CASSANDRA - CREATE KEYSPACE

Creating a Keyspace using Cqlsh

A keyspace in Cassandra is a namespace that defines data replication on nodes. A cluster contains one keyspace per node. Given below is the syntax for creating a keyspace using the statement **CREATE KEYSPACE**.

Syntax

```
CREATE KEYSPACE <identifier> WITH <properties>
```

i.e.

```
CREATE KEYSPACE “KeySpace Name”
WITH replication = {'class': ‘Strategy name’, 'replication_factor' : ‘No.Of   replicas’};

CREATE KEYSPACE “KeySpace Name”
WITH replication = {'class': ‘Strategy name’, 'replication_factor' : ‘No.Of  replicas’}

AND durable_writes = ‘Boolean value’;
```

The CREATE KEYSPACE statement has two properties: **replication** and **durable_writes**.

Replication

The replication option is to specify the **Replica Placement strategy** and the number of replicas wanted. The following table lists all the replica placement strategies.

Strategy name	Description
Simple Strategy'	Specifies a simple replication factor for the cluster.
Network Topology Strategy	Using this option, you can set the replication factor for each data-center independently.
Old Network Topology Strategy	This is a legacy replication strategy.

Using this option, you can instruct Cassandra whether to use **commitlog** for updates on the current KeySpace. This option is not mandatory and by default, it is set to true.

Example

Given below is an example of creating a KeySpace.

- Here we are creating a KeySpace named **TutorialsPoint**.
- We are using the first replica placement strategy, i.e.., **Simple Strategy**.
- And we are choosing the replication factor to **1 replica**.

```
cqlsh.> CREATE KEYSPACE tutorialspoint
WITH replication = {'class':'SimpleStrategy', 'replication_factor' : 3};
```

Verification

You can verify whether the table is created or not using the command **Describe**. If you use this command over keyspaces, it will display all the keyspaces created as shown below.

```
cqlsh> DESCRIBE keyspaces;

tutorialspoint system system_traces
```

Here you can observe the newly created KeySpace **tutorialspoint.**

Durable_writes

By default, the durable_writes properties of a table is set to **true,** however it can be set to false. You cannot set this property to **simplex strategy.**

Example

Given below is the example demonstrating the usage of durable writes property.

```
cqlsh> CREATE KEYSPACE test
... WITH REPLICATION = { 'class' : 'NetworkTopologyStrategy', 'datacenter1' : 3 }
... AND DURABLE_WRITES = false;
```

Verification

You can verify whether the durable_writes property of test KeySpace was set to false by querying the System Keyspace. This query gives you all the KeySpaces along with their properties.

```
cqlsh> SELECT * FROM system.schema_keyspaces;

  keyspace_name | durable_writes |                                              strategy_class
| strategy_options
----------------+----------------+--------------------------------------------------------
--+----------------------------

            test |          False | org.apache.cassandra.locator.NetworkTopologyStrategy |
{"datacenter1" : "3"}

 tutorialspoint |           True |          org.apache.cassandra.locator.SimpleStrategy |
{"replication_factor" : "4"}

          system |           True |           org.apache.cassandra.locator.LocalStrategy |
{ }

  system_traces |           True |          org.apache.cassandra.locator.SimpleStrategy |
{"replication_factor" : "2"}

(4 rows)
```

Here you can observe the durable_writes property of test KeySpace was set to false.

Using a Keyspace

You can use a created KeySpace using the keyword **USE**. Its syntax is as follows:

```
Syntax:USE <identifier>
```

Example

In the following example, we are using the KeySpace **tutorialspoint.**

```
cqlsh> USE tutorialspoint;
cqlsh:tutorialspoint>
```

Creating a Keyspace using Java API

You can create a Keyspace using the **execute** method of **Session** class. Follow the steps given

below to create a keyspace using Java API.

Step1: Create a Cluster Object

First of all, create an instance of **Cluster.builder** class of **com.datastax.driver.core** package as shown below.

```
//Creating Cluster.Builder object

Cluster.Builder builder1 = Cluster.builder();
```

Add a contact point *IPaddressofthenode* using **addContactPoint** method of **Cluster.Builder** object. This method returns **Cluster.Builder**.

```
//Adding contact point to the Cluster.Builder object

Cluster.Builder builder2 = build.addContactPoint( "127.0.0.1" );
```

Using the new builder object, create a cluster object. To do so, you have a method called **build** in the **Cluster.Builder** class. The following code shows how to create a cluster object.

```
//Building a cluster
Cluster cluster = builder.build();
```

You can build a cluster object in a single line of code as shown below.

```
Cluster cluster = Cluster.builder().addContactPoint("127.0.0.1").build();
```

Step 2: Create a Session Object

Create an instance of **Session** object using the **connect** method of **Cluster** class as shown below.

```
Session session = cluster.connect( );
```

This method creates a new session and initializes it. If you already have a keyspace, you can set it to the existing one by passing the keyspace name in string format to this method as shown below.

```
Session session = cluster.connect(“ Your keyspace name ” );
```

Step 3: Execute Query

You can execute **CQL** queries using the **execute** method of **Session** class. Pass the query either in string format or as a **Statement** class object to the **execute** method. Whatever you pass to this method in string format will be executed on the **cqlsh**.

In this example, we are creating a KeySpace named **tp.** We are using the first replica placement strategy, i.e., Simple Strategy, and we are choosing the replication factor to 1 replica.

You have to store the query in a string variable and pass it to the execute method as shown below.

```
String query = "CREATE KEYSPACE tp WITH replication "
   + "= {'class':'SimpleStrategy', 'replication_factor':1}; ";
session.execute(query);
```

Step4 : Use the KeySpace

You can use a created KeySpace using the execute method as shown below.

```
execute(“ USE tp ” );
```

Given below is the complete program to create and use a keyspace in Cassandra using Java API.

```
import com.datastax.driver.core.Cluster;
import com.datastax.driver.core.Session;

public class Create_KeySpace {

   public static void main(String args[]){

      //Query
      String query = "CREATE KEYSPACE tp WITH replication "
         + "= {'class':'SimpleStrategy', 'replication_factor':1};";

      //creating Cluster object
      Cluster cluster = Cluster.builder().addContactPoint("127.0.0.1").build();

      //Creating Session object
      Session session = cluster.connect();

      //Executing the query
      session.execute(query);

      //using the KeySpace
      session.execute("USE tp");
      System.out.println("Keyspace created");
   }
}
```

Save the above program with the class name followed by .java, browse to the location where it is saved. Compile and execute the program as shown below.

```
$javac Create_KeySpace.java
$java Create_KeySpace
```

Under normal conditions, it will produce the following output:

```
Keyspace created
```