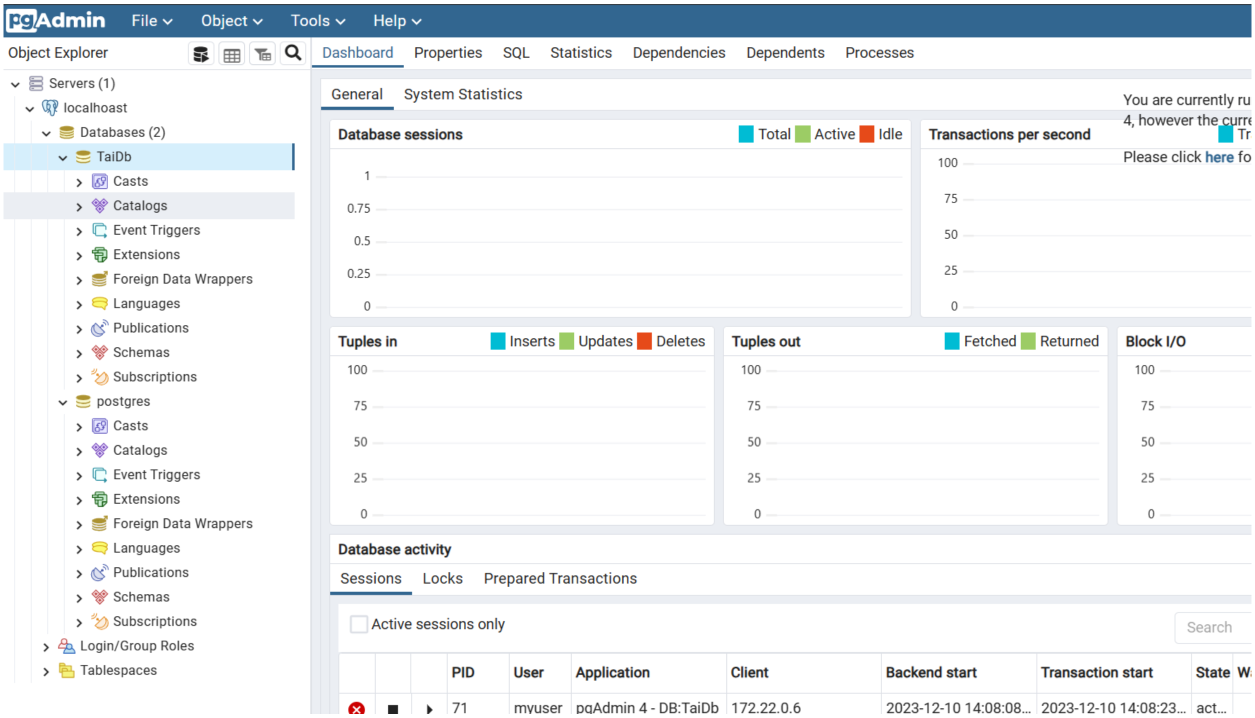
- It makes it easy for local development setup. A simple abstraction using a one-line setup for getting your database set up!
- It does not require you to have any databases installed at all (or a specific version of the database) on your computer.
- It isolates the database environment. It allows me to run multiple versions and instances at the same time without getting in trouble.



```yaml
services:
  ########### postgres ################
  postgres:
    container_name: container-pg
    image: "postgres:16.0"
    hostname: localhost
    ports:
      - "5432:5432"
    environment:
      POSTGRES_USER: myuser
      POSTGRES_PASSWORD: data
      POSTGRES_DB: TaiDb
    networks:
      - roachnet
    restart: unless-stopped

  ########### pgadmin ################
  pgadmin:
    container_name: container-pgadmin
    image: "dpage/pgadmin4:7.8"
    environment:
      PGADMIN_DEFAULT_EMAIL: admin@admin.com
      PGADMIN_DEFAULT_PASSWORD: root
    depends_on:
      - postgres
    ports:
      - "5050:80"
    networks:
      - roachnet
    restart: unless-stopped
```



```
cf388310369e   hello-world                      "/hello"                  2 weeks ago      Exited (0) 2 weeks ago                                                                          infallible_archimedes
PS D:\Documents\Unob_7\STC\STC_code\Cockroach\Tai_cockroach> docker ps -a
CONTAINER ID   IMAGE                            COMMAND                   CREATED          STATUS                   PORTS                                                                   NAMES
7fe398c591a5   dpage/pgadmin4:7.8               "/entrypoint.sh"          5 minutes ago    Up 5 minutes             443/tcp, 0.0.0.0:5050->80/tcp                                           container-pgadmin
5d4cdab26150   postgres:16.0                    "docker-entrypoint.s…"    5 minutes ago    Up 5 minutes             0.0.0.0:5432->5432/tcp                                                  container-pg
3498c7bb66cd   cockroachdb/cockroach:v23.1.11   "/cockroach/cockroac…"    41 hours ago     Up 26 minutes            8080/tcp, 0.0.0.0:8081->8081/tcp, 26257/tcp, 0.0.0.0:26258->26258/tcp   roach2
88ed1ad36fc1   cockroachdb/cockroach:v23.1.11   "/cockroach/cockroac…"    41 hours ago     Up 26 minutes            0.0.0.0:8080->8080/tcp, 0.0.0.0:26257->26257/tcp                        roach1
b01ec6864769   cockroachdb/cockroach:v23.1.11   "/cockroach/cockroac…"    41 hours ago     Up 26 minutes            8080/tcp, 0.0.0.0:8082->8082/tcp, 26257/tcp, 0.0.0.0:26259->26259/tcp   roach3
cf388310369e   hello-world                      "/hello"                  2 weeks ago      Exited (0) 2 weeks ago                                                                           infallible_archimedes
PS D:\Documents\Unob_7\STC\STC_code\Cockroach\Tai_cockroach> 
```



You are currently ru
4, however the curre
Please click here fo

□ Active sessions only

			PID	User	Application	Client	Backend start	Transaction start	State	W
⊗	■	▸	71	myuser	pgAdmin 4 - DB:TaiDb	172.22.0.6	2023-12-10 14:08:08...	2023-12-10 14:08:23...	act...	

3.2 Cockroach

CockroachDB is a distributed SQL database built on a transactional and strongly-consistent key-value store. It scales horizontally; you can quickly and simply add nodes to your cluster to handle more traffic.

A wire protocol is the format for interactions between a database server and its clients. It encompasses authentication, sending queries, receiving responses, and so on. It is a description of the exact bytes sent and received by servers and clients. It does NOT encompass the actual query language itself, let alone database semantics.

CockroachDB is wire-compatible with PostgreSQL clients, meaning that if you have a PostgreSQL client library for your favorite programming language, you can use that to connect to CockroachDB.
The database features and SQL dialect itself are not 100% compatible with PostgreSQL, and likely never will be.
Also, even if all the SQL you do want to use is supported, it is likely a sub-optimal idea to just port some SQL over to the CockroachDB platform without additional investigation. Some issues, like interleaving values from different tables, should be examined to ensure good performance on CockroachDB.
And even setting aside issues like that, the access patterns you'd use for a multi-region distributed database are going to be a bit different than a high-availability PostgreSQL cluster. Data locality and all that.
The wide availability of client drivers just means you can get started quickly on popular platforms.

CockroachDB architecture terms

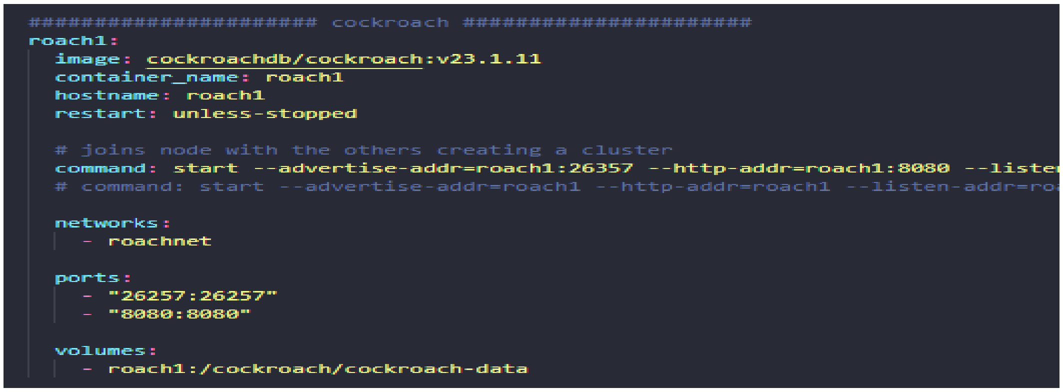
Cluster

- A group of interconnected CockroachDB nodes that function as a single distributed SQL database server. Nodes collaboratively organize transactions, and rebalance workload and data storage to optimize performance and fault-tolerance.
- Each cluster has its own authorization hierarchy, meaning that users and roles must be defined on that specific cluster.
- A CockroachDB cluster can be run in CockroachDB Cloud, within a customer Organization, or can be self-hosted.

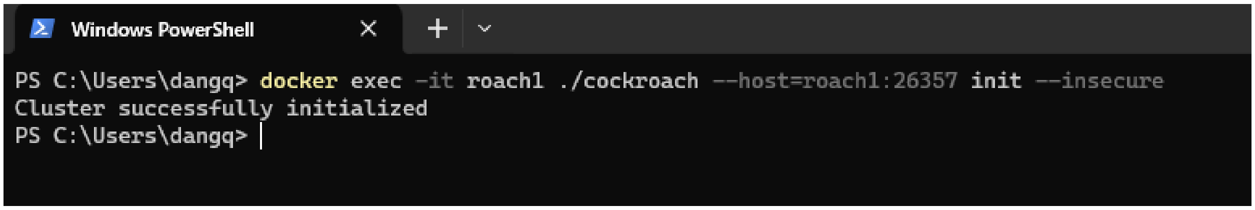
Node: An individual instance of CockroachDB. One or more nodes form a cluster.
Range: CockroachDB stores all user data (tables, indexes, etc.) and almost all system data in a sorted map of key-value pairs. This keyspace is divided into contiguous chunks called ranges, such that every key is found in one range.
Replica: A copy of a range stored on a node. By default, there are three replicas of each range on different nodes.
Leaseholder: The replica that holds the "range lease." This replica receives and coordinates all read and write requests for the range. For most types of tables and queries, the leaseholder is the only replica that can serve consistent reads (reads that return "the latest" data).

Raft protocol: The consensus protocol employed in CockroachDB that ensures that your data is safely stored on multiple nodes and that those nodes agree on the current state even if some of them are temporarily disconnected.
Raft leader: For each range, the replica that is the "leader" for write requests. The leader uses the Raft protocol to ensure that a majority of replicas (the leader and enough followers) agree, based on their Raft logs, before committing the write. The Raft leader is almost always the same replica as the leaseholder.
Raft log: A time-ordered log of writes to a range that its replicas have agreed on. This log exists on-disk with each replica and is the range's source of truth for consistent replication.

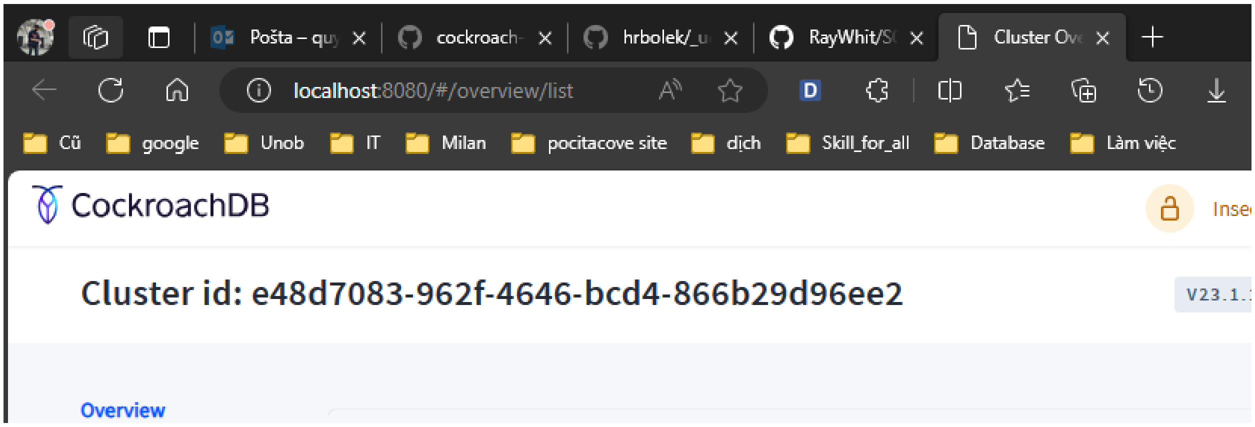
Docker file



```
###################### cockroach ######################
roach1:
  image: cockroachdb/cockroach:v23.1.11
  container_name: roach1
  hostname: roach1
  restart: unless-stopped

  # joins node with the others creating a cluster
  command: start --advertise-addr=roach1:26357 --http-addr=roach1:8080 --listen
  # command: start --advertise-addr=roach1 --http-addr=roach1 --listen-addr=roa

  networks:
    - roachnet

  ports:
    - "26257:26257"
    - "8080:8080"

  volumes:
    - roach1:/cockroach/cockroach-data
```

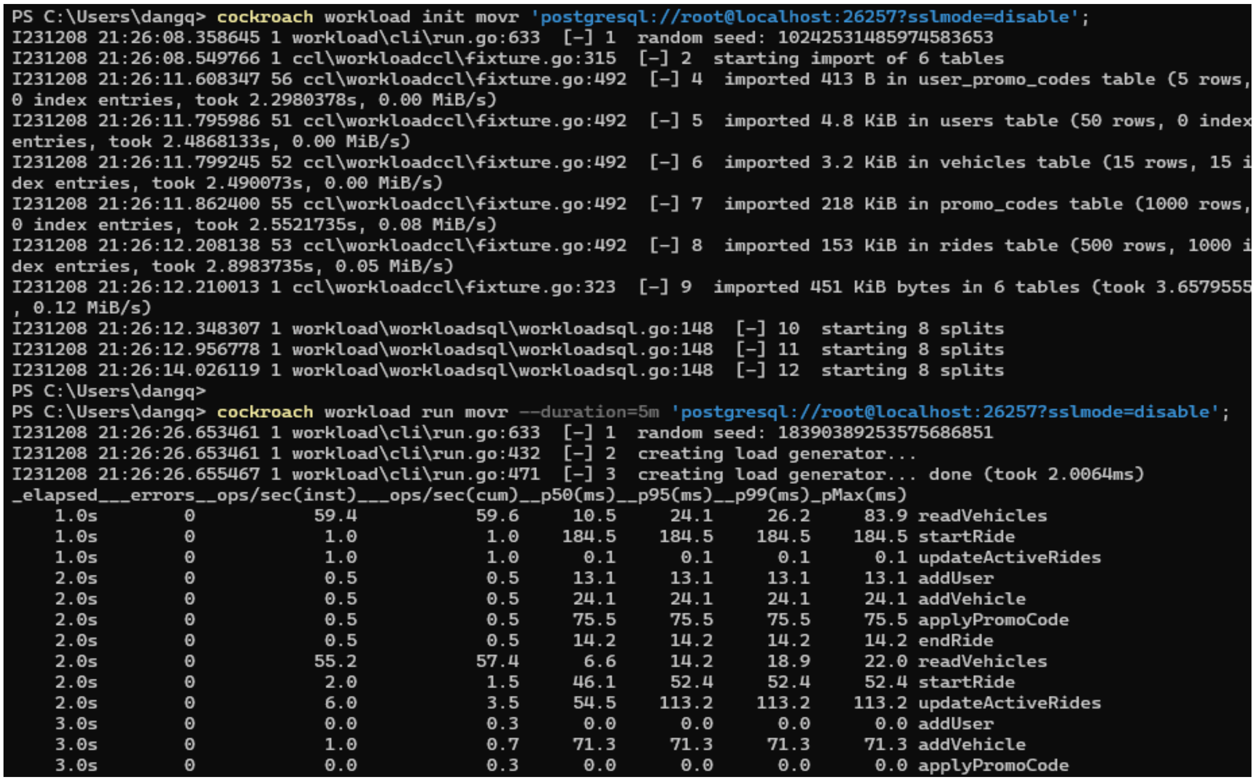


```
PS C:\Users\dangq> docker exec -it roach1 ./cockroach --host=roach1:26357 init --insecure
Cluster successfully initialized
PS C:\Users\dangq>
```

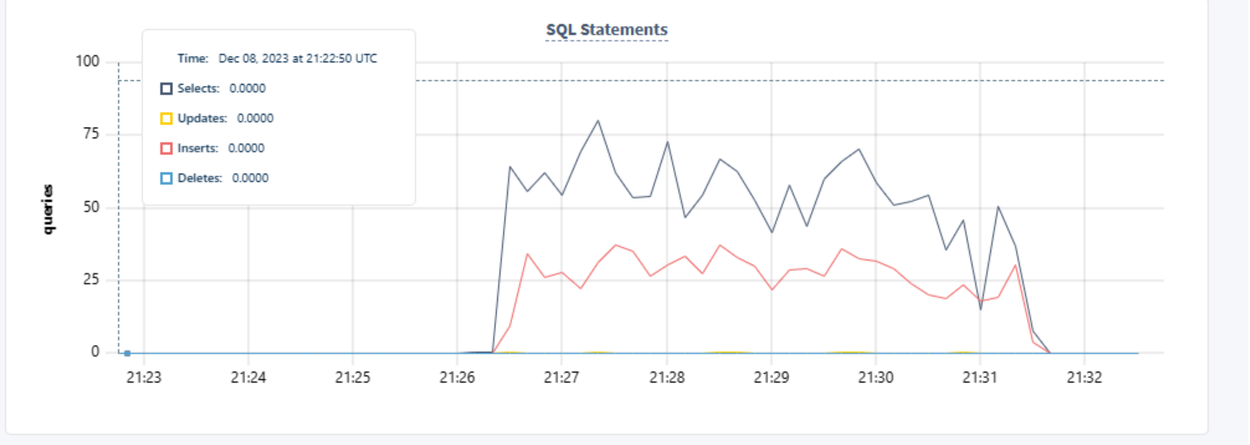


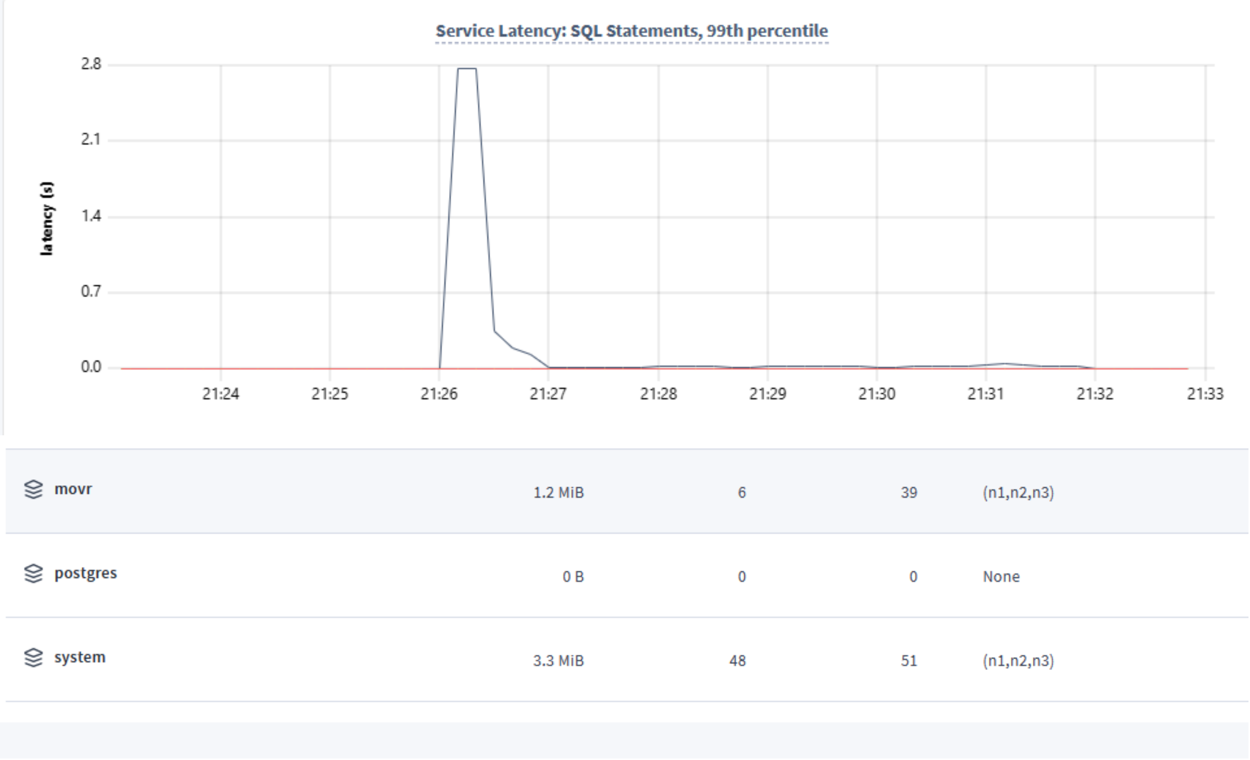
Cluster id: e48d7083-962f-4646-bcd4-866b29d96ee2

Run the workload in 5mins and see the UI

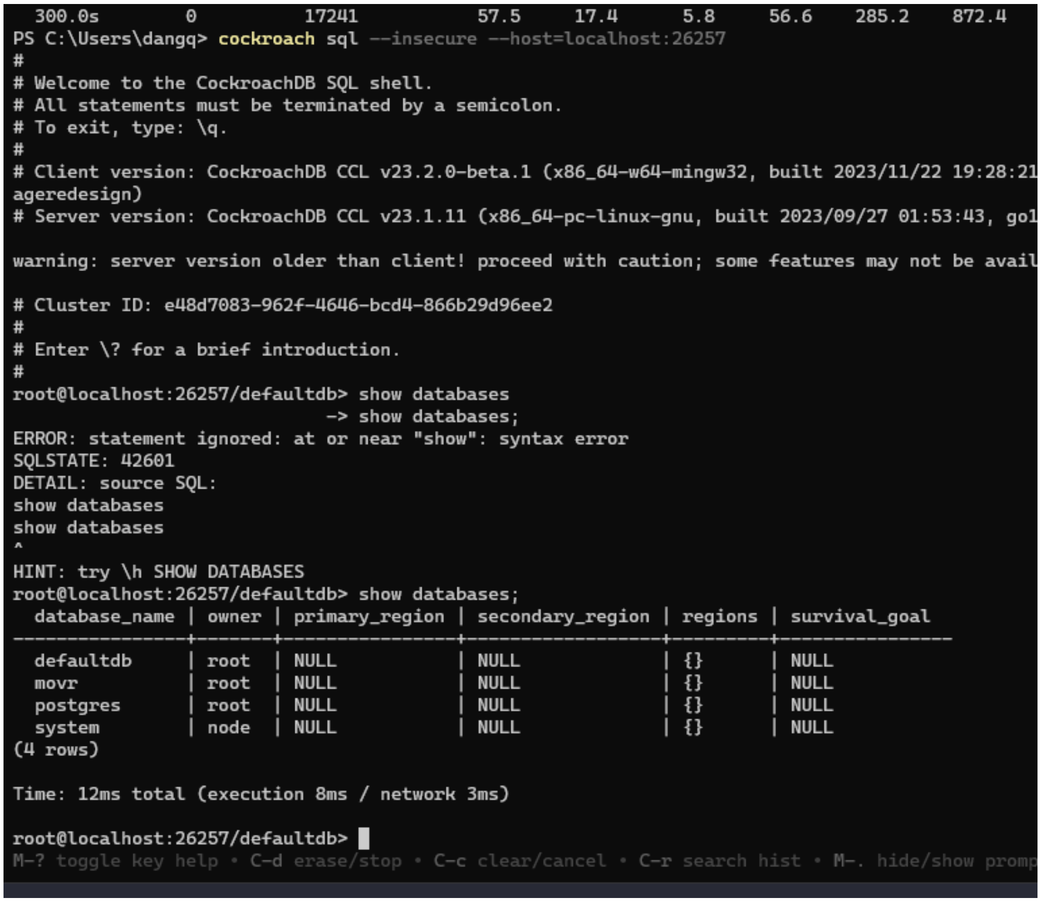


```
PS C:\Users\dangq> cockroach workload init movr 'postgresql://root@localhost:26257?sslmode=disable';
I231208 21:26:08.358645 1 workload\cli\run.go:633  [-] 1  random seed: 1024253148597458653
I231208 21:26:08.549766 1 ccl\workloadccl\fixture.go:315  [-] 2  starting import of 6 tables
I231208 21:26:11.608347 56 ccl\workloadccl\fixture.go:492  [-] 4  imported 413 B in user_promo_codes table (5 rows,
0 index entries, took 2.2980378s, 0.00 MiB/s)
I231208 21:26:11.795986 51 ccl\workloadccl\fixture.go:492  [-] 5  imported 4.8 KiB in users table (50 rows, 0 index
entries, took 2.4868133s, 0.00 MiB/s)
I231208 21:26:11.799245 52 ccl\workloadccl\fixture.go:492  [-] 6  imported 3.2 KiB in vehicles table (15 rows, 15 i
dex entries, took 2.490073s, 0.00 MiB/s)
I231208 21:26:11.862400 55 ccl\workloadccl\fixture.go:492  [-] 7  imported 218 KiB in promo_codes table (1000 rows,
0 index entries, took 2.5521735s, 0.08 MiB/s)
I231208 21:26:12.208138 53 ccl\workloadccl\fixture.go:492  [-] 8  imported 153 KiB in rides table (500 rows, 1000 i
dex entries, took 2.8983735s, 0.05 MiB/s)
I231208 21:26:12.210013 1 ccl\workloadccl\fixture.go:323  [-] 9  imported 451 KiB bytes in 6 tables (took 3.6579555
, 0.12 MiB/s)
I231208 21:26:12.348307 1 workload\workloadsql\workloadsql.go:148  [-] 10  starting 8 splits
I231208 21:26:12.956778 1 workload\workloadsql\workloadsql.go:148  [-] 11  starting 8 splits
I231208 21:26:14.026119 1 workload\workloadsql\workloadsql.go:148  [-] 12  starting 8 splits
PS C:\Users\dangq>
PS C:\Users\dangq> cockroach workload run movr --duration=5m 'postgresql://root@localhost:26257?sslmode=disable';
I231208 21:26:26.653461 1 workload\cli\run.go:633  [-] 1  random seed: 1839038925357568685 1
I231208 21:26:26.653461 1 workload\cli\run.go:432  [-] 2  creating load generator...
I231208 21:26:26.655467 1 workload\cli\run.go:471  [-] 3  creating load generator... done (took 2.0064ms)
_elapsed___errors__ops/sec(inst)___ops/sec(cum)__p50(ms)__p95(ms)__p99(ms)_pMax(ms)
    1.0s        0           59.4           59.6     10.5     24.1     26.2     83.9 readVehicles
    1.0s        0            1.0            1.0    184.5    184.5    184.5    184.5 startRide
    1.0s        0            1.0            1.0      0.1      0.1      0.1      0.1 updateActiveRides
    2.0s        0            0.5            0.5     13.1     13.1     13.1     13.1 addUser
    2.0s        0            0.5            0.5     24.1     24.1     24.1     24.1 addVehicle
    2.0s        0            0.5            0.5     75.5     75.5     75.5     75.5 applyPromoCode
    2.0s        0            0.5            0.5     14.2     14.2     14.2     14.2 endRide
    2.0s        0           55.2           57.4      6.6     14.2     18.9     22.0 readVehicles
    2.0s        0            2.0            1.5     46.1     52.4     52.4     52.4 startRide
    2.0s        0            6.0            3.5     54.5    113.2    113.2    113.2 updateActiveRides
    3.0s        0            0.0            0.3      0.0      0.0      0.0      0.0 addUser
    3.0s        0            1.0            0.7     71.3     71.3     71.3     71.3 addVehicle
    3.0s        0            0.0            0.3      0.0      0.0      0.0      0.0 applyPromoCode
```





movr	1.2 MiB	6	39	(n1,n2,n3)
postgres	0 B	0	0	None
system	3.3 MiB	48	51	(n1,n2,n3)



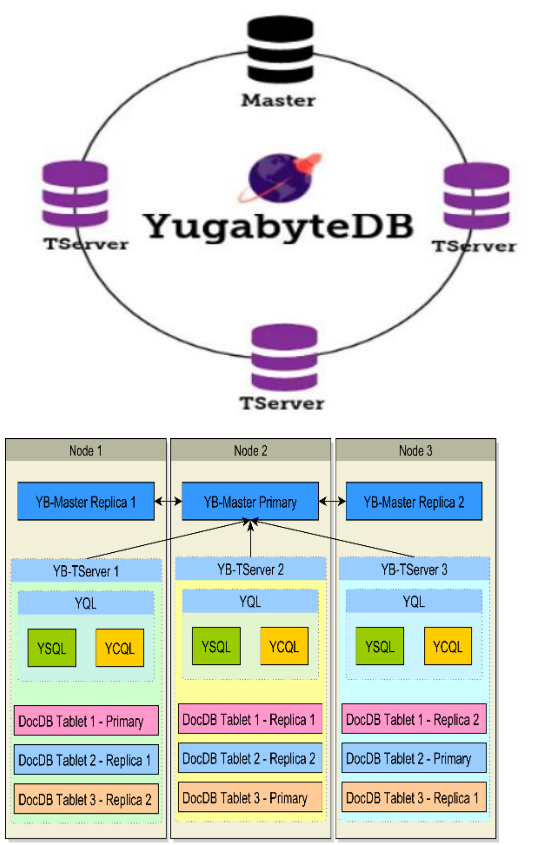
```
  300.0s        0          17241          57.5     17.4      5.8     56.6    285.2    872.4
PS C:\Users\dangq> cockroach sql --insecure --host=localhost:26257
#
# Welcome to the CockroachDB SQL shell.
# All statements must be terminated by a semicolon.
# To exit, type: \q.
#
# Client version: CockroachDB CCL v23.2.0-beta.1 (x86_64-w64-mingw32, built 2023/11/22 19:28:21
ageredesign)
# Server version: CockroachDB CCL v23.1.11 (x86_64-pc-linux-gnu, built 2023/09/27 01:53:43, go1

warning: server version older than client! proceed with caution; some features may not be avail

# Cluster ID: e48d7083-962f-4646-bcd4-866b29d96ee2
#
# Enter \? for a brief introduction.
#
root@localhost:26257/defaultdb> show databases
                             -> show databases;
ERROR: statement ignored: at or near "show": syntax error
SQLSTATE: 42601
DETAIL: source SQL:
show databases
show databases
^
HINT: try \h SHOW DATABASES
root@localhost:26257/defaultdb> show databases;
  database_name | owner | primary_region | secondary_region | regions | survival_goal
----------------+-------+----------------+------------------+---------+----------------
  defaultdb     | root  | NULL           | NULL             | {}      | NULL
  movr          | root  | NULL           | NULL             | {}      | NULL
  postgres      | root  | NULL           | NULL             | {}      | NULL
  system        | node  | NULL           | NULL             | {}      | NULL
(4 rows)

Time: 12ms total (execution 8ms / network 3ms)

root@localhost:26257/defaultdb> 
M-? toggle key help • C-d erase/stop • C-c clear/cancel • C-r search hist • M-. hide/show promp
```

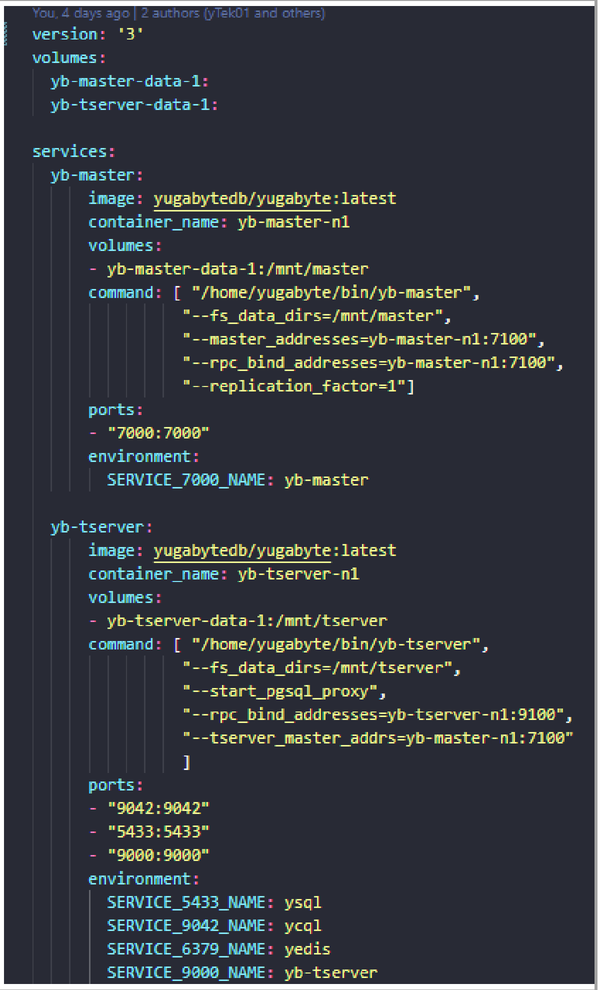
YugabyteDB

YugabyteDB is distributed PostgreSQL for enterprise applications delivered as a flexible cloud service.

PostgreSQL Compatibility YugabyteDB is more than just wire compatible with PostgreSQL, it is code compatible. YugabyteDB achieves this by reusing PostgreSQL's query layer to achieve a high degree of compatibility with existing PostgreSQL applications or those that can be migrated to PostgreSQL.

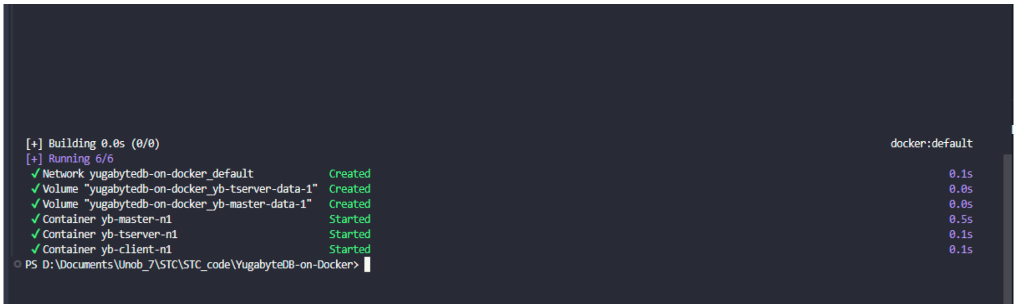
YugabyteDB top-level architecture



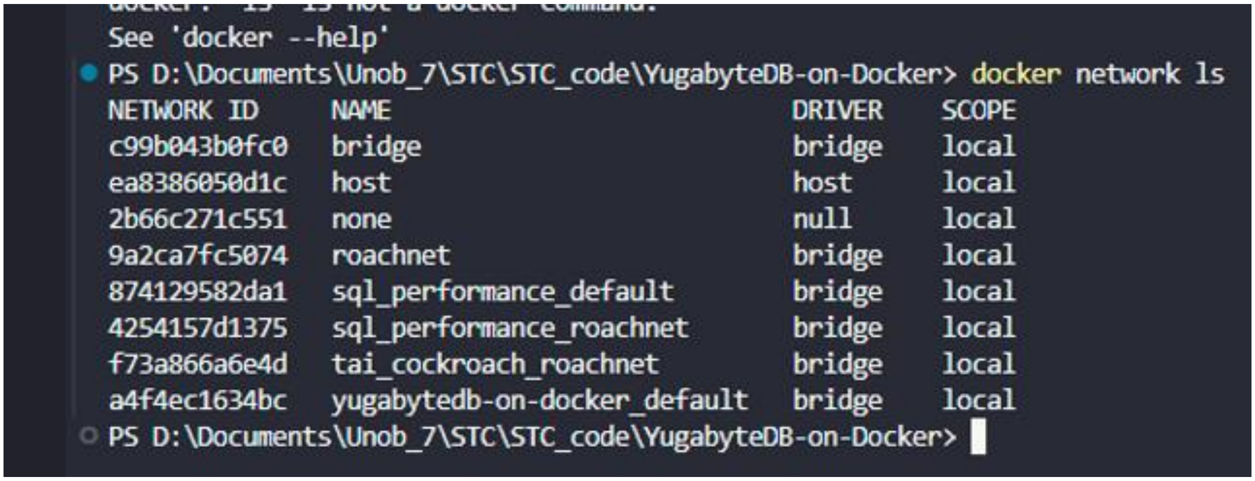


```
version: '3'
volumes:
  yb-master-data-1:
  yb-tserver-data-1:

services:
  yb-master:
      image: yugabytedb/yugabyte:latest
      container_name: yb-master-n1
      volumes:
      - yb-master-data-1:/mnt/master
      command: [ "/home/yugabyte/bin/yb-master",
                "--fs_data_dirs=/mnt/master",
                "--master_addresses=yb-master-n1:7100",
                "--rpc_bind_addresses=yb-master-n1:7100",
                "--replication_factor=1"]
      ports:
      - "7000:7000"
      environment:
        SERVICE_7000_NAME: yb-master

  yb-tserver:
      image: yugabytedb/yugabyte:latest
      container_name: yb-tserver-n1
      volumes:
      - yb-tserver-data-1:/mnt/tserver
      command: [ "/home/yugabyte/bin/yb-tserver",
                "--fs_data_dirs=/mnt/tserver",
                "--start_pgsql_proxy",
                "--rpc_bind_addresses=yb-tserver-n1:9100",
                "--tserver_master_addrs=yb-master-n1:7100"
                ]
      ports:
      - "9042:9042"
      - "5433:5433"
      - "9000:9000"
      environment:
        SERVICE_5433_NAME: ysql
        SERVICE_9042_NAME: ycql
        SERVICE_6379_NAME: yedis
        SERVICE_9000_NAME: yb-tserver
```

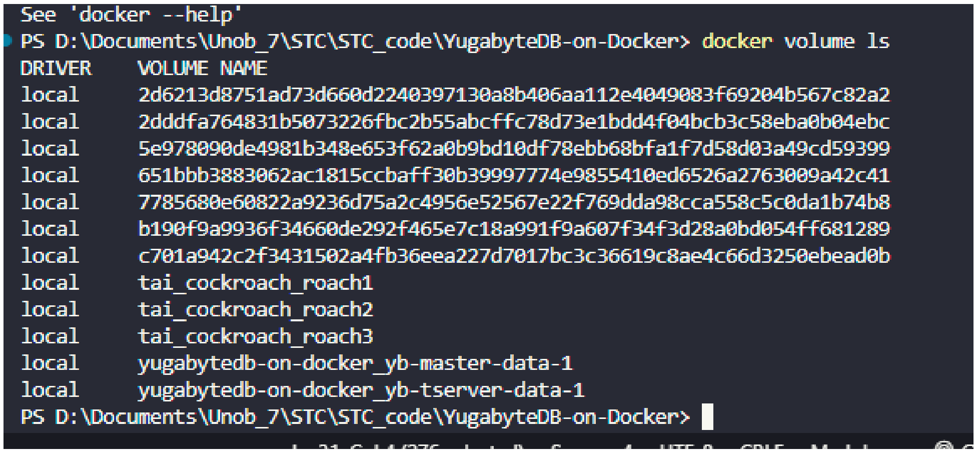
Start the Cluster: docker-compose -f ./docker-compose.Yugabyte.yaml up -d



```
[+] Building 0.0s (0/0)                                                                              docker:default
[+] Running 6/6
 ✓ Network yugabytedb-on-docker_default              Created                                                   0.1s
 ✓ Volume "yugabytedb-on-docker_yb-tserver-data-1"   Created                                                   0.0s
 ✓ Volume "yugabytedb-on-docker_yb-master-data-1"    Created                                                   0.0s
 ✓ Container yb-master-n1                            Started                                                   0.5s
 ✓ Container yb-tserver-n1                           Started                                                   0.1s
 ✓ Container yb-client-n1                            Started                                                   0.1s
○ PS D:\Documents\Unob_7\STC\STC_code\YugabyteDB-on-Docker>
```

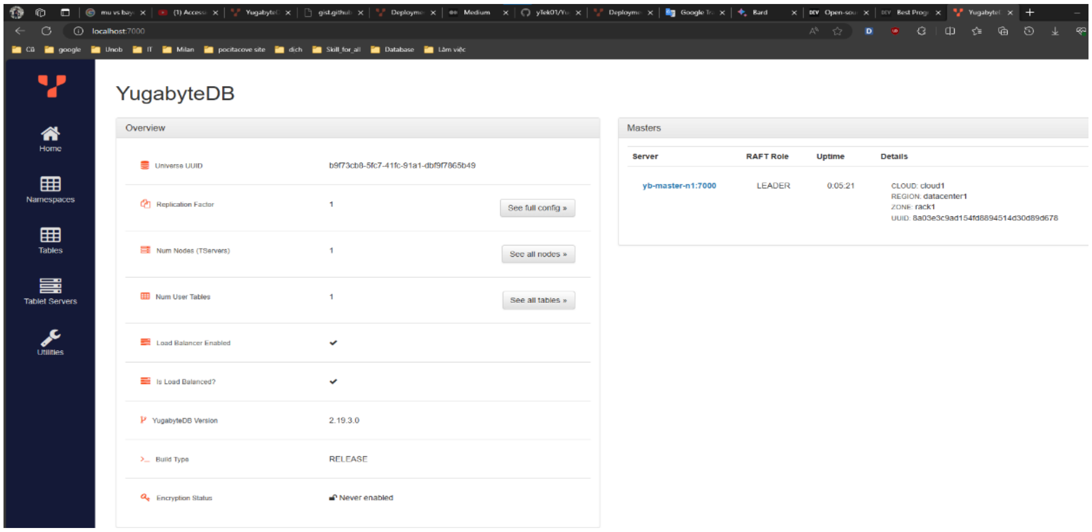


```
See 'docker --help'
PS D:\Documents\Unob_7\STC\STC_code\YugabyteDB-on-Docker> docker network ls
NETWORK ID     NAME                          DRIVER    SCOPE
c99b043b0fc0   bridge                        bridge    local
ea8386050d1c   host                          host      local
2b66c271c551   none                          null      local
9a2ca7fc5074   roachnet                      bridge    local
874129582da1   sql_performance_default       bridge    local
4254157d1375   sql_performance_roachnet      bridge    local
f73a866a6e4d   tai_cockroach_roachnet        bridge    local
a4f4ec1634bc   yugabytedb-on-docker_default  bridge    local
PS D:\Documents\Unob_7\STC\STC_code\YugabyteDB-on-Docker>
```



```
See 'docker --help'
PS D:\Documents\Unob_7\STC\STC_code\YugabyteDB-on-Docker> docker volume ls
DRIVER    VOLUME NAME
local     2d6213d8751ad73d660d2240397130a8b406aa112e4049083f69204b567c82a2
local     2dddfa764831b5073226fbc2b55abcffc78d73e1bdd4f04bcb3c58eba0b04ebc
local     5e978090de4981b348e653f62a0b9bd10df78ebb68bfa1f7d58d03a49cd59399
local     651bbb3883062ac1815ccbaff30b39997774e9855410ed6526a2763009a42c41
local     7785680e60822a9236d75a2c4956e52567e22f769dda98cca558c5c0da1b74b8
local     b190f9a9936f34660de292f465e7c18a991f9a607f34f3d28a0bd054ff681289
local     c701a942c2f3431502a4fb36eea227d7017bc3c36619c8ae4c66d3250ebead0b
local     tai_cockroach_roach1
local     tai_cockroach_roach2
local     tai_cockroach_roach3
local     yugabytedb-on-docker_yb-master-data-1
local     yugabytedb-on-docker_yb-tserver-data-1
PS D:\Documents\Unob_7\STC\STC_code\YugabyteDB-on-Docker>
```

Test the APIs: Clients can now connect to the YSQL API at localhost:5433 and the YCQL API at localhost:9042. The yb-master admin service is available at http://localhost:7000.



YugabyteDB

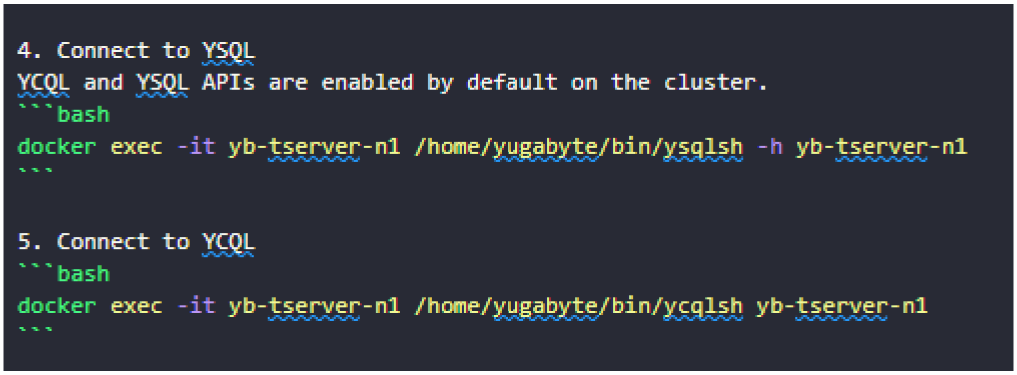
Overview

Universe UUID	b9f73cb8-5fc7-41fc-91a1-dbf9f7865b49
Replication Factor	1
Num Nodes (TServers)	1
Num User Tables	1
Load Balancer Enabled	✓
Is Load Balanced?	✓
YugabyteDB Version	2.19.3.0
Build Type	RELEASE
Encryption Status	Never enabled

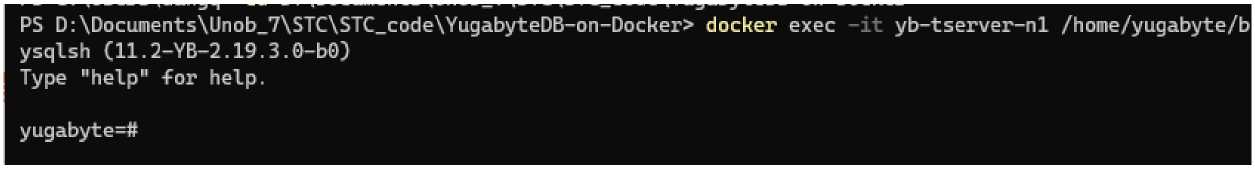
Masters

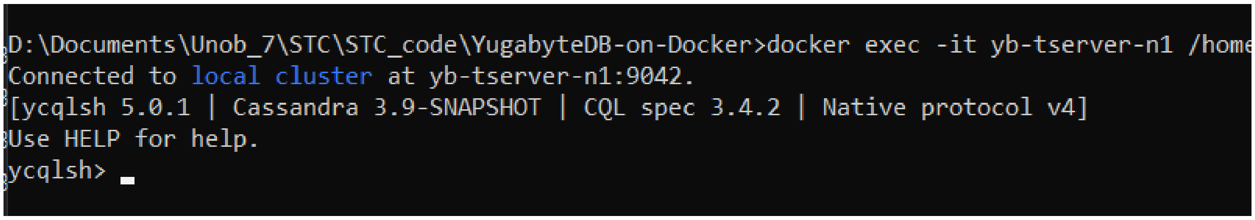
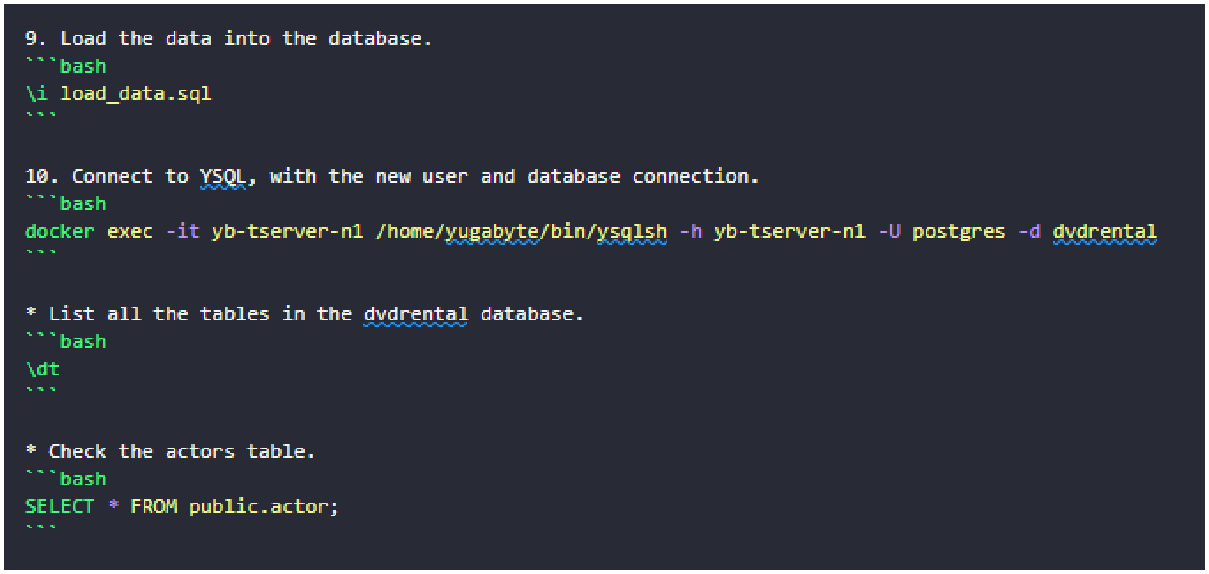
Server	RAFT Role	Uptime	Details
yb-master-n1:7000	LEADER	0:05:21	CLOUD: cloud1 REGION: datacenter1 ZONE: rack1 UUID: 8a03e3c9ad154fd8894514d30d89d678

YugabyteDB metrics | YugabyteDB Docs



~~~
4. Connect to YSQL
YCQL and YSQL APIs are enabled by default on the cluster.
```bash
docker exec -it yb-tserver-n1 /home/yugabyte/bin/ysqlsh -h yb-tserver-n1
~~~

5. Connect to YCQL
```bash
docker exec -it yb-tserver-n1 /home/yugabyte/bin/ycqlsh yb-tserver-n1
```
```



```
PS D:\Documents\Unob_7\STC\STC_code\YugabyteDB-on-Docker> docker exec -it yb-tserver-n1 /home/yugabyte/b
ysqlsh (11.2-YB-2.19.3.0-b0)
Type "help" for help.

yugabyte=#
```



```
D:\Documents\Unob_7\STC\STC_code\YugabyteDB-on-Docker>docker exec -it yb-tserver-n1 /home
Connected to local cluster at yb-tserver-n1:9042.
[ycqlsh 5.0.1 | Cassandra 3.9-SNAPSHOT | CQL spec 3.4.2 | Native protocol v4]
Use HELP for help.
ycqlsh>
```



```
9. Load the data into the database.
```bash
\i load_data.sql
```

10. Connect to YSQL, with the new user and database connection.
```bash
docker exec -it yb-tserver-n1 /home/yugabyte/bin/ysqlsh -h yb-tserver-n1 -U postgres -d dvdrental
```

* List all the tables in the dvdrental database.
```bash
\dt
```

* Check the actors table.
```bash
SELECT * FROM public.actor;
```
```
```