Contents

[Introduction 1](#_Toc366423729)

[Creating the Visual Studio project 1](#_Toc366423730)

[Configuring the HDFS file system 4](#_Toc366423731)

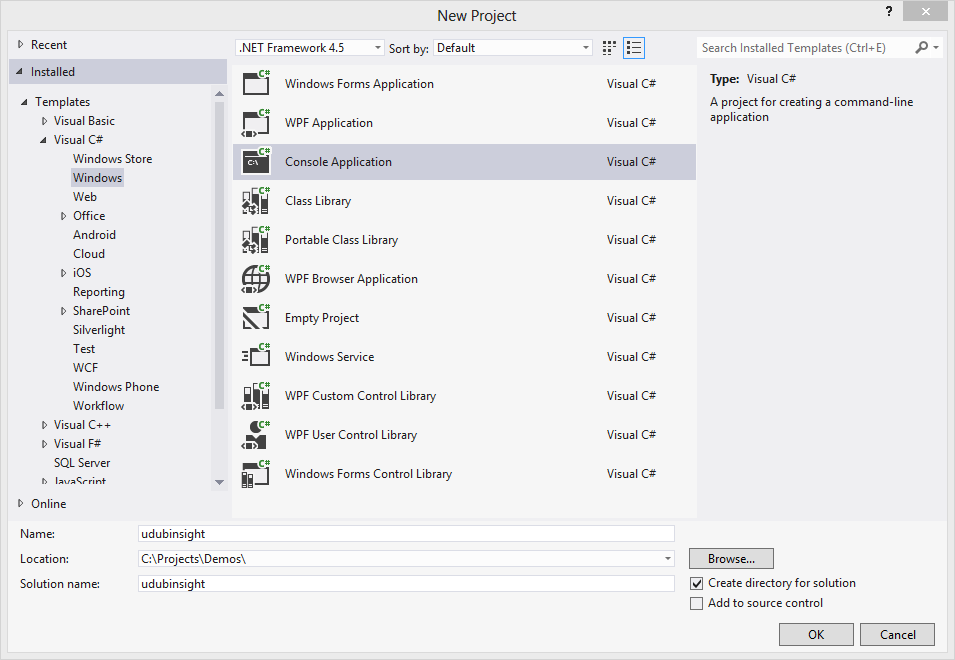
[Update the Mapper 4](#_Toc366423732)

# Introduction

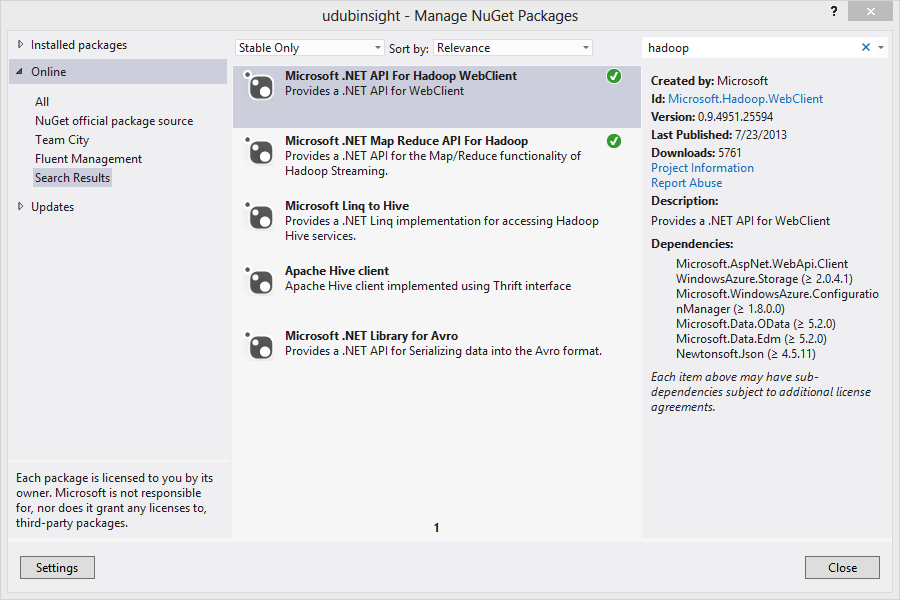
The purpose of this demo is to show how map reduce can be used to analyse a set of data and determine some aggregations of that data through querying.

# Creating the Visual Studio project

To begin we’ll create a console application which will contain the data needed for the



After creating a console application add a reference through nuget for the HDInsight SDK. Search for Microsoft.Hadoop.WebClient and Microsoft.Hadoop.MapReduce to find the packages.



Create three new files called **CountryMapper.cs** and **CountryReducer.cs** and **CountryJob.cs** .

At the top of each of the files add the following namespace declaration.

using Microsoft.Hadoop.MapReduce;

In CountryMapper.cs enter the following:

   public class CountryMapper : MapperBase

    {

        #region Overrides of MapperBase

        public override void Map(string inputLine, MapperContext context)

        {

            string[] terms = inputLine.Split('\t');

            // add a sanity check in case we have a data quality issue

            if (terms.Length != 6)

                return;

            // get the country part out

            string country = terms[3];

            context.EmitKeyValue(country, "1");

        }

        #endregion

    }

In CountryReducer.cs enter the following:

 public class CountryReducer : ReducerCombinerBase

    {

        #region Overrides of ReducerCombinerBase

        public override void Reduce(string key, IEnumerable<string> values, ReducerCombinerContext context)

        {

            context.EmitKeyValue(key, values.Count().ToString(CultureInfo.InvariantCulture));

        }

        #endregion

    }

Next we’ll add the following to the **CountryJob.cs** class.

 public class CountryJob : HadoopJob<CountryMapper, CountryReducer>

    {

        #region Overrides of HadoopJob

        public override HadoopJobConfiguration Configure(ExecutorContext context)

        {

            var configuration = new HadoopJobConfiguration

            {

                InputPath = context.Arguments[0],

                OutputFolder = context.Arguments[1]

            };

            return configuration;

        }

        #endregion

    }

Lastly add the following to the **main** method. Copy the 916kb.log file into the local directory as we’re going to debug this now. We’ll be looking at using StreamingUnit to test our assumptions before testing the cluster.

bool debug = true;

if (debug)

{

using (var reader = new StreamReader("916kb.log"))

{

string sample = reader.ReadToEnd();

var output = StreamingUnit.Execute<CountryMapper, CountryReducer>(sample.Split(new[] {'\r', '\n'}));

foreach (string result in output.ReducerResult)

{

Console.WriteLine(result);

}

Console.ReadLine();

}

}

else

{

 Environment.SetEnvironmentVariable("JAVA\_HOME", "this\_is\_a\_BUG", EnvironmentVariableTarget.Process);

                Environment.SetEnvironmentVariable("HADOOP\_HOME", "this\_is\_a\_BUG", EnvironmentVariableTarget.Process);

**var** azureCluster = **new** Uri("https://<the cluster name>.azurehdinsight.net:563");

**var** clusterUsername = "<your user name>";

**var** clusterPassword = "<your password>";

**var** hadoopUserName = "hadoop";

**var** storageAccount = "<your storage account name>.blob.core.windows.net";

**var** storageKey = "<your storage account key>";

**var** storageContainer = "<storage container name>";

**bool** createContainerIfNotExist = **true**;

**var** hadoop = Hadoop.Connect(azureCluster,clusterUsername, hadoopUserName, clusterPassword, storageAccount, storageKey,storageContainer, createContainerIfNotExist);

**var** result = hadoop.MapReduceJob.ExecuteJob<CountryJob>();

}

Let’s run streaming unit and test now. This will show us an output like the following.

|  |  |
| --- | --- |
| France | 234 |
| Germany | 3774 |
| UK | 1020 |
| USA | 9972 |

# Configuring the HDFS file system

Use the remote desktop session you created earlier and enter the following at the Hadoop Cmd prompt (a link for this can be found from the desktop).

* hadoop fs -mkdir countries

Copy the 916kb.log which can be found in the demo directory onto the Hadoop name node. Then use the location to copy this into HDFS/ASV.

* hadoop fs -copyFromLocal "C:\916kb.log" "countries/input.log"

Check to ensure that the folder has been made.

* hadoop fs -ls countries

Make the point that HDFS and ASV are two discrete filesystems.

We’re only interested in putting this into ASV.

* hadoop dfs -copyFromLocal "C:\916kb.log" asv:///countries/input.log

This will store this in a container in the name of the cluster you’ve setup.

# Run the project

Run the project you’ve created. Debug this rather than Run.

You should see the following output in the StandardError that’s returned from the MapReduceResult.

*13/09/14 17:37:35 INFO util.NativeCodeLoader: Loaded the native-hadoop library*

*13/09/14 17:37:35 WARN snappy.LoadSnappy: Snappy native library not loaded*

*13/09/14 17:37:35 INFO mapred.FileInputFormat: Total input paths to process : 1*

*13/09/14 17:37:35 INFO streaming.StreamJob: getLocalDirs(): [c:\hdfs\mapred\local]*

*13/09/14 17:37:35 INFO streaming.StreamJob: Running job: job\_201309131434\_0006*

*13/09/14 17:37:35 INFO streaming.StreamJob: To kill this job, run:*

*13/09/14 17:37:35 INFO streaming.StreamJob: C:\apps\dist\hadoop-1.1.0-SNAPSHOT/bin/hadoop job -Dmapred.job.tracker=jobtrackerhost:9010 -kill job\_201309131434\_0006*

*13/09/14 17:37:35 INFO streaming.StreamJob: Tracking URL: http://headnodehost:50030/jobdetails.jsp?jobid=job\_201309131434\_0006*

*13/09/14 17:37:36 INFO streaming.StreamJob: map 0% reduce 0%*

*13/09/14 17:37:55 INFO streaming.StreamJob: map 100% reduce 0%*

*13/09/14 17:38:07 INFO streaming.StreamJob: map 100% reduce 100%*

*13/09/14 17:38:10 INFO streaming.StreamJob: Job complete: job\_201309131434\_0006*

*13/09/14 17:38:10 INFO streaming.StreamJob: Output: asv:///countrycount*

You should see the inputs and outputs as below:



You can remote desktop onto the machine and check using the following:

* hadoop dfs -lsr asv:///countrycount

Which should list two files. We can view the output like so.

* hadoop dfs -cat asv:///countrycount/part-0000

# Update the Mapper

We’re going to open up MapperBase and replace with the following code to the Mapper method. This will filter out the purchases that we’re interested in.

const string purchaseConst = "Purchase";

string[] terms = inputLine.Split('\t');

// get the country part out

string country = terms[3];

string purchase = terms[5];

if (purchase == purchaseConst)

context.EmitKeyValue(country, "1");

Copy the contents of the directory to the name node as above. You should see results similar to this.

In the CountryJob change the output folder to countrycount2 like so:

OutputFolder = "asv:///countrycount2";

|  |  |
| --- | --- |
| Germany | 3718 |
| UK | 4 |
| USA | 42 |

Check the output of the job.

* hadoop dfs -cat asv:///countrycount2/part-0000

Copyright 2013 Microsoft Corporation. All rights reserved.

Except where otherwise noted, these materials are licensed under the terms of the Apache License, Version 2.0. You may use it according to the license as is most appropriate for your project on a case-by-case basis. The terms of this license can be found in <http://www.apache.org/licenses/LICENSE-2.0>.