# Run a Hadoop job using DocumentDB and HDInsight

This tutorial shows you how to run [Apache Hive](http://hive.apache.org/), [Apache Pig](http://pig.apache.org/), and [Apache Hadoop](http://hadoop.apache.org/) MapReduce jobs on Azure HDInsight with DocumentDB’s Hadoop connector. DocumentDB’s Hadoop connector allows DocumentDB to act as both a source and sink for Hive, Pig, and MapReduce jobs. This tutorial will use DocumentDB as both the data source and destination for Hadoop jobs.

After completing this tutorial, you’ll be able to answer the following questions:

* How do I load data from DocumentDB using a Hive, Pig, or MapReduce job?
* How do I store data in DocumentDB using a Hive, Pig, or MapReduce job?

We recommend getting started by watching the following video, where we run through a Hive job using DocumentDB and HDInsight.

[AZURE.VIDEO use-azure-documentdb-hadoop-connector-with-azure-hdinsight]

Then, return to this article, where you’ll receive the full details on how you can run analytics jobs on your DocumentDB data.

[AZURE.TIP] This tutorial assumes that you have prior experience using Apache Hadoop, Hive, and/or Pig. If you are new to Apache Hadoop, Hive, and Pig, we recommend visiting the [Apache Hadoop documentation](http://hadoop.apache.org/docs/current/). This tutorial also assumes that you have prior experience with DocumentDB and have a DocumentDB account. If you are new to DocumentDB or you do not have a DocumentDB account, please check out our [Getting Started](/documentation/articles/documentdb-get-started) page.

Don’t have time to complete the tutorial and just want to get the full sample PowerShell scripts for Hive, Pig, and MapReduce? Not a problem, get them [here](http://portalcontent.blob.core.windows.net/samples/documentdb-hdinsight-samples.zip). The download also contains the hql, pig, and java files for these samples.

## Newest Version

<tr><th>Hadoop Connector Version</th>  
 <td>1.2.0</td></tr>  
<tr><th>Script Uri</th>  
 <td>https://portalcontent.blob.core.windows.net/scriptaction/documentdb-hadoop-installer-v04.ps1</td></tr>  
<tr><th>Date Modified</th>  
 <td>04/26/2016</td></tr>  
<tr><th>Supported HDInsight Versions</th>  
 <td>3.1, 3.2</td></tr>  
<tr><th>Change Log</th>  
 <td>Updated DocumentDB Java SDK to 1.6.0</br>  
 Added support for partitioned collections as both a source and sink</br>  
 </td></tr>

## Prerequisites

Before following the instructions in this tutorial, ensure that you have the following:

* A DocumentDB account, a database, and a collection with documents inside. For more information, see [Getting Started with DocumentDB](/documentation/articles/documentdb-get-started). Import sample data into your DocumentDB account with the [DocumentDB import tool](/documentation/articles/documentdb-import-data).
* Throughput. Reads and writes from HDInsight will be counted towards your allotted request units for your collections. For more information, see [Provisioned throughput, request units, and database operations](/documentation/articles/documentdb-manage#ProvThroughput).
* Capacity for an additional stored procedure within each output collection. The stored procedures are used for transferring resulting documents. For more information, see [Collections and provisioned throughput](/documentation/articles/documentdb-manage#IndexOverhead).
* Capacity for the resulting documents from the Hive, Pig, or MapReduce jobs. For more information, see [Manage DocumentDB capacity and performance](/documentation/articles/documentdb-manage#Collections).
* [*Optional*] Capacity for an additional collection. For more information, see [Provisioned document storage and index overhead](/documentation/articles/documentdb-manage#IndexOverhead).

[AZURE.WARNING] In order to avoid the creation of a new collection during any of the jobs, you can either print the results to stdout, save the output to your WASB container, or specify an already existing collection. In the case of specifying an existing collection, new documents will be created inside the collection and already existing documents will only be affected if there is a conflict in *ids*. **The connector will automatically overwrite existing documents with id conflicts**. You can turn off this feature by setting the upsert option to false. If upsert is false and a conflict occurs, the Hadoop job will fail; reporting an id conflict error.

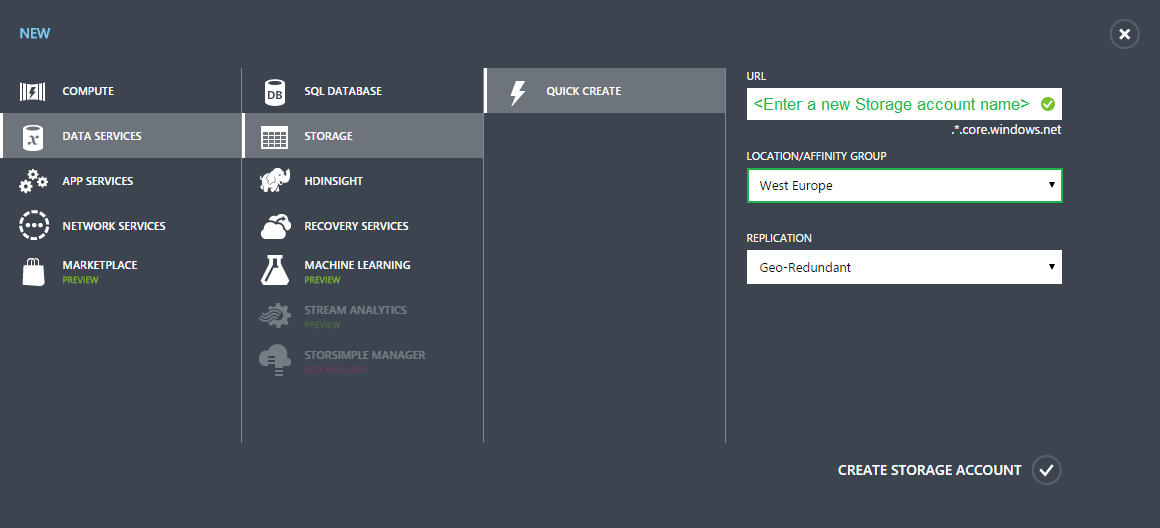
## Step 1: Create an Azure Storage account

[AZURE.IMPORTANT] If you **already** have an Azure Storage Account and you are willing to create a new blob container within that Account, you can skip to [Step 2: Create a customized HDInsight cluster](#ProvisionHDInsight).

Azure HDInsight uses Azure Blob Storage for storing data. It is called *WASB* or *Azure Storage - Blob*. WASB is Microsoft’s implementation of HDFS on Azure Blob storage. For more information see [Use Azure Blob storage with HDInsight](/documentation/articles/hdinsight-hadoop-use-blob-storage).

When you provision an HDInsight cluster, you specify an Azure Storage account. A specific Blob storage container from that account is designated as the default file system, just like in HDFS. The HDInsight cluster is by default provisioned in the same data center as the storage account you specify.

**To create an Azure Storage account**

1. Sign into the [Azure Classic Portal](https://manage.windowsazure.com/).
2. Click **+ NEW** on the lower left corner, point to **DATA SERVICES**, point to **STORAGE**, and then click **QUICK CREATE**. 
3. Enter the **URL**, select the **LOCATION** and **REPLICATION** values, and then click **CREATE STORAGE ACCOUNT**. Affinity groups are not supported.

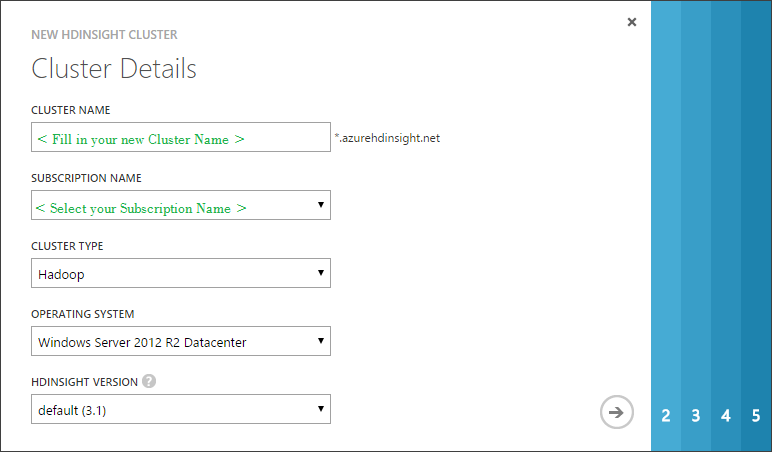
* You will see the new storage account in the storage list.
* [AZURE.IMPORTANT] For best performance, make sure your storage account, HDInsight cluster, and DocumentDB account are located in the same Azure region.

1. Wait until the **STATUS** of the new storage account is changed to **Online**.

## Step 2: Create a customized HDInsight cluster

This tutorial uses Script Action from the Azure Classic Portal to customize your HDInsight cluster. In this tutorial we will use the Azure Classic Portal to create your customized cluster. For instructions on how to use PowerShell cmdlets or the HDInsight .NET SDK, check out the [Customize HDInsight clusters using Script Action](/documentation/articles/hdinsight-provision-clusters#powershell) article.

1. Sign in to the [Azure Classic Portal](https://manage.windowsazure.com/). You may be already signed in from the previous step.
2. Click **+ NEW** on the bottom of the page, click **DATA SERVICES**, click **HDINSIGHT**, and then click **CUSTOM CREATE**.
3. On the **Cluster Details** page, type or choose the following values:

* 
* Provide Hadoop HDInsight initial cluster details
* <tr><th>Property</th><th>Value</th></tr>  
  <tr><td>Cluster name</td><td>Name the cluster.<br/>  
   DNS name must start and end with an alpha numeric character, and may contain dashes.<br/>  
   The field must be a string between 3 and 63 characters.</td></tr>  
  <tr><td>Subscription Name</td>  
   <td>If you have more than one Azure Subscription, select the subscription corresponding to the storage account from <strong>Step 1</strong>. </td></tr>  
  <tr><td>Cluster Type</td>  
   <td>For cluster type, select <strong>Hadoop</strong>.</td></tr>  
  <tr><td>Operating System</td>  
   <td>For operating system, select <strong>Windows Server 2012 R2 Datacenter</strong>.</td></tr>  
  <tr><td>HDInsight version</td>  
   <td>Choose the version. </br>Select <Strong>HDInsight version 3.1</strong>.</td></tr>  
  </table>
* Enter or select the values as shown in the table and then click the right arrow.

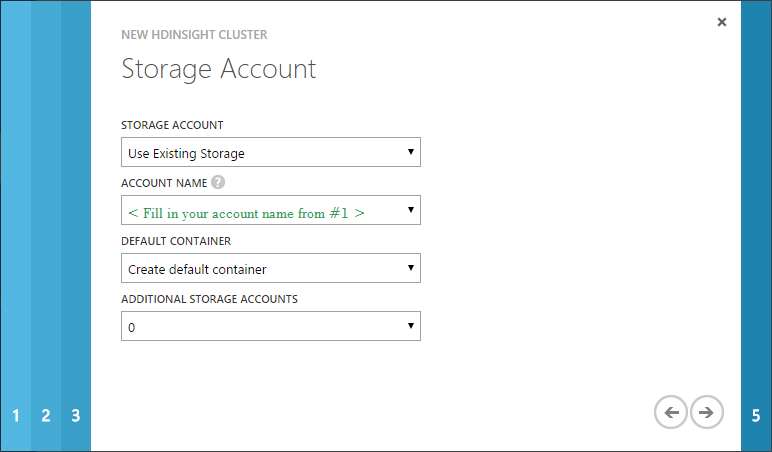
1. On the **Configure Cluster** page, enter or select the following values:

* Name
* Value
* Data nodes
* Number of data nodes you want to deploy. Take note that HDInsight’s data nodes are associated with both performance and pricing.
* Region/Virtual network
* Choose the same region as your newly created Storage account and your DocumentDB account. HDInsight requires the storage account be located in the same region. Later in the configuration, you can only choose a storage account that is in the same region as you specified here.
* Click the right arrow.

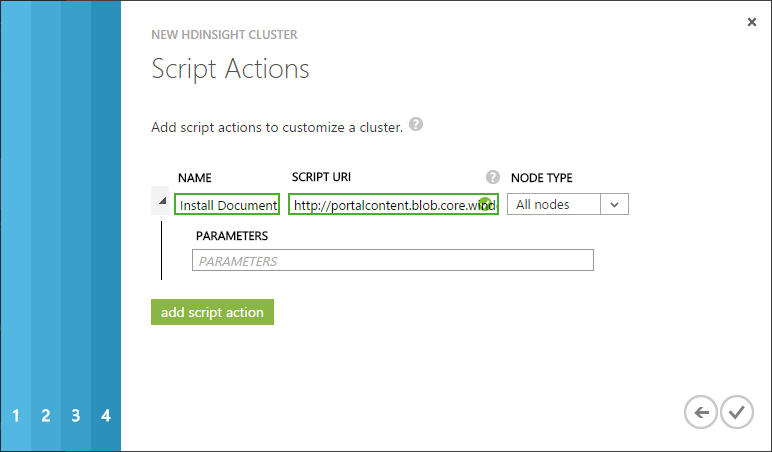
1. On the **Configure Cluster User** page, provide the following values:

* <tr><th>Property</th><th>Value</th></tr>  
  <tr><td>User name</td>  
   <td>Specify the HDInsight cluster user name.</td></tr>  
  <tr><td>Password/Confirm Password</td>  
   <td>Specify the HDInsight cluster user password.</td></tr>
* Click the right arrow.

1. On the **Storage Account** page, provide the following values:

* 
* Provide storage account for Hadoop HDInsight cluster
* <tr><th>Property</th><th>Value</th></tr>  
  <tr><td>Storage Account</td>  
   <td>Specify the Azure storage account that will be used as the default file system for the HDInsight cluster. You can choose one of the three options: Use Existing Storage, Create New Storage, or Use Storage from Another Subscription</br></br>  
   Please select <strong>Use Existing Storage</strong>.  
   </td>  
   </td></tr>  
  <tr><td>Account Name</td>  
   <td>  
   For <strong>Account name</strong>, select the account created in <strong>Step 1</strong>. The drop-down only lists the storage accounts under the same Azure subscription that are located in the same data center where you chose to provision the cluster.  
   </td></tr>  
  <tr><td>Default container</td>  
   <td>Specifies the default container on the storage account that is used as the default file system for the HDInsight cluster. If you choose <strong>Use Existing Storage</strong> for the <strong>Storage Account</strong> field, and there are no existing containers in that account, the container is created by default with the same name as the cluster name. If a container with the name of the cluster already exists, a sequence number will be appended to the container name.  
  </td></tr>  
  <tr><td>Additional Storage Accounts</td>  
   <td>HDInsight supports multiple storage accounts. There is no limit on the additional storage accounts that can be used by a cluster. However, if you create a cluster using the Azure Classic Portal, you have a limit of seven due to the UI constraints. Each additional storage account you specify adds an extra Storage Account page to the wizard where you can specify the account information.</td></tr>
* Click the right arrow.

1. On the **Script Actions** page, click **add script action** to provide details about PowerShell script that you will run to customize your cluster, as the cluster is being created. The PowerShell script will install the DocumentDB Hadoop Connector onto your HDInsight clusters during cluster creation.

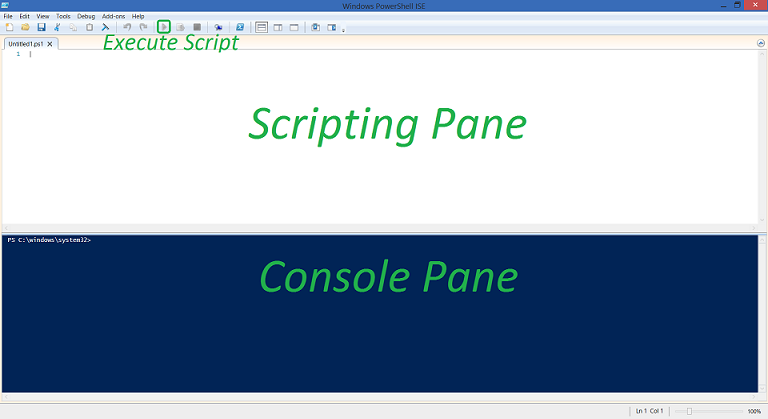
* 
* Configure Script Action to customize an HDInsight cluster
* <tr><th>Property</th><th>Value</th></tr>  
  <tr><td>Name</td>  
   <td>Specify a name for the script action.</td></tr>  
  <tr><td>Script URI</td>  
   <td>Specify the URI to the script that is invoked to customize the cluster.</br></br>  
   Please enter: </br> <strong>https://portalcontent.blob.core.windows.net/scriptaction/documentdb-hadoop-installer-v03.ps1</strong>.</td></tr>  
  <tr><td>Node Type</td>  
   <td>Specifies the nodes on which the customization script is run. You can choose <b>All Nodes</b>, <b>Head nodes only</b>, or <b>Worker nodes</b> only.</br></br>  
   Please select <strong>All Nodes</strong>.</td></tr>  
  <tr><td>Parameters</td>  
   <td>Specify the parameters, if required by the script.</br></br>  
   <strong>No Parameters needed</strong>.</td></tr>
* Click the check mark to complete the cluster creation.

## Step 3: Install and configure Azure PowerShell

1. Install Azure PowerShell. Instructions can be found [here](/documentation/articles/powershell-install-configure).

* [AZURE.NOTE] Alternatively, just for Hive queries, you can use HDInsight’s online Hive Editor. To do so, sign in to the [Azure Classic Portal](https://manage.windowsazure.com/), click **HDInsight** on the left pane to view a list of your HDInsight clusters. Click the cluster you want to run Hive queries on, and then click **Query Console**.

1. Open the Azure PowerShell Integrated Scripting Environment:
   * On a computer running Windows 8 or Windows Server 2012 or higher, you can use the built-in Search. From the Start screen, type **powershell ise** and click **Enter**.
   * On a computer running a version earlier than Windows 8 or Windows Server 2012, use the Start menu. From the Start menu, type **Command Prompt** in the search box, then in the list of results, click **Command Prompt**. In the Command Prompt, type **powershell\_ise** and click **Enter**.
2. Add your Azure Account.
   1. In the Console Pane, type **Add-AzureAccount** and click **Enter**.
   2. Type in the email address associated with your Azure subscription and click **Continue**.
   3. Type in the password for your Azure subscription.
   4. Click **Sign in**.
3. The following diagram identifies the important parts of your Azure PowerShell Scripting Environment.

* 
* Diagram for Azure PowerShell

## Step 4: Run a Hive job using DocumentDB and HDInsight

[AZURE.IMPORTANT] All variables indicated by < > must be filled in using your configuration settings.

1. Set the following variables in your PowerShell Script pane.

* # Provide Azure subscription name, the Azure Storage account and container that is used for the default HDInsight file system.  
  $subscriptionName = "<SubscriptionName>"  
  $storageAccountName = "<AzureStorageAccountName>"  
  $containerName = "<AzureStorageContainerName>"  
    
  # Provide the HDInsight cluster name where you want to run the Hive job.  
  $clusterName = "<HDInsightClusterName>"
* Let’s begin constructing your query string. We’ll write a Hive query that takes all documents’ system generated timestamps (\_ts) and unique ids (\_rid) from a DocumentDB collection, tallies all documents by the minute, and then stores the results back into a new DocumentDB collection.
* First, let’s create a Hive table from our DocumentDB collection. Add the following code snippet to the PowerShell Script pane after the code snippet from #1. Make sure you include the optional DocumentDB.query parameter to trim our documents to just \_ts and \_rid.
* [AZURE.NOTE] **Naming DocumentDB.inputCollections was not a mistake.** Yes, we allow adding multiple collections as an input: ‘*DocumentDB.inputCollections*’ = ‘*<DocumentDB Input Collection Name 1>*,*<DocumentDB Input Collection Name 2>*’ The collection names are separated without spaces, using only a single comma.
* # Create a Hive table using data from DocumentDB. Pass DocumentDB the query to filter transferred data to \_rid and \_ts.  
  $queryStringPart1 = "drop table DocumentDB\_timestamps; " +   
   "create external table DocumentDB\_timestamps(id string, ts BIGINT) " +  
   "stored by 'com.microsoft.azure.documentdb.hive.DocumentDBStorageHandler' " +  
   "tblproperties ( " +   
   "'DocumentDB.endpoint' = '<DocumentDB Endpoint>', " +  
   "'DocumentDB.key' = '<DocumentDB Primary Key>', " +  
   "'DocumentDB.db' = '<DocumentDB Database Name>', " +  
   "'DocumentDB.inputCollections' = '<DocumentDB Input Collection Name>', " +  
   "'DocumentDB.query' = 'SELECT r.\_rid AS id, r.\_ts AS ts FROM root r' ); "

1. Next, let’s create a Hive table for the output collection. The output document properties will be the month, day, hour, minute, and the total number of occurrences.

* [AZURE.NOTE] **Yet again, naming DocumentDB.outputCollections was not a mistake.** Yes, we allow adding multiple collections as an output: ‘*DocumentDB.outputCollections*’ = ‘*<DocumentDB Output Collection Name 1>*,*<DocumentDB Output Collection Name 2>*’ The collection names are separated without spaces, using only a single comma. Documents will be distributed round-robin across multiple collections. A batch of documents will be stored in one collection, then a second batch of documents will be stored in the next collection, and so forth.
* # Create a Hive table for the output data to DocumentDB.  
  $queryStringPart2 = "drop table DocumentDB\_analytics; " +  
   "create external table DocumentDB\_analytics(Month INT, Day INT, Hour INT, Minute INT, Total INT) " +  
   "stored by 'com.microsoft.azure.documentdb.hive.DocumentDBStorageHandler' " +   
   "tblproperties ( " +   
   "'DocumentDB.endpoint' = '<DocumentDB Endpoint>', " +  
   "'DocumentDB.key' = '<DocumentDB Primary Key>', " +   
   "'DocumentDB.db' = '<DocumentDB Database Name>', " +  
   "'DocumentDB.outputCollections' = '<DocumentDB Output Collection Name>' ); "

1. Finally, let’s tally the documents by month, day, hour, and minute and insert the results back into the output Hive table.

* # GROUP BY minute, COUNT entries for each, INSERT INTO output Hive table.  
  $queryStringPart3 = "INSERT INTO table DocumentDB\_analytics " +  
   "SELECT month(from\_unixtime(ts)) as Month, day(from\_unixtime(ts)) as Day, " +  
   "hour(from\_unixtime(ts)) as Hour, minute(from\_unixtime(ts)) as Minute, " +  
   "COUNT(\*) AS Total " +  
   "FROM DocumentDB\_timestamps " +  
   "GROUP BY month(from\_unixtime(ts)), day(from\_unixtime(ts)), " +  
   "hour(from\_unixtime(ts)) , minute(from\_unixtime(ts)); "

1. Add the following script snippet to create a Hive job definition from the previous query.

* # Create a Hive job definition.  
  $queryString = $queryStringPart1 + $queryStringPart2 + $queryStringPart3  
  $hiveJobDefinition = New-AzureHDInsightHiveJobDefinition -Query $queryString
* You can also use the -File switch to specify a HiveQL script file on HDFS.

1. Add the following snippet to save the start time and submit the Hive job.

* # Save the start time and submit the job to the cluster.  
  $startTime = Get-Date  
  Select-AzureSubscription $subscriptionName  
  $hiveJob = Start-AzureHDInsightJob -Cluster $clusterName -JobDefinition $hiveJobDefinition

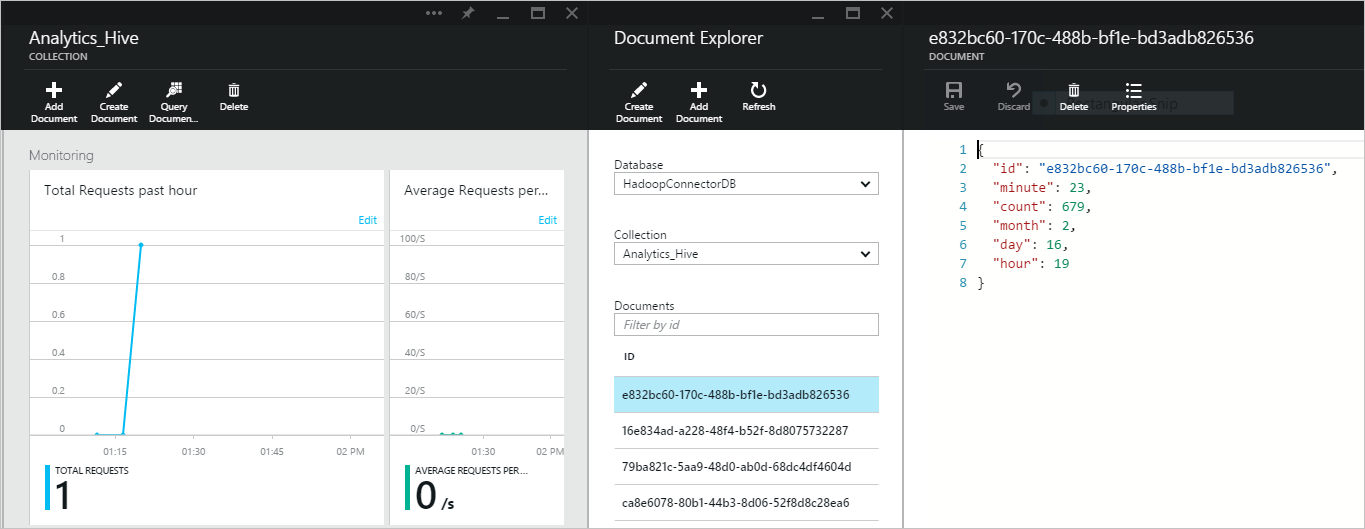
1. Add the following to wait for the Hive job to complete.

* # Wait for the Hive job to complete.  
  Wait-AzureHDInsightJob -Job $hiveJob -WaitTimeoutInSeconds 3600

1. Add the following to print the standard output and the start and end times.

* # Print the standard error, the standard output of the Hive job, and the start and end time.  
  $endTime = Get-Date  
  Get-AzureHDInsightJobOutput -Cluster $clusterName -JobId $hiveJob.JobId -StandardOutput  
  Write-Host "Start: " $startTime ", End: " $endTime -ForegroundColor Green

1. **Run** your new script! **Click** the green execute button.
2. Check the results. Sign into the [Azure Portal](https://portal.azure.com/).
   1. Click Browse on the left-side panel.
   2. Click everything at the top-right of the browse panel.
   3. Find and click DocumentDB Accounts.
   4. Next, find your DocumentDB Account, then DocumentDB Database and your DocumentDB Collection associated with the output collection specified in your Hive query.
   5. Finally, click Document Explorer underneath Developer Tools.

* You will see the results of your Hive query.
* 
* Hive query results

## Step 5: Run a Pig job using DocumentDB and HDInsight

[AZURE.IMPORTANT] All variables indicated by < > must be filled in using your configuration settings.

1. Set the following variables in your PowerShell Script pane.

* # Provide Azure subscription name.  
  $subscriptionName = "Azure Subscription Name"  
    
  # Provide HDInsight cluster name where you want to run the Pig job.  
  $clusterName = "Azure HDInsight Cluster Name"
* Let’s begin constructing your query string. We’ll write a Pig query that takes all documents’ system generated timestamps (\_ts) and unique ids (\_rid) from a DocumentDB collection, tallies all documents by the minute, and then stores the results back into a new DocumentDB collection.
* First, load documents from DocumentDB into HDInsight. Add the following code snippet to the PowerShell Script pane after the code snippet from #1. Make sure to add a DocumentDB query to the optional DocumentDB query parameter to trim our documents to just \_ts and \_rid.
* [AZURE.NOTE] Yes, we allow adding multiple collections as an input: ‘*<DocumentDB Input Collection Name 1>*,*<DocumentDB Input Collection Name 2>*’ The collection names are separated without spaces, using only a single comma.
* Documents will be distributed round-robin across multiple collections. A batch of documents will be stored in one collection, then a second batch of documents will be stored in the next collection, and so forth.
* # Load data from DocumentDB. Pass DocumentDB query to filter transferred data to \_rid and \_ts.  
  $queryStringPart1 = "DocumentDB\_timestamps = LOAD '<DocumentDB Endpoint>' USING com.microsoft.azure.documentdb.pig.DocumentDBLoader( " +  
   "'<DocumentDB Primary Key>', " +  
   "'<DocumentDB Database Name>', " +  
   "'<DocumentDB Input Collection Name>', " +  
   "'SELECT r.\_rid AS id, r.\_ts AS ts FROM root r' ); "

1. Next, let’s tally the documents by the month, day, hour, minute, and the total number of occurrences.

* # GROUP BY minute and COUNT entries for each.  
  $queryStringPart2 = "timestamp\_record = FOREACH DocumentDB\_timestamps GENERATE `$0#'id' as id:int, ToDate((long)(`$0#'ts') \* 1000) as timestamp:datetime; " +  
   "by\_minute = GROUP timestamp\_record BY (GetYear(timestamp), GetMonth(timestamp), GetDay(timestamp), GetHour(timestamp), GetMinute(timestamp)); " +  
   "by\_minute\_count = FOREACH by\_minute GENERATE FLATTEN(group) as (Year:int, Month:int, Day:int, Hour:int, Minute:int), COUNT(timestamp\_record) as Total:int; "

1. Finally, let’s store the results into our new output collection.

* [AZURE.NOTE] Yes, we allow adding multiple collections as an output: ‘<DocumentDB Output Collection Name 1>,<DocumentDB Output Collection Name 2>’ The collection names are separated without spaces, using only a single comma. Documents will be distributed round-robin across the multiple collections. A batch of documents will be stored in one collection, then a second batch of documents will be stored in the next collection, and so forth.
* # Store output data to DocumentDB.  
  $queryStringPart3 = "STORE by\_minute\_count INTO '<DocumentDB Endpoint>' " +  
   "USING com.microsoft.azure.documentdb.pig.DocumentDBStorage( " +  
   "'<DocumentDB Primary Key>', " +  
   "'<DocumentDB Database Name>', " +  
   "'<DocumentDB Output Collection Name>'); "

1. Add the following script snippet to create a Pig job definition from the previous query.

* # Create a Pig job definition.  
  $queryString = $queryStringPart1 + $queryStringPart2 + $queryStringPart3  
  $pigJobDefinition = New-AzureHDInsightPigJobDefinition -Query $queryString -StatusFolder $statusFolder
* You can also use the -File switch to specify a Pig script file on HDFS.

1. Add the following snippet to save the start time and submit the Pig job.

* # Save the start time and submit the job to the cluster.  
  $startTime = Get-Date  
  Select-AzureSubscription $subscriptionName  
  $pigJob = Start-AzureHDInsightJob -Cluster $clusterName -JobDefinition $pigJobDefinition

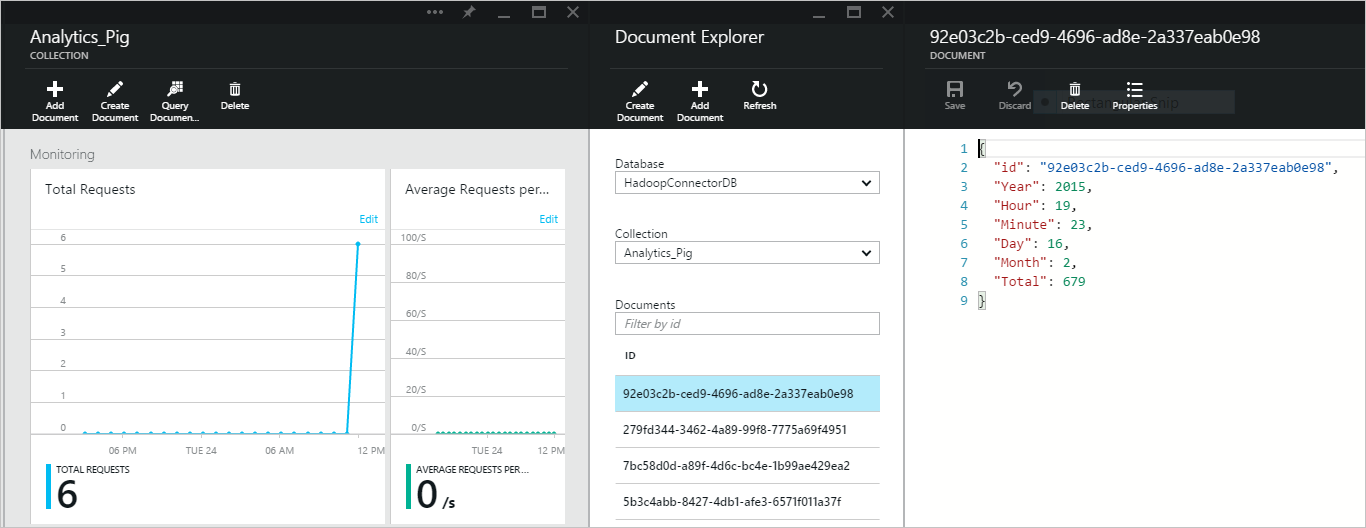
1. Add the following to wait for the Pig job to complete.

* # Wait for the Pig job to complete.  
  Wait-AzureHDInsightJob -Job $pigJob -WaitTimeoutInSeconds 3600

1. Add the following to print the standard output and the start and end times.

* # Print the standard error, the standard output of the Hive job, and the start and end time.  
  $endTime = Get-Date  
  Get-AzureHDInsightJobOutput -Cluster $clusterName -JobId $pigJob.JobId -StandardOutput  
  Write-Host "Start: " $startTime ", End: " $endTime -ForegroundColor Green

1. **Run** your new script! **Click** the green execute button.
2. Check the results. Sign into the [Azure Portal](https://portal.azure.com/).
   1. Click Browse on the left-side panel.
   2. Click everything at the top-right of the browse panel.
   3. Find and click DocumentDB Accounts.
   4. Next, find your DocumentDB Account, then DocumentDB Database and your DocumentDB Collection associated with the output collection specified in your Pig query.
   5. Finally, click Document Explorer underneath Developer Tools.

* You will see the results of your Pig query.
* 
* Pig query results

## Step 6: Run a MapReduce job using DocumentDB and HDInsight

1. Set the following variables in your PowerShell Script pane.

* $subscriptionName = "<SubscriptionName>" # Azure subscription name  
  $clusterName = "<ClusterName>" # HDInsight cluster name

1. We’ll execute a MapReduce job that tallies the number of occurrences for each Document property from your DocumentDB collection. Add this script snippet **after** the snippet above.

* # Define the MapReduce job.  
  $TallyPropertiesJobDefinition = New-AzureHDInsightMapReduceJobDefinition -JarFile "wasb:///example/jars/TallyProperties-v01.jar" -ClassName "TallyProperties" -Arguments "<DocumentDB Endpoint>","<DocumentDB Primary Key>", "<DocumentDB Database Name>","<DocumentDB Input Collection Name>","<DocumentDB Output Collection Name>","<[Optional] DocumentDB Query>"
* [AZURE.NOTE] TallyProperties-v01.jar comes with the custom installation of the DocumentDB Hadoop Connector.

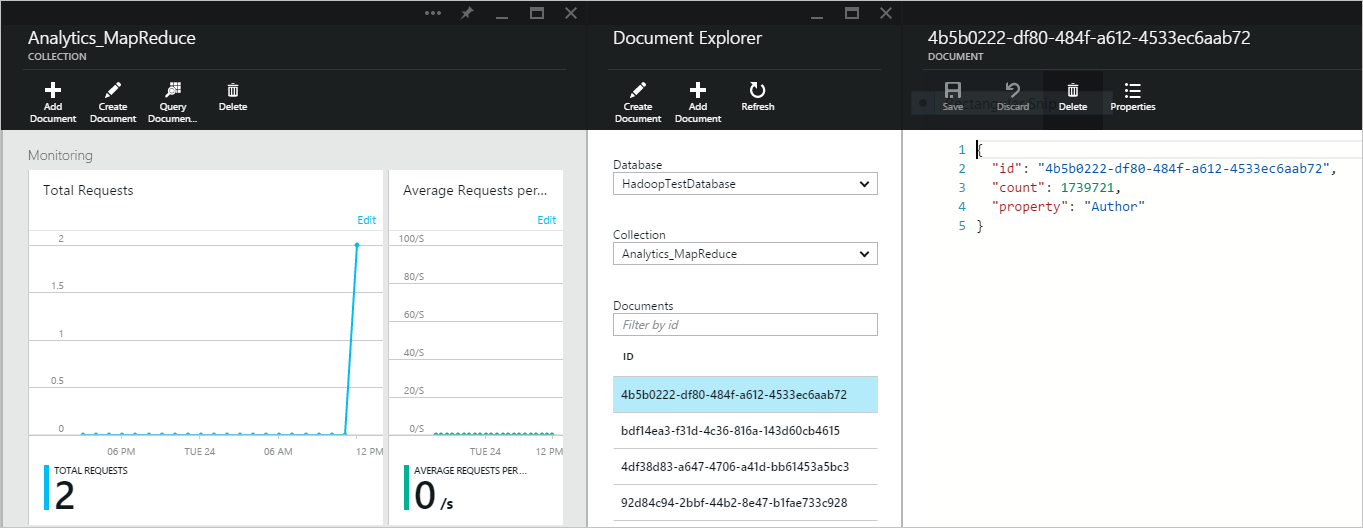
1. Add the following command to submit the MapReduce job.

* # Save the start time and submit the job.  
  $startTime = Get-Date  
  Select-AzureSubscription $subscriptionName  
  $TallyPropertiesJob = Start-AzureHDInsightJob -Cluster $clusterName -JobDefinition $TallyPropertiesJobDefinition | Wait-AzureHDInsightJob -WaitTimeoutInSeconds 3600
* In addition to the MapReduce job definition, you also provide the HDInsight cluster name where you want to run the MapReduce job, and the credentials. The Start-AzureHDInsightJob is an asynchronized call. To check the completion of the job, use the *Wait-AzureHDInsightJob* cmdlet.

1. Add the following command to check any errors with running the MapReduce job.

* # Get the job output and print the start and end time.  
  $endTime = Get-Date  
  Get-AzureHDInsightJobOutput -Cluster $clusterName -JobId $TallyPropertiesJob.JobId -StandardError  
  Write-Host "Start: " $startTime ", End: " $endTime -ForegroundColor Green

1. **Run** your new script! **Click** the green execute button.
2. Check the results. Sign into the [Azure Portal](https://portal.azure.com/).
   1. Click Browse on the left-side panel.
   2. Click everything at the top-right of the browse panel.
   3. Find and click DocumentDB Accounts.
   4. Next, find your DocumentDB Account, then DocumentDB Database and your DocumentDB Collection associated with the output collection specified in your MapReduce job.
   5. Finally, click Document Explorer underneath Developer Tools.

* You will see the results of your MapReduce job.
* 
* MapReduce query results

## Next Steps

Congratulations! You just ran your first Hive, Pig, and MapReduce jobs using Azure DocumentDB and HDInsight.

We have open sourced our Hadoop Connector. If you’re interested, you can contribute on [GitHub](https://github.com/Azure/azure-documentdb-hadoop).

To learn more, see the following articles:

* [Develop a Java application with Documentdb](/documentation/articles/documentdb-java-application)
* [Develop Java MapReduce programs for Hadoop in HDInsight](/documentation/articles/hdinsight-develop-deploy-java-mapreduce)
* [Get started using Hadoop with Hive in HDInsight to analyze mobile handset use](/documentation/articles/hdinsight-hadoop-tutorial-get-started-windows)
* [Use MapReduce with HDInsight](/documentation/articles/hdinsight-use-mapreduce)
* [Use Hive with HDInsight](/documentation/articles/hdinsight-use-hive)
* [Use Pig with HDInsight](/documentation/articles/hdinsight-use-pig)
* [Customize HDInsight clusters using Script Action](/documentation/articles/hdinsight-hadoop-customize-cluster)