# Submit Hadoop jobs in HDInsight

Learn how to use Azure PowerShell to submit MapReduce and Hive jobs, and how to use the HDInsight .NET SDK to submit MapReduce, Hadoop streaming, and Hive jobs.

## Prerequisites

Before you begin this article, you must have the following:

* **An Azure HDInsight cluster**. For instructions, see [Get started with HDInsight](/documentation/articles/hdinsight-get-started) or [Create Hadoop clusters in HDInsight](/documentation/articles/hdinsight-provision-clusters).
* **A workstation with Azure PowerShell**. See [Install and use Azure PowerShell](/documentation/articles/install-configure-powershell).

## Submit MapReduce jobs by using Azure PowerShell

Azure PowerShell is a powerful scripting environment that you can use to control and automate the deployment and management of your workloads in Azure. For more information about using Azure PowerShell with HDInsight, see [Manage HDInsight by using PowerShell](/documentation/articles/hdinsight-administer-use-powershell).

Hadoop MapReduce is a software framework for writing applications that process vast amounts of data. HDInsight clusters come with a JAR file (located at *-mapreduce-examples.jar*), which contains several MapReduce examples.

One of the examples is for counting word frequencies in source files. In this session, you will learn how to use Azure PowerShell from a workstation to run the word count sample. For more information about developing and running MapReduce jobs, see [Use MapReduce with HDInsight](/documentation/articles/hdinsight-use-mapreduce).

**To run the word count MapReduce program by using Azure PowerShell**

1. Open **Azure PowerShell**. For instructions about how to open the Azure PowerShell console window, see [Install and configure Azure PowerShell](/documentation/articles/install-configure-powershell).
2. Set the following variables by running these Azure PowerShell commands:

* $subscriptionName = "<SubscriptionName>"  
  $clusterName = "<HDInsightClusterName>"
* The subscription name is the one you used to create the HDInsight cluster. The HDInsight cluster is the one you want to use to run the MapReduce job.

1. Run the following commands to create a MapReduce job definition:

* # Define the word count MapReduce job  
  $wordCountJobDefinition = New-AzureHDInsightMapReduceJobDefinition -JarFile "wasb:///example/jars/hadoop-mapreduce-examples.jar" -ClassName "wordcount" -Arguments "wasb:///example/data/gutenberg/davinci.txt", "wasb:///example/data/WordCountOutput"
* There are two arguments. The first one is the source file name, and the second is the output file path. For more information about the wasb:// prefix, see [Use Azure Blob storage with HDInsight](/documentation/articles/hdinsight-use-blob-storage).

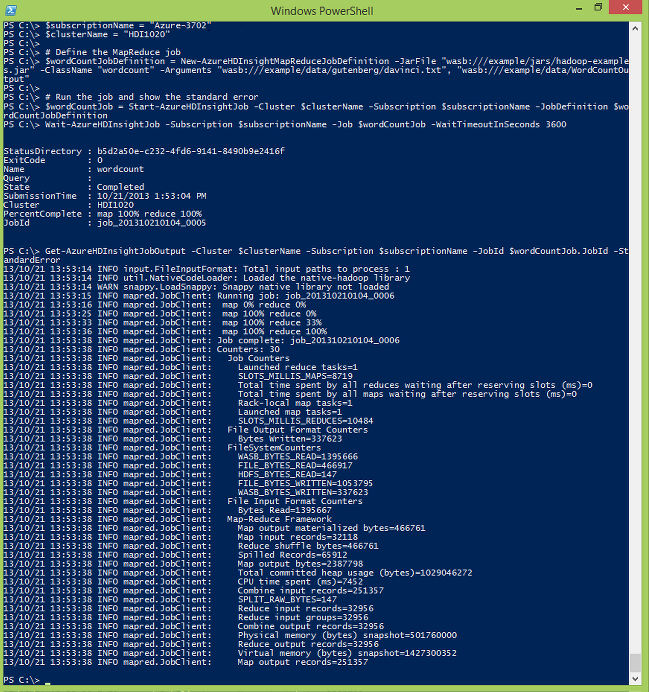
1. Run the following command to run the MapReduce job:

* # Submit the MapReduce job  
  Select-AzureSubscription $subscriptionName  
  $wordCountJob = Start-AzureHDInsightJob -Cluster $clusterName -JobDefinition $wordCountJobDefinition
* In addition to the MapReduce job definition, you also provide the HDInsight cluster name for where you want to run the MapReduce job.

1. Run the following command to check the completion of the MapReduce job:

* # Wait for the job to complete  
  Wait-AzureHDInsightJob -Job $wordCountJob -WaitTimeoutInSeconds 3600

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

* # Get the job standard error output  
  Get-AzureHDInsightJobOutput -Cluster $clusterName -JobId $wordCountJob.JobId -StandardError
* The following screenshot shows the output of a successful run. Otherwise, you will see some error messages.
* 
* HDI.GettingStarted.RunMRJob

**To retrieve the results of the MapReduce job**

1. Open **Azure PowerShell**.
2. Set the following variables by running these Azure PowerShell commands:

* $subscriptionName = "<SubscriptionName>"  
  $storageAccountName = "<StorageAccountName>"  
  $containerName = "<ContainerName>"
* The Storage account name is the Azure storage account that you specified during the HDInsight cluster creation. The storage account is used to host the blob container that is used as the default HDInsight cluster file system. The container name usually share the same name as the HDInsight cluster unless you specify a different name when you create the cluster.

1. Run the following commands to create an Azure Blob storage context object:

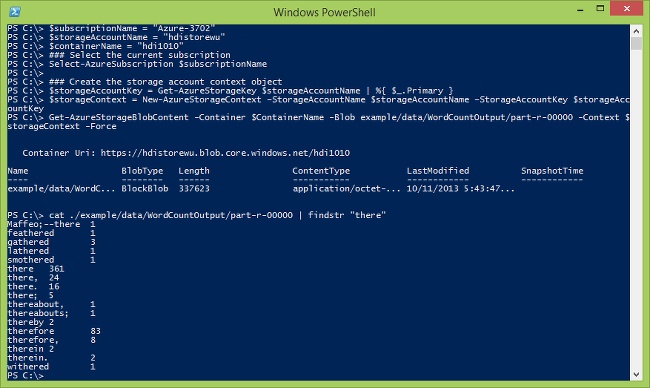
* # Create the storage account context object  
  Select-AzureSubscription $subscriptionName  
  $storageAccountKey = Get-AzureStorageKey $storageAccountName | %{ $\_.Primary }  
  $storageContext = New-AzureStorageContext -StorageAccountName $storageAccountName -StorageAccountKey $storageAccountKey
* **Select-AzureSubscription** is used to set the current subscription if you have multiple subscriptions, and the default subscription is not the one to use.

1. Run the following command to download the MapReduce job output from the blob container to the workstation:

* # Get the blob content  
  Get-AzureStorageBlobContent -Container $ContainerName -Blob example/data/WordCountOutput/part-r-00000 -Context $storageContext -Force
* The *example/data/WordCountOutput* folder is the output folder specified when you run the MapReduce job. *part-r-00000* is the default file name for MapReduce job output. The file will be download to the same folder structure in the local folder. For example, in the following screenshot, the current folder is the C: root folder. The file will be downloaded to:

\*C:\*

1. Run the following command to print the MapReduce job output file:

* cat ./example/data/WordCountOutput/part-r-00000 | findstr "there"
* 
* HDI.GettingStarted.MRJobOutput
* The MapReduce job produces a file named *part-r-00000*, and it contains the words and the counts. The script uses the **findstr** command to list all of the words that contains “there.”

[AZURE.NOTE] If you open ./example/data/WordCountOutput/part-r-00000 (a multiline output from a MapReduce job) in Notepad, you will notice the line breaks do not render correctly. This is expected.

## Submit Hive jobs by using Azure PowerShell

[Apache Hive](http://hive.apache.org/) provides a means of running MapReduce job through an SQL-like scripting language, called *HiveQL*, which can be applied to summarize, query, and analyze large volumes of data.

HDInsight clusters come with a sample Hive table called *hivesampletable*. In this session, you will use Azure PowerShell to run a Hive job to list some data from the Hive table.

**To run a Hive job by using Azure PowerShell**

1. Open **Azure PowerShell**. For instructions about how to open the Azure PowerShell console window, see [Install and configure Azure PowerShell](/documentation/articles/install-configure-powershell).
2. Set the first two variables in the following commands, and then run the commands:

* $subscriptionName = "<SubscriptionName>"  
  $clusterName = "<HDInsightClusterName>"  
  $querystring = "SELECT \* FROM hivesampletable WHERE Country='China';"
* The $querystring is the HiveQL query.

1. Run the following command to select the Azure subscription and the cluster to run the Hive job:

* Select-AzureSubscription -SubscriptionName $subscriptionName

1. Run the following commands to submit the hive job:

* Use-AzureHDInsightCluster $clusterName  
  Invoke-Hive -Query $queryString
* You can use the **-File** switch to specify a HiveQL script file in the Hadoop distributed file system (HDFS).

For more information about Hive, see [Use Hive with HDInsight](/documentation/articles/hdinsight-use-hive).

## Submit Hive jobs by using Visual Studio

See [Get started using HDInsight Hadoop Tools for Visual Studio](/documentation/articles/hdinsight-hadoop-visual-studio-tools-get-started).

## Submit Sqoop jobs by using Azure PowerShell

See [Use Sqoop with HDInsight](/documentation/articles/hdinsight-use-sqoop).

## Submit MapReduce jobs using HDInsight .NET SDK

The HDInsight .NET SDK provides .NET client libraries, which makes it easier to work with HDInsight clusters from .NET. HDInsight clusters come with a JAR file (located at *-mapreduce-examples.jar*), which contains several MapReduce examples. One of the examples is for counting word frequencies in source files. In this session, you will learn how to create a .NET application to run the word count sample. For more information about developing and running MapReduce jobs, see [Use MapReduce with HDInsight](/documentation/articles/hdinsight-use-mapreduce).

**To Submit the wordcount MapReduce job**

1. Create a C# console application in Visual Studio.
2. From the Nuget Package Manager Console, run the following command.

* Install-Package Microsoft.Azure.Management.HDInsight.Job -Pre

1. Use the following using statements in the Program.cs file:

* using System;  
  using System.Collections.Generic;  
  using Microsoft.Azure.Management.HDInsight.Job;  
  using Microsoft.Azure.Management.HDInsight.Job.Models;  
  using Hyak.Common;

1. Add the following code into the Main() function.

* var ExistingClusterName = "<HDInsightClusterName>";  
  var ExistingClusterUri = ExistingClusterName + ".azurehdinsight.cn";  
  var ExistingClusterUsername = "<HDInsightClusterHttpUsername>";  
  var ExistingClusterPassword = "<HDInsightClusterHttpUserPassword>";  
  HDInsightJobManagementClient \_hdiJobManagementClient;  
    
  List<string> arguments = new List<string> { { "wasb:///example/data/gutenberg/davinci.txt" }, { "wasb:///example/data/WordCountOutput" } };  
    
  var clusterCredentials = new BasicAuthenticationCloudCredentials { Username = ExistingClusterUsername, Password = ExistingClusterPassword };  
  \_hdiJobManagementClient = new HDInsightJobManagementClient(ExistingClusterUri, clusterCredentials);  
    
  var parameters = new MapReduceJobSubmissionParameters  
  {  
   UserName = ExistingClusterUsername,  
   JarFile = "wasb:///example/jars/hadoop-mapreduce-examples.jar",  
   JarClass = "wordcount",  
   Arguments = ConvertArgsToString(arguments)  
  };  
    
  System.Console.WriteLine("Submitting the MapReduce job to the cluster...");  
  var response = \_hdiJobManagementClient.JobManagement.SubmitMapReduceJob(parameters);  
  System.Console.WriteLine("Validating that the response is as expected...");  
  System.Console.WriteLine("Response status code is " + response.StatusCode);  
  System.Console.WriteLine("Validating the response object...");  
  System.Console.WriteLine("JobId is " + response.JobSubmissionJsonResponse.Id);  
  Console.WriteLine("Press ENTER to continue ...");  
  Console.ReadLine();

1. Add the following helper function in the the class.

* private static string ConvertArgsToString(List<string> args)  
  {  
   if (args.Count == 0)  
   {  
   return null;  
   }  
    
   return string.Join("&arg=", args.ToArray());  
  }

1. Press **F5** to run the application.

## Submit Hadoop streaming jobs using HDInsight .NET SDK

HDInsight clusters come with a word-counting Hadoop stream program, which is developed in C#. The mapper program is */example/apps/cat.exe*, and the reduce program is */example/apps/wc.exe*. In this session, you will learn how to create a .NET application to run the word-counting sample.

For the details about creating a .NET application for submitting MapReduce jobs, see [Submit MapReduce jobs using HDInsight .NET SDK](#mapreduce-sdk).

For more information about developing and deploying Hadoop streaming jobs, see [Develop C# Hadoop streaming programs for HDInsight](/documentation/articles/hdinsight-hadoop-develop-deploy-streaming-jobs).

**To Submit the wordcount MapReduce job**

1. From the Visual Studio Package Manager Console, run the following Nuget command to import the package.

* Install-Package Microsoft.Azure.Management.HDInsight.Job -Pre

1. Use the following using statements in the Program.cs file:

* using System;  
  using System.Collections.Generic;  
  using Microsoft.Azure.Management.HDInsight.Job;  
  using Microsoft.Azure.Management.HDInsight.Job.Models;  
  using Hyak.Common;

1. Add the following code into the Main() function.

* var ExistingClusterName = "<HDInsightClusterName>";  
  var ExistingClusterUri = ExistingClusterName + ".azurehdinsight.cn";  
  var ExistingClusterUsername = "<HDInsightClusterHttpUsername>";  
  var ExistingClusterPassword = "<HDInsightClusterHttpUserPassword>";  
    
  List<string> arguments = new List<string> { { "/example/apps/cat.exe" }, { "/example/apps/wc.exe" } };  
    
  HDInsightJobManagementClient \_hdiJobManagementClient;  
  var clusterCredentials = new BasicAuthenticationCloudCredentials { Username = ExistingClusterUsername, Password = ExistingClusterPassword };  
  \_hdiJobManagementClient = new HDInsightJobManagementClient(ExistingClusterUri, clusterCredentials);  
    
  var parameters = new MapReduceStreamingJobSubmissionParameters  
  {  
   UserName = ExistingClusterUsername,  
   File = ConvertArgsToString(arguments),  
   Mapper = "cat.exe",  
   Reducer = "wc.exe",  
   Input = "/example/data/gutenberg/davinci.txt",  
   Output = "/tutorials/wordcountstreaming/output"  
  };  
    
  System.Console.WriteLine("Submitting the MapReduce job to the cluster...");  
  var response = \_hdiJobManagementClient.JobManagement.SubmitMapReduceStreamingJob(parameters);  
  System.Console.WriteLine("Validating that the response is as expected...");  
  System.Console.WriteLine("Response status code is " + response.StatusCode);  
  System.Console.WriteLine("Validating the response object...");  
  System.Console.WriteLine("JobId is " + response.JobSubmissionJsonResponse.Id);  
  Console.WriteLine("Press ENTER to continue ...");  
  Console.ReadLine();

1. Add a help function to the class.

* private static string ConvertArgsToString(List<string> args)  
  {  
   if (args.Count == 0)  
   {  
   return null;  
   }  
    
   return string.Join("&arg=", args.ToArray());  
  }

1. Press **F5** to run the application.

## Submit Hive jobs by using HDInsight .NET SDK

HDInsight clusters come with a sample Hive table called *hivesampletable*. In this session, you will create a .NET application to run a Hive job to list the Hive tables that are created in an HDInsight cluster. For more information about using Hive, see [Use Hive with HDInsight](/documentation/articles/hdinsight-use-hive).

The following procedures are needed to create an HDInsight cluster by using the SDK:

* Install the HDInsight .NET SDK
* Create a console application
* Run the application

**To install the HDInsight .NET SDK** You can install the latest published build of the SDK from [NuGet](http://nuget.codeplex.com/wikipage?title=Getting%20Started). The instructions will be shown in the next procedure.

**To create a Visual Studio console application**

1. Open Visual Studio 2013 or 2015.
2. Create a new project with the following settings:

|  |  |
| --- | --- |
| * Property | * Value |
| * Template | * Templates/Visual C#/Windows/Console Application |
| * Name | * SubmitHiveJob |

1. From the **Tools** menu, click **Nuget Package Manager**, and then click **Package Manager Console**.
2. Run the following command in the console to install the packages:

* Install-Package Microsoft.Azure.Common.Authentication -pre  
  Install-Package Microsoft.Azure.Management.HDInsight -Pre  
  Install-Package Microsoft.Azure.Management.HDInsight.Job -Pre
* These commands add .NET libraries and references to them to the current Visual Studio project.

1. From Solution Explorer, double-click **Program.cs** to open it, paste the following code, and provide values for the variables:

* using System.Collections.Generic;  
  using System.Linq;  
  using Microsoft.Azure.Management.HDInsight.Job;  
  using Microsoft.Azure.Management.HDInsight.Job.Models;  
  using Hyak.Common;  
    
  namespace SubmitHiveJob  
  {  
   class Program  
   {  
   private static HDInsightJobManagementClient \_hdiJobManagementClient;  
    
   private const string ExistingClusterName = "<HDINSIGHT CLUSTER NAME>";  
   private const string ExistingClusterUri = ExistingClusterName + ".azurehdinsight.cn";  
    
   private const string ExistingClusterUsername = "<HDINSIGHT HTTP USER NAME>"; //The default name is admin.  
   private const string ExistingClusterPassword = "<HDINSIGHT HTTP USER PASSWORD>";  
    
   private static void Main(string[] args)  
   {  
    
   var clusterCredentials = new BasicAuthenticationCloudCredentials { Username = ExistingClusterUsername, Password = ExistingClusterPassword };  
   \_hdiJobManagementClient = new HDInsightJobManagementClient(ExistingClusterUri, clusterCredentials);  
    
   SubmitHiveJob();  
   }  
    
   private static void SubmitHiveJob()  
   {  
   Dictionary<string, string> defines = new Dictionary<string, string> { { "hive.execution.engine", "ravi" }, { "hive.exec.reducers.max", "1" } };  
   List<string> args = new List<string> { { "argA" }, { "argB" } };  
   var parameters = new HiveJobSubmissionParameters  
   {  
   UserName = ExistingClusterUsername,  
   Query = "SHOW TABLES",  
   Defines = ConvertDefinesToString(defines),  
   Arguments = ConvertArgsToString(args)  
   };  
    
   System.Console.WriteLine("Submitting the Hive job to the cluster...");  
   var response = \_hdiJobManagementClient.JobManagement.SubmitHiveJob(parameters);  
   System.Console.WriteLine("Validating that the response is as expected...");  
   System.Console.WriteLine("Response status code is " + response.StatusCode);  
   System.Console.WriteLine("Validating the response object...");  
   System.Console.WriteLine("JobId is " + response.JobSubmissionJsonResponse.Id);  
   System.Console.WriteLine("Press ENTER to continue ...");  
   System.Console.ReadLine();  
   }  
    
   private static string ConvertDefinesToString(Dictionary<string, string> defines)  
   {  
   if (defines.Count == 0)  
   {  
   return null;  
   }  
    
   return string.Join("&define=", defines.Select(x => x.Key + "%3D" + x.Value).ToArray());  
   }  
   private static string ConvertArgsToString(List<string> args)  
   {  
   if (args.Count == 0)  
   {  
   return null;  
   }  
    
   return string.Join("&arg=", args.ToArray());  
   }  
   }  
  }

1. Press **F5** to run the application.

## Submit jobs using the HDInsight Tools for Visual Studio

Using the HDInsight Tools for Visual Studio, you can run Hive queries and Pig scripts. See [Get started using Visual Studio Hadoop tools for HDInsight](/documentation/articles/hdinsight-hadoop-visual-studio-tools-get-started).

## Next steps

In this article, you have learned several ways to create an HDInsight cluster. To learn more, see the following articles:

* [Get started with Azure HDInsight](/documentation/articles/hdinsight-get-started)
* [Create Hadoop clusters in HDInsight](/documentation/articles/hdinsight-provision-clusters)
* [Manage HDInsight by using PowerShell](/documentation/articles/hdinsight-administer-use-powershell)
* [HDInsight Cmdlet Reference Documentation](https://msdn.microsoft.com/zh-cn/library/dn858087.aspx)
* [Use Hive with HDInsight](/documentation/articles/hdinsight-use-hive)
* [Use Pig with HDInsight](/documentation/articles/hdinsight-use-pig)