Beginning Map-Reduce

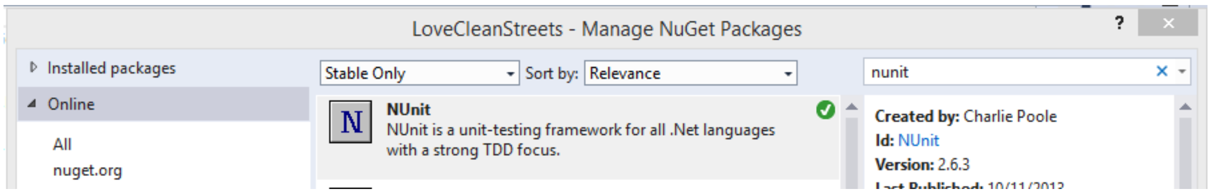
Begin by downloading a dataset of 50,000 events from lovecleanstreets.com an Azure service which collects data from members of the public on behalf of the local authority in the UK. The dataset can be downloaded from here:

http://elastastorage.blob.core.windows.net/hdinsight/LoveCleanStreets.csv

Creating the project

To begin we'll create a project and add dependencies. Open Visual Studio and create a console application. It should be called LoveCleanStreets and also a library project called LoveCleanStreetsLib. LoveCleanStreets should be dependent on LoveCleanStreetsLib. All of the files in this section should be added to **LoveCleanStreetsLib**.

When the project has been created right-click on the references and a reference to Nunit.


LoveCleanStreets - Manage NuGet Packages

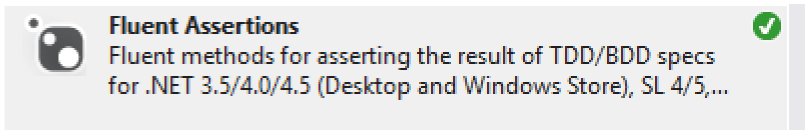
Installed packages
Online
All
nuget.org

Stable Only | Sort by: Relevance | nunit

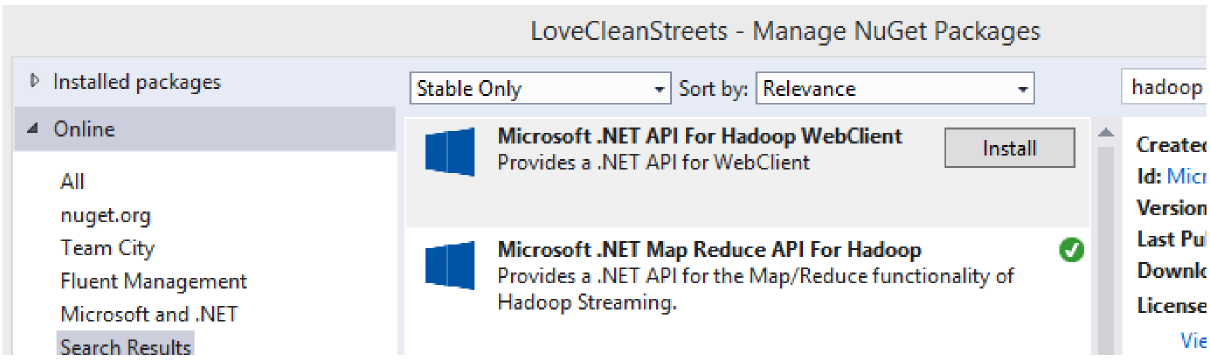
NUnit
NUnit is a unit-testing framework for all .Net languages with a strong TDD focus.

Created by: Charlie Poole
Id: NUnit
Version: 2.6.3

To FluentAssertions:


Fluent Assertions
Fluent methods for asserting the result of TDD/BDD specs for .NET 3.5/4.0/4.5 (Desktop and Windows Store), SL 4/5,...

And to Microsoft.Hadoop.MapReduce:


LoveCleanStreets - Manage NuGet Packages

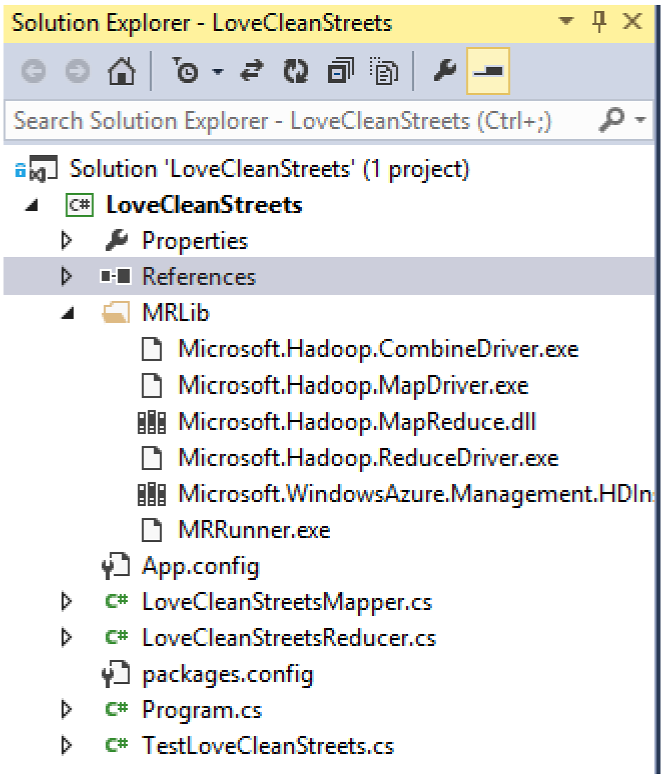
Installed packages
Online
All
nuget.org
Team City
Fluent Management
Microsoft and .NET
Search Results

Stable Only | Sort by: Relevance | hadoop

Microsoft .NET API For Hadoop WebClient
Provides a .NET API for WebClient
Install

Microsoft .NET Map Reduce API For Hadoop
Provides a .NET API for the Map/Reduce functionality of Hadoop Streaming.

When the project has been created you'll see a directory full of assemblies. This directory is the SDK to allow you to stream CLR types to Hadoop.


Solution Explorer - LoveCleanStreets
Search Solution Explorer - LoveCleanStreets (Ctrl+;)

```
Solution 'LoveCleanStreets' (1 project)
  LoveCleanStreets
    Properties
    References
    MRLib
      Microsoft.Hadoop.CombineDriver.exe
      Microsoft.Hadoop.MapDriver.exe
      Microsoft.Hadoop.MapReduce.dll
      Microsoft.Hadoop.ReduceDriver.exe
      Microsoft.WindowsAzure.Management.HDIn
      MRRunner.exe
    App.config
    LoveCleanStreetsMapper.cs
    LoveCleanStreetsReducer.cs
    packages.config
    Program.cs
    TestLoveCleanStreets.cs
```

Open the dataset in Excel.

ReportId, ReportItemId, Latitude, Longitude, ReportItemDateTime, ItemNote, ReportDateTime, Completed, CompletedDate, CategoryId, CategoryName, ReportNotes, FormattedAddress, LA, Ward, ImageUrl

For the purposes of this lab we're interested in only **LA** and **ReportItemDate**.

We'll begin by creating a Mapper class as follows. The file should be called **LoveCleanStreetsMapper**.

Add the following using statement to the top of the C# file:

```
using Microsoft.Hadoop.MapReduce;
```

Then add the following class definition.

```
public class LoveCleanStreetsMapper : MapperBase
    {
        #region Overrides of MapperBase

        public override void Map(string inputLine, MapperContext context)
        {
            var parts = inputLine.Split(',');
            if (parts.Length != 16)
                return;
            var key = String.Format("{0}", parts[13]);
            context.EmitKeyValue(key, "1");
        }

        #endregion
    }
```

Now we'll create a class called **LoveCleanStreetsReducer** adding the using statement as above.

```
public class LoveCleanStreetsReducer : 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
    }
```

Writing Tests

We'll be testing both the mapper and reducer using the **StreamingUnit** framework.

We'll be using NUnit to test the Mapper and Reducer with the following class definition. This should be in a new file called **TestLoveCleanStreets**.

```
  [TestFixture]
    public class TestLoveCleanStreets
    {

    }
```

We're going to create 4 sample reports and add them to the class.

```
private const string Report1 = "1,2,3,4,2013-10-21,5,,2013-10-
21,Cleared,1,,,,Leicester,,";
private const string Report2 = "1,2,3,4,2013-10-21,5,,2013-10-
21,Cleared,1,,,,London,,";
private const string Report3 = "1,2,3,4,2013-10-21,5,,2013-10-
21,Cleared,1,,,,London,,";
// this report has an invalid length of 15
private const string Report4 = "1,2,3,4,2013-10-21,5,,2013-10-
21,Cleared,1,,,,London,";
```

The first 3 contain LA's of London or Leicester and the 4th report is invalid as each report should have a column length of 16.

We'll add an initial test to ensure that the Mapper filters out the invalid record.

```
public void LoveCleanStreetsMapper_FilterLA_ThreeItemsFromMapper()
        {
            //ReportId      ReportItemId  Latitude      Longitude
       ReportItemDateTime   ItemNotes
            //ReportDateTime       Completed     CompletedDate CategoryId
       CategoryName  ReportNotes   FormattedAddress
            //LA     Ward    ImageUrl
            var result = StreamingUnit.Execute<LoveCleanStreetsMapper,
LoveCleanStreetsReducer>(new[]
            {
                Report1,
                Report2,
                Report3,
                Report4
            });

            result.MapperResult.Count.Should().Be(3);
        }
```

You should be able to see that **StreamingUnit** can be used to test the mapper and reducer with a collection as if it was running on an **HDInsight** cluster.

As our Mapper is keying on the LA and there are only two LAs we should have two items from our reducer. Add the following test to check this.

```
[Test]
        public void LoveCleanStreetsMapper_FilterLA_TwoItemsFromReducer()
        {
            //ReportId      ReportItemId  Latitude      Longitude
       ReportItemDateTime   ItemNotes
            //ReportDateTime       Completed     CompletedDate CategoryId
       CategoryName  ReportNotes   FormattedAddress
            //LA     Ward   ImageUrl
            var result = StreamingUnit.Execute<LoveCleanStreetsMapper,
LoveCleanStreetsReducer>(new[]
            {
                Report1,
                Report2,
                Report3,
                Report4
            });

            result.ReducerResult.Count.Should().Be(2);
        }
```

Let's check to see whether we have a key separated by a Tab character and one or more values separated by non-Tabs.

```
[Test]
        public void LoveCleanStreetsMapper_FilterLA_CheckReducerTwoPartsToLine()
        {
            //ReportId      ReportItemId  Latitude      Longitude
       ReportItemDateTime   ItemNotes
            //ReportDateTime       Completed     CompletedDate CategoryId
       CategoryName  ReportNotes   FormattedAddress
            //LA     Ward   ImageUrl
            var result = StreamingUnit.Execute<LoveCleanStreetsMapper,
LoveCleanStreetsReducer>(new[]
            {
                Report1,
                Report2,
                Report3,
                Report4
            });

            result.ReducerResult[0].Split('\t').Length.Should().Be(2);
        }
```

We'll now check that we have a Leicester and London output with 1 record count and 2 record counts respectively.

```
       [Test]
        public void LoveCleanStreetsMapper_FilterLA_CheckReducerCountReports()
        {
            //ReportId      ReportItemId  Latitude      Longitude
        ReportItemDateTime   ItemNotes
            //ReportDateTime       Completed     CompletedDate CategoryId
        CategoryName  ReportNotes   FormattedAddress
            //LA     Ward    ImageUrl
            var result = StreamingUnit.Execute<LoveCleanStreetsMapper,
LoveCleanStreetsReducer>(new[]
            {
                Report1,
                Report2,
                Report3,
                Report4
            });

            result.ReducerResult[0].Split('\t')[0].Should().Be("Leicester");
            int.Parse(result.ReducerResult[0].Split('\t')[1]).Should().Be(1);
            result.ReducerResult[1].Split('\t')[0].Should().Be("London");
            int.Parse(result.ReducerResult[1].Split('\t')[1]).Should().Be(2);
        }
```

Now we're going to enhance the mapper and enrich the key output so that break the results down by **LA** and **ReportItemDateTime**.

Let's update the Mapper:

```
var key = String.Format("{0},{1}", parts[13], DateTime.Parse(parts[4]).ToString("yyyy-
MM-dd"));
```

We'll add another report to the Test file that can test a variation in date:

```
// this report will occur on a different day
        private const string Report5 = "1,2,3,4,2013-10-22,5,,2013-10-
21,Cleared,1,,,,Leicester,,";
```

Lastly we'll add another test which can be used to test the variation in date.

```
       [Test]
        public void LoveCleanStreetsMapper_FilterLA_CheckReducerCountReportsByDay()
        {
            //ReportId      ReportItemId  Latitude      Longitude
        ReportItemDateTime   ItemNotes
            //ReportDateTime       Completed     CompletedDate CategoryId
        CategoryName  ReportNotes   FormattedAddress
            //LA     Ward    ImageUrl
            var result = StreamingUnit.Execute<LoveCleanStreetsMapper,
LoveCleanStreetsReducer>(new[]
            {
                Report1,
                Report2,
                Report3,
                Report4,
                Report5
            });

            result.ReducerResult[0].Split('\t')[0].Should().Be("Leicester,2013-10-
21");
            int.Parse(result.ReducerResult[0].Split('\t')[1]).Should().Be(1);
            result.ReducerResult[1].Split('\t')[0].Should().Be("Leicester,2013-10-
22");
            int.Parse(result.ReducerResult[1].Split('\t')[1]).Should().Be(1);
            result.ReducerResult[2].Split('\t')[0].Should().Be("London,2013-10-21");
            int.Parse(result.ReducerResult[2].Split('\t')[1]).Should().Be(2);
        }
```

Creating the cluster

Now that the tests are complete we'll create a job which will allow us to submit to a real cluster.

Create a new file called LoveCleanStreetsJob.cs and add the following to it.

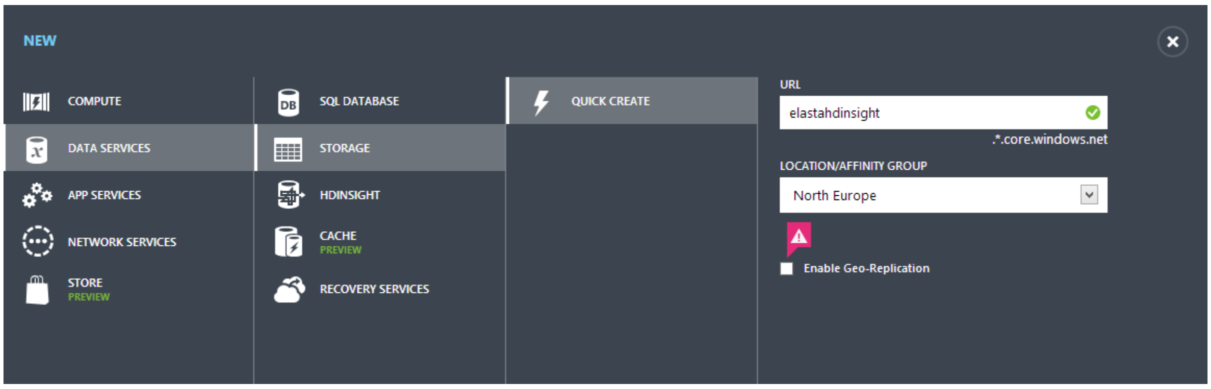
```
public class LoveCleanStreetsJob : HadoopJob<LoveCleanStreetsMapper,
LoveCleanStreetsReducer>
    {
        #region Overrides of HadoopJob

        public override HadoopJobConfiguration Configure(ExecutorContext context)
        {
            return new HadoopJobConfiguration()
            {
                InputPath = "/input/LoveCleanStreets.csv",
                OutputFolder = "/output/LoveCleanStreets"
            };
        }

        #endregion
    }
```

Following on from this we'll create a Windows Azure Storage account. To do this let's navigate to the Windows Azure Portal and upload the LoveCleanStreets dataset.

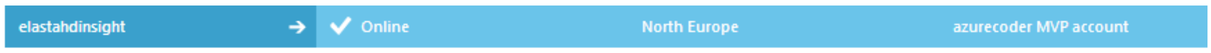
I'll use QuickCreate from the portal like so:



URL: elastahdinsight .*.core.windows.net

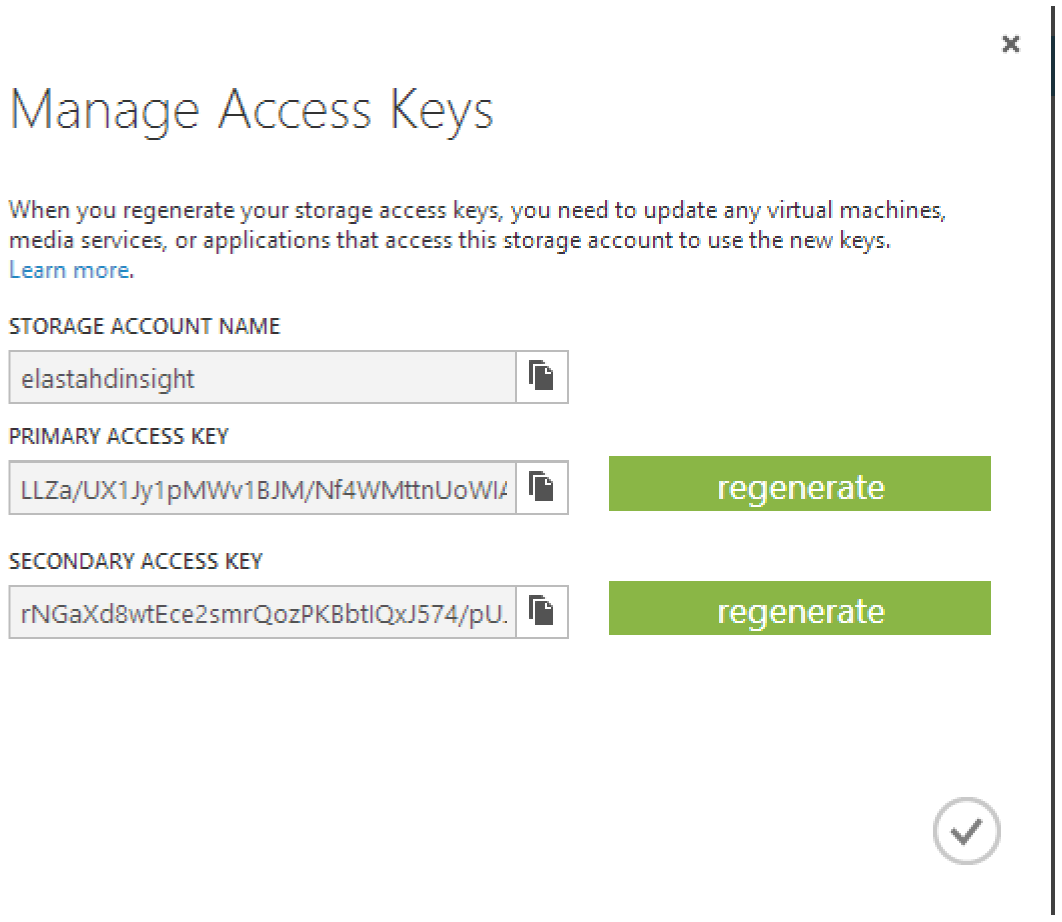
LOCATION/AFFINITY GROUP: North Europe

It should take no more than a minute and the account will be available.



elastahdinsight | Online | North Europe | azurecoder MVP account

You should be able to highlight it and select the Manage Access Keys option from the AppBar menu.



Manage Access Keys

When you regenerate your storage access keys, you need to update any virtual machines, media services, or applications that access this storage account to use the new keys. Learn more.

STORAGE ACCOUNT NAME

elastahdinsight

PRIMARY ACCESS KEY

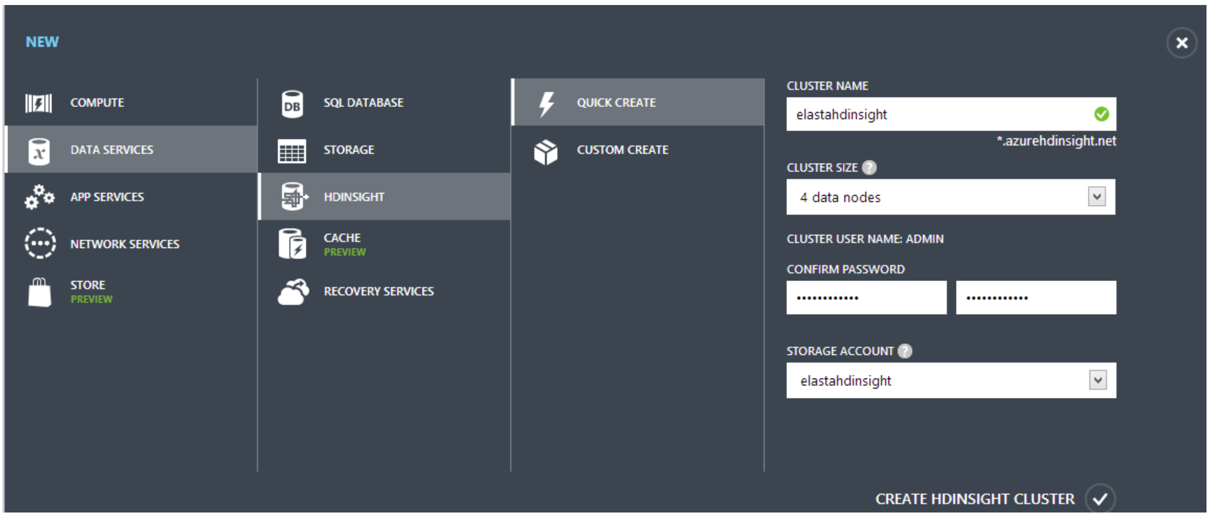
LLZa/UX1Jy1pMWv1BJM/Nf4WMttnUoWI/

SECONDARY ACCESS KEY

rNGaXd8wtEce2smrQozPKBbtIQxJ574/pU.

Copy the primary key using the symbol to the right of the textbox.

Create an HDInsight cluster as follows:



CLUSTER NAME: elastahdinsight .azurehdinsight.net

CLUSTER SIZE: 4 data nodes

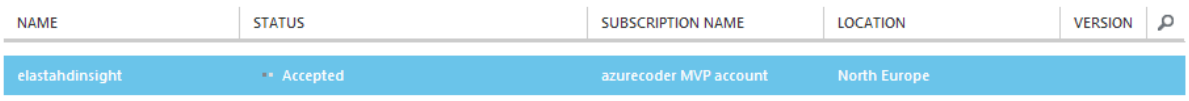
CLUSTER USER NAME: ADMIN

CONFIRM PASSWORD: ············ ············

STORAGE ACCOUNT: elastahdinsight

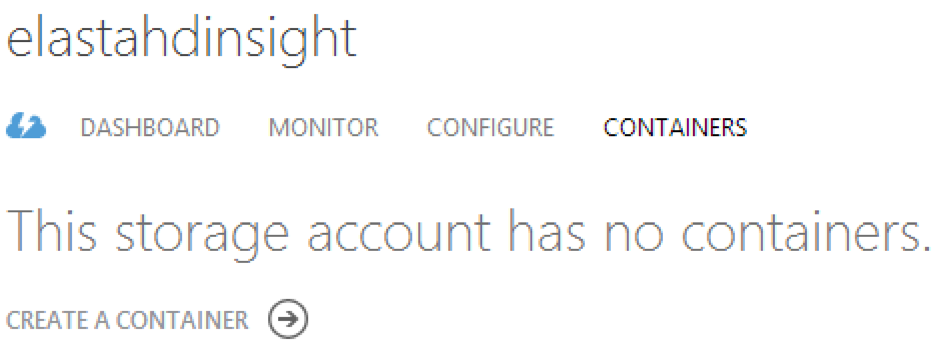
By default the username is Admin. Stick with a 4 node cluster which should cost around £1.20/hour.

The cluster will take a little longer than the storage account to be created. Around 12-15 minutes is a good rule of thumb.

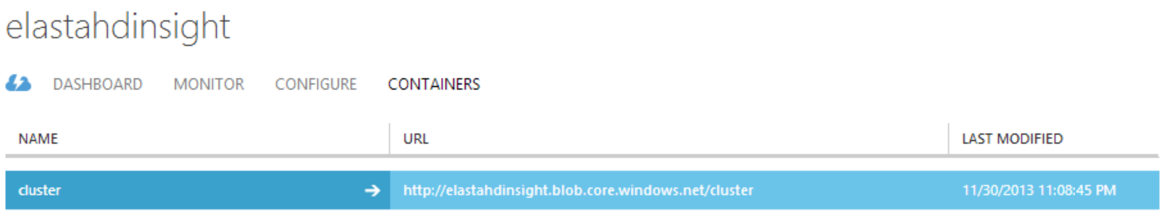


NAME	STATUS	SUBSCRIPTION NAME	LOCATION	VERSION
elastahdinsight	Accepted	azurecoder MVP account	North Europe	

Whilst this is occurring navigate to the Storage Account in the portal and add a container called cluster. You can leave the access set to the default private.



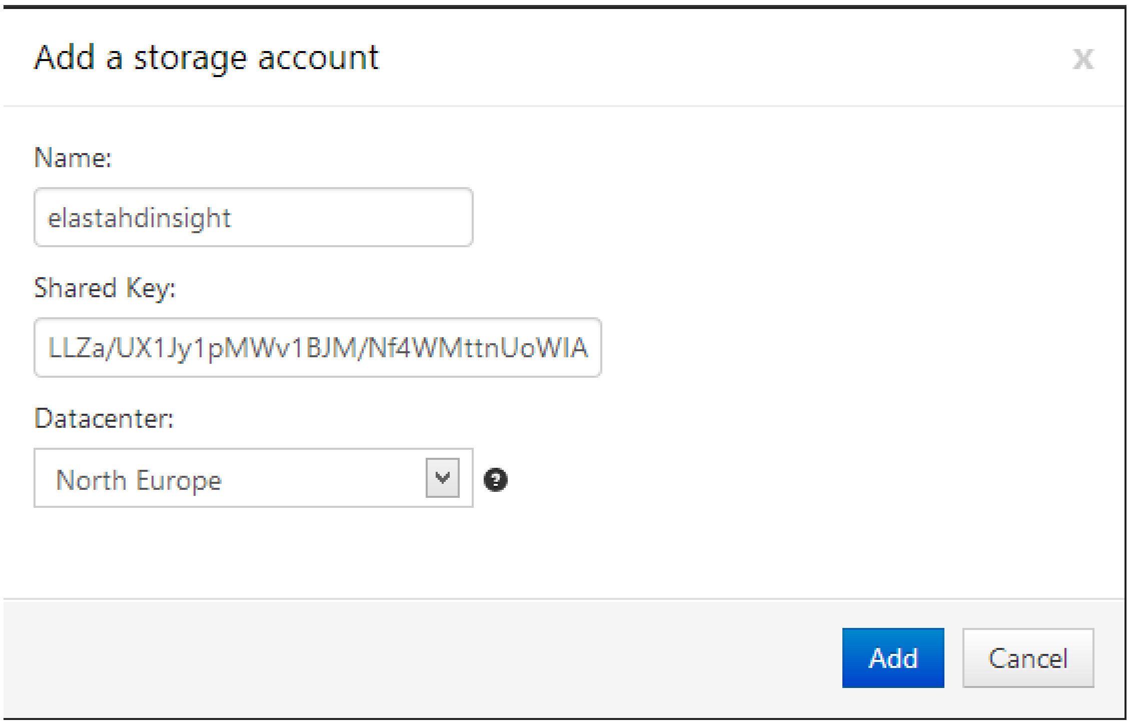
You should see the following when complete.



NAME	URL	LAST MODIFIED
cluster	http://elastahdinsight.blob.core.windows.net/cluster	11/30/2013 11:08:45 PM

You'll need to use a storage management tool like zud.io or Azure Management Studio to upload the LCS dataset to the input

The example shows zud.io but you can also use AMS if you're used to that. Both are equally good.



Name:

elastahdinsight

Shared Key:

LLZa/UX1Jy1pMWv1BJM/Nf4WMttnUoWIA

Datacenter:

North Europe

Use the storage management tool to upload **LoveCleanStreets.csv** to an input folder in the cluster container. The file is 22MB so it may take a few minutes.

Running the Job

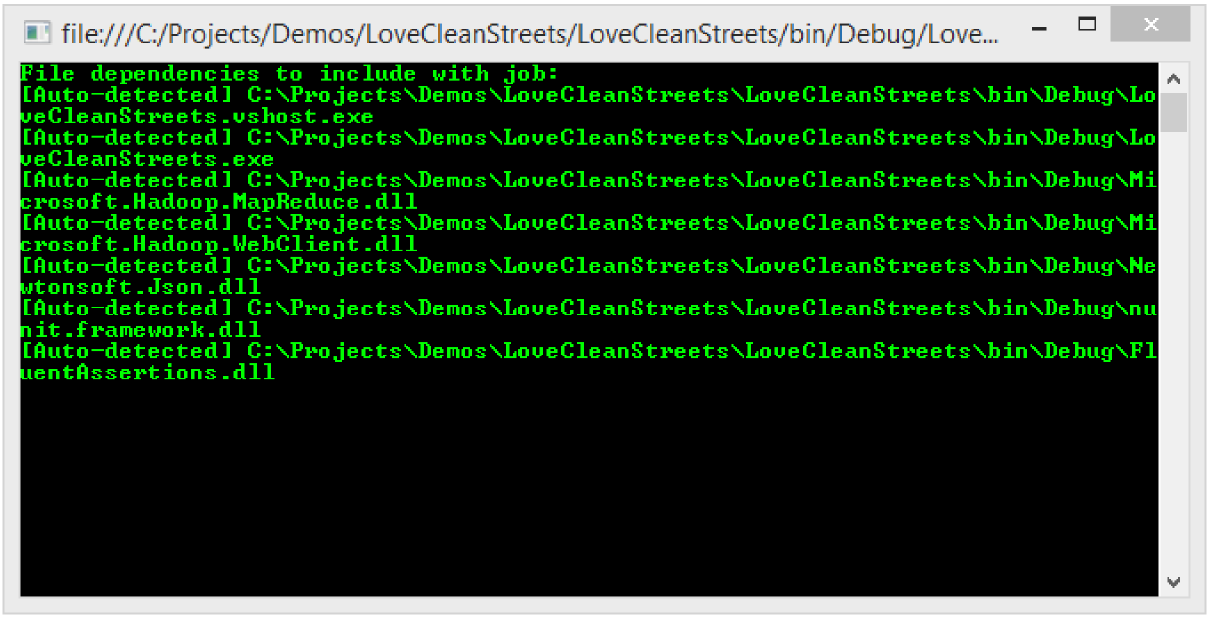
In the LoveCleanStreets project In the Main method in Program.cs add the following.

```
  class Program
    {
        static void Main(string[] args)
        {
            var hadoop = Hadoop.Connect(new
Uri("https://elastahdinsight.azurehdinsight.net"), "admin", "hadoop", "@azurecodeR1",
                "elastahdinsight",
"LLZa/UX1Jy1pMWv1BJM/Nf4WMttnUoWIAFidgUVOwZSxW633JL3XUiEKIKzBbzf0NrEhGhvXQ3yWqhXs6JYVo
w==", "cluster", false);
            var result = hadoop.MapReduceJob.ExecuteJob<LoveCleanStreetsJob>();
            if (result.Info.ExitCode != 0)
            {
                Console.WriteLine("Returned with unexpected exit code of " +
result.Info.ExitCode);
            }
            Console.ReadLine();
        }
    }
```

Replace the above with your own cluster Uri, password, storage account name and key. All other parameters can remain the same.

By now the cluster creation should have completed.

Run the code in Debug mode. The command Window should show something like the following:


file:///C:/Projects/Demos/LoveCleanStreets/LoveCleanStreets/bin/Debug/Love...

```
File dependencies to include with job:
[Auto-detected] C:\Projects\Demos\LoveCleanStreets\LoveCleanStreets\bin\Debug\Lo
veCleanStreets.vshost.exe
[Auto-detected] C:\Projects\Demos\LoveCleanStreets\LoveCleanStreets\bin\Debug\Lo
veCleanStreets.exe
[Auto-detected] C:\Projects\Demos\LoveCleanStreets\LoveCleanStreets\bin\Debug\Mi
crosoft.Hadoop.MapReduce.dll
[Auto-detected] C:\Projects\Demos\LoveCleanStreets\LoveCleanStreets\bin\Debug\Mi
crosoft.Hadoop.WebClient.dll
[Auto-detected] C:\Projects\Demos\LoveCleanStreets\LoveCleanStreets\bin\Debug\Ne
wtonsoft.Json.dll
[Auto-detected] C:\Projects\Demos\LoveCleanStreets\LoveCleanStreets\bin\Debug\nu
nit.framework.dll
[Auto-detected] C:\Projects\Demos\LoveCleanStreets\LoveCleanStreets\bin\Debug\Fl
uentAssertions.dll
```

The job should return with an error code of 1. On inspection we have the following failure:

```
Unhandled Exception: System.FormatException: The string was not recognized
as a valid DateTime. There is an unknown word starting at index 0.
   at System.DateTime.Parse(String s)
   at LoveCleanStreetsLib.LoveCleanStreetsMapper.Map(String inputLine,
MapperContext context)
```

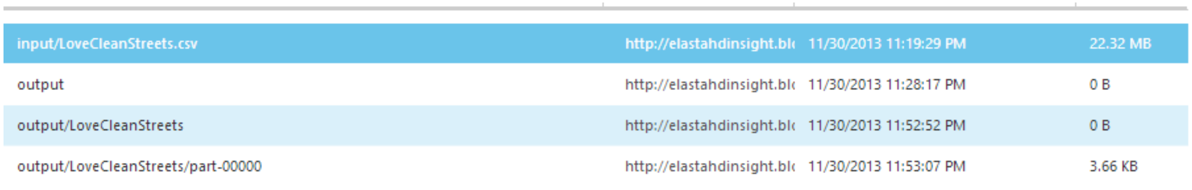
When the cluster fails it cannot be recovered so we'll ensure that the job doesn't fail by handling the exception. The exception is occurring because the first line of the file lists all of the column headings so cannot be parsed as a DateTime. As such we'll update the mapper to suppress the exception.

```
            try
            {
                var parts = inputLine.Split(',');
                if (parts.Length != 16)
                    return;
                var key = String.Format("{0},{1}", parts[13],
DateTime.Parse(parts[4]).ToString("yyyy-MM-dd"));
                context.EmitKeyValue(key, "1");
            }
            catch (Exception)
            {
            }
```

Then run the job again.

Confirming the output

When the job complete we can inspect the output by looking at the Windows Azure portal and downloading the output file from Blob Storage.



input/LoveCleanStreets.csv	http://elastahdinsight.bl	11/30/2013 11:19:29 PM	22.32 MB
output	http://elastahdinsight.bl	11/30/2013 11:28:17 PM	0 B
output/LoveCleanStreets	http://elastahdinsight.bl	11/30/2013 11:52:52 PM	0 B
output/LoveCleanStreets/part-00000	http://elastahdinsight.bl	11/30/2013 11:53:07 PM	3.66 KB

We should have a 3.66kB file called **output/LoveCleanStreets/part-00000**

When we inspect the output we can see that we have an empty LA. Data generally contains noise and we need to determine whether this is meaningful or noisy to our output. A subsequent update of the Mapper can filter these out if we believe them to be meaningless to what we're trying to show.

```
,2011-08-12	1
,2011-09-11	1
,2011-12-04	1
```

Also look at the data for Lewisham. An additional exercise could be done by enriching the data. A quick scan of dates will show that there is a higher incidence of reports over the weekends. This could be because more crime happens over the weekend or more people are around to report it.

```
Lewisham London Borough,2011-07-04	1
Lewisham London Borough,2011-07-11	3
Lewisham London Borough,2011-08-11	6
Lewisham London Borough,2011-09-11	4
Lewisham London Borough,2011-10-11	4
Lewisham London Borough,2012-01-04	4
Lewisham London Borough,2012-01-05	27
Lewisham London Borough,2012-02-04	22
Lewisham London Borough,2012-02-05	18
Lewisham London Borough,2012-03-04	33
Lewisham London Borough,2012-04-04	22
Lewisham London Borough,2012-04-05	1
Lewisham London Borough,2012-05-04	30
Lewisham London Borough,2012-06-04	2
Lewisham London Borough,2012-06-08	1
Lewisham London Borough,2012-07-04	3
Lewisham London Borough,2012-08-04	3
Lewisham London Borough,2012-08-05	2
Lewisham London Borough,2012-09-04	7
Lewisham London Borough,2012-10-04	50
Lewisham London Borough,2012-11-04	23
Lewisham London Borough,2012-12-04	28
```

Challenge

We've looked at simple usage of Map-Reduce. In this challenge we want you to enhance the mapper and reducer to enable the determination of the average time of resolution of a reported incident per LA and per LA/Ward (two jobs).