CockroachDB

Emanuel Kranjec & Felix Tröbinger

May 4, 2020

Contents

1 Setting up a CockroachDB Cluster with Docker

1.1 Creating a bridge network

Even though all created nodes – or rather containers – in this example will run on the same host, they still need to be connected via a network of some sorts. As a solution this example will use a Docker *bridge network*.

A bridge network is a software bridge, which allows containers on the same host to communicate with each other. Features of docker bridge networks are that user-defined bridges provide automatic DNS resolution between containers and that containers are isolated in a better way.[1]

The following command will create a bridge network called `roachnet`. This name will be passed to each node on startup/creation.

```
docker network create -d bridge roachnet
```

1.2 Creating Nodes

This example will create a cluster of three connected nodes. The first one is created with the following command.

```
docker run -d \
--name=roach1 \
--hostname=roach1 \
--net=roachnet \
-p 26257:26257 -p 8080:8080  \
-v "${PWD}/roach1:/cockroach/cockroach-data"  \
cockroachdb/cockroach:latest start \
--insecure \
--join=roach1,roach2,roach3
```

The node is connected to the previously created bridge network and given a name, `roach1` in this case.

Port 8080 of the host machine is mapped to port 8080 of the container which is later used for accessing the web-interface (See chapter 1.4). Port 26257 of the container is mapped to the same port on the host. This port is used for communication.

Adding a volume to the container will give the user the possibility to see what CockroachDB stores on disk and save data on container restart. To

completely reset the node, the user has to delete the volume. If no volume is mapped on container start, all data will be lost when the container stops.

The Docker image in use is called `cockroachdb/cockroach` and this example uses the latest version. The command `start` is passed, which will start the CockroachDB inside the container.

For the time being this example will use the insecure mode of CockroachDB which does not require certificates. More on that in chapter 2.

Finally the command will use the `join` argument to connect the three nodes in the cluster by specifying their `hostname`.

For this example two more nodes will be created (nodes `roach2` and `roach3`) using a very similar command as the one above, only changing the hostname and the volume directory on the host machine. The last thing that differs for the last two nodes is that the ports will not be mapped as witch `roach1`. This is because when the web-interface in this example will be accessed we will access the first node we created. For more information about the web-interface see chapter 1.4.

As the next step the cluster has to be initialized. This is done with the following command:

```
docker exec -it roach1 ./cockroach init --insecure
```

[2]

1.3 SQL Command Line

Using the CockroachDBs SQL client can be done via the following command. In this example we use the SQL client by accessing the first node we created, `roach1`.

```
docker exec -it roach1 ./cockroach sql --insecure
```

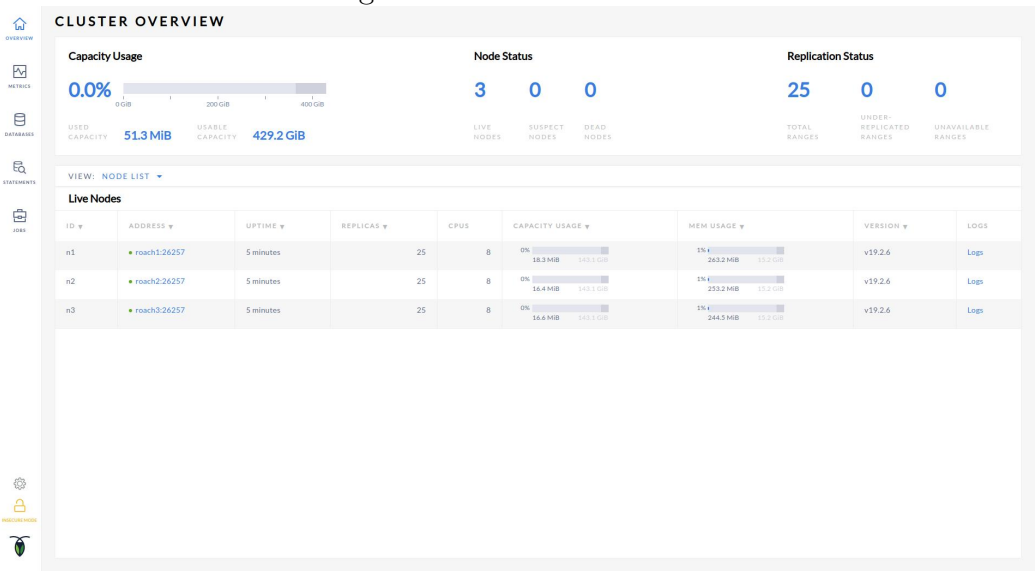
CockroachDB uses the *PostgreSQL* dialect.

1.4 Web-Interface

Accessing the admin web-interface user interface is as easy as navigating to `localhost:8080` in a browser. Port 8080 is used because the host port 8080 is mapped to port 8080 on `roach1`. If this is already in use by another application running on the host machine, port 8080 within the container can be remapped to a different port on the host machine. This would be done

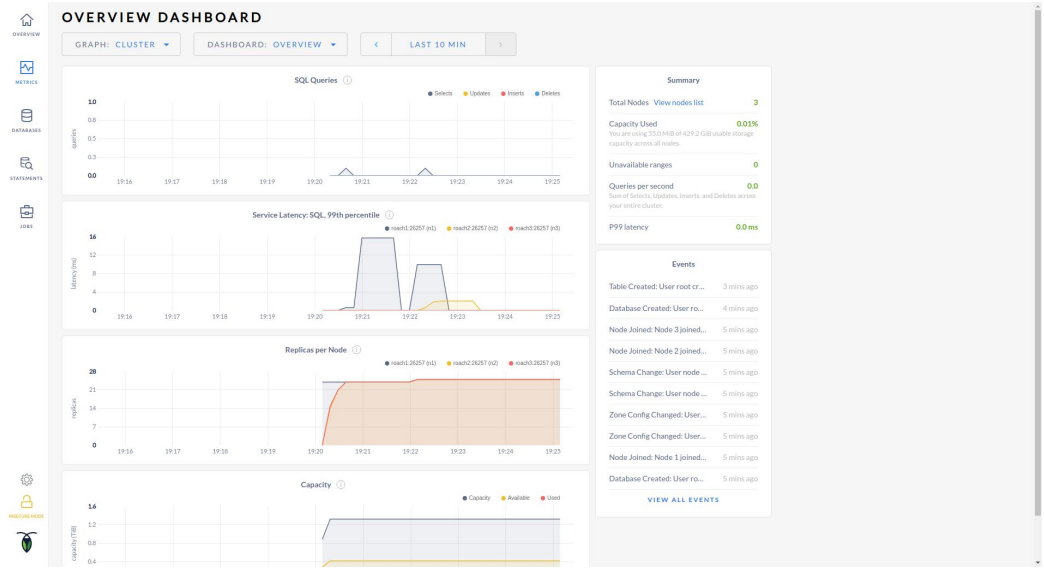
by changing `-p 8080:8080` to `-p <YOUR-PORT>:8080` in the command that started the first node.

Figure 1: Cluster Overview



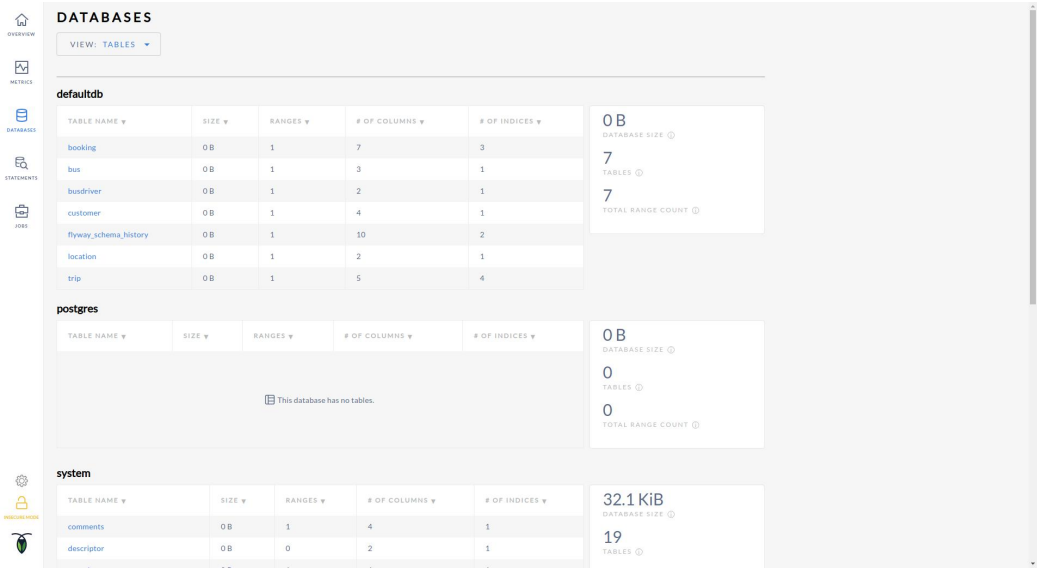
The first view the user is greeted with is the overview of the cluster which can be seen in Figure 1. This view shows all nodes connected to the cluster and the cluster capacity as well as the capacities for every single node.

Figure 2: The CockroachDB Web-Interface Dashboard



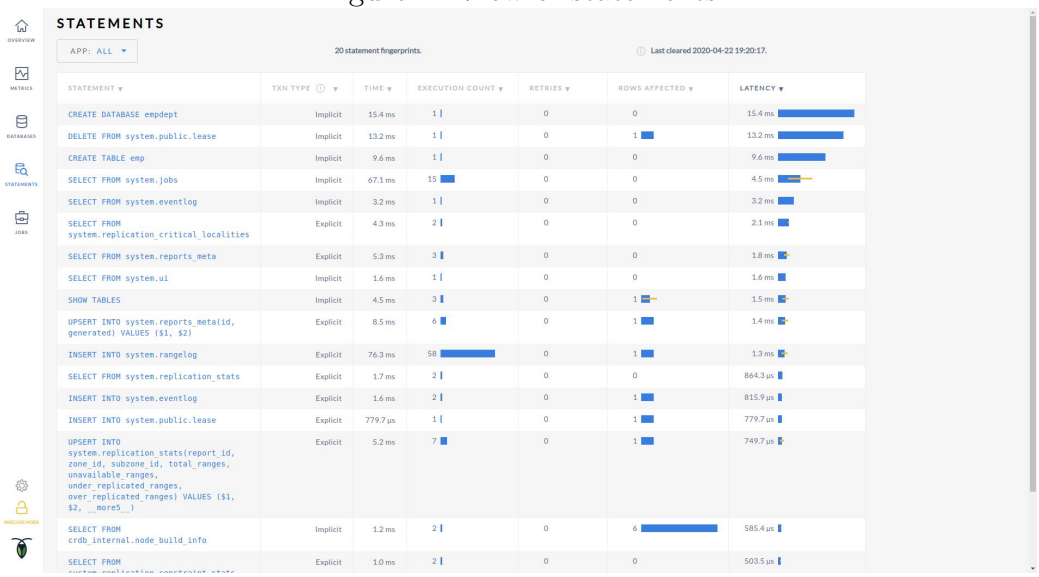
On the Dashboard view (also labeled *Metrics* in the web UI) users can see graphs of SQL Queries ran on different nodes as well as latency and replication information. In this example we created tables and inserted data on the first node. This is visible in the *Replicas per node* graph where the line representing `roach1` immediately jumps to the value whereas the other two nodes take a little bit of time replicating the data on their system.

Figure 3: View of databases



The *Databases* view show a list of all databases and their tables. Note that it does not display contents of the database.

Figure 4: View of statements



The *Statements* view list all executed SQL commands/statements which can be filtered and sorted.

2 The secure way

In the previous example all nodes have been started with the command line option `--insecure`. In the bottom of the web-interface the user is also reminded that CockroachDB is running in insecure mode by the golden lock icon. This is obviously a bad idea to be used in production or for any database, really.

Running CockroachDB in secure mode requires using certificates. These can be created by running either the `cockroach cert` command or using `opensll`. For this example the Cockroach command will be used.

```
cockroach cert create-ca \
    --certs-dir=certs \
    --ca-key=my-safe-directory/ca.key
```

The above command generates

1. a CA (= **C**ertificate **A**uthority) certificate and clients/node certificates
2. a CA key

When creating secure nodes or client certificates the CA key is required and referenced during the creation process.

Creating certificates and key pairs for a node can be done with the following command. (Replace *hostname* with your hostname/container name.)

change commands to use docker commands

```
cockroach cert create-node \
    localhost $(hostname) \
    --certs-dir=certs \
    --ca-key=my-safe-directory/ca.key
```

Then we create a certified client. In this case for the root user.

```
cockroach cert create-client root \
    --certs-dir=certs \
    --ca-key=my-safe-directory/ca.key
```

References

[1] docker docs. Use bridge networks.

[2] Cockroach Labs. Start a cluster in docker (insecure).