**Create a Cosmos DB Account with Cassandra API**

Problem:

You are a Data Engineer at a BioTech Corp. Your company has kicked off a trial for a new device that will allow humans to communicate with animals. However, candidates must have a certain blood type to be selected for the trial. More than 100 million candidates have been sourced, and their data has been stored in a data lake. Knowing that there will be a variety of column names and data types among the records, the team has decided to process the data in a NoSQL database.

As the lead Data Engineer on the project, you have been assigned to research how your team can use Azure Cosmos DB as the database solution for the consolidated records. For your research you will use the Azure portal to create an Azure Cosmos DB account for the Cassandra API. Your team has developed prototype scripts that use the Python SDK for Cassandra. You’ll use these scripts to create keyspaces and tables, generate test data, and query the database.

1. Log into the Azure portal using the credentials and button to the right of these instructions.

**Note:** If you get a Tour pop-up, close it.

1. In the search bar at the top of the window type Cosmos DB and click **Azure Cosmos DB**.
2. On the Azure Cosmos DB dashboard, click **Create Azure Cosmos DB account**.
3. In the Select API option window, under **Cassandra** click **Create**.
4. On the Create Azure Cosmos DB Account - Cassandra form, click into the field labeled **Resource Group**. Select the existing resource group that begins with **azure-lab**.
5. Enter the resource group name in the **Account Name** field.
6. For the remaining fields, cofirm the have their default values:
   * Location = (US) West US
   * Capacity mode = Provisioned throughput
7. Change **Apply Free Tier Discount** to **Do Not Apply**.
8. At the bottom of the page, click the **Next: Global Distribution** button.
9. On the Global Distribution screen, select the following options:
   * Geo-Redundency = Enable
   * Multi-region Writes = Enable
10. At the bottom of the page, click the **Review + create** button.
11. On the review screen, note the estimated creation time. It may take up to 15 minutes for the database to be created. Also, double check that the **Account Name** is exactly the same as the **Resource Group** name. On this screen, the Account Name will be prefaced with **(new)**.
12. Click **Create**.

After clicking Create, a Deployment is in progress window appears that allows you to monitor the deployment. Wait for the deployment to complete. As mentioned, this may take 15 minutes.

1. When the screen displays **Your deployment is complete**, click **Go to resource** to view the Azure Cosmos DB account

**Configure Cloud Shell to Connect to the Cassandra Database**

With Cloud Shell, you’ll use a Bash terminal to configure the environment to connect to the Cassandra database. Scripts that set up the environment and connect to the database have been provided in a GitHub repository that you will clone to Cloud Shell.

1. Open a **Cloud Shell** by clicking the shell icon in the upper right hand corner of the Azure portal toolbar. The Cloud Shell icon is the first icon to the right of the search bar.

**Note:** Leave the Azure portal at the **Azure Cosmos DB account**'s **Overview** page, as you will return to that in the next challenge.

1. Clicking the Cloud Shell icon opens a pane on the bottom half of the browser window. In the **Welcome to Azure Cloud Shell** pop-up, click **Bash**.
2. A second pop-up opens indicating that no storage is mounted. Click the **Create storage** button. It will take a few minutes for the storage to be created. Once the storage is in place the pane is replaced with a Bash terminal.
3. Clone the materials into your terminal with the following command:

git clone <https://github.com/ps-interactive/lab_azure_build-cassandra-app-with-python-sdk-azure-cosmos-db.git>

A screenshot of a computer

Description automatically generated

1. Move into the cloned directory by entering the following command:

cd lab\_azure\_build-cassandra-app-with-python-sdk-azure-cosmos-db

1. Configure the environment by sourcing a bash script with the following command:

source 0-set-cosmosdb-environment.sh

The script will output that the environment variables DB\_HOSTNAME, DB\_USERNAME, and DB\_PASSWORD have been set.

1. Create and activate a Python virtual environment with the following two commands:

python -m venv lab

source ./lab/bin/activate

**Note:** After running the source command, the right side of the terminal prompt should now display **(lab)**.

1. Update pip and install the dependencies needed to run the scripts with the following commands:

pip install -U pip

pip install -U -r requirements.txt

A screen shot of a computer screen

Description automatically generated

1. After the environment has been set and all dependencies have been installed, confirm that your Cloud Shell has been successfully configured to connect to your Cassandra database with the following command:

python 1-test-connection.py

The output from the connection test should display lines ending with

**Connection test was successful!**

**Create a Cassandra Key space and Tables**

In this challenge you’ll be running commands in the Cloud Shell and observing the results in the Azure portal. The best approach will be to keep both the Cloud Shell and portal open.

If you close the Cloud Shell and reopen it, you’ll need to first reconfigure the environment by changing into the lab directory, sourcing the environment configuration script, and sourcing the virtual environment activation script (see the previous challenge).

1. In the Azure portal, at the **Azure Cosmos DB account**'s **Overview** page, click **Data Explorer**.

**Note:** On the Data Explorer panel, note that there are no keyspaces or tables created.

1. In the Cloud Shell, run the following command to create the customers keyspace and tables belonging to the keyspace:

python 2-create-keyspace-and-tables.py

A computer screen with a black screen

Description automatically generated

The output should indicate that the keyspace and tables were successfully created.

1. In the portal, click the circular **Refresh tree** arrow next to **CASSANDRA API** to refresh the keyspace tree.
2. Expand the **candidates** keyspace and each table under the keyspace. Under each table, click **Rows**. Note that the tables don't have any data.
3. In the Cloud Shell, run the following command to generate 500 customer records filled with fake data:

python 3-generate-data.py

1. After starting the command, in the portal under **blood\_group** click **Rows**, and toward the top of the page click the **CQL Query Text** button.

This creates a default query to grab all rows from the table.

1. Click **Run Query**.

A screenshot of a computer

Description automatically generated

1. Repeat tasks 6 and 7 for the other two tables.

The display is updated with 100 rows of the sample data. You could click the numbers below the data to view the next 100 rows. To load more data, you could click Load more.

**Running Queries with Python Scripts**

1. The provided Python scripts use the **session.execute()** function to run queries against the Cassandra database. Take a moment to review the queries that each script will run against the database. Note that some of the scripts will use one query while others will use multiple queries.

grep session.execute 4-count-records.py

grep session.execute 5-count-records-by-gender.py

grep session.execute 6-count-blood-groups-by-gender.py

grep session.execute 7-count-records-by-blood-group.py

1. Run the script that reports a count of the records in the database and review the output:

python 4-count-records.py

1. Run the script that reports a count of the records by gender and review the output:

python 5-count-records-by-gender.py

1. Run the script that reports a count of the blood groups by gender and review the output:

python 6-count-blood-groups-by-gender.py

1. Run the script that reports a count of the records by blood group and review the output:

python 7-count-records-by-blood-group.py

This last script will display results similar to this table:

+----+----+----+----+----+-----+-----+

| A- | B+ | B- | O+ | O- | AB+ | AB- |

+----+----+----+----+----+-----+-----+

| 70 | 59 | 70 | 62 | 55 | 58 | 67 |

+----+----+----+----+----+-----+-----+

A computer screen shot of a computer screen

Description automatically generated

References:

<https://learn.microsoft.com/en-us/azure/cosmos-db/cassandra/manage-data-python>