Microsoft

SQL Server 2016 CTP2 JSON HOL

Contents

Overview and setup

Estimated time to complete lab is 50-55 minutes.

Overview

JSON is a popular, language-independent data-interchange format used in modern web and mobile applications, as well for storing unstructured data. JSON is an alternate to XML and is more compact than that format, and as such has become the first choice for many applications that need to move data around on the web. One of the biggest reasons JSON is becoming more important than XML is that XML has to be parsed with an XML parser, while JSON can be parsed by a standard JavaScript function. This makes it easier and faster than working with XML. JSON is also the storage format used in several DB/NoSQL engines, including Azure DocumentDB.

This hands on lab will familiarize you with the JSON support in SQL Server and help you understand how to implement it. In particular, you will learn:

1. How to query data in SQL tables and format the output as JSON, using the FOR JSON PATH and FOR JSON AUTO syntax options;
2. How to control the JSON output structure, including creating path hierarchies, handling NULL values, and creating root keys;
3. How to move data from a relational format in SQL Server to a JSON format and insert it to a DocumentDB no-sql database.

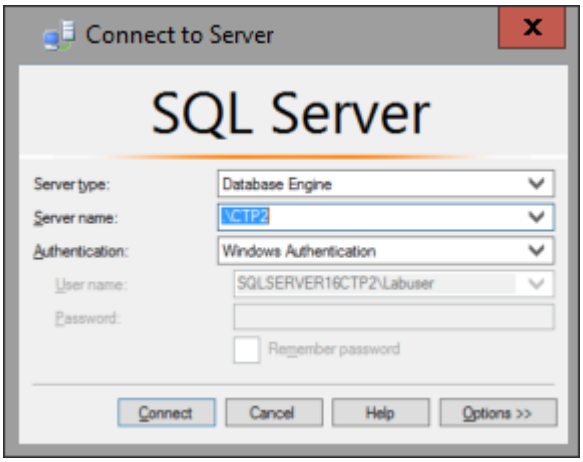
This lab is divided into 3 parts to help you learn how to query relational data in SQL Server and output it as JSON. First, we will walk through an example of a simple JSON query with FOR JSON AUTO and examine the general structure of JSON data. Second, we will examine how we can exercise more control over the JSON output and how to control the treatment of null values. Finally, we will create a new Azure DocumentDB and move data into it using the FOR JSON syntax. At the end of this lab, you will have worked through several of the most common scenarios involved with the new JSON feature of SQL Server 2016 CTP2.

Simple query with FOR JSON AUTO

In this lab, we will explore how we can output data from a simple SQL query in a JSON format and what JSON data looks like. We'll also add a root key to wrap around the JSON data. Let's get started!

Query SQL data and output as JSON

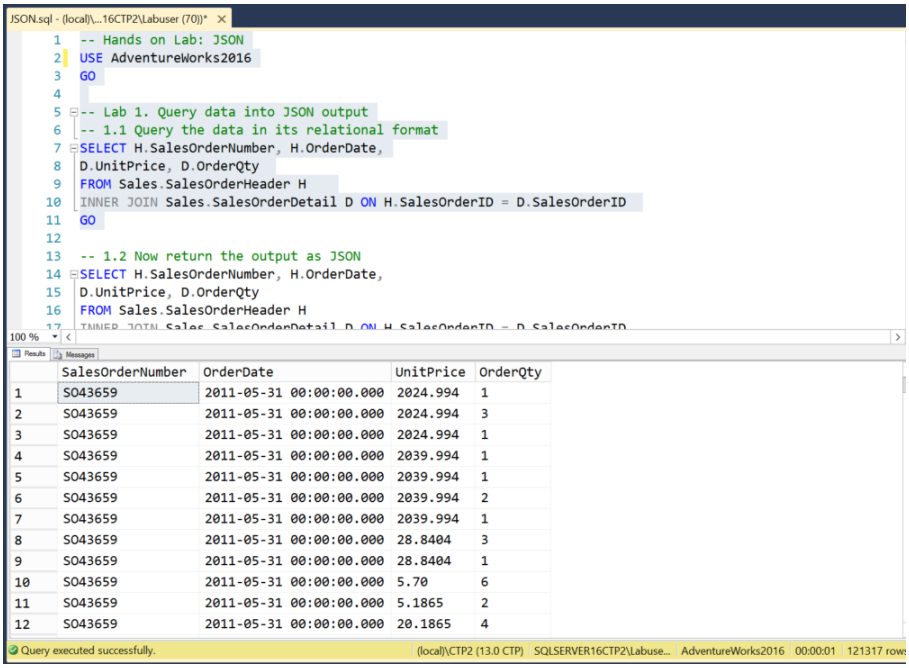
1. Open **SQL Server Management Studio** and connect the **.\CTP2** instance.



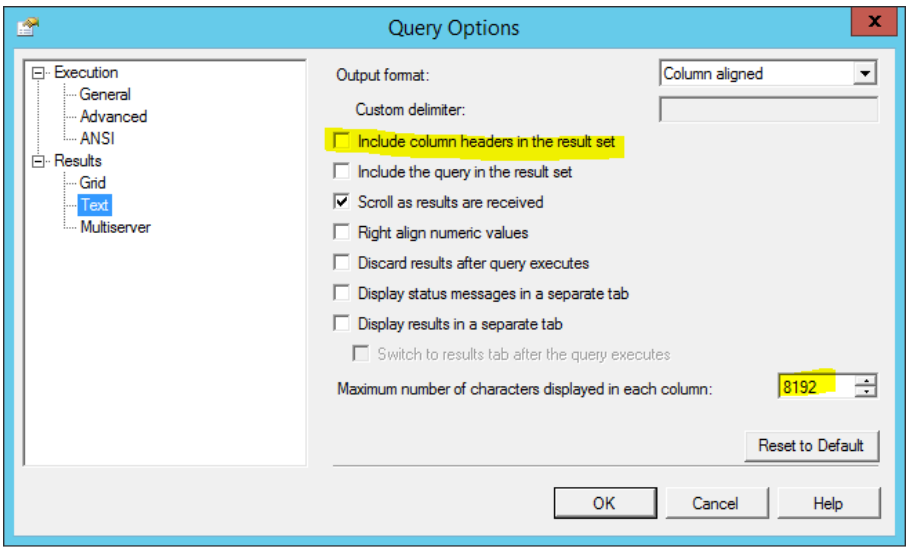
2. Expand out the **Databases** folder and click on the **AdventureWorks2016** database.

3. Press **Ctrl+O** and open the file Open the **C:\SQL Server 2016 CPT2 HOLs\JSON.sql**

4. Select the following query text and execute the code to show the output of the results you will see in JSON format in the next step.

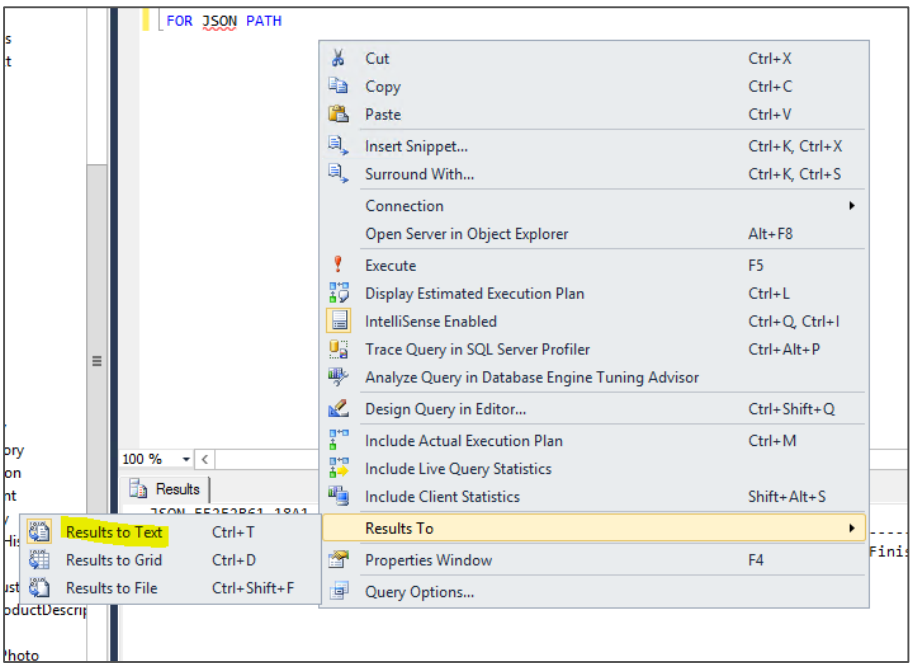


5. Right click in the query editor and select **Query Options**. Under **Results**, select **Text**. Uncheck "**Include column headers in the result set**" and change the maximum number of characters displayed to **8192**.

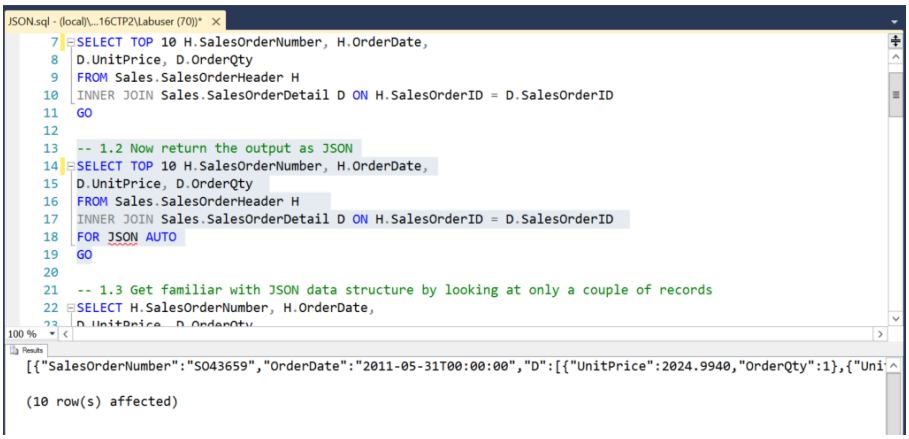


6. Click on in the editor and click **Results To** and choose **Results to Text** and click **OK**.

   Note: The default maximum number of characters displayed in each column is 256. The maximum allowed value is 8192.



7. Back in the query editor, select the next SQL statement that includes **FOR JSON AUTO** and execute it.



Notice that there are the same number of rows, but that the data is formatted as JSON.

Note: A common use for this construct would be to return the JSON results into a local variable, or to wrap the query in a user-defined function to make it reusable.

Explore JSON data structure

Note that XML is implemented as a datatype in SQL Server. This is not how JSON is handled in SQL Server 2016 CTP2. Instead, it is created in the querying process. If you want to store JSON data, you should use nvarchar as the datatype.

JSON syntax is simple and human-readable. JSON values consist of name/value pairs and individual values are separated by commas. Objects are containers of one or more name/value pairs and are contained within curly brackets. JSON arrays can contain multiple objects and arrays are contained within square brackets.

1. Select the next T-SQL SELECT statement that includes a **WHERE** clause to the query and execute it to examine the results:

```
21  -- 1.3 Get familiar with JSON data structure by looking at only a couple of records
22  SELECT H.SalesOrderNumber, H.OrderDate,
23  D.UnitPrice, D.OrderQty
24  FROM Sales.SalesOrderHeader H
25  INNER JOIN Sales.SalesOrderDetail D ON H.SalesOrderID = D.SalesOrderID
26  WHERE H.SalesOrderID IN (43660, 43669)
27  FOR JSON AUTO
28  GO
```

```
[{"SalesOrderNumber":"SO43660","OrderDate":"2011-05-31T00:00:00","D":[{"UnitPrice":419.4589,"OrderQty":1},{"Unit

(3 row(s) affected)
```

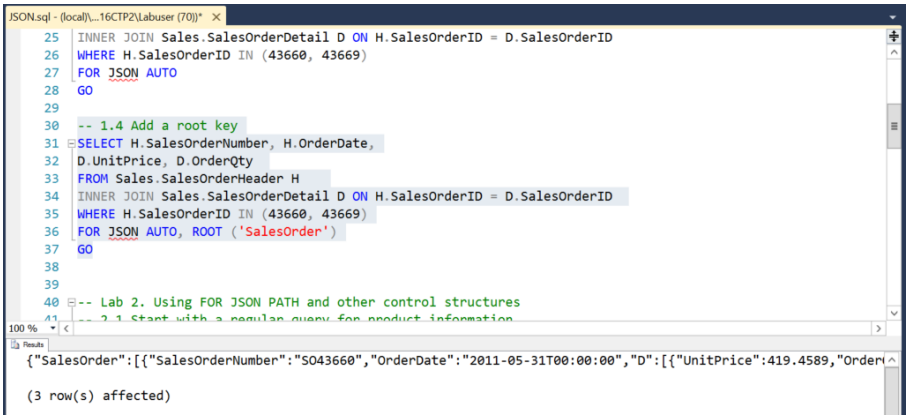
Notice that the outside of the result set is contained within 2 square brackets, since this is an array of 2 orders. The first order has an array of order details (also within square brackets) and each name/value pair is contained within curly brackets.

Notice also that SSMS returns "3 rows affected" since one of the orders had 2 associated detail records, even though the JSON results show 2 sales order objects.

Add a root key to JSON output

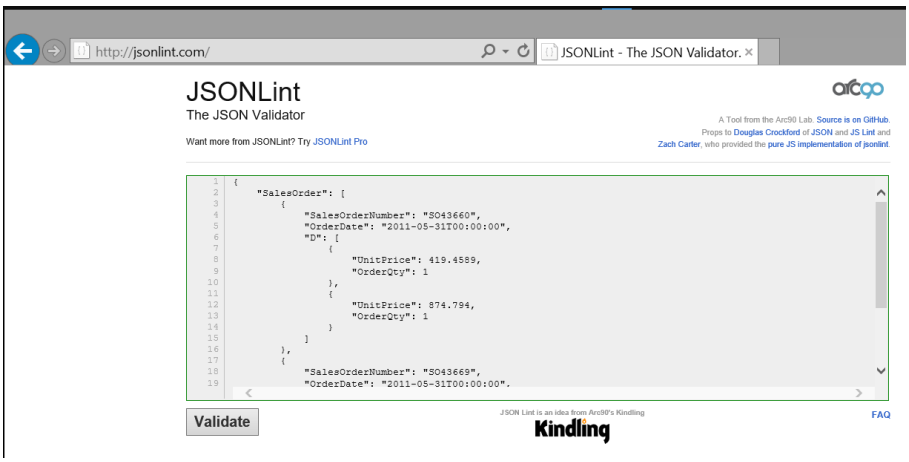
Now let's add a root key for all the sales orders that we are looking at. SQL Server 2016 CTP2 has a special syntax for this.

1. Select the next query that as the "**ROOT**" element (we'll name the root "**SalesOrder**") to the **FOR JSON** clause after a comma and execute it to see the results.



2. If we manually format the structure with line returns and indentations, it is easier to see how the data is laid out. You can also use an online formatter like http://jsonlint.com/.

```
{"SalesOrder":
  [
    {"SalesOrderNumber":"SO43660","OrderDate":"2011-05-
      31T00:00:00",
      "D":
      [{"UnitPrice":419.4589,"OrderQty":1},
       {"UnitPrice":874.7940,"OrderQty":1}
      ]},
    {"SalesOrderNumber":"SO43669","OrderDate":"2011-05-
      31T00:00:00",
      "D":
      [{"UnitPrice":714.7043,"OrderQty":1}
      ]}
  ]
}
```



Note: Various online JSON formatters are available. Search for "JSON validator" to find several others.

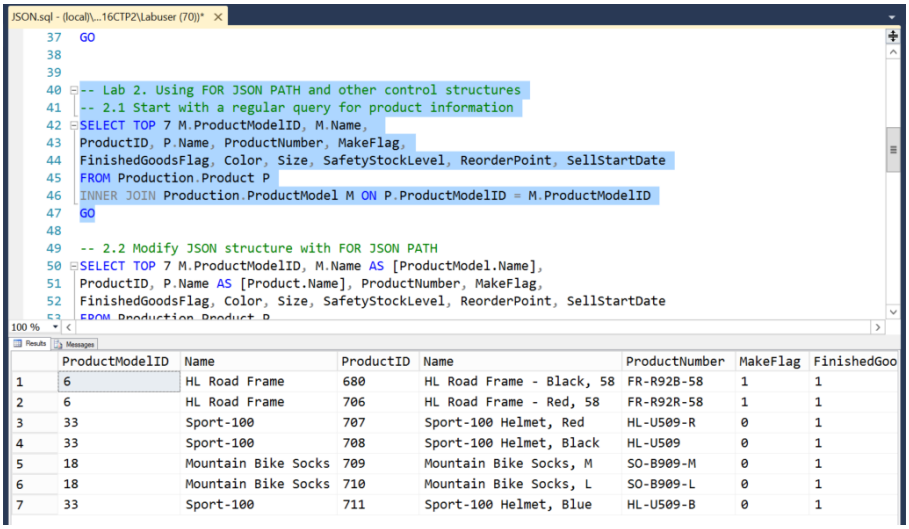
Using FOR JSON PATH

In this exercise, we will build upon what we learned concerning FOR JSON and exercise more control over the output. We will modify the JSON path structure using the FOR JSON PATH syntax. Additionally, we will look at how to modify the treatment of NULL values in JSON data and learn about an alternate syntax form.

In this scenario we are going to work with Product data from the AdventureWorks2016 database so we can start to get it into a hypothetically defined JSON structure that we will use in the next part of the lab to import to a JSON-based Azure DocumentDB no-sql database.

Modify JSON structure with FOR JSON PATH

1. Let's start with a regular query that returns ProductModels (groups of related products), plus information about related products. Right click in the query editor and choose Results To | Results to Grid. Run the query. Note that some of the Size attributes have NULL values.



	ProductModelID	Name	ProductID	Name	ProductNumber	MakeFlag	FinishedGoo
1	6	HL Road Frame	680	HL Road Frame - Black, 58	FR-R92B-58	1	1
2	6	HL Road Frame	706	HL Road Frame - Red, 58	FR-R92R-58	1	1
3	33	Sport-100	707	Sport-100 Helmet, Red	HL-U509-R	0	1
4	33	Sport-100	708	Sport-100 Helmet, Black	HL-U509	0	1
5	18	Mountain Bike Socks	709	Mountain Bike Socks, M	SO-B909-M	0	1
6	18	Mountain Bike Socks	710	Mountain Bike Socks, L	SO-B909-L	0	1
7	33	Sport-100	711	Sport-100 Helmet, Blue	HL-U509-B	0	1

2. Add the FOR JSON PATH clause at the end and try to execute the query. You will get an error that says JSON will not allow the same attribute name more than once. Here we have the Name field in both tables. We will fix this by adding an alias for each of these. The column names define the path hierarchical structure with a Parent.Child syntax, so modify the column aliases for the 2 Name columns to [ProductModel.Name] and [Product.Name].

```
SELECT TOP 7 M.ProductModelID, M.Name AS
[ProductModel.Name],
ProductID, P.Name AS [Product.Name], ProductNumber,
MakeFlag,
FinishedGoodsFlag, Color, Size, SafetyStockLevel,
ReorderPoint, SellStartDate
FROM Production.Product P
INNER JOIN Production.ProductModel M ON P.ProductModelID
= M.ProductModelID
FOR JSON PATH
```

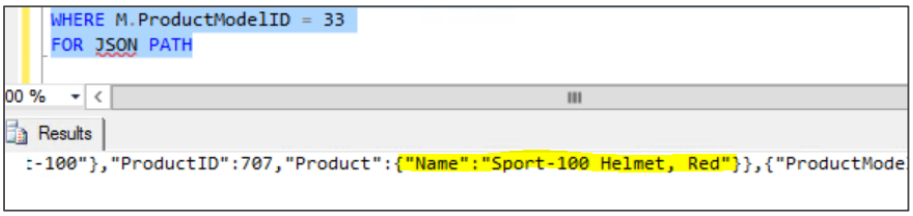
Note that the dot notation for column names reflects the hierarchy of the JSON dataset. Also note that the column names in JSON are case sensitive.

Handling NULL values in JSON data

Note that JSON by default does not display attribute values where those values are NULL. Some of the Size attributes have NULL values and those attribute value pairs are not displayed in the above result set. We can force the display of NULL values with the INCLUDE_NULL_VALUES clause.

1. First look at a small JSON result set where there are NULLs in the underlying data field for Size. Notice that the Size attribute is not displayed. We'll use a simplified version of the query from above.

```
SELECT M.ProductModelID, M.Name AS [ProductModel.Name],
ProductID, P.Name AS [Product.Name], Size
FROM Production.Product P
INNER JOIN Production.ProductModel M ON P.ProductModelID
= M.ProductModelID
WHERE M.ProductModelID = 33
FOR JSON PATH
```



2. Now add INCLUDE_NULL_VALUES to the FOR JSON PATH after a comma. Note that the Size attribute is returned.

```
WHERE M.ProductModelID = 33
FOR JSON PATH, INCLUDE_NULL_VALUES
```

100 %

Results

```
:-100"},"ProductID":707,"Product":{"Name":"Sport-100 Helmet, Red"},"Size":null},{"Pro
```

Alternate JSON structure with nested queries

An alternate method for returning hierarchical result sets as JSON data is to use nested FOR JSON subqueries. This syntax is shown below. Run the query and compare the results with those above.

```
SELECT M.ProductModelID, M.Name AS [ProductModel.Name],
      (SELECT ProductID, P.Name AS [Product.Name], Size
       FROM Production.Product P
       WHERE P.ProductModelID = M.ProductModelID
       FOR JSON PATH) AS P
FROM Production.ProductModel M
WHERE M.ProductModelID = 33
FOR JSON PATH
```

Note: This alternate structure works with both FOR JSON PATH and FOR JSON AUTO.

Export JSON data from SQL to Azure DocDB

In this exercise, we will create a new DocumentDB database on Azure and move data from SQL Server into it, using the FOR JSON construct. DocumentDB is a fully-managed document database-as-a-service with rich query and indexing capabilities over a schema-free JSON data model. It offers configurable and reliable performance, native JavaScript transactional processing, and is built for the cloud with elastic scale.

Azure account requirements

To conduct this part of the hands on lab you will be making use of the the **Azure Preview portal** at https://portal.azure.com/.

To perform this lab, you will require a Microsoft Azure account.

If you do not have an Azure account, you can request a free trial version by going to http://azure.microsoft.com/en-us/pricing/free-trial/.

Within the one-month trial version, you can perform other SQL Server 2016 hands on labs along with other tutorials available on Azure.

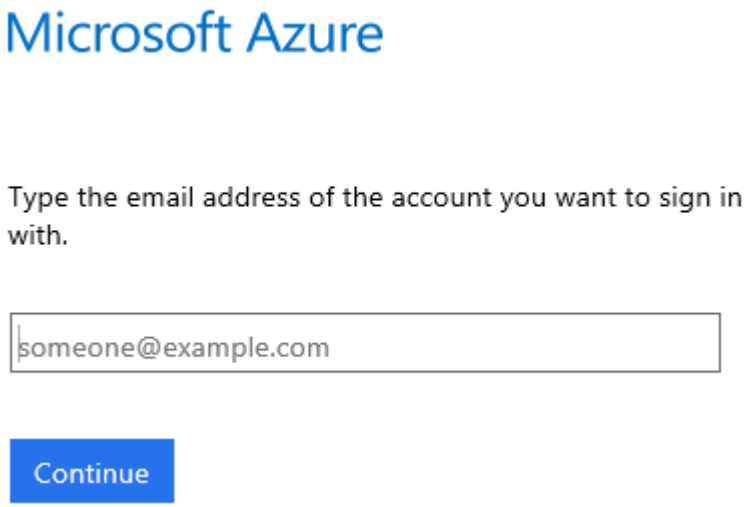
Note, to sign up for a free trial, you will need a mobile device that can receive text messages and a valid credit card.

Be sure to follow the ***Roll back Azure changes*** section at the end of this exercise after creating the Azure database so that you can make the most use of your $200 free Azure credit.

Sign in to the Azure preview portal

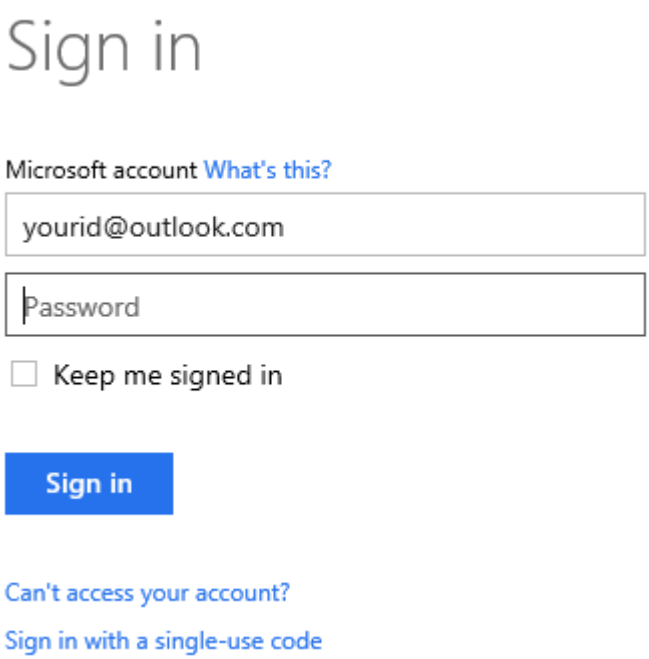
1. Click on the IE toolbar button at the bottom of the desktop to launch IE with the Azure Management Portal.

2. Enter in your Microsoft ID that is associated with your Azure subscription and click **Continue**.

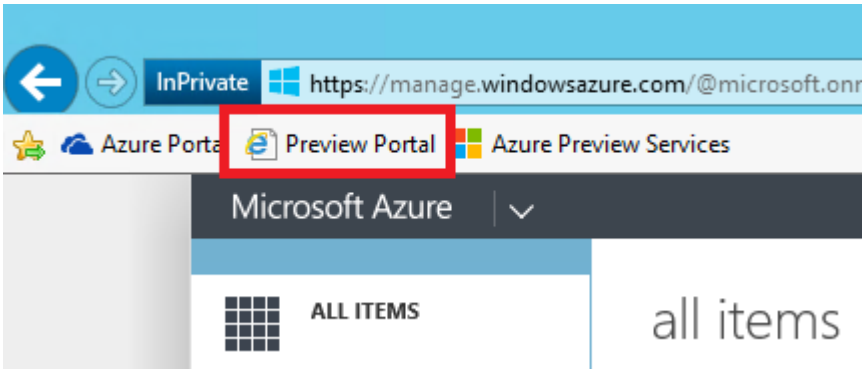


This process validates your Microsoft ID to verify that it is associated with an Azure subscription.

3. Enter in your password for the Sign in page and click **Sign in** to continue.

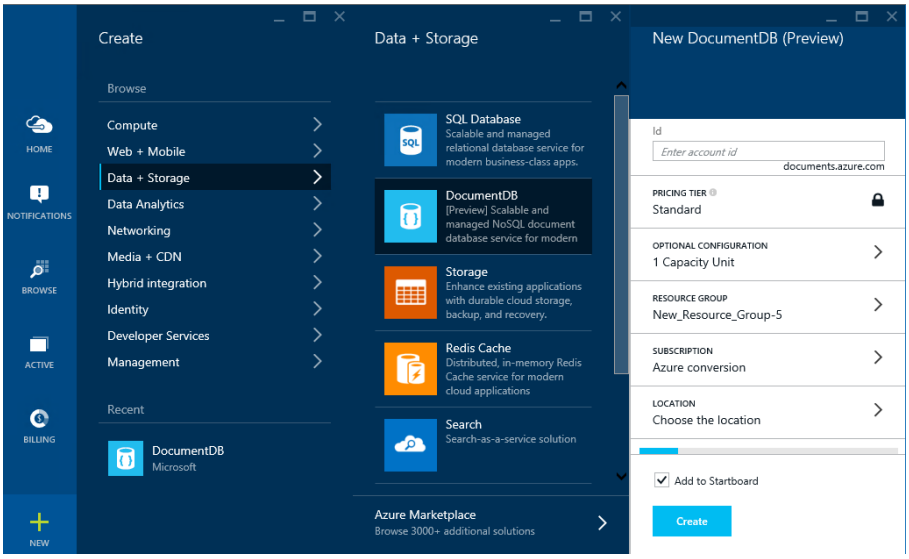


4. You should now be logged into the Azure Management portal. In order to create a DocumentDB database, you need to use the new Azure portal at https://portal.azure.com. Click on the **Preview Portal** in the IE favorites bar to go to the new portal.



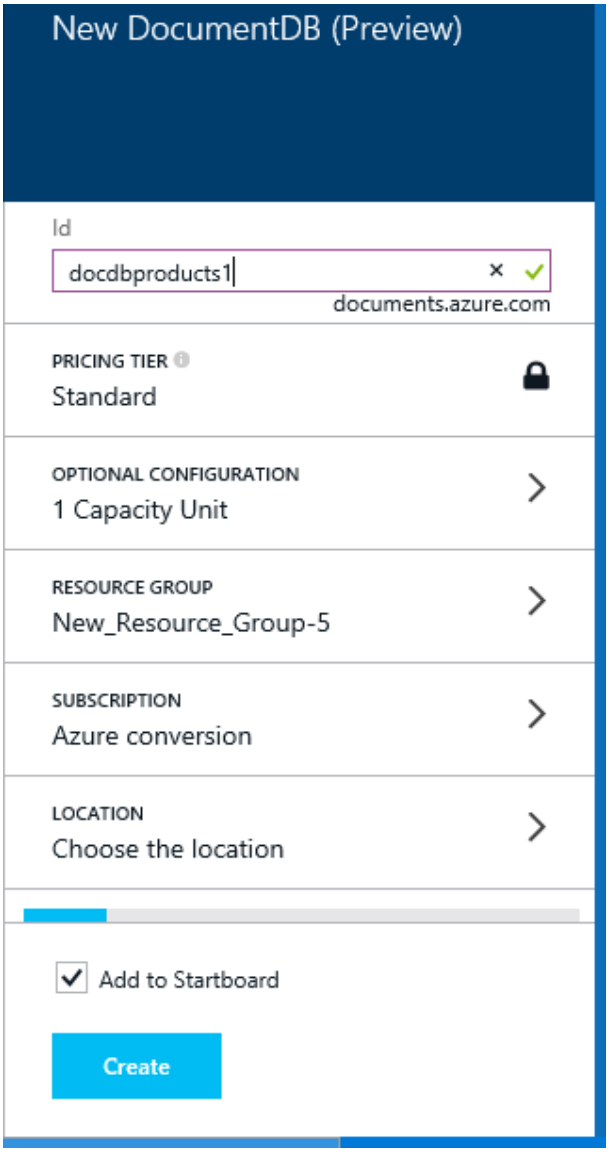
Create a DocumentDB account

1. Within the portal, click on the **New** button at the lower left portion of the page and then click Data + Storage and click on **DocumentDB**.

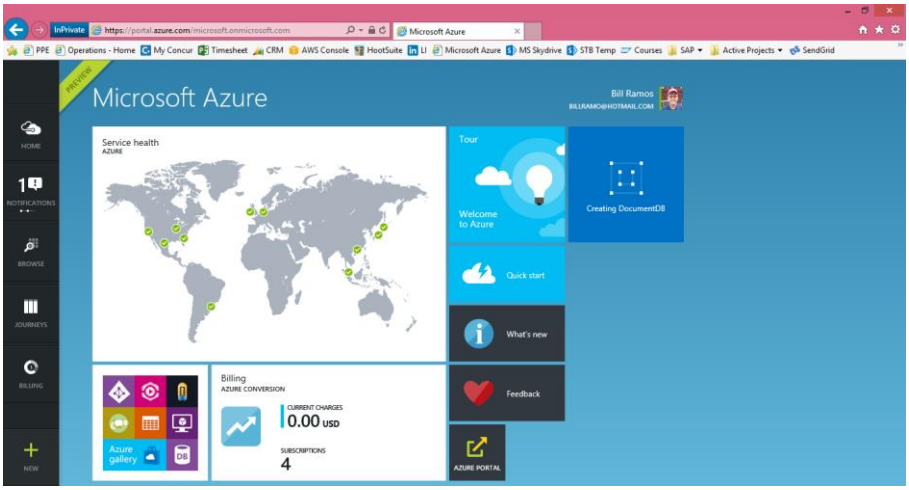


2. The Azure portal then displays the New DocumentDB (Preview) blade to enter in the database specifics. For the **ID**, enter in the name of your database such as **docdbproducts1**. The green smile icon will indicate that the **ID** is valid and not a duplicate of an existing name. You may need to change the name if yours is not unique.

   Note: The Id may contain only lowercase letters, numbers, and the '-' character, and must be between 3 and 50 characters.

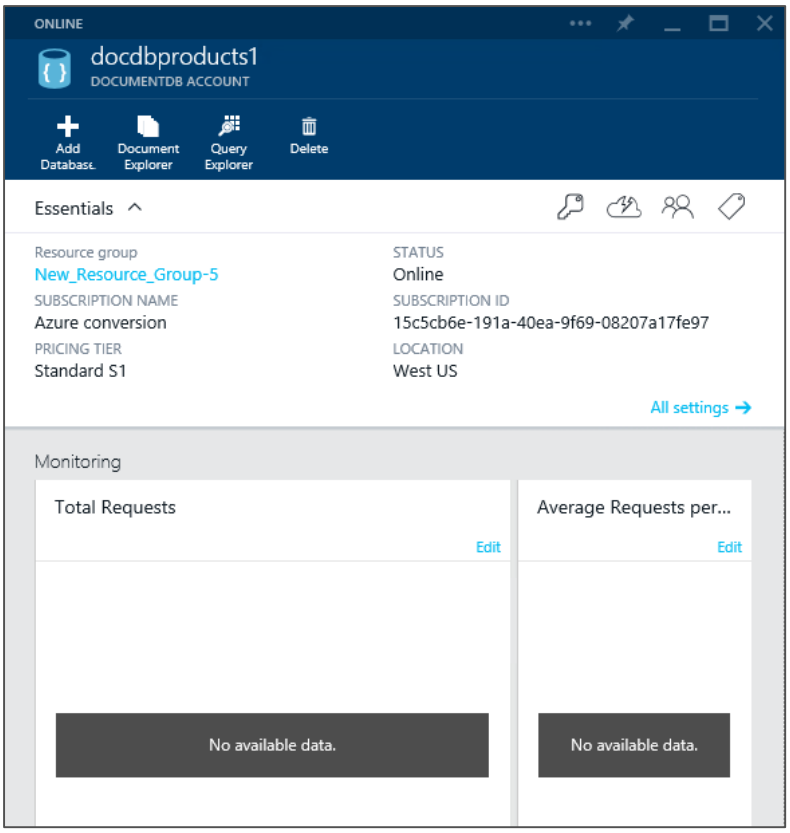


3. Change the **LOCATION** to **West US** and leave the other items with the provided defaults. Click **Create** to create the database.



The provisioning process will begin creating your database account. When the provisioning process is complete – this can take up to 10 minutes – you should see a notification appear in the notifications area of the portal and the tile on your start screen will change to show the completed action.

4. Once provisioning is complete, clicking the DocumentDB tile from the start screen will bring up the main blade for this newly created DocumentDB account, if it doesn't open automatically.



Export relational data from SQL Server to a JSON file

1. Back in SQL Server, right click on the query editor and set the Results To option to “Results to File.” Also click on Query Options and verify that there is no check mark on “Include column headers in the result set.”

2. Execute the FOR JSON PATH query from the previous part of the lab, that begins with SELECT TOP 7. Add a root element by placing ROOT ('ProductModel') in the FOR JSON PATH clause after a comma.

   Note: The JSON import will fail without a root element.

```
SELECT TOP 7 M.ProductModelID, M.Name AS
[ProductModel.Name],
ProductID, P.Name AS [Product.Name], ProductNumber,
MakeFlag,
FinishedGoodsFlag, Color, Size, SafetyStockLevel,
ReorderPoint, SellStartDate
FROM Production.Product P
INNER JOIN Production.ProductModel M ON
P.ProductModelID = M.ProductModelID
FOR JSON PATH, ROOT ('ProductModel')
```

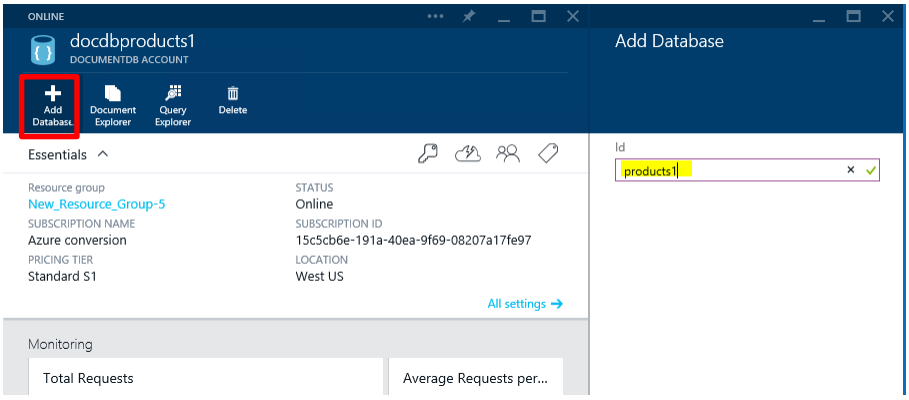
3. In the Save Results dialog window, change Save as type to All files and name the file products.json. Note the location where you saved the file so you can find it later.

4. We need to clean up one thing on the JSON file. Open it with either Visual Studio or Notepad. Highlight everything after the final curly bracket and delete all content and carriage returns after the last curly bracket, including the “(7 rows(s) affected).” Save your changes and close the file.

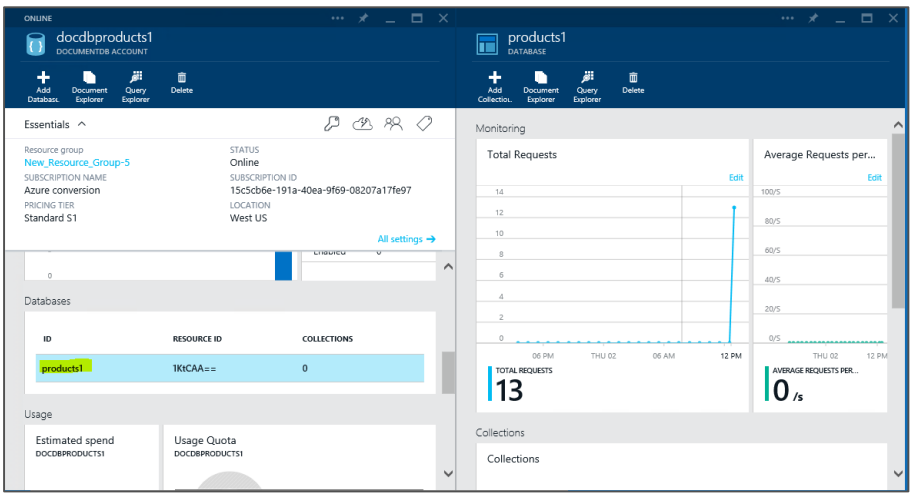
   Note: The JSON import will fail if these extra carriage returns and extraneous text is left in the file.

Import the JSON data into DocumentDB

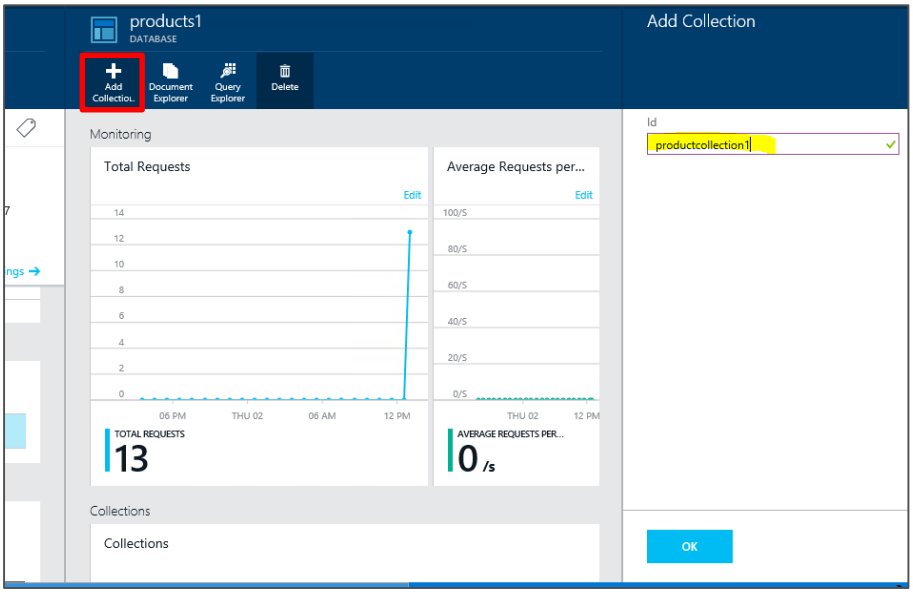
1. Back on the Azure Portal on the DocumentDB you just created, click Add Database. Type in a unique name. Use lower case letters and numbers.



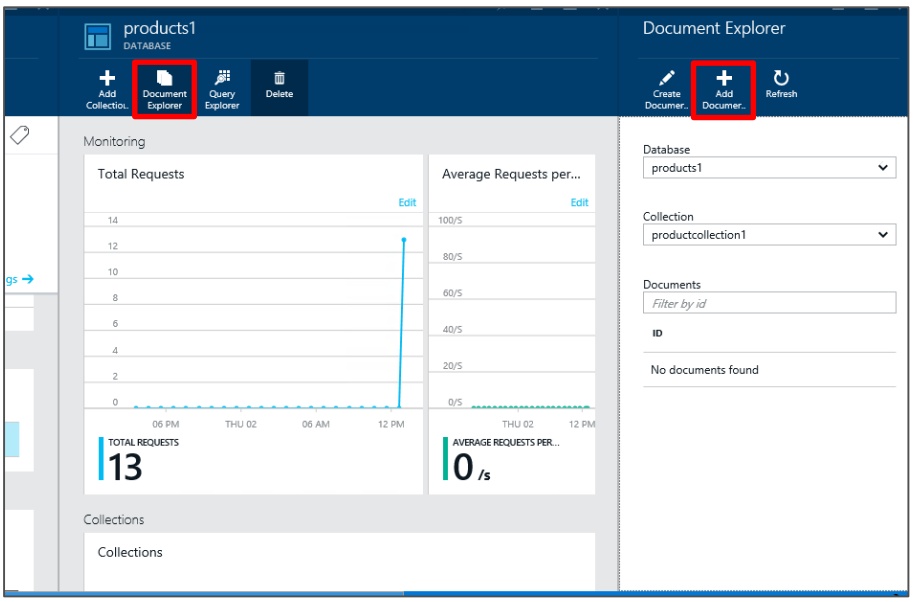
2. On the DocumentDB account tile (the one on the left) scroll down below the monitoring section until you come to the “Databases” section. Click on the database you just created.



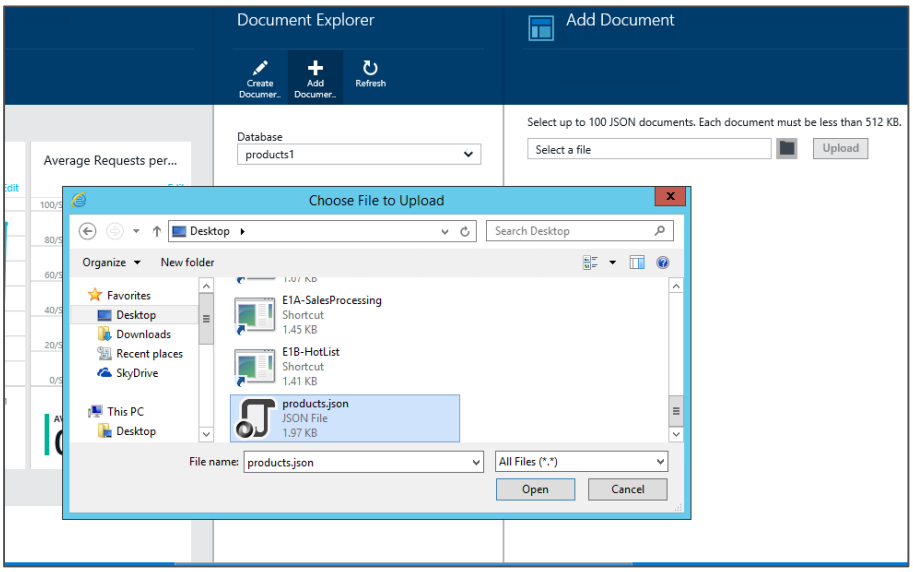
3. On the top of the right tile, you should now see an “Add Collection” button. Click that and type in an Id using lower case letters or numbers.

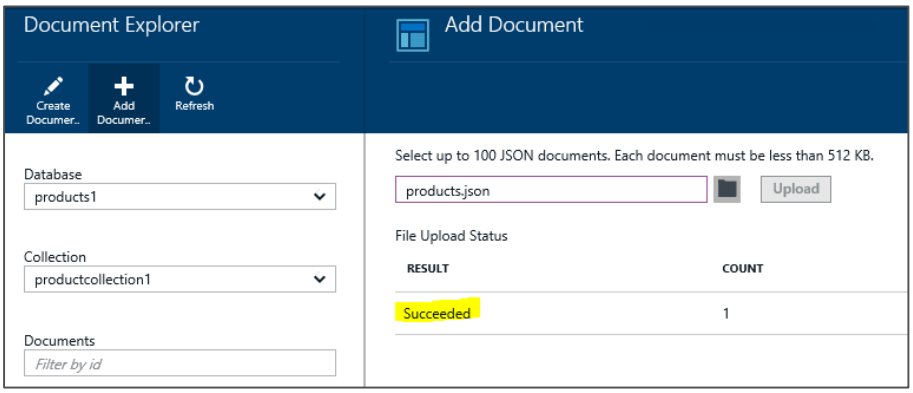


4. Click on the Document Explorer in the Database tile. Select Add Document.



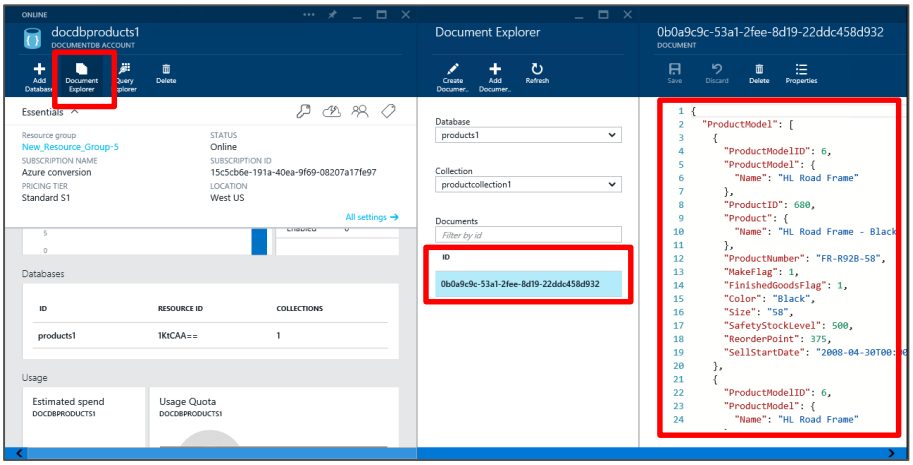
5. Navigate to the products.json file you created and click open. Then click Upload.





6. You can close all of the tiles except for the DocumentDB Account tile and select the Document Explorer. Then click

the document's ID (it is a GUID) and you can view the JSON data in the Document tile.

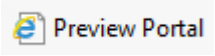


7. You just successfully created JSON data from relational data in SQL Server and uploaded it to Azure DocumentDB.

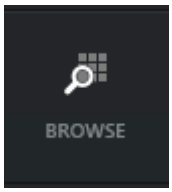
Roll back Azure changes

Let's clean up the assets we have used during this hands on lab. Here are the items you should be delete from your subscription:

Delete the Azure Website

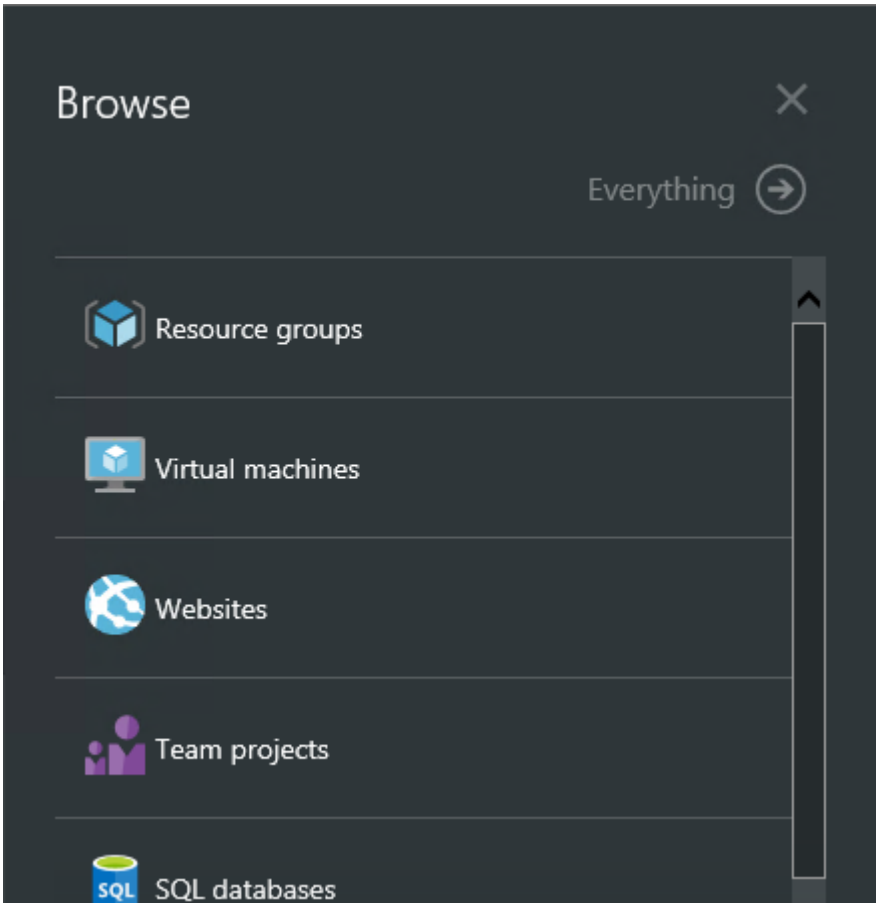
1. Go to the Azure Preview Portal by clicking on the **Preview Portal** link 

on the IE favorites bar.

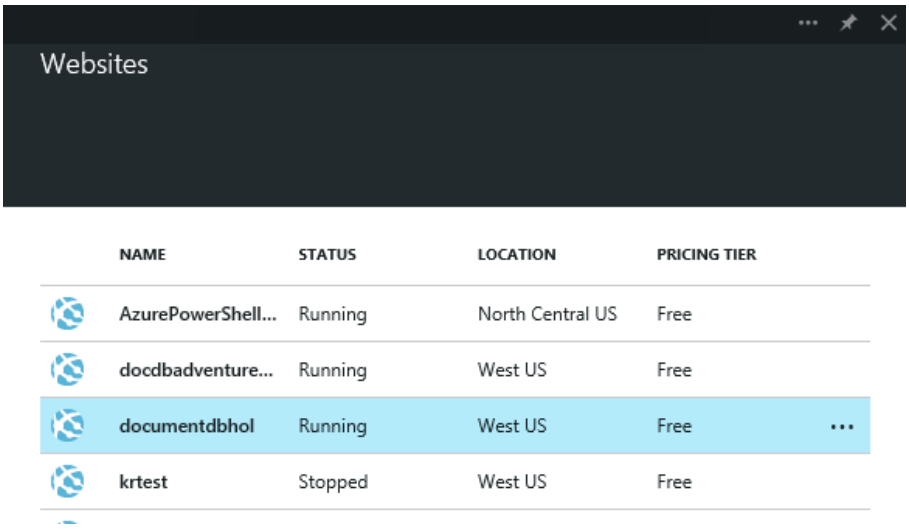
2. Click **BROWSE** 

on the left pane.

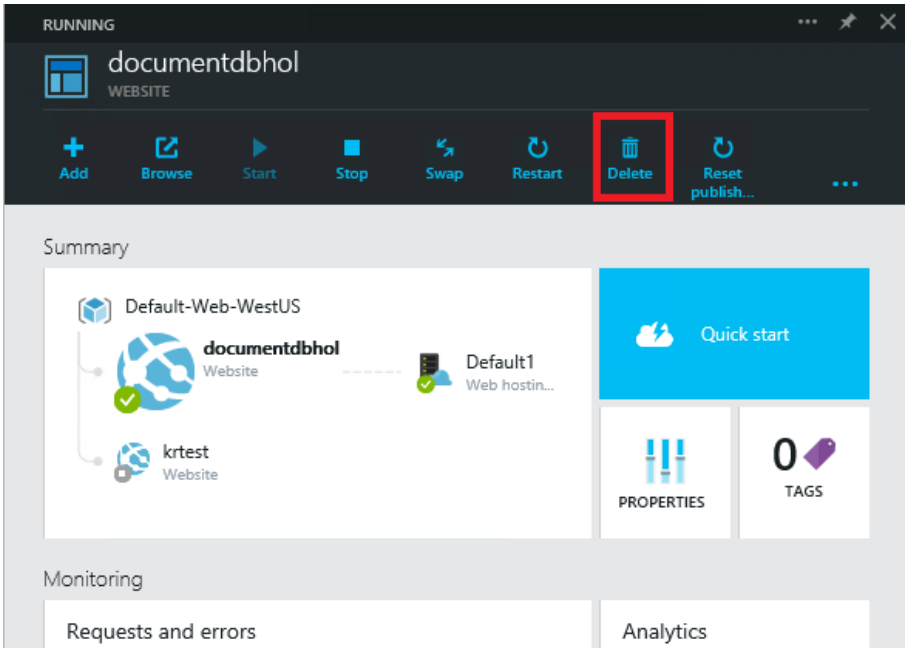
3. Click **Websites** in the Browse control.



4. Click in the Website name you created in the **Websites** blade.



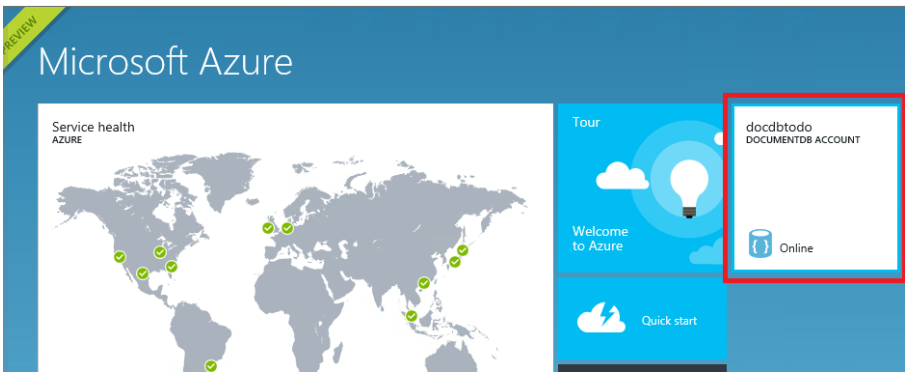
5. Click on the **Delete** button within the properties blade for your Website.



6. Click **Yes** to confirm.

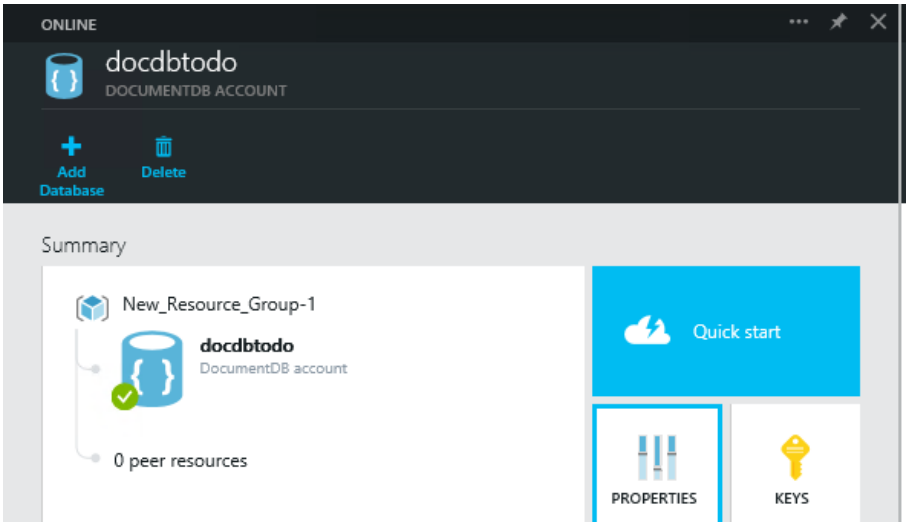
Delete DocumentDB

1. Scroll the preview back to the main Portal blade.



2. Click on your DocumentDB tile as shown above.

3. Click on the Delete icon displayed in the DocumentDB database properties blade.



4. Click **Yes** to confirm.

   You can now close the lab environment.

Terms of use

By using this Hands-on Lab, you agree to the following terms:

The technology/functionality described in this Hands-on Lab is provided by Microsoft Corporation in a “sandbox” testing environment for purposes of obtaining your feedback and to provide you with a learning experience. You may only use the Hands-on Lab to evaluate such technology features and functionality and provide feedback to Microsoft. You may not use it for any other purpose. You may not modify, copy, distribute, transmit, display, perform, reproduce, publish, license, create derivative works from, transfer, or sell this Hands-on Lab or any portion thereof.

COPYING OR REPRODUCTION OF THE HANDS-ON LAB (OR ANY PORTION OF IT) TO ANY OTHER SERVER OR LOCATION FOR FURTHER REPRODUCTION OR REDISTRIBUTION IS EXPRESSLY PROHIBITED.

THIS HANDS-ONLAB PROVIDES CERTAIN SOFTWARE TECHNOLOGY/PRODUCT FEATURES AND FUNCTIONALITY, INCLUDING POTENTIAL NEW FEATURES AND CONCEPTS, IN A SIMULATED ENVIRONMENT WITHOUT COMPLEX SET-UP OR INSTALLATION FOR THE PURPOSE DESCRIBED ABOVE. THE TECHNOLOGY/CONCEPTS REPRESENTED IN THIS HANDS-ON LAB MAY NOT REPRESENT FULL FEATURE FUNCTIONALITY AND MAY NOT WORK THE WAY A FINAL VERSION MAY WORK. WE ALSO MAY NOT RELEASE A FINAL VERSION OF

SUCH FEATURES OR CONCEPTS. YOUR EXPERIENCE WITH USING SUCH FEATURES AND FUNCITONALITY IN A PHYSICAL ENVIRONMENT MAY ALSO BE DIFFERENT.

FEEDBACK. If you give feedback about the technology features, functionality and/or concepts described in this Hands-on Lab to Microsoft, you give to Microsoft, without charge, the right to use, share and commercialize your feedback in any way and for any purpose. You also give to third parties, without charge, any patent rights needed for their products, technologies and services to use or interface with any specific parts of a Microsoft software or service that includes the feedback. You will not give feedback that is subject to a license that requires Microsoft to license its software or documentation to third parties because we include your feedback in them. These rights survive this agreement.

MICROSOFT CORPORATION HEREBY DISCLAIMS ALL WARRANTIES AND CONDITIONS WITH REGARD TO THE HANDS-ON LAB , INCLUDING ALL WARRANTIES AND CONDITIONS OF MERCHANTABILITY, WHETHER EXPRESS, IMPLIED OR STATUTORY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. MICROSOFT DOES NOT MAKE ANY ASSURANCES OR REPRESENTATIONS WITH REGARD TO THE ACCURACY OF THE RESULTS, OUTPUT THAT DERIVES FROM USE OF THE VIRTUAL LAB, OR SUITABILITY OF THE INFORMATION CONTAINED IN THE VIRTUAL LAB FOR ANY PURPOSE.

DISCLAIMER

This lab contains only a portion of new features and enhancements in Microsoft SQL Server 2016 CTP2. Some of the features might change in future releases of the product. In this lab, you will learn about some, but not all, new features.