Assigning HBase Resources in HDInsight

Lab 6 - Getting Started with HBase

Overview

In this lab, you will provision an HDInsight HBase cluster. You will then create an HBase table and use it to store data.

What You’ll Need

To complete the labs, you will need the following:

* A web browser
* A Microsoft account
* A Microsoft Azure subscription
* A Microsoft Windows computer containing: o Microsoft Visual Studio/VS code

o The lab doc for this course

**Note**: To set up the required environment for the lab, follow the instructions in the **Setup** document forthis course. Specifically, you must have signed up for an Azure subscription.

Provisioning an HDInsight HBase Cluster

The first task you must perform is to provision an HDInsight HBase cluster.

**Note**: The Microsoft Azure portal is continually improved in response to customer feedback. The steps inthis exercise reflect the user interface of the Microsoft Azure portal at the time of writing, but may not match the latest design of the portal exactly.

Provision an HDInsight Cluster

1. In a web browser, navigate to [http://portal.azure.com,](http://portal.azure.com/) and if prompted, sign in using the Microsoft account that is associated with your Azure subscription.
2. In the Microsoft Azure portal, add a new HDInsight cluster with the following settings:
   * **Cluster Name**:*Enter a unique name (and make a note of it!)*
   * **Cluster Type**: HBase 1.1.2(HDI 3.6) [As of writing this Lab]
   * **Operating System**:*Choose the latest version of Linux Server*
   * **Subscription**:*Your Azure subscription*
   * **Resource Group**:*Create a new resource group with a unique name*
   * **Credentials**:
     + **Cluster Login Username**:*Enter a user name of your choice (and make a note of it!)*
     + **Cluster Login Password**:*Enter and confirm a strong password (and make a note of it!)*
     + **Enable Remote Desktop:** Yes
     + **Expires on**:*Select the date a week from now*
     + **SSH Username:** *Enter another user name of your choice (and make a**note of it!)*
     + **SSH Password:** *Enter and confirm a strong password (and make a note**of it!)*
   * **Data Source**:
     + **Selection Method**: From all subscriptions
     + **Create a new storage account**:*Enter a unique name for your storage account (and**make a note of it!)*
     + **Choose Default Container**:*The name of your cluster*
     + **Location**:*Select any available region*.
   * **Optional Configuration**:
     + **HDInsight Version**:*Select the most recent version available*
     + **Virtual Network**: Not Configured
     + **External Metastores**: Not Configured
     + **Script Actions**: Not Configured
     + **Azure Storage Keys**: Not Configured
   * **Node Pricing Tiers**:
     + **Number of Worker nodes**: 1
     + **Worker Nodes Pricing Tier**:*Leave the default selection*
     + **Head Node Pricing Tier**:*Leave the default selection*
   * **Pin to Startboard**:*Not selected*
3. In the Azure portal, view **Notifications** to verify that deployment has started. Then wait for the cluster to be deployed (this can take a long time – often 30 minutes or more. Now may be a good time to go and have a cup of coffee!)

**Note**: As soon as an HDInsight cluster is running, the credit in your Azure subscription will start to becharged. The free-trial subscription includes a credit limit of approximately $100 (or local equivalent) that you can spend over a period of 30 days, which is enough to complete the labs in this course as long as clusters are deleted when not in use. If you decide not to complete this lab, follow the instructions in the *Clean Up* procedure at the end of the lab to delete your cluster in order to avoid using your Azure credit unnecessarily.

Creating an HBase Table

Now that you have provisioned an HDInsight HBase cluster, you can create HBase tables and store data in them.

Open a Remote Desktop Connection to the Cluster

1. In the Azure portal, browse to the HBase cluster you just created.
2. Open a SSH connection to the cluster using the SSH username and password you specified when provisioning the cluster.
3. When the SSH connection connects, open the **Hadoop Command Line** console and view the syntax documentation for the Hadoop command line tool.

Create an HBase Table

**Note**: The commands in this procedure arecase-sensitive.

1. In the Hadoop Command Line console window, enter the following command to change the current directory to the HBase installation directory:

cd %HBASE\_HOME%\bin

1. Enter the following command to start the HBase shell. hbase shell
2. Enter the following command to create a table named **Stocks** with two column families named **Current** and **Closing**.

create 'Stocks', 'Current', 'Closing'

1. Enter the following command to insert a field for a record with the key **ABC** and a value of **97.3** for a column named **Price** in the **Current** column family.

put 'Stocks', 'ABC', 'Current:Price', '97.3'

1. Enter the following command to insert a field for record **ABC** and a value of **95.7** for a column named **Price** in the **Closing** column family.

put 'Stocks', 'ABC', 'Closing:Price', '95.7'

1. Enter the following command to return all rows from the table. scan 'Stocks'
2. Verify that the output shows the two values you entered for the row ABC, as shown here:

ROW

ABC

ABC

COLUMN+CELL column=Closing:Price, column=Current:Price,

timestamp=*nnn*, value=95.7 timestamp=*nnn*, value=97.3

1. Enter the following command to insert a field for record **ABC** and a value of **Up** for a column named **Status** in the **Current** column family.

put 'Stocks', 'ABC', 'Current:Status', 'Up'

1. Enter the following command to return the values for row ABC. get 'Stocks', 'ABC'
2. Verify that the output shows the values of all cells for row ABC, as shown here:

COLUMNCELL

Closing:Pricetimestamp=*nnn*, value=95.7

Current:Pricetimestamp=*nnn*, value=97.3

Current:Statustimestamp=*nnn*, value=Up

1. Enter the following command to set the **Price** column in the **Current** column family of row **ABC** to **99.1**.

put 'Stocks', 'ABC', 'Current:Price', '99.1'

1. Enter the following command to return the values for row ABC.

get 'Stocks', 'ABC'

1. Verify that the output shows the updated values of all cells for row ABC, as shown here:

COLUMNCELL

Closing:Pricetimestamp=*nnn*, value=95.7

Current:Pricetimestamp=*nnn*, value=99.1

Current:Statustimestamp=*nnn*, value=Up

1. Note the **timestamp** value for the **Current:Price** cell. Then enter the following command to retrieve the previous version of the cell value by replacing ***nnn-1*** with the timestamp for **Current:Price** minus 1 (for example, if the timestamp for **Current:Price** in the results above is144012345678, replace ***nnn-1*** with 144012345677.)

get 'Stocks', 'ABC', {TIMERANGE=>[0,***nnn-1***]}

1. Verify that the output shows previous **Current:Price** value, as shown here:

COLUMNCELL

Closing:Pricetimestamp=*nnn*, value=95.7

Current:Pricetimestamp=*nnn*, value=97.3

Current:Statustimestamp=*nnn*, value=Up

1. Enter the following command to delete the **Status** column in the **Current** column family of row **ABC**.

delete 'Stocks', 'ABC', 'Current:Status'

1. Enter the following command to return the values for row ABC. get 'Stocks', 'ABC'
2. Verify that the **Current:Status** cell has been deleted as shown here:

COLUMNCELL

Closing:Pricetimestamp=*nnn*, value=95.7

Current:Pricetimestamp=*nnn*, value=99.1

1. Enter the following command to exit the HBase shell and return to the Hadoop command line. quit
2. Minimize the remote desktop window (you will return to the Hadoop Command Line later.)

Bulk Load Data into an HBase Table

1. In the C:\HDRTLabs\Lab01 folder, double-click **stocks.txt** to open the file in Notepad. Note that this file contains tab-delimited records of closing and current prices for a variety of stocks. Then close Notepad without saving any changes.
2. In the C:\HDRTLabs\Lab01 folder, right-click **stocks.txt** and click **Copy** to copy the file to the clipboard.
3. Maximize the remote desktop window, open File Explorer, and view the content of the **C:\** folder on your cluster head node. Then paste the copied stocks.txt file into C:\.
4. In the **Hadoop Command Line** window, enter the following commands to create a folder named **/data** in the HDFS file system for the cluster, and copy the stocks.txt file from the local C:\ folderto the new **/data** folder.

hadoop fs -mkdir /data

hadoop fs -copyFromLocal c:\stocks.txt /data/stocks.txt

1. In the **Hadoop Command Line** window, enter the following command (on a single line) to transform the tab-delimited stocks.txt data to the HBase StoreFile format.

hbase org.apache.hadoop.hbase.mapreduce.ImportTsv - Dimporttsv.columns="HBASE\_ROW\_KEY,Closing:Price, Current:Price" - Dimporttsv.bulk.output="/data/storefile" Stocks /data/stocks.txt

1. Enter the following command (on a single line) to load the transformed data into the **Stocks** table you created previously.

hbase org.apache.hadoop.hbase.mapreduce.LoadIncrementalHFiles /data/storefile Stocks

1. Enter the following command to start the HBase shell. hbase shell
2. Enter the following command to return all rows from the table. scan 'Stocks'
3. Verify that the output includes rows for the ABC stock you entered previously and the stocks in the stocks.txt file you imported, as shown here:

ROW

AAA

AAA

ABC

ABC

BBB

BBB

CBA

CBA

GDM

GDM

COLUMN+CELL column=Closing:Price, column=Current:Price, column=Closing:Price, column=Current:Price, column=Closing:Price, column=Current:Price, column=Closing:Price, column=Current:Price, column=Closing:Price, column=Current:Price,

timestamp=*nnn*, value=12.8 timestamp=*nnn*, value=14.2 timestamp=*nnn*, value=95.7 timestamp=*nnn*, value=99.1 timestamp=*nnn*, value=30.1 timestamp=*nnn*, value=30.1 timestamp=*nnn*, value=120.3 timestamp=*nnn*, value=120.3 timestamp=*nnn*, value=126.7 timestamp=*nnn*, value=135.2

...

1. Enter the following command to return only the **Current:Price** column for each row: scan 'Stocks', {COLUMNS => 'Current:Price'}
2. Verify that the output includes a row for each stock with only the Current:Price column, as shown here:

ROW

AAA

ABC

BBB

CBA

GDM

COLUMN+CELL column=Current:Price, column=Current:Price, column=Current:Price, column=Current:Price, column=Current:Price,

timestamp=*nnn*, value=14.2 timestamp=*nnn*, value=99.1 timestamp=*nnn*, value=30.1 timestamp=*nnn*, value=120.3 timestamp=*nnn*, value=135.2

...

1. Enter the following command to return only the first three rows: scan 'Stocks', {LIMIT => 3}
2. Verify that the output includes data for only three rows (there are two columns per row), as shown here:

ROW

AAA

AAA

ABC

ABC

BBB

BBB

COLUMN+CELL column=Closing:Price, column=Current:Price, column=Closing:Price, column=Current:Price, column=Closing:Price, column=Current:Price,

timestamp=*nnn*, value=12.8 timestamp=*nnn*, value=14.2 timestamp=*nnn*, value=95.7 timestamp=*nnn*, value=99.1 timestamp=*nnn*, value=30.1 timestamp=*nnn*, value=30.1

1. Enter the following command to return only the rows for with key values between C and H: scan 'Stocks', {STARTROW=>'C', STOPROW=>'H'}
2. Verify that the output includes only rows for stocks with stock codes between ‘C’ and ‘H’, as shown here:

ROW

CBA

CBA

GDM

GDM

COLUMN+CELL column=Closing:Price, column=Current:Price, column=Closing:Price, column=Current:Price,

timestamp=*nnn*, value=120.3 timestamp=*nnn*, value=120.3 timestamp=*nnn*, value=126.7 timestamp=*nnn*, value=135.2

16. Minimize the remote desktop window (you will return to the HBase Shell in the next exercise.)

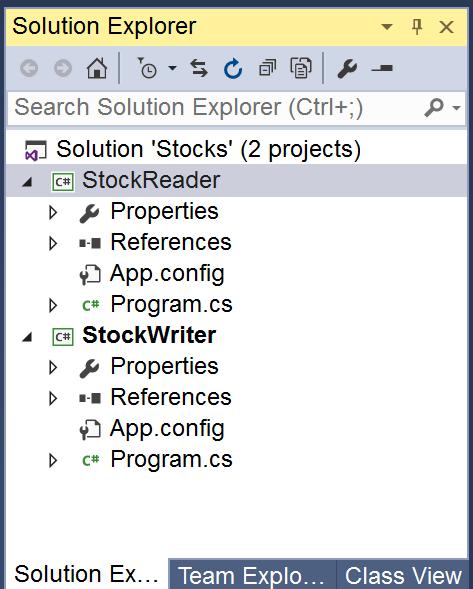
Using the Microsoft .NET HBase REST API Client

The Microsoft .NET HBase REST API is a .NET wrapper around the HTTP REST-based interface for Base. This API makes it easier to create .NET client applications for HBase than programming directly against the REST interface.

In this exercise, you will create two simple .NET console applications; one to constantly update stock data in HBase, and the other to read price data for a specific stock symbol.

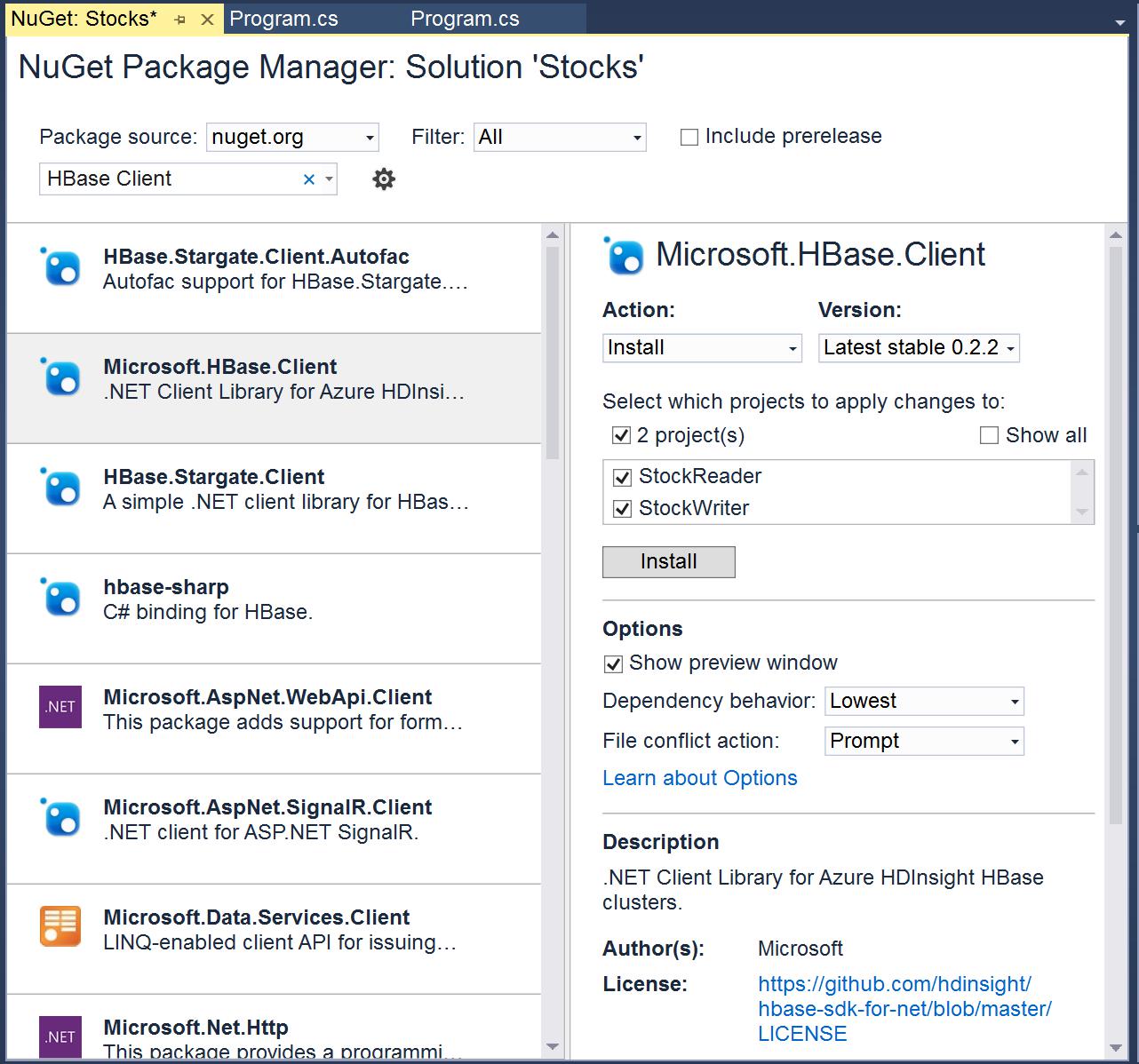
Create Projects for the StockWriter and StockReader Applications

1. Start Visual Studio and on the **File** menu, point to **New**, and click **Project**. Then create a new project based on the Visual C# **Console Application** template. Name the project **StockWriter**, name the solution **Stocks**, and save it in the C:\HDRTLabs\Lab01 folder.
2. On the **File** menu, point to **Add** and click **New Project**. Then add a Visual C# **Console Application** project named **StockReader** to the solution.
3. If the Solution Explorer pane is not visible, on the **View** menu, click **Solution Explorer**; and then verify that your **Stocks** solution contains two projects named **StockReader** and **StockWriter** as shown here:



Add the .NET HBase Client Package

1. On the **Tools** menu, point to **NuGet Package Manager**, and click **Manage NuGet Packages for** **Solution**.
2. In the NuGet Package Manager window, search nuget.org for *HBase Client*, and in the list of results, select **Microsoft.HBase.Client**. Then configure the following settings and click **Install**, as shown below:
   * **Action**: Install
   * **Version**: Latest stable x.x.x
   * **Select which projects to apply changes to**:*Select* ***StockReader*** *and* ***StockWriter***.



1. If you are prompted to review changes, click **OK**.
2. When the package has been installed, close the NuGet Package Manager window.

Implement the StockWriter Application

1. In Solution Explorer, under **StockWriter**, double-click **Program.cs** to open the main code file for the **StockWriter** project.
2. At the top of the code file, replace all of the existing using statements with the following code. You can copy and paste this code from **StockWriter.txt** in the C:\HDRTLabs\Lab01 folder.

using System; using System.Text;

using Microsoft.HBase.Client;

using org.apache.hadoop.hbase.rest.protobuf.generated;

1. In the static void **Main** function, add the following code, replacing the values for the **clusterURL**, **userName**, and **password** variables with the appropriate values for your HDInsight cluster. Youcan copy and paste this code from **StockWriter.txt** in the C:\HDRTLabs\Lab01 folder.

while (true)

{

Random rnd = new Random(); Console.Clear();

string clusterURL = "https://***hb12345***.azurehdinsight.net"; string userName = "***HDUser***";

string password = "***HDPa$$w0rd***";

// Connect to HBase cluster

ClusterCredentials creds =new ClusterCredentials(

new Uri(clusterURL),

userName, password);

HBaseClient hbaseClient = new HBaseClient(creds);

// Get all stocks

Scanner scanSettings = new Scanner()

{

batch = 10,

startRow = Encoding.UTF8.GetBytes("AAA"), endRow = Encoding.UTF8.GetBytes("ZZZ")

};

ScannerInformation stockScanner = hbaseClient.CreateScanner("Stocks", scanSettings);

CellSet stockCells = null;

while ((stockCells = hbaseClient.ScannerGetNext(stockScanner))

!= null)

{

foreach (var row in stockCells.rows)

{

string stock = Encoding.UTF8.GetString(row.key); Double currentPrice = Double.Parse(

Encoding.UTF8.GetString(row.values[1].data)); Double newPrice = currentPrice +

(rnd.NextDouble() \* (1 - -1) + -1); Cell c = new Cell { column =

Encoding.UTF8.GetBytes("Current:Price"),

data =

Encoding.UTF8.GetBytes(newPrice.ToString())};

row.values.Insert(2, c);

Console.WriteLine(stock + ": " + currentPrice.ToString() + " := " + newPrice.ToString());

}

hbaseClient.StoreCells("Stocks", stockCells);

}

}

**Note**: This code performs the following actions:

* 1. Connects to your HBase cluster using the URL and credentials in your code.
  2. Uses a **Scanner** object to retrieve a cellset containing all stock records from the **Stocks** HBase table you created earlier in this lab. This is a wrapper around the **scan** HBase command.
  3. Loops through each stock record, incrementing the value of the first column (**Current:Price**) by a random value between -1 and 1.
  4. Stores the updated cells back to the table.

1. Save **Program.cs** and close it.

Implement the StockReader Application

1. In Solution Explorer, under **StockReader**, double-click **Program.cs** to open the main code file for the **StockReader** project.
2. At the top of the code file, replace all of the existing using statements with the following code. You can copy and paste this code from **StockReader.txt** in the C:\HDRTLabs\Lab01 folder.

using System; using System.Text;

using Microsoft.HBase.Client;

using org.apache.hadoop.hbase.rest.protobuf.generated;

1. In the static void **Main** function, add the following code, replacing the values for the **clusterURL**, **userName**, and **password** variables with the appropriate values for your HDInsight cluster. Youcan copy and paste this code from **StockReader.txt** in the C:\HDRTLabs\Lab01 folder.

bool quit = false; while (!quit)

{

Console.ResetColor();

Console.WriteLine("Enter a stock code, or enter 'quit' to exit");

// Connect to HBase cluster

string clusterURL = "https://***hb12345***.azurehdinsight.net"; string userName = "***HDUser***";

string password = "***HDPa$$w0rd***"; ClusterCredentials creds = new ClusterCredentials

(new Uri(clusterURL),

userName,

password);

HBaseClient hbaseClient = new HBaseClient(creds);

string input = Console.ReadLine();

if (input.ToLower() == "quit")

{

quit = true;

}

else

{

CellSet cellSet = hbaseClient.GetCells("Stocks", input); var row = cellSet.rows[0];

Double currentPrice = Double.Parse( Encoding.UTF8.GetString(row.values[1].data));

Double closingPrice = Double.Parse( Encoding.UTF8.GetString(row.values[0].data));

if (currentPrice > closingPrice)

{

Console.ForegroundColor = ConsoleColor.Green;

}

else if (currentPrice < closingPrice)

{

Console.ForegroundColor = ConsoleColor.Red;

}

Console.WriteLine(input + ": " + currentPrice.ToString());

}

}

**Note**: This code performs the following actions:

* 1. Connects to your HBase cluster using the URL and credentials in your code.
  2. Reads the input from the command line (which is assumed to be either a stock code or the command “quit”).
  3. Uses the **GetCells** method to retrieve the record in the HBase **Stocks** table for the specified stock code key value. This is a wrapper around the **get** HBase command.
  4. Reads the first (**Current:Price**) and second (**Closing:Price**) cells from the cellset.
  5. Displays the current stock price, with color coding to indicate whether it is higher or lower than the closing price.

1. Save **Program.cs** and close it.

Build and Test the Applications

1. On the **Build** menu, click **Build Solution**.
2. When both projects have been built, in Solution Explorer, right-click **StockReader**, point to **Debug**, and click **Start new instance**. This opens a console window for the StockReaderapplication.
3. In the StockReader console window, enter **AAA**, and note that the current stock price for stock **AAA** is displayed.

**Note**: The code in the application has been kept deliberately simple to make it easy tounderstand the key methods for working with HBase. The application contains no error handling, and will crash if an invalid stock code is entered. In a production application, you would add error handling code to prevent this.

1. In Solution Explorer, right-click **StockWriter**, point to **Debug**, and click **Start new instance**. This opens a console window for the StockWriter application.
2. Observe the StockWriter console window, noting that it displays a constant sequence of updated stock prices.
3. In the StockReader console window, enter **AAA**, and note that the latest current stock price for stock **AAA** is displayed.
4. In the StockReader console window, enter **BBB**, and note that the latest current stock price for stock **BBB** is displayed.
5. Repeat the previous two steps, noting that the latest price for the specified stock is always retrieved from HBase.
6. In the StockReader console window, enter **quit** to close the console window. Then close the StockWriter console window to end the application.
7. Close Visual Studio, saving your work of prompted.

Clean Up

Now that you have finished using HBase, you can delete your cluster and the associated storage account. This ensures that you avoid being charged for cluster resources when you are not using them. If you are using a trial Azure subscription that includes a limited free credit value, deleting the cluster maximizes your credit and helps to prevent using it all before the free trial period has ended.

Delete the Resource Group

1. Close the browser tab containing the HDInsight Query Console if it is open.
2. In the Azure portal, view your **Resource groups** and select the resource group you created for your cluster. This resource group contains your cluster and the associated storage account.
3. In the blade for your resource group, click **Delete**. When prompted to confirm the deletion, enter the resource group name and click **Delete**.
4. Wait for a notification that your resource group has been deleted.
5. Close the browser.