

HBase Lab: Analyze real-time Twitter Sentiment with HBase in HDInsight

*This lab has been adapted and extended by [Data Science Dojo](#) from Microsoft's support documentation for HBase, originally contributed by [Mumian](#). The original documentation can be [here](#).

Contents

Lab 1: Preface.....	3
Exercise 1: Lab Prerequisites.....	3
Exercise 2: Understanding the App.....	4
Lab 2: Configuring Twitter.....	6
Exercise 1: Getting a Twitter Account.....	6
Exercise 2: Converting Twitter into a Streaming App.....	7
Lab 3: Setting Up an HBase Cluster.....	9
Exercise 1: Create an Azure Storage Account.....	9
Exercise 2: Provisioning an HDInsight HBase Cluster.....	10
Lab 4: Creating a .Net Twitter Stream Service.....	11
Exercise 1: Setting up the Solution File.....	11
Exercise 2: Installing Dependencies & Packages.....	12
Exercise 3: Loading the HBase Writer Class.....	1617
Exercise 4: Loading Program.cs.....	1819
Exercise 5: Configuring your App.....	1920
Exercise 6: Running your App to Collect Data.....	2526
Lab 5: Create an Azure Website to visualize Twitter sentiment.....	2728
Exercise 1: Setting up the Solution File.....	2829
Exercise 2: Installing Dependencies & Packages.....	3132
Exercise 3: HBase Reader Class.....	3334
Exercise 4: Tweets Controller.....	3536
Exercise 5: Heat Map Library.....	3738
Exercise 6: Website Core Functionalities.....	3839
Exercise 7: HTML & CSS.....	3940
Lab 6: Web Deployment.....	4243
Exercise 1: Testing Your Webpage.....	4243
Exercise 2: Publishing Your Webpage.....	4546

Lab 1: Preface

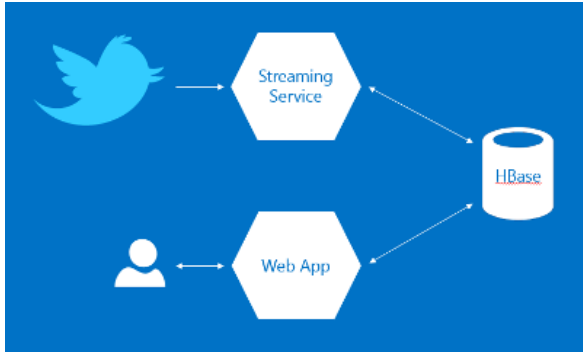
Exercise 1: Lab Prerequisites

1. Windows Operating System (or Windows Virtual Machine)
2. Visual Studio (Free Trial, or Community Edition also works) (Lab 4, Exercise 1)
3. A Twitter Account (Lab 2, Exercise 1)
4. An Azure Account (Lab 3, Exercise 1)

Exercise 2: Understanding the App

Learn how to do real-time [sentiment analysis](#) of big data using HBase in an HDInsight (Hadoop) cluster.

Social web sites are one of the major driving forces for Big Data adoption. Public APIs provided by sites like Twitter are a useful source of data for analyzing and understanding popular trends. In this tutorial, you will develop a console streaming service application and an ASP.NET web application to perform the following:

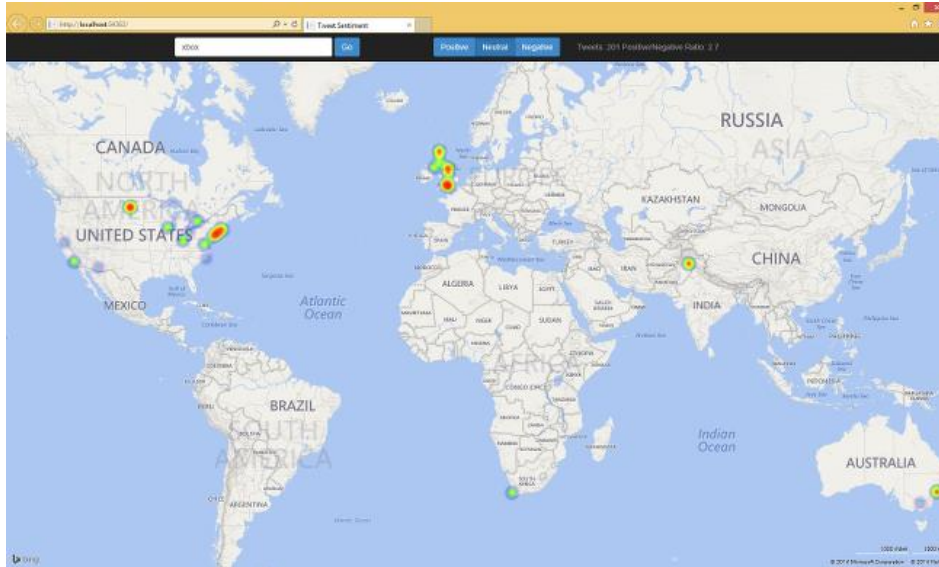


1. The streaming application
 - a) Get geo-tagged Tweets in real-time using the Twitter streaming API.
 - b) Evaluate the sentiment of these Tweets.
 - c) Store the sentiment information in HBase.

```
file:///C:/tutorials/TweetSentimentStreaming/TweetSentimentStreaming/bin/D...
501729043868180480: English "Omikareyesss: @kieferravenna aint effective." ohh..
Location: 125.167834, 6.125631
Tweets/sec: 123
Rows written: 5127
501729047794425856: Undefined @SofiaSifre QUE
Location: -66.071136, 18.365115
Tweets/sec: 113
501729052496252929: English I #vote5sos for the 2014 MTV #UMA Best Lyric Video!
See who's in the lead and vote: http://t.co/8JYMtCPtSH x69
Location: 17.923249, 59.390403
Tweets/sec: 121
Rows written: 5118
501729056237178880: Japanese ?????????????????????????????????????????? http://t.co/
SUJX0uhh9o
Location: 139.06788, 36.400123
Tweets/sec: 129
501729058929926144: Undefined lemas dlm lemak2 perut uwais hahahha http://t.co/4
apPcj3UuE
Location: 101.545247, 3.150171
Tweets/sec: 111
```

2. The Azure Web application

- a) Plot the real-time statistical results on Bing maps using an ASP.NET Web application. A visualization of the tweets will look like this:



You will be able to query tweets with certain keywords to get a sense of whether the expressed opinion of a tweet is positive, negative, or neutral.

An example of the completed web application can be found here: <http://tweetsentiment.azurewebsites.net/>.

Lab 2: Configuring Twitter

Exercise 1: Getting a Twitter Account

We will be taking a conventional Twitter account and turning it into a web app that will live stream incoming tweets from around the world. To do this you must first have a Twitter account. **If you already have a Twitter account, go ahead and skip to Exercise 2.**

1. Go to <http://twitter.com> and find the sign up box, or go directly to <https://twitter.com/signup>.
2. Enter your **full name**, **email address**, and a **password**.
3. Click **Sign up for Twitter**.
4. On the next page, you can select a **username** (usernames are unique identifiers on Twitter) — type your own or choose one we've suggested. We'll tell you if the username you want is available.
5. **Double-check** your name, email address, password, and username.
6. Click **Create my account**. You may be asked to complete a Captcha to let us know that you're human.
7. Twitter will send a **confirmation email** to the email address you entered. Click the link in that email to confirm your email address and account.

Exercise 2: Converting Twitter into a Streaming App

The Twitter Streaming APIs use [OAuth](#) to authorize requests. The first step to use OAuth is to create a new application on the Twitter Developer site.

To create Twitter application ID and secrets:

1. Sign in to <https://apps.twitter.com/>.
2. Click **Create New App**.
3. Enter **Name**, **Description**, **Website**. The Website field is not really used. It **does not** have to be a valid URL. The following table shows some sample values to use:

FIELD	VALUE
Name	MyGloballyUniqueTwitterAppName
Description	<YourDescription>
Website	http://www.WhateverYouWant.com

NOTE: The name Twitter application name must be a unique name.

1. Check **Yes, I agree**, and then click **Create your Twitter application**.
2. Click the **Permissions** tab. The default permission is **Read only**. This is sufficient for this tutorial.
3. Click the **Keys and Access Tokens** tab.
4. Click **Create my access token**.
5. Click **Test OAuth** in the upper right corner of the page.
6. Write down (save to a notepad) **Consumer key**, **Consumer secret**, **Access token**, and **Access token secret**. You will need the values later in the tutorial.



OAuth Tool

OAuth Settings

Consumer key: *

G5jxFAKRIYTbJauIBXVI9jrbC

Consumer secret: *

skNtbrZfmQU08JN4FCqcintFLNg77N0kA0nLXDesBCAwB8Q0gZ

Remember this should not be shared.

Access token:

23087070-eGMoPyH7x2r2jP9IfwpXSrUAd5gd4EnR71QY9YQab

Access token secret:

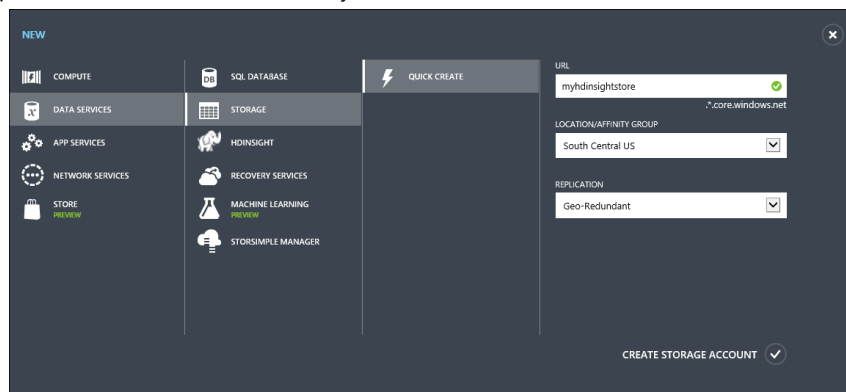
J78XjeowwyHNmgkROS6OQQKPY4GSo3el2m3iGuYMqTyFN

Lab 3: Setting Up an HBase Cluster

Exercise 1: Create an Azure Storage Account

Since HBase runs on the Hadoop framework, it will require you to link it to an external data storage unit. In this case we will be using an Azure Blob storage.

1. Sign into your Azure account if you have one.
 - a) If not, sign up for a free trial Azure account.
<http://azure.microsoft.com/en-us/pricing/free-trial/>
2. Create an Azure Storage (sometimes called Azure Storage Vault). HDInsight and Hadoop do not deal with the storage or persistence of data and instead reference a data store.
 - b) Once you are logged into your Azure portal (<https://manage.windowsazure.com/>), click **New>Data Services>Storage>Quick Create**
 - c) URL: The URL will be the name of your storage and serve as a pseudo primary key for your storage name. Assign a unique name to it. It's called a URL because the storage account can be referenced directly via HTTP, ex: <https://mystorageaccount.blob.core.windows.net/>
 - d) Location: Select a region that is closest to you, your users, or your Hadoop clusters.
 - e) Replication: Geo-Redundant or Locally Redundant will suffice.

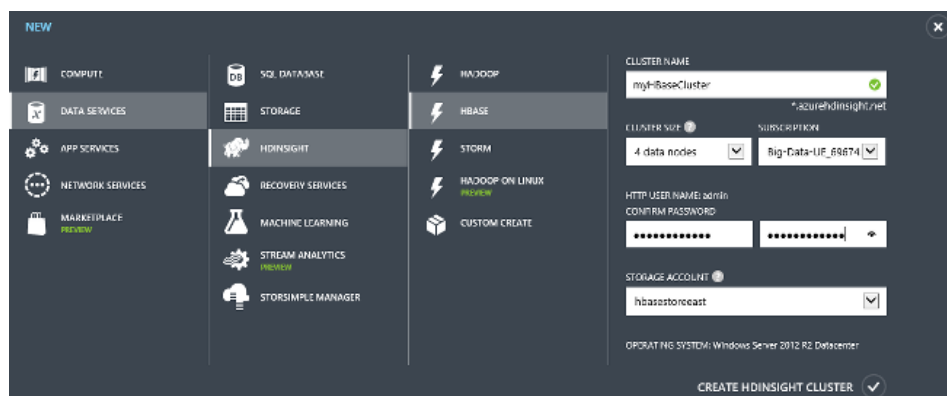


- f) Provisioning a storage will take ~2 minutes after hitting the check mark button.

Exercise 2: Provisioning an HDInsight HBase Cluster

To provision an HBase cluster by using the Azure portal

- 1) Sign in to the [Azure portal](#).
- 2) Click **NEW** in the lower left, and then click **DATA SERVICES > HDINSIGHT > HBASE**
- 3) Enter
 - a) **CLUSTER NAME:** Set a globally unique name for your cluster. Imagine that you are picking a name for a website.
 - b) **CLUSTER SIZE:** 1 data node.
 - c) **CLUSTER USER PASSWORD:** <YourPassword>
 - d) **STORAGE ACCOUNT:** Reference the storage account you created in the previous exercise.



The screenshot shows the 'NEW' page in the Azure portal. On the left, a navigation pane lists various services under categories like COMPUTE, DATA SERVICES, and APP SERVICES. The 'HDINSIGHT' service is selected under 'DATA SERVICES'. On the right, the configuration form for an HDInsight HBase cluster is displayed. The form includes fields for 'CLUSTER NAME' (set to 'my-hbase-cluster'), 'CLUSTER SIZE' (set to '4 data nodes'), 'CLUSTER USER PASSWORD' (with a confirmation field), and 'STORAGE ACCOUNT' (set to 'hbasestorageaccount'). The 'OPERATING SYSTEM' is set to 'Windows Server 2012 R2 Datacenter'. A 'CREATE HDINSIGHT CLUSTER' button with a checkmark icon is at the bottom right.

The default HTTP USER NAME is admin.

Click the checkmark icon in the lower right to create the HBase cluster.

The cluster will; take ~17 minutes to provision. So please move onto the next lab if possible.

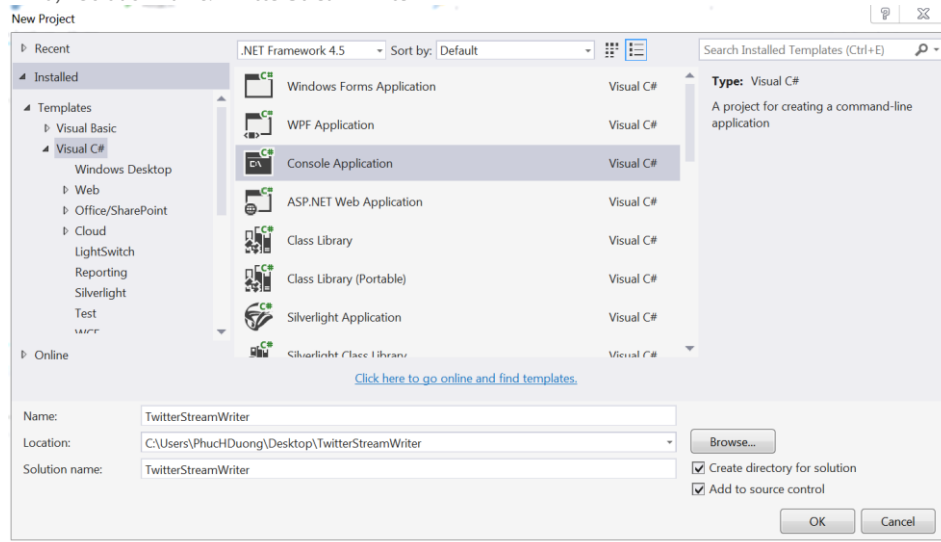
Lab 4: Creating a .Net Twitter Stream Service

We will now build a C# application that will intercept tweet streams from our Twitter account. The C# app will also take those tweets, parse them, and then write them to the HBase cluster. To do this we will be using Visual Studio.

If you do not have Visual Studio, please navigate to the following link (windows only) and download the 90 day free trial: <https://www.visualstudio.com/en-us/downloads/download-visual-studio-vs.aspx>

Exercise 1: Setting up the Solution File

- 1) **To create the Visual Studio solution:**
- 2) Open **Visual Studio**.
- 3) From the **File** menu, point to **New**, and then click **Project**.
- 4) Type or select the following values:
 - a) Template: **Visual C# / Windows Desktop / Console Application**
 - b) Name: **TwitterStreamWriter**
 - c) Location: **C:\YourDesktop\TwitterStreamWriter**
 - d) Solution name: **TwitterStreamWriter**

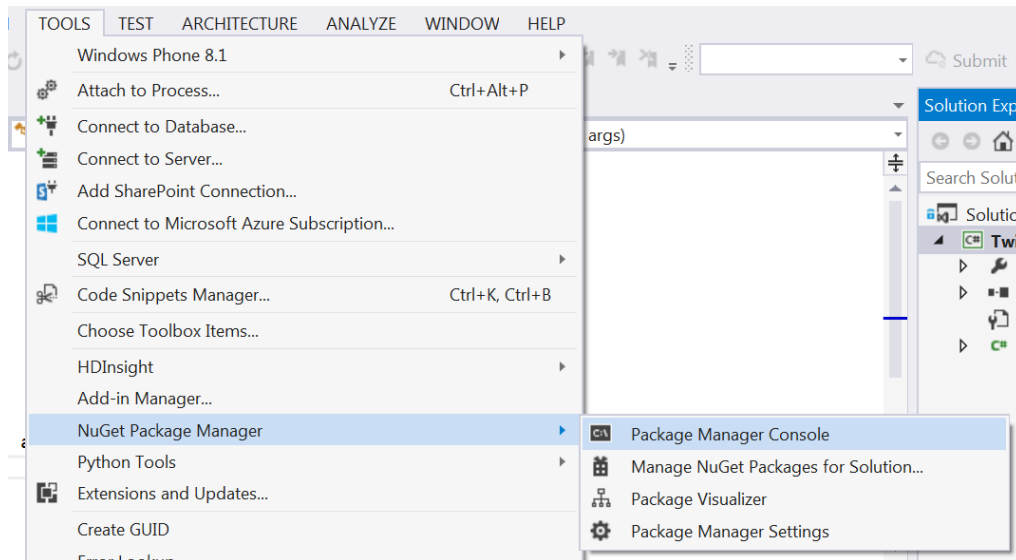


- 5) Click **OK** to continue.

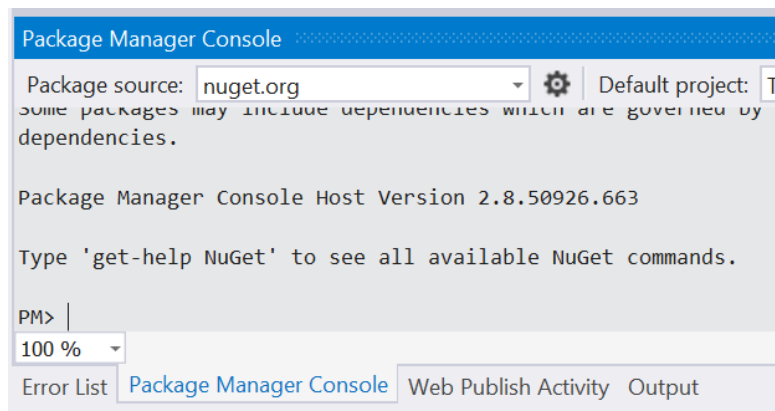
Exercise 2: Installing Dependencies & Packages

This lab uses preexisting libraries written for the Twitter API. There also exists libraries associated with HBase connectivity. We will install and reference these libraries/packages using the **NuGet Package Manager**.

- 1) From the **Tools** menu, click **NuGet Package Manager**, and then click **Package Manager Console**. The console panel will open at the bottom of the page.



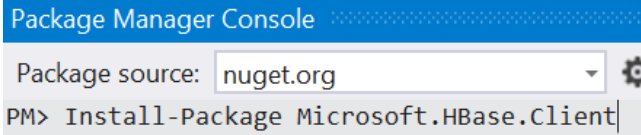
- 2) Wait for the Package Manager Console to initialize. Look for a "PM>" at the very bottom.



- 3) Use the following commands to install the [HBase.NET SDK](#) package, which is client library used to access HBase clusters, and the [Tweetinvi](#) package, which is used to access the Twitter API.

- a) Install the HBase client: type in the following command into the Package Manager Console.

```
PM> Install-Package Microsoft.HBase.Client
```

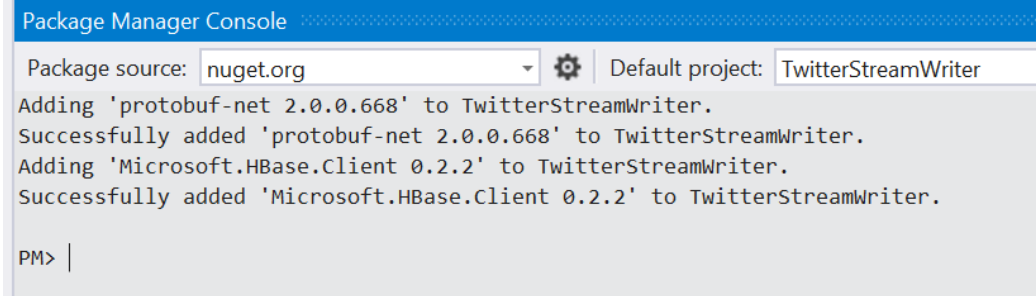


Package Manager Console

Package source:

PM> Install-Package Microsoft.HBase.Client

Success Output:



Package Manager Console

Package source: Default project:

Adding 'protobuf-net 2.0.0.668' to TwitterStreamWriter.
Successfully added 'protobuf-net 2.0.0.668' to TwitterStreamWriter.
Adding 'Microsoft.HBase.Client 0.2.2' to TwitterStreamWriter.
Successfully added 'Microsoft.HBase.Client 0.2.2' to TwitterStreamWriter.

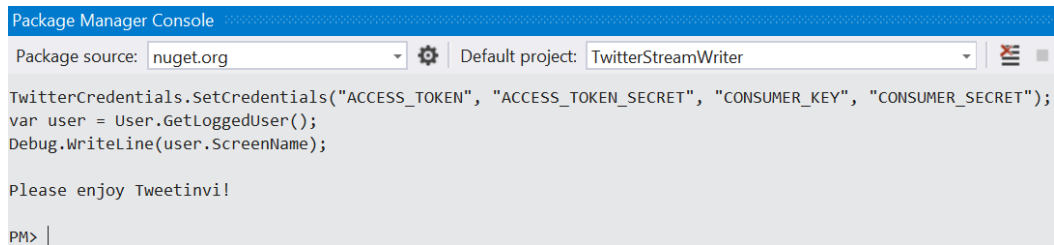
PM> |

- b) Install the TwitterInvi package: type the following command into the Package Manager Console.

```
PM> Install-Package TweetinviAPI
```

```
PM> Install-Package TweetinviAPI
```

On success output:



Package Manager Console

Package source: Default project:

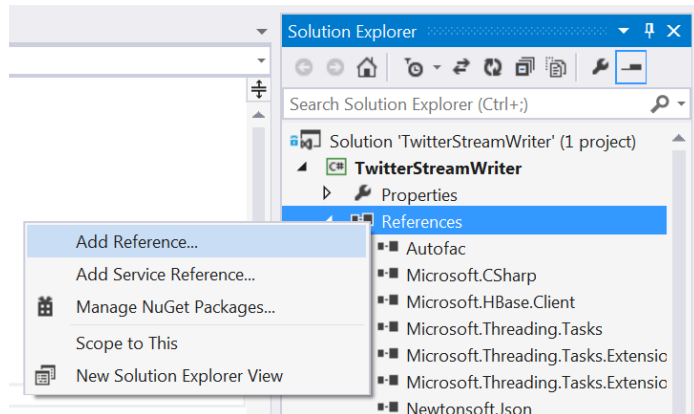
TwitterCredentials.SetCredentials("ACCESS_TOKEN", "ACCESS_TOKEN_SECRET", "CONSUMER_KEY", "CONSUMER_SECRET");
var user = User.GetLoggedInUser();
Debug.WriteLine(user.ScreenName);

Please enjoy Tweetinvi!

PM> |

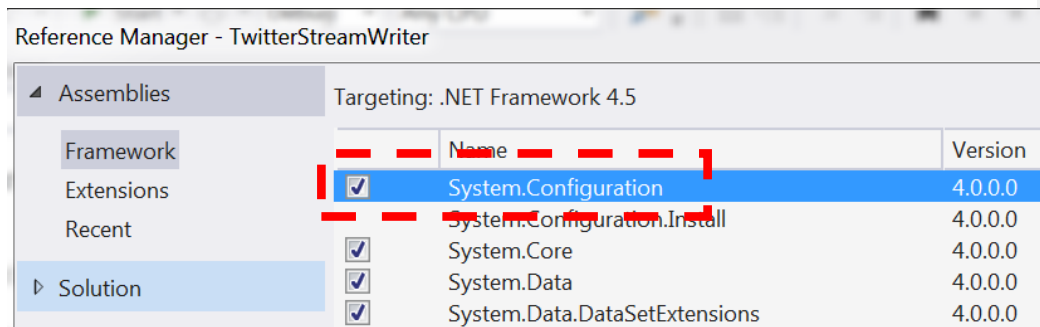
4) **Installing System.Configuration:**

5) From **Solution Explorer**, right-click **References**, and then click **Add Reference**.

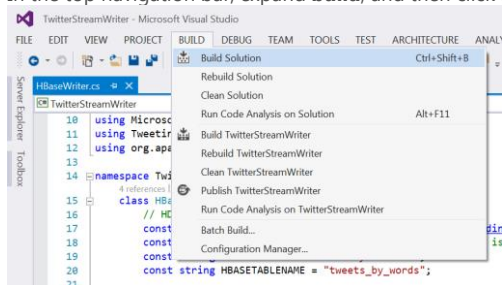


6) In the left pane, expand **Assemblies**, and then click **Framework**.

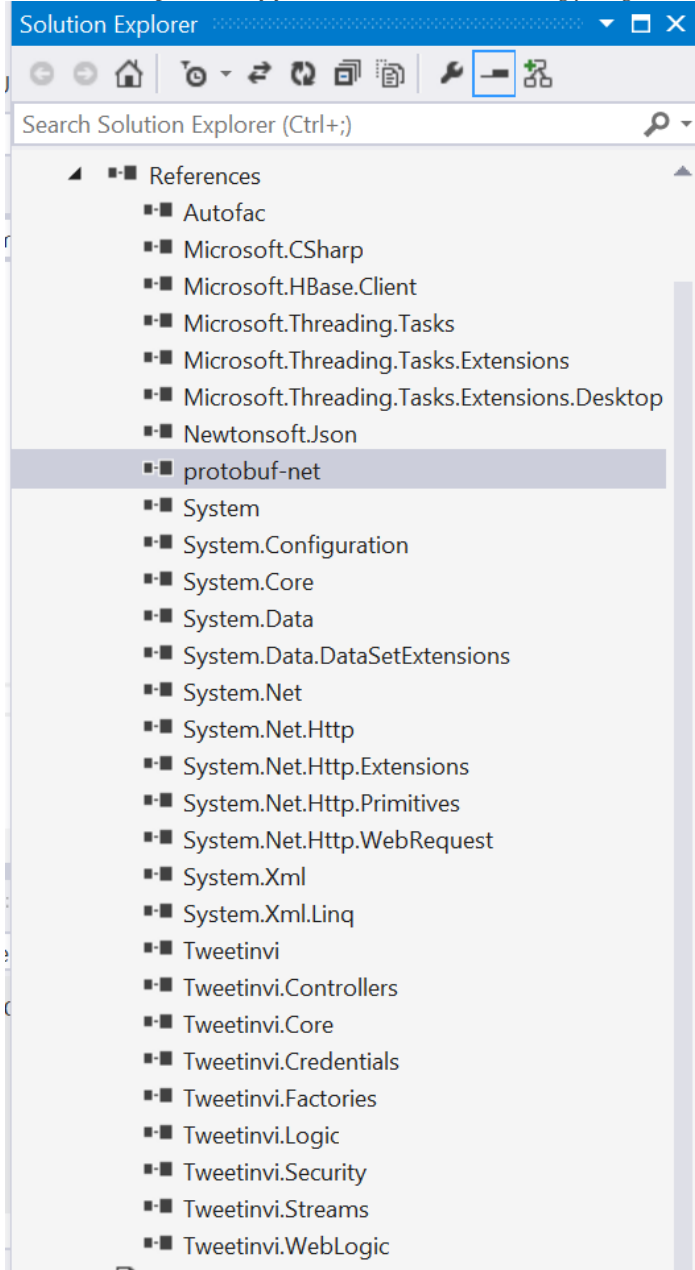
7) In the right pane, select the checkbox in front of **System.Configuration**, and then click **OK**.



8) In the top navigation bar, expand **build**, and then click **Build Solution**.



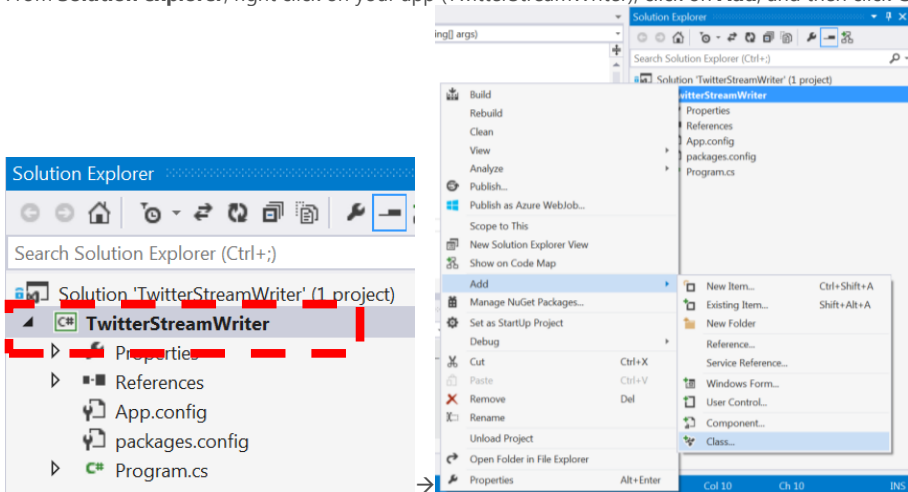
- 9) From **Solution Explorer**, verify your solution contains the following packages.



Exercise 3: Loading the HBase Writer Class

The HBase writer that will take incoming tweet payloads and write them to the HBase cluster.

- 1) From **Solution explorer**, right click on your app (TwitterStreamWriter), click on **Add**, and then click **Class**.

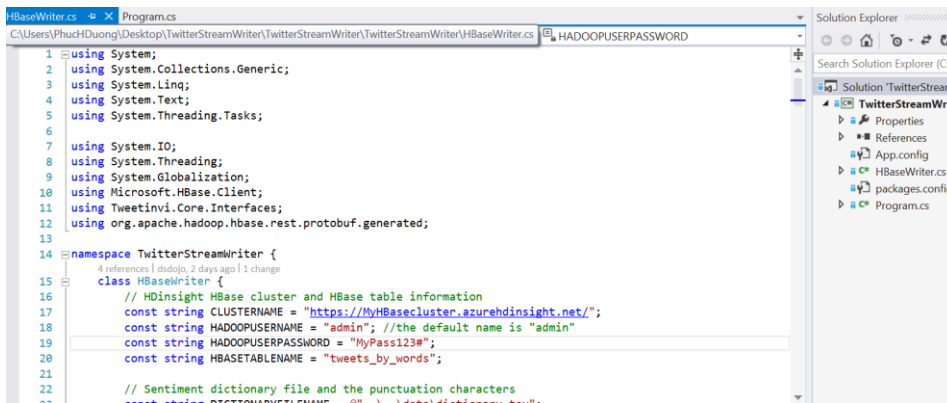


- 2) In the bottom textbox labeled **Name**, type **HBaseWriter.cs** and then click **Add**.

- 3) We have written a template HBase writer for you, please view the following link:

- a) <https://github.com/datasciencedojo/TwitterHBaseStreamWriter/blob/master/TwitterStreamWriter/HBaseWriter.cs>

- b) Copy and paste the entirety of the file, then paste it into your own HBaseWriter.cs.



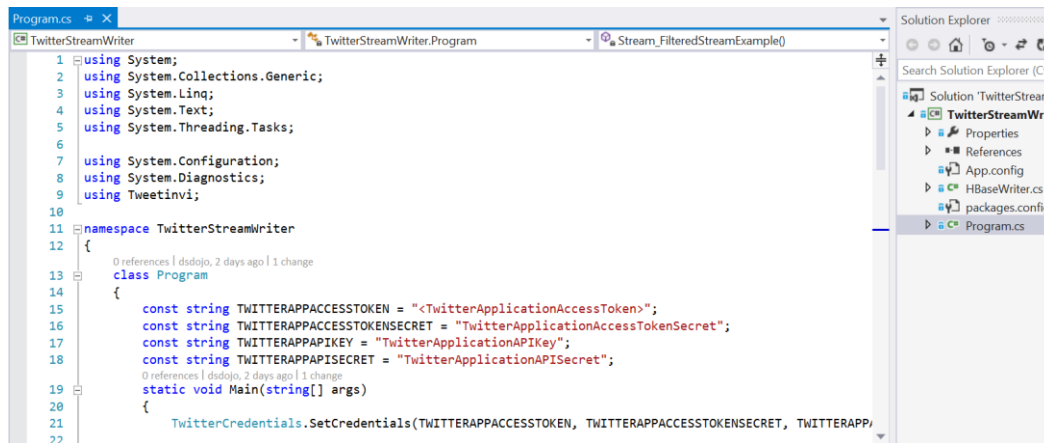
The HBaseWriter.cs provides the following functionality:

- **Connect to Hbase [HBaseWriter()]:** Use the HBase SDK to create a *ClusterCredentials* object with the cluster URL and the Hadoop user credential, and then create a *HBaseClient* object using the *ClusterCredentials* object.
- **Create HBase table [HBaseWriter()]:** The method call is *HBaseClient.CreateTable()*.
 - In this case it creates a table called "*tweets_by_words*" to store all of our incoming tweets.
- **Write to HBase table [WriterThreadFunction()]:** The method call is *HBaseClient.StoreCells()*.
- **Calculate the sentiment of the tweet:** *CalcSentimentScore()* will calculate whether the tweet is a negative, positive or neutral sentiment tweet based upon a dictionary that we will feed it later.

Exercise 4: Loading Program.cs

Now that the HBaseWriter class is defined, our app has been given the functionality to parse tweets, write Tweets to HBase, and calculate each tweet's sentiment. However we don't have a file that executes anything yet. Let's define our program.cs in order to use our newly defined functions.

- 1) Open your **program.cs** file within your solution explorer.
- 2) To save time we have pre-written a program.cs file for you:
 - a) <https://github.com/datasciencedojo/TwitterHBaseStreamWriter/blob/master/TwitterStreamWriter/TwitterStreamWriter/Program.cs>
 - b) Copy and paste the following GitHub code file above, and overwrite your program.cs file.



```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 using System.Configuration;
8 using System.Diagnostics;
9 using Tweetinvi;
10
11 namespace TwitterStreamWriter
12 {
13     0 references | dsdojo, 2 days ago | 1 change
14     class Program
15     {
16         const string TWITTERAPPACCESSTOKEN = "<TwitterApplicationAccessToken>";
17         const string TWITTERAPPACCESSTOKENSECRET = "TwitterApplicationAccessTokenSecret";
18         const string TWITTERAPPAPIKEY = "TwitterApplicationAPIKey";
19         const string TWITTERAPPAPISECRET = "TwitterApplicationAPISecret";
20         0 references | dsdojo, 2 days ago | 1 change
21         static void Main(string[] args)
22         {
23             TwitterCredentials.SetCredentials(TWITTERAPPACCESSTOKEN, TWITTERAPPACCESSTOKENSECRET, TWITTERAPP
```

Program.cs will make the direct connection to your Twitter account, open up the live stream feed, and then write incoming tweets to the HBase cluster using the HBaseWriter.cs class we defined earlier.

Exercise 5: Configuring your App

Now we must configure the App to use your own HBase cluster, as well as your own twitter account. To do this we will have to give it the correct credentials to access your HBase cluster and Twitter account.

1) Referencing your HBase cluster:

- a) There are 4 constants defined at the top of the HBaseWriter class within your HBaseWriter.cs file that will directly link to your HBase cluster, we just need to point them in the correct direction.
- i) **"CLUSTERNAME"** (line 17): This will be the direct URL connection string to your HBase cluster. Go to your Azure Management Portal (manage.windowsazure.com), click on **HDInsight > YourCluster -> Dashboard**.

The screenshot shows the Microsoft Azure portal interface. The top navigation bar includes 'Microsoft Azure', 'CREDIT STATUS', 'Subscriptions', and a user profile. The left sidebar contains various service icons, with 'myhbasecluster' highlighted. The main content area displays the 'myhbasecluster' dashboard, which includes tabs for 'DASHBOARD', 'MONITOR', 'CONFIGURATION', and 'SCALE'. The dashboard shows a 'usage' section with a bar chart indicating '18 Cores' and '23% of 60 HDINSIGHT CORES'. A 'quick glance' section provides summary information: 'STATUS: Running', 'LOCATION: West US', 'CLUSTER OPERATING SYSTEM: Windows Server 2012 R2 Datacenter', 'CLUSTER CONNECTION STRING: https://myhbasecluster.azurehdinsight.net', and 'CREATION DATE'. A 'linked resources' table lists 'dojoattendeestorage' as a 'Storage Account' with a status of 'Online'. A yellow arrow originates from the 'CLUSTERNAME' constant in the code below and points to the 'CLUSTER CONNECTION STRING' value in the dashboard.

```
10 using Microsoft.HBase.Client;
11 using Tweetinvi.Core.Interfaces;
12 using org.apache.hadoop.hbase.rest.protobuf.generated;
13
14 namespace TwitterStreamWriter {
15     4 references | dsdojo, 2 days ago | 1 change
16     class HBaseWriter {
17         // HDInsight HBase cluster and HBase table information
18         const string CLUSTERNAME = "https://MyHBasecluster.azurehdinsight.net/";
19         const string HADOOPUSERNAME = "admin"; //the default name is "admin"
20         const string HADOOPUSERPASSWORD = "MyPass123#";
21         const string HBASETABLENAME = "tweets_by_words";
```

- ii) **"HADOOPUSERNAME"**: Leave the username as "admin".
- iii) **"HADOOPUSERPASSWORD"**: Insert the password that you set for the HBase cluster when you provisioned it.
- iv) **"HBASETABLENAME"**: Leave it as "tweets_by_words";

2) Referencing your Twitter account:

- a) We will now reference the Twitter app that we created from your own personal Twitter account in Lab 2, Exercise 2.
- b) Navigate to: <https://apps.twitter.com/>
 - i) Click on your App.
 - ii) Click on "Keys and Access Tokens"

- iii) Under **Program.cs** has 4 constants defined at the top of the program class that defines your Twitter API keys and tokens. Please fill in the twitter key constants with your own twitter keys using the diagram below:

Application Settings

Keep the "Consumer Secret" a secret. This key should never be human-readable in your application.

Consumer Key (API Key)	JqyH4BF9JozhYgAXYoljbWw8H
Consumer Secret (API Secret)	At0dRtyjqHlvJtI32nggqwzAiWG25TEVHfe7QsR9LBivs425c
Access Level	Read-only (modify app permissions)
Owner	DataScienceDojo
Owner ID	1318985240

```
StreamWriter
TwitterStreamWriter.Program

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

using System.Configuration;
using System.Diagnostics;
using Tweetinvi;

namespace TwitterStreamWriter
{
    0 references | dsdojo, 2 days ago | 1 change
    class Program
    {
        const string TWITTERAPPACcesstoken = "<TwitterApplicationAcc
        const string TWITTERAPPACcesstokensecret = "TwitterApplicati
        const string TWITTERAPPAPIKEY = "TwitterApplicationAPIKey";
        const string TWITTERAPPAPISECRET = "TwitterApplicationAPISec
        0 references | dsdojo, 2 days ago | 1 change
        static void Main(string[] args)
        {
            TwitterCredentials.SetCredentials(TWITTERAPPACcesstoken,
```

Your Access Token

This access token can be used to make API requests on your own account's behalf. Do not share your access token.

Access Token	1318985240-yYIz4hVWNmbvbpNeMvRkPNPlwxjCO2XqFOb4leQ
Access Token Secret	FBteOpkARK8kN3dpy6aVvk3qgQMaKA7OC2xDsyDxU5pmjz
Access Level	Read-only
Owner	DataScienceDojo
Owner ID	1318985240

3) Configuring and loading the sentiment dictionary.

The HBaseWriter class will try to quantify whether or not each incoming tweet is negative, positive, or neutral before writing it to the HBase cluster. It does this by reading from a sentiment dictionary, which is a predefined table that lists whether or not a word is positive or negative based upon its context usage within the English language.

a) Obtaining the dictionary:

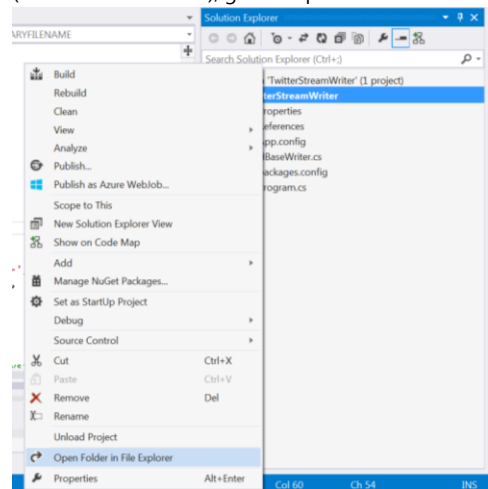
- i) Visit the following link, save the file as "dictionary.tsv".

<https://dojoattendeestorage.blob.core.windows.net/datasets/dictionary.tsv>

weaksubj	1	abandoned	adj	n	negative
weaksubj	1	abandonment	noun	n	negative
weaksubj	1	abandon verb	y		negative
strongsubj	1	abase verb	y		negative
strongsubj	1	abasement	anypos	y	negative
strongsubj	1	abash verb	y		negative
weaksubj	1	abate verb	y		negative
weaksubj	1	abdicate	verb	y	negative
strongsubj	1	aberration	adj	n	negative
strongsubj	1	aberration	noun	n	negative
strongsubj	1	abhor anypos	y		negative
strongsubj	1	abhor verb	y		negative
strongsubj	1	abhorred	adj	n	negative
strongsubj	1	abhorrence	noun	n	negative
strongsubj	1	abhorrent	adj	n	negative
strongsubj	1	abhorrently	anypos	n	negative
strongsubj	1	abhors adj	n		negative
strongsubj	1	abhors noun	n		negative
strongsubj	1	abidance	adj	n	positive
strongsubj	1	abidance	noun	n	positive
strongsubj	1	abide anypos	y		positive
strongsubj	1	abject adj	n		negative
strongsubj	1	abjectly	adverb	n	negative
weaksubj	1	abjure verb	y		negative
weaksubj	1	abilities	noun	n	positive
weaksubj	1	ability noun	n		positive
weaksubj	1	able adj	n		positive

ii)

- b) Moving your dictionary into your App folder:
- i) Within your visual studio solution for the twitter streaming app, right click on your app (TwitterStreamWriter), go to "Open Folder in File Explorer".

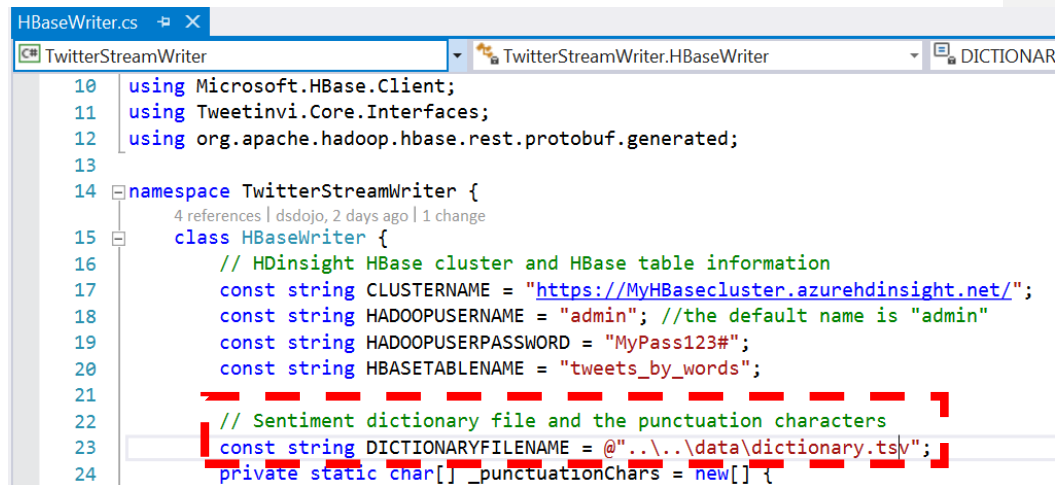


- ii) Create a new folder called "data" and drag the dictionary.tsv into it.

bin	3/20/2015 5:08 PM	File folder
data	3/20/2015 6:35 PM	File folder
obj	3/20/2015 5:08 PM	File folder
Properties	3/20/2015 5:08 PM	File folder
App.config	3/20/2015 5:23 PM	XML Configuration ...
HBaseWriter.cs	3/20/2015 6:37 PM	Visual C# Source file
packages.config	3/20/2015 5:23 PM	XML Configuration ...
Program.cs	3/20/2015 6:39 PM	Visual C# Source file
TwitterStreamWriter.csproj	3/20/2015 6:28 PM	Visual C# Project file

c) Referencing your dictionary within your App:

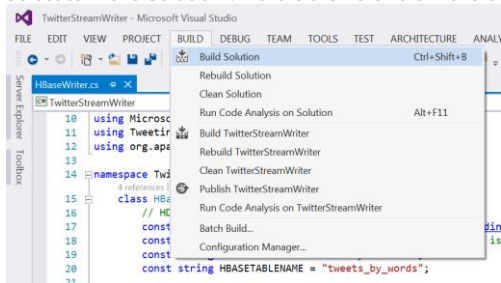
- i) Within your HBaseWriter.cs, there should be a constant defined as "DICTIONARYFILENAME", make sure it is referenced correctly to the correct folder, and it's named correctly. (it should already be referenced correctly named and referenced but just in case, check).



```
10 using Microsoft.HBase.Client;
11 using Tweetinvi.Core.Interfaces;
12 using org.apache.hadoop.hbase.rest.protobuf.generated;
13
14 namespace TwitterStreamWriter {
15     4 references | dsdojo, 2 days ago | 1 change
16     class HBaseWriter {
17         // HDinsight HBase cluster and HBase table information
18         const string CLUSTERNAME = "https://MyHBasecluster.azurehdinsight.net/";
19         const string HADOOPUSERNAME = "admin"; //the default name is "admin"
20         const string HADOOPUSERPASSWORD = "MyPass123#";
21         const string HBASETABLENAME = "tweets_by_words";
22
23         // Sentiment dictionary file and the punctuation characters
24         const string DICTIONARYFILENAME = @"\"..\\..\\data\\dictionary.tsv\";
25         private static char[] _punctuationChars = new[] {
```

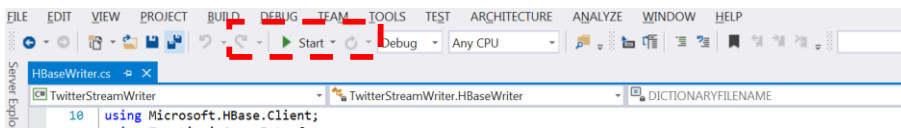

Exercise 6: Running your App to Collect Data

1. Build your solution to ensure there are no errors. At the top navigation bar, there should be a build button. Select to "Build Solution". If there are no errors move on, if not, then please troubleshoot.

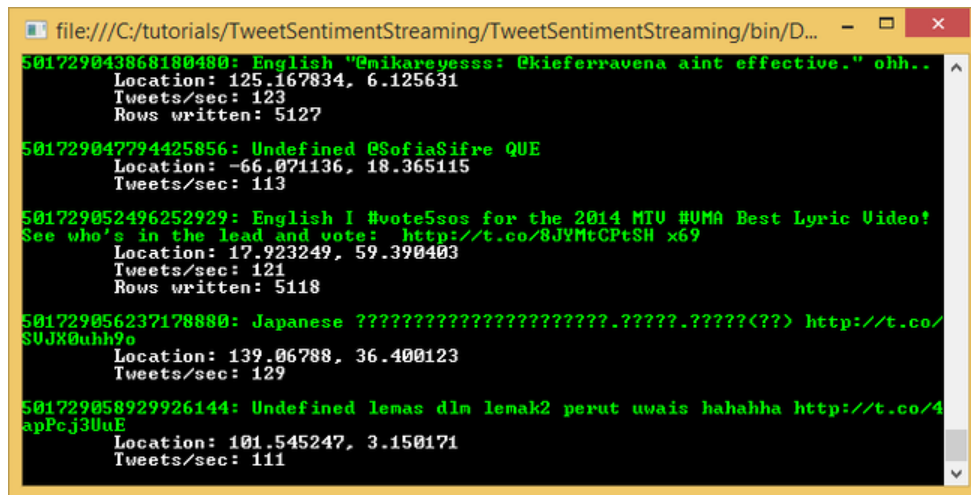


2. Run your program for the first time:

- a. Click the "Start" button.

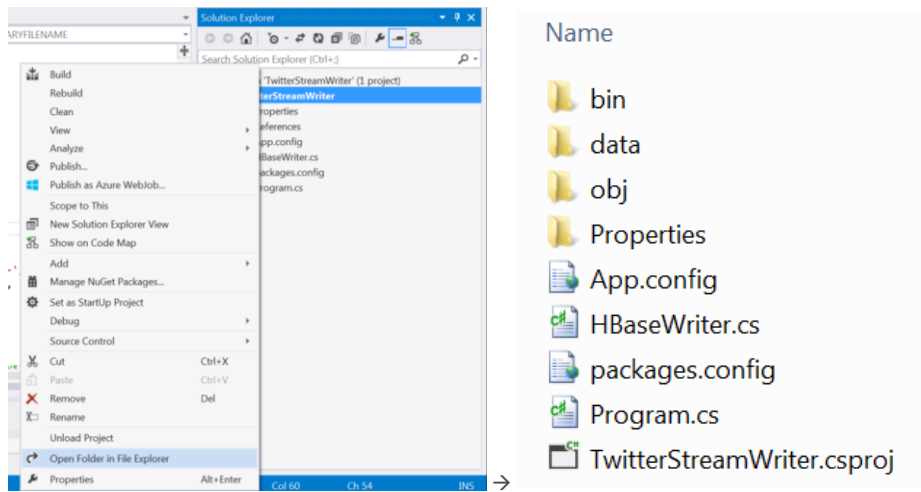


- b. Your app will now open up. It'll try to create a table. After which, it'll start streaming tweets over the console. Please make sure your output looks like the output below before moving on to the next step.



3. Run your app outside of visual studio.

- a. We want to keep this app on so that it can collect tweets for us. Ideally we'd be running this on commodity hardware such as Raspberry Pi. In all cases, we want to run this outside of visual studio to reduce ram usage.
- b. Make sure you've run you've built and successfully ran the app within visual studio before moving onto this step. Within your solution explorer, right click on your App, then go to "Open folder in File Explorer". This will open up your app folder.



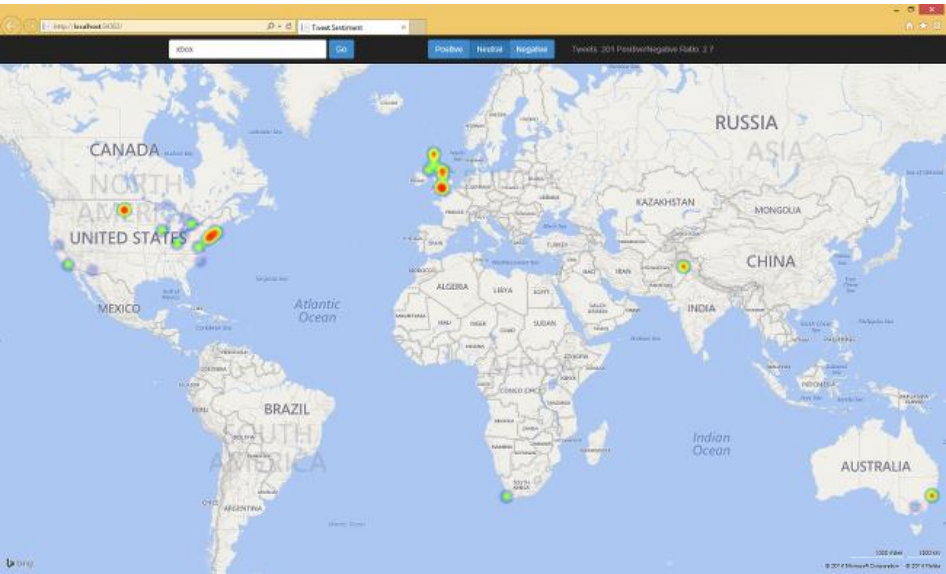
- c. Navigate: **bin > Debug > YourAppName.exe**

Name	Date modified	Type
TwitterStreamWriter.exe	3/20/2015 6:32 PM	Application
TwitterStreamWriter.vshost.exe	3/23/2015 7:41 AM	Application
Autofac.dll	3/20/2015 5:23 PM	Application extension
Microsoft HBase Client.dll	3/20/2015 5:19 PM	Application extension

- d. Click on it to execute it. You may create desktop shortcut of this file so you won't have to navigate through your folder directories again.
- e. Leave this app running in the background to accumulate twitter data.

Lab 5: Create an Azure Website to visualize Twitter sentiment

In this section, you will create an ASP.NET MVC Web application to read the real-time sentiment data from HBase and plot the data on Bing maps. We will also upload the website to an opened domain where you can show others. We will also be using a JavaScript heat map library to plot our tweets on a world map.

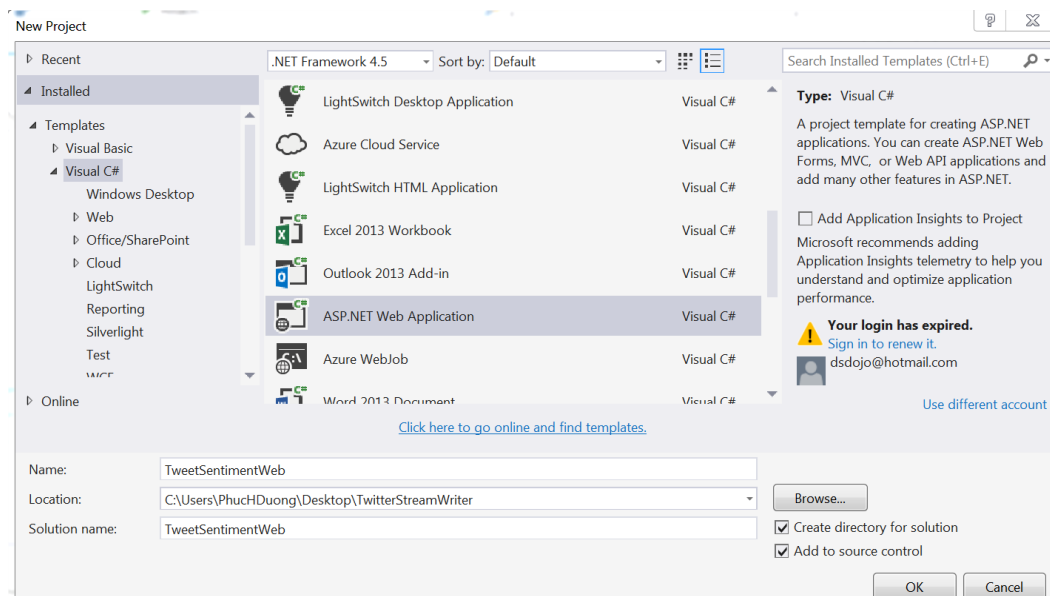


Exercise 1: Setting up the Solution File

We will be creating a new solution file for our new project. It is recommended that you create folder first manually to house this project file. For this example we created a Desktop folder that would house both apps, the twitter stream writer and the twitter heat map web app. Then created a separate folder within for each of the two apps.

To create an ASP.NET MVC Web application:

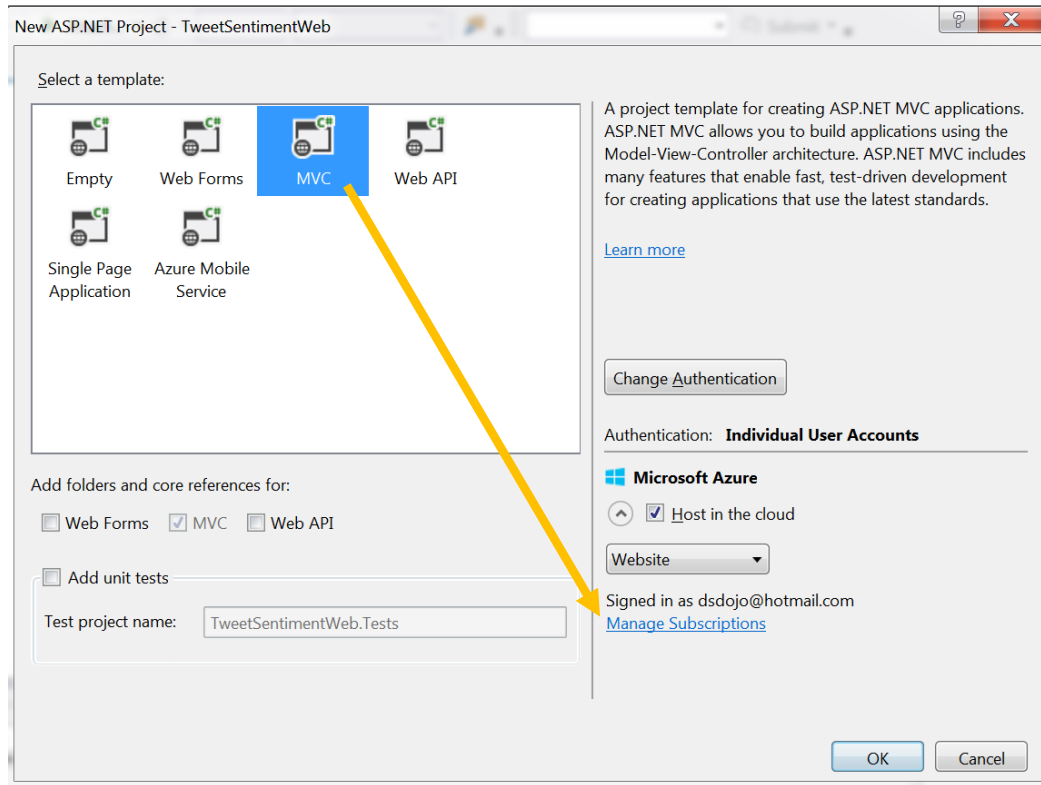
- 1) Open Visual Studio.
- 2) Click **File**, click **New**, and then click **Project**.
- 3) Type or enter the following:
 - a) Template category: **Visual C#/Web**
 - b) Template: **ASP.NET Web Application**
 - c) Name: **TweetSentimentWeb**
 - d) Location: **C:\Users\MyUserName\Desktop\TwitterStreamWriter**



- 4) Click **OK**.

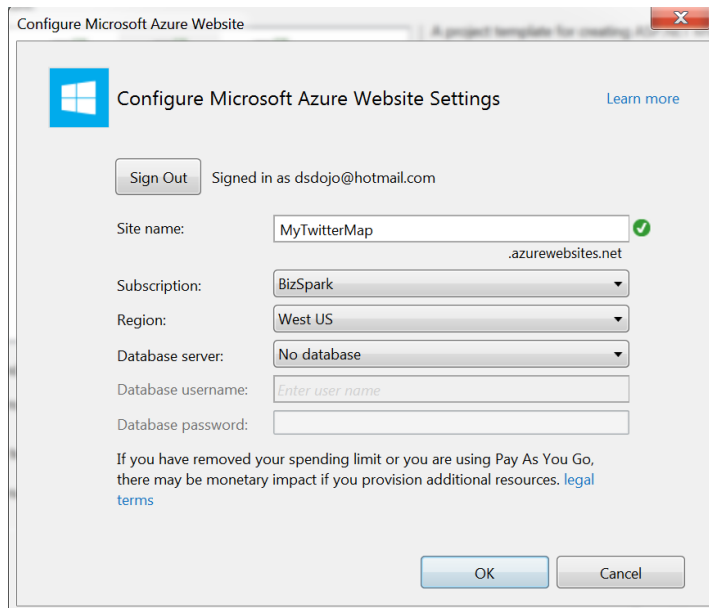
5) In **Select a template**, click **MVC**.

6) In **Microsoft Azure**, click **Manage Subscriptions**.



7) From **Manage Microsoft Azure Subscriptions**, click **Sign in**.


- 8) Enter your Azure credential. Your Azure subscription information will be shown on the Accounts tab.
- 9) Click **Close** to close the Manage Microsoft Azure Subscriptions window.
- 10) From **New ASP.NET Project - TweetSentimentWeb**, Click **OK**.



Configure Microsoft Azure Website

Configure Microsoft Azure Website Settings [Learn more](#)

Sign Out Signed in as dsdojo@hotmail.com

Site name: MyTwitterMap 
.azurewebsites.net

Subscription: BizSpark

Region: West US

Database server: No database

Database username: Enter user name

Database password:

If you have removed your spending limit or you are using Pay As You Go, there may be monetary impact if you provision additional resources. [legal terms](#)

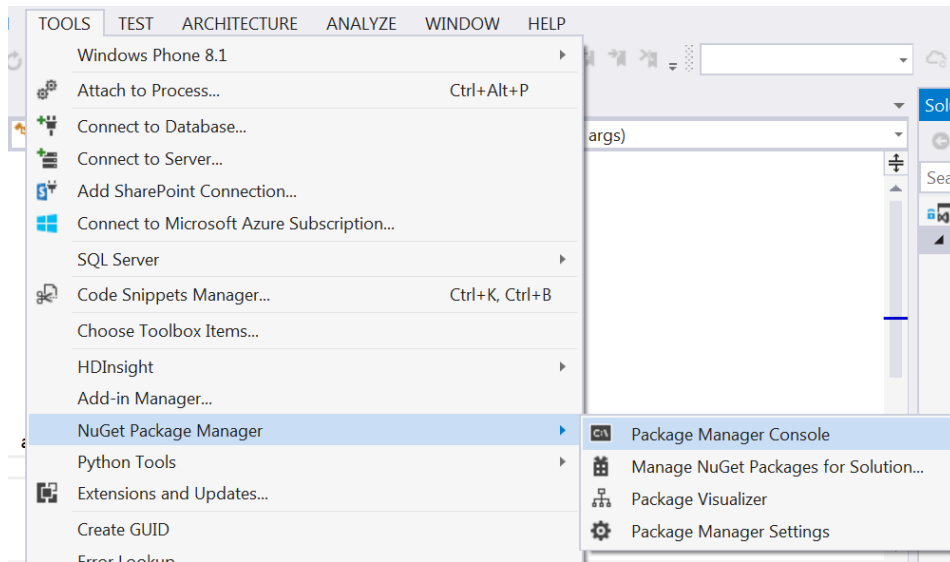
OK Cancel

- 11) **Name your website, this will be a public domain, accessible by anyone.** This will create an Azure Website and domain for you automatically.
- 12) From **Configure Microsoft Azure Site Settings**, select the **Region** that is closer to you. You don't need to specify a database server since we will be using an HBase cluster.

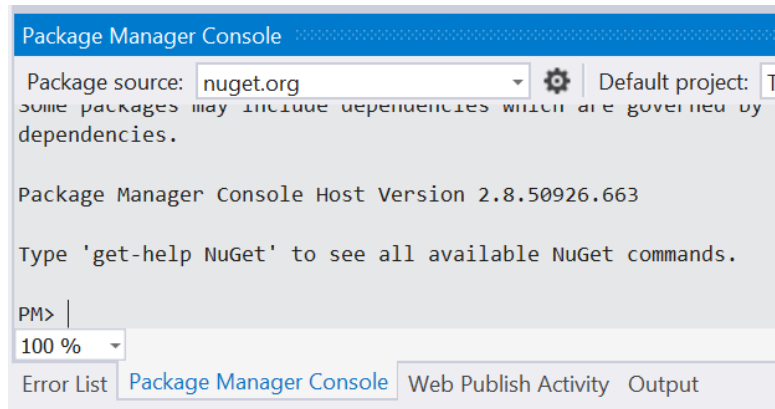
Exercise 2: Installing Dependencies & Packages

This lab will leverage from preexisting libraries associated with HBase connectivity. We will install and reference these libraries/packages using the **NuGet Package Manager**.

- 1) From the **Tools** menu, click **NuGet Package Manager**, and then click **Package Manager Console**. The console panel will open at the bottom of the page.



- 2) The Package Manager Console will open up somewhere your Visual Studio workspace. Wait for the Package Manager Console to initialize. You can tell that it is done when it says "PM>" at the very bottom.



- 3) Use the following commands to install the [HBase.NET SDK](#) package, which is client library used to access HBase clusters, and the [Tweetinvi](#) package, which is used to access the Twitter API.
- a) Install the HBase client: type in the following command into the Package Manager Console.

```
PM> Install-Package Microsoft.HBase.Client
```

Package Manager Console

Package source:

```
PM> Install-Package Microsoft.HBase.Client
```

Success Output:

Package Manager Console

Package source:

Default project:

Installing 'Microsoft.HBase.Client 0.2.2'.

Successfully installed 'Microsoft.HBase.Client 0.2.2'.

Adding 'protobuf-net 2.0.0.668' to TweetSentimentWeb.

Successfully added 'protobuf-net 2.0.0.668' to TweetSentimentWeb.

Adding 'Microsoft.HBase.Client 0.2.2' to TweetSentimentWeb.

Successfully added 'Microsoft.HBase.Client 0.2.2' to TweetSentimentWeb.

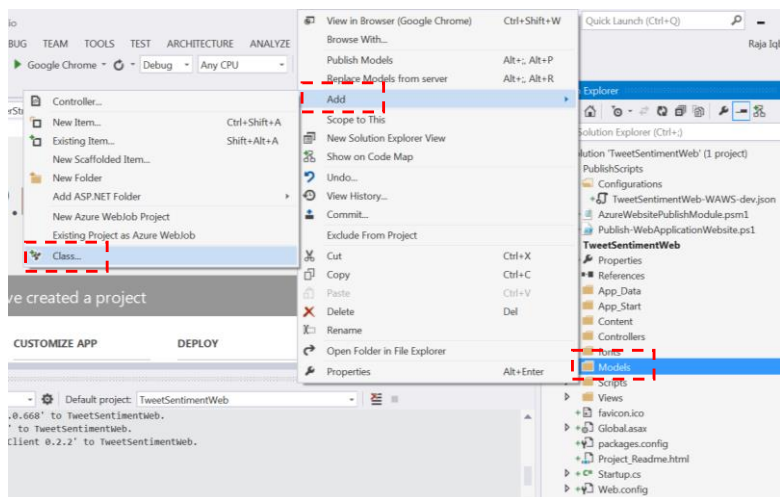
```
PM> |
```


Exercise 3: HBase Reader Class

Since we are not reading from a traditional database, we can't connect to a database like we normally would and specify an ODATA or ODBC driver to connect automatically. Instead we must specify our own HBase writer much like we did in the HBase Twitter Stream Writer app.

1) Adding an HBase Reader class:

- From **Solution Explorer**, expand **TweetSentiment**.
- Right-click **Models**, click **Add**, and then click **Class**.
- In Name, enter **HBaseReader.cs**, and then click **Add**.



- To save time, we have written an HBase reader for you. Visit the following link and copy and paste the code over to your own HBaseReader.cs.
<https://raw.githubusercontent.com/datasciencedojo/TwitterSentimentWeb/master/TweetSentimentWeb/Models/HBaseReader.cs>

2) Referencing your HBase cluster:

- a) There are 4 constants defined at the top of the HBaseWriter class within your HBaseWriter.cs file that will directly link to your HBase cluster, we just need to point them in the correct direction.
 - i) **"CLUSTERNAME"** (line 17): This will be the direct URL connection string to your HBase cluster. Go to your Azure Management Portal (manage.windowsazure.com), click on **HDInsight > YourCluster -> Dashboard**.

The screenshot shows the Microsoft Azure portal interface for an HDInsight cluster named 'myhbasecluster'. The left sidebar contains a navigation menu with icons for various Azure services. The main content area displays the cluster's dashboard, including tabs for DASHBOARD, MONITOR, CONFIGURATION, and SCALE. The dashboard shows a usage bar chart, a 'quick glance' section with status (Running), location (West US), and cluster operating system (Windows Server 2012 R2 Datacenter). A 'linked resources' table lists the storage account 'dojathendestorage'. A red arrow points to the cluster name 'myhbasecluster' in the sidebar. A yellow arrow points from the 'CLUSTERNAME' constant in the code to the 'CLUSTER CONNECTION STRING' field in the 'quick glance' section, which contains the URL 'https://myhbasecluster.azurehdinsight.net'.

```
using org.apache.hadoop.hbase.rest.protobuf.generated;
namespace TweetSentimentWeb.Models
{
    1 reference | 0 authors | 0 changes
    public class HBaseReader
    {
        // For reading Tweet sentiment data from HDInsight HBase
        HBaseClient client;

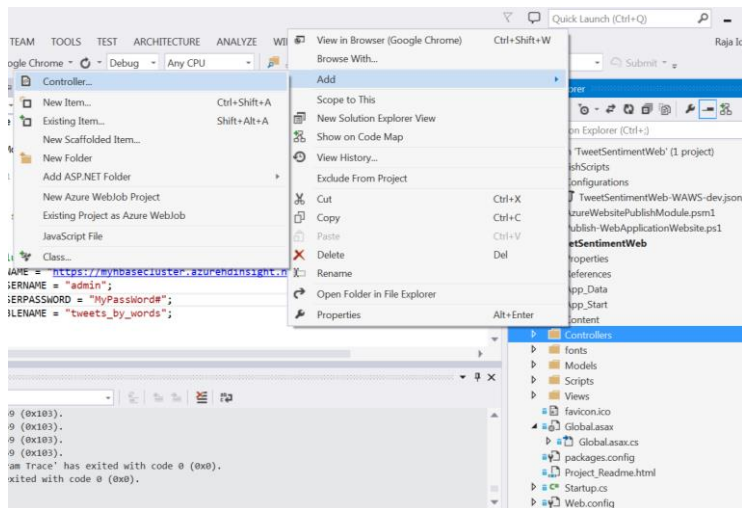
        // HDInsight HBase cluster and HBase table information
        const string CLUSTERNAME = "<HBaseClusterName>";
        const string HADOOPUSERNAME = "admin";
        const string HADOOPUSERPASSWORD = "<HBaseClusterUserPassword>";
        const string HBASETABLENAME = "tweets_by_words";
    }
}
```

- ii) **"HADOOPUSERNAME"**: Leave the username as "admin".
- iii) **"HADOOPUSERPASSWORD"**: Insert the password that you set for the HBase cluster when you provisioned it.
- iv) **"HBASETABLENAME"**: Leave it as "tweets_by_words";

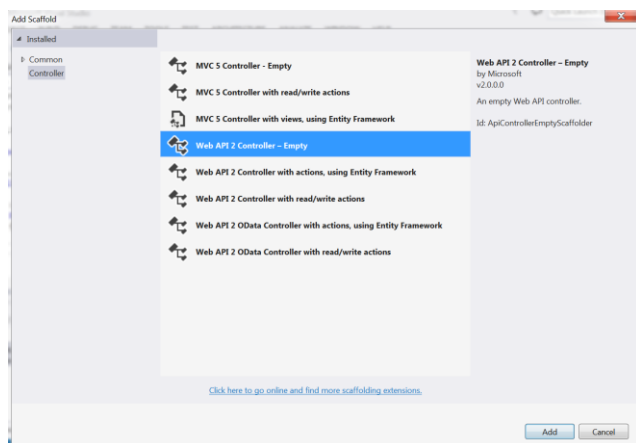
Exercise 4: Tweets Controller

We will now add application logic to our website by adding a TweetsController to the webpage. This controller will deal specifically with opening an HBase cluster connection, and sending queries by utilizing our HBase reader class. Upon a successful query, the results will be sent back to our frontend JavaScript code via AJAX.

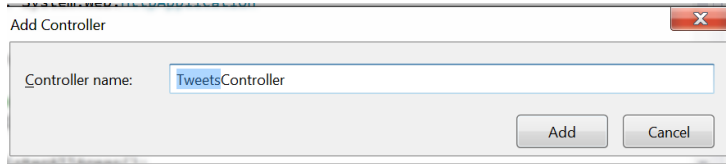
- 1) From **Solution Explorer**, expand **TweetSentimentWeb**.
- 2) Right-click **Controllers**, click **Add**, and then click **Controller**.



- 3) Click **Web API 2 Controller - Empty**, and then click **Add**.



- 4) In Controller name, type **TweetsController**, and then click **Add**.



- 5) From **Solution Explorer**, double-click TweetsController.cs to open the file.

- 6) Modify the file, so it looks like the following:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Net.Http;
using System.Web.Http;

using System.Threading.Tasks;
using TweetSentimentWeb.Models;

namespace TweetSentimentWeb.Controllers
{
    public class TweetsController : ApiController
    {
        HBaseReader hbase = new HBaseReader();

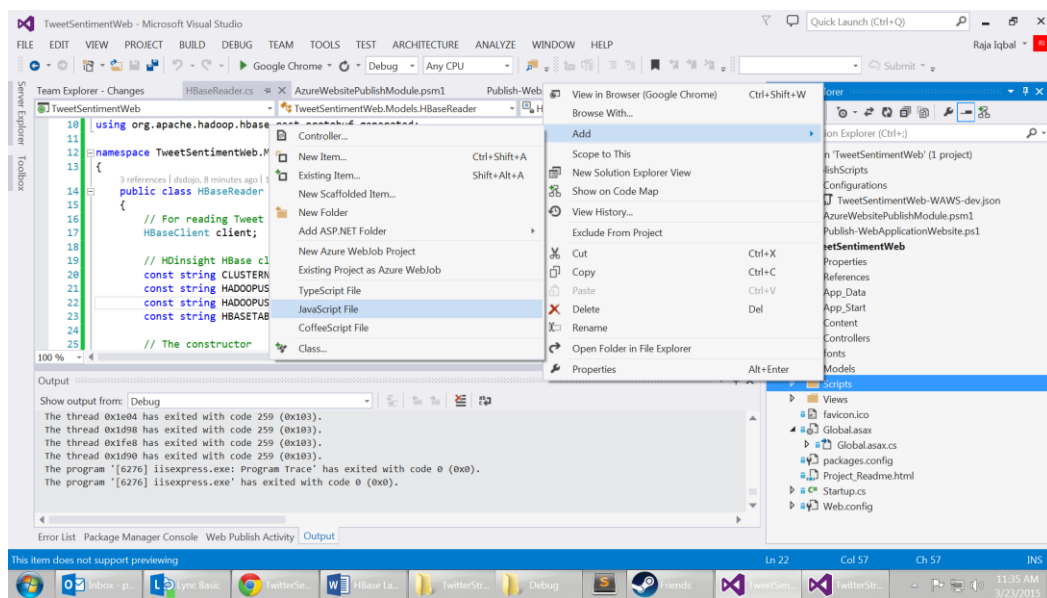
        public async Task<IEnumerable<Tweet>> GetTweetsByQuery(string query)
        {
            return await hbase.QueryTweetsByKeywordAsync(query);
        }
    }
}
```

Exercise 5: Heat Map Library

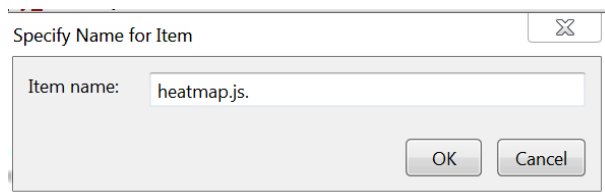
To get Heat Map functionally, we will leverage a JavaScript library written by Alastair Aitchison. The library will use a Bing world map and can plot tweets by longitude/latitude coordinates then color code them by frequencies and magnitude. The library is pretty versatile and can be adapted to other projects such as mapping crime in cities. For information about this library visit the following link:

<https://alastaira.wordpress.com/2011/04/15/bing-maps-ajax-v7-heatmap-library/>

- 1) From **Solution Explorer**, expand **TweetSentimentWeb**.
- 2) Right-click **Scripts**, click **Add**, click **JavaScript File**.



- 3) In Item name, enter **heatmap.js**.

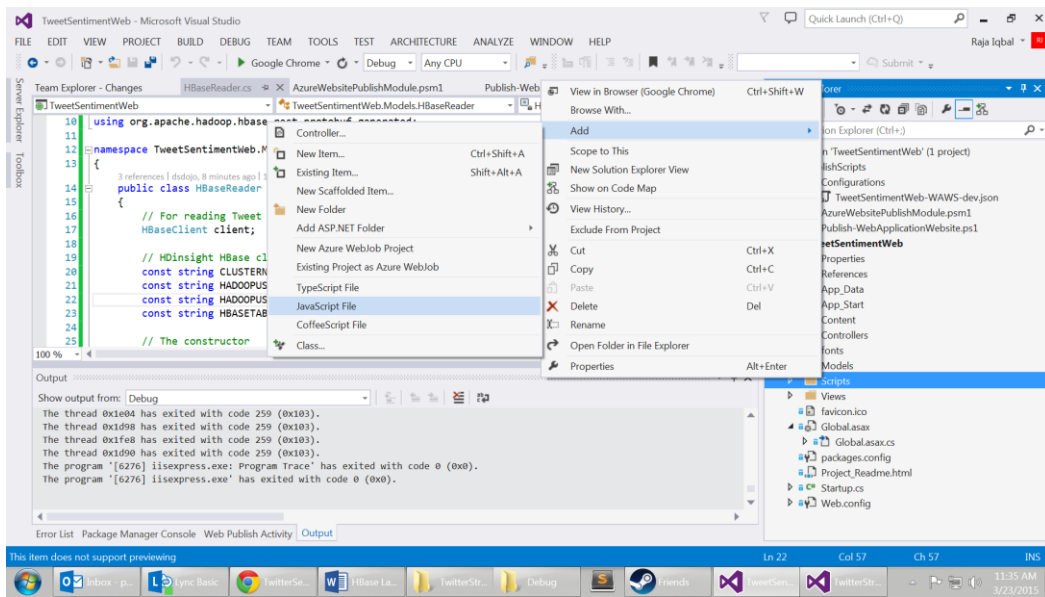


- 4) Visit the following link, copy and paste over this heatmap.js into your heatmap.js
<https://raw.githubusercontent.com/datasciencedojo/TwitterSentimentWeb/master/TweetSentimentWeb/Scripts/heatmap.js>

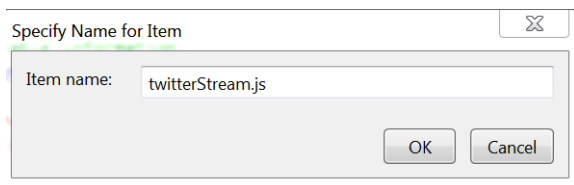
Exercise 6: Website Core Functionalities

We will now setup a file called, "tweetStream.js", which will be the core of our website's functionality. First it'll initialize the prewritten Microsoft Bing world map. It also submits the queries from the front end via AJAX to our controller to send off to our HBase cluster. Once the results from the query are received, tweetStream.js will plot all the tweets by longitude/latitude, then color code data points by tweet counts and sentiment. It does this by using the heatmap.js library that we created in the previous exercise.

- 1) From **Solution Explorer**, expand **TweetSentimentWeb**.
- 2) Right-click **Scripts**, click **Add**, click **JavaScript File**.



- 3) In Item name, enter **twitterStream.js**.



- 4) Visit the following link, paste this twitterStream.js into your twitterStream.js:
<https://raw.githubusercontent.com/datasciencedojo/TwitterSentimentWeb/master/TweetSentimentWeb/Scripts/twitterStream.js>

Exercise 7: Enabling Plugins

Now we will modify the registration of the API routes to make Web API controller work inside of the MVC application.

1) From **Solution Explorer**, expand **TweetSentimentWeb**, and then double-click **Global.asax**.

2) Add the following using statement below "using System.Web.Routing":

```
using System.Web.Http;
```

3) Add the following lines inside the **Application Start()** function:

```
GlobalConfiguration.Configure(WebApiConfig.Register);
```

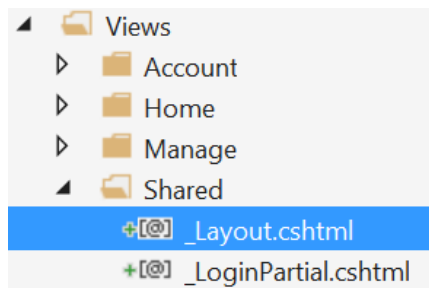
Exercise 8: HTML & CSS

Time to build our homepage using .Net template language. First we must build our HTML page. By default Azure website homepages will have a prepopulated "template" page (Index.html). We must overwrite this page, also we will take advantage of the .Net template language.

1) **_Layout.cshtml**:

This will be our base page without the map. Though we will specify for the template, a special place to insert the map.

a) From **Solution Explorer**, expand **TweetSentimentWeb**, expand **Views**, expand **Shared**, and then double-click **_Layout.cshtml**.



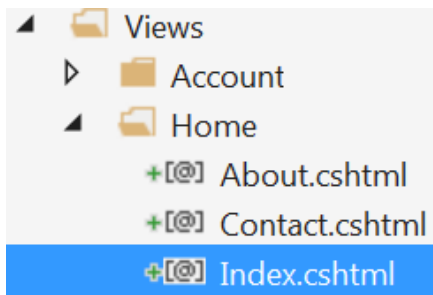
b) Visit the following link for the _Layout.cshtml file, copy over the code in this file and overwrite your _Layout.cshtml file:

https://raw.githubusercontent.com/datasciencedojo/TwitterSentimentWeb/master/TweetSentimentWeb/Views/Shared/_Layout.cshtml

2) Index.cshtml

This is where our heatmap will go. It'll automatically be inserted into the layout.cshtml.

- a) From **Solution Explorer**, expand **TweetSentimentWeb**, expand **Views**, expand **Home**, and then double-click **Index.cshtml**.



- b) Replace the content with the following:

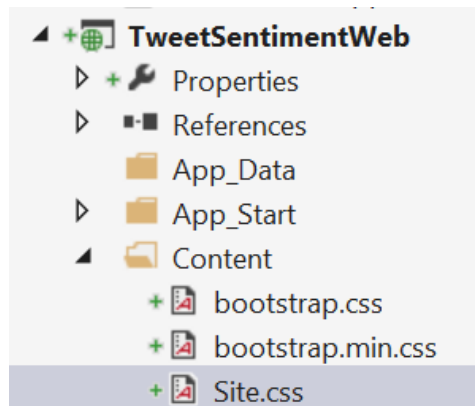
```
@{
    ViewBag.Title = "Tweet Sentiment";
}

<div class="map_container">
    <div id="map_canvas"/>
</div>
```

3) Site.css

Let's add minimal CSS styling for our website to look nicer. Specifically to make the map full screen.

- a) From **Solution Explorer**, expand **TweetSentimentWeb**, expand **Content**, and then double-click **Site.css**.



b) Append site.css with the following code:

```
/* make container, and thus map, 100% width */
.map_container {
  width: 100%;
  height: 100%;
}

#map_canvas{
  height:100%;
}

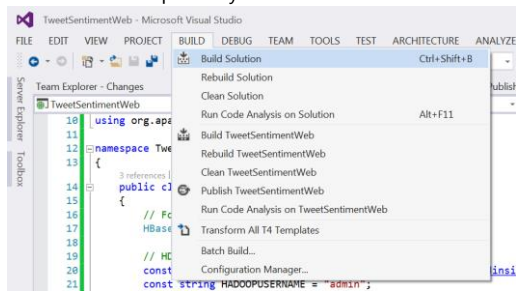
#tweets{
  position: absolute;
  top: 60px;
  left: 75px;
  z-index:1000;
  font-size: 30px;
}
```

Lab 6: Web Deployment

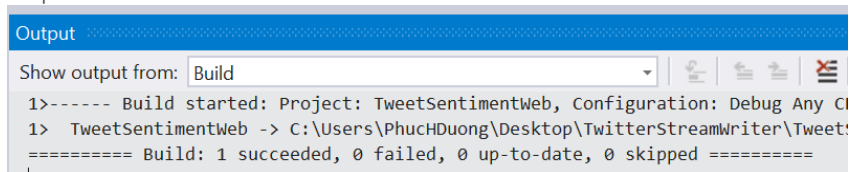
We are now able to deploy our website.

Exercise 1: Testing Your Webpage

- 1) **Build your solution** to ensure there are no errors. At the top navigation bar, there should be a build button. Select to "Build Solution", this will compile all your packages together into one neat packages and ensure that there are no compatibility issues.

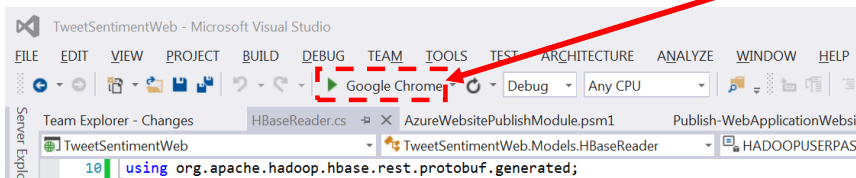


Output:



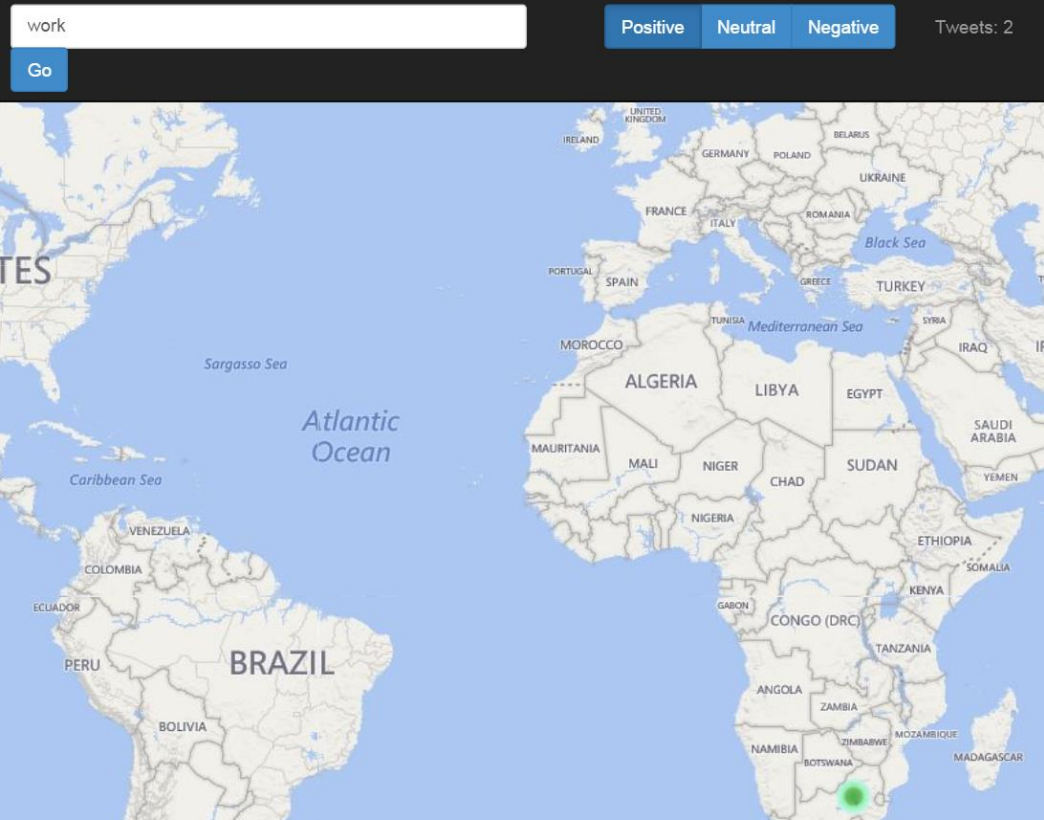
- 2) **Run Your Webserver**

We will now launch a local webserver and test our webpage. Click on the play button.

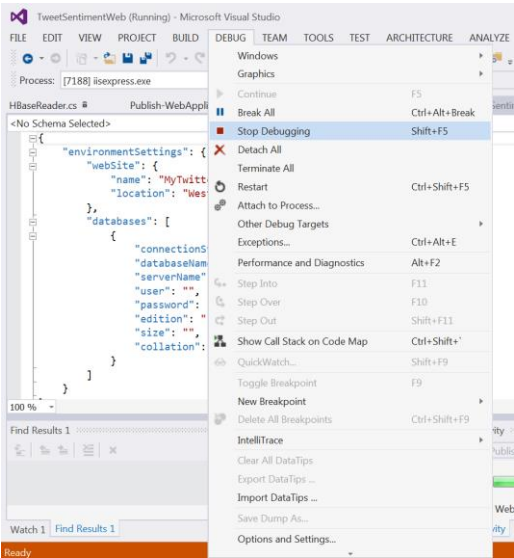


3) **Test Queries**

Assuming you've left your HBase Twitter Stream app on to collect data, you should very easily be able to query and see the results. In the below example, the word "work" was queried against the tweets repository in the HBase cluster under "positive" sentiments, and two tweets came back and were plotted in South Africa. Popular words on twitter include "cheese", "iPhone", or "work".



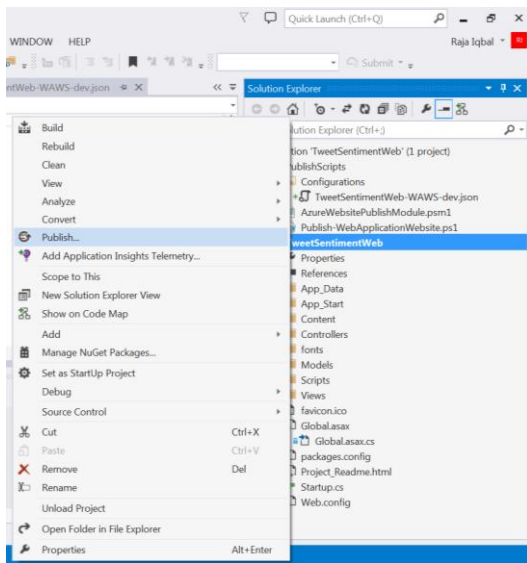
4) Stop Debug and Turn Off Webserver



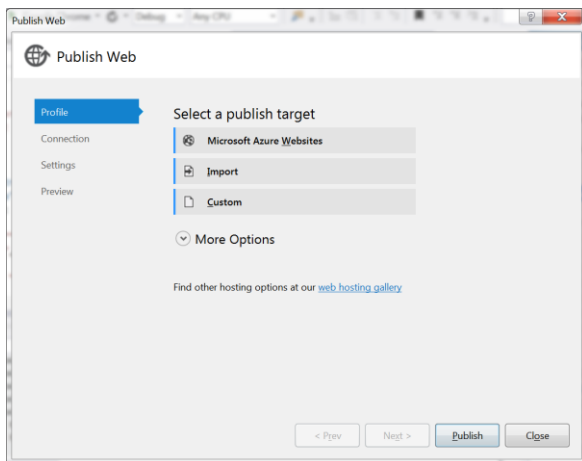
Exercise 2: Publishing Your Webpage

Now that we've confirmed that our website works, we can now publish it to a live web domain and share it with the world. The following exercise will show you how to publish to an Azure website, but if you know how to send via FFTP connection, you may send it to whatever webhost you want. Just make sure that the webhost supports .NET web servers.

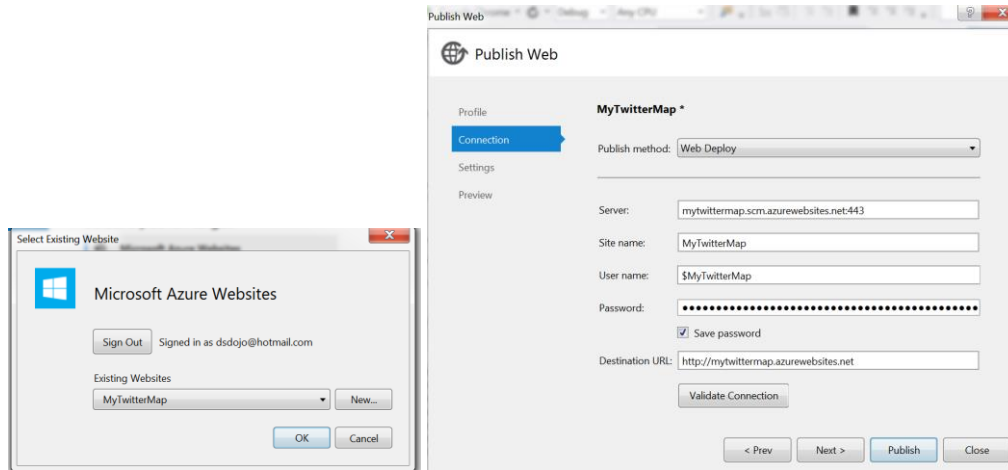
- 1) Right click on your WebApp within your the Solution Explorer. In the example below the WebApp is named "TweetSentimentWeb". Click on Publish



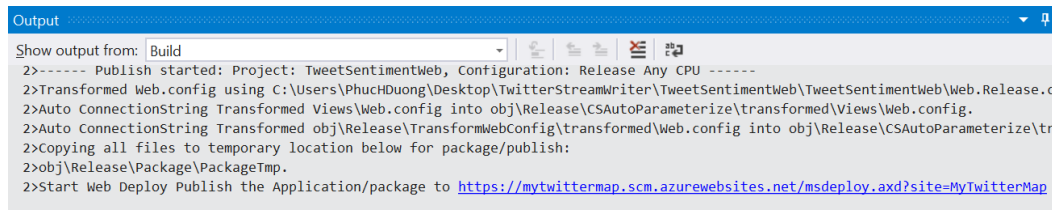
- 2) Select Microsoft Azure Websites



- 3) Sign into your Azure account. Select your website domain. Make a new domain if you do not have one. It should populate your publish credentials. Click publish.



Output (start of upload):



- 4) You can now share this app with others:



- 5) Source Control: If you're happy with your touches, it is recommended that you submit your webpage to GitHub. Be careful not to submit with your passwords to a public Git repository.

Warning: Please do not leave your HBase cluster own. Turn it off when you're done with it or else you will eat away use all of your free-subscription funds.