**Scénario : cassandra : run cassandra dans Docker :**

**Step 1**

**Starting a Single-Node Cassandra Cluster in Docker**

If you have a Docker environment installed on your machine, it's extremely simple to start a Cassandra node. First, let's obtain the Apache Cassandra image:

docker pull cassandra

This command pulls the Docker image marked with the tag *latest* from the [Docker Hub](https://hub.docker.com/_/cassandra/).

Let's create a network for our cluster. We don't technically need it right away for a single node, but it will be useful when we add nodes later.

docker network create cass-network

Now you can start an instance of Cassandra with default options and a container name of my-cassandra:

docker run -d --name my-cassandra --network cass-network cassandra

The -d option starts the container in the background without printing out the logs.

You used the --name option to specify a name for the container, which allows you to reference the container by name when using other Docker commands.

If you don't provide a name for the container, the Docker runtime will assign a randomly selected name such as breezy\_ensign. Docker also creates a unique identifier for each container that is returned from the initial run command. Either the name or ID may be used to reference a specific container in Docker commands.

You can use docker ps to see status information about the container you've created:

docker ps

In order to access a node from outside Docker for CQL queries, you'll want to make sure the standard CQL port is exposed when the node is created:

docker start cassandra -p 9042:9042

There are several other configuration options available for running Cassandra in Docker, which are documented on the Docker Hub page referenced above.

**Step 2**

**Running cqlsh on a Cassandra Node in Docker**

To start an instance of cqlsh, the simplest way is to use the copy inside your Docker instance by executing a command on the instance:

docker exec -it my-cassandra cqlsh

This will give you a cqlsh prompt, with which you could execute similar commands to those you learn in other labs in this series such as Get to Know CQL or Query Options in CQL.

If you get an error message when starting cqlsh, Cassandra is still starting up. You can wait a minute and try again, or run the following command to watch the startup progress (use Ctrl-C to exit):

docker logs -f my-cassandra

To create a simple schema in your cluster, you'll use the CREATE KEYSPACE and CREATE TABLE commands:

CREATE KEYSPACE my\_keyspace WITH replication = {'class': 'SimpleStrategy', 'replication\_factor': 1};

CREATE TABLE my\_keyspace.user (first\_name text, last\_name text, title text, PRIMARY KEY (last\_name, first\_name));

You can use the DESCRIBE command to verify the schema was created:

DESCRIBE KEYSPACE my\_keyspace;

To get a list of other cqlsh commands, use HELP:

HELP

When you're done using the shell, you can EXIT:

EXIT

Make sure to EXIT now. You'll need to be back at the Bash shell prompt to run the commands in the next step.

#### Step 3

# Creating a Multinode Cassandra Cluster in Docker

One exercise you might find interesting is to launch multiple nodes in Docker to create a small cluster.

Let's two additional nodes using the same network:

docker run --name my-cassandra-2 --network cass-network -d -e CASSANDRA\_SEEDS=my-cassandra cassandra

docker run --name my-cassandra-3 --network cass-network -d -e CASSANDRA\_SEEDS=my-cassandra cassandra

With these commands, you've started two additional nodes on a single Docker instance, providing the first node as a seed node. Once these new nodes start up, they will connect with the original and take over responsibility for a portion of the cluster's data. The schema you've created are automatically replicated to these new nodes.

You might find it interesting to watch the startup progress of one of these new nodes as it joins the cluster and negotiates what data it will be responsible for storing (use Ctrl-C to exit):

docker logs -f my-cassandra-2

If you want to create a cluster spanning containers on multiple Docker instances, you'll need to do a bit more networking, such as making sure that ports Cassandra uses for internode communication are accessible. It's also possible to configure the image to mount an external directory for Cassandra to store its data files. You can find these details in the documentation for the Cassandra Docker image.

# Monitoring a Cassandra Cluster in Docker

nodetool is a command-line program that offers a rich array of ways to look at your cluster, understand its activity, and modify it. nodetool lets you get statistics about the cluster, see the ranges each node maintains, and a variety of management tasks such as moving data from one node to another, decommissioning nodes, repairing nodes, and more.

You can use the nodetool status command on any of the nodes to see the status of the nodes in the cluster.

docker exec -it my-cassandra nodetool status

The nodetool info command provides additional details:

docker exec -it my-cassandra nodetool info

To see details of which nodes own which portions of the token ring, use nodetool ring:

docker exec -it my-cassandra nodetool ring

You can get a list of other available commands using nodetool help:

docker exec -it my-cassandra nodetool help

#### Step 5

# Stopping Cassandra Docker Containers

You can stop the containers you've started by using the docker stop command:

docker stop my-cassandra

docker stop my-cassandra-2

docker stop my-cassandra-3