**CSCIE-90 Cloud ComputingAssignment 10**

Submitted by Pamela Lim (60889651)

Problem 1(a)

Architectural diagram of a system using queues, blob, web roles and worker roles to prevent two or more customers from booking the same seat on the same flight. Basically, when a customer comes in to book a seat that is available, the seat is locked for 15 seconds, enough for the applications to process. During this time lapse, if there is another customer who comes and book the seat, it will be shown as unavailable.



Problem 1(b)

To notify customers of available airline seats using Queue-Centric-Workflow, we need a **Web Role**and a **Worker Role**, a Reliable **Azure Storage Queue** and a Durable/Persistent Storage in the **Blobs and Tables**.

1. The Web Role receives user requests to view available seats and places the request on the Secured Queue.
2. The Worker Role pulls the request from the Secured Queue for processing.
3. The Worker Role checks if the message is valid.
4. If the message is valid, it will go on and process it. Otherwise, it will check if it is a poison message.
5. The Database returns the availability of seats.
6. If the message has been dequeued N times. A message that has been dequeued more than a specified N number of times is considered a poison message. It is then added to the error message.
7. The outcome of the inquiry is written as a Callback message on the Secured Queue. The request message is deleted.

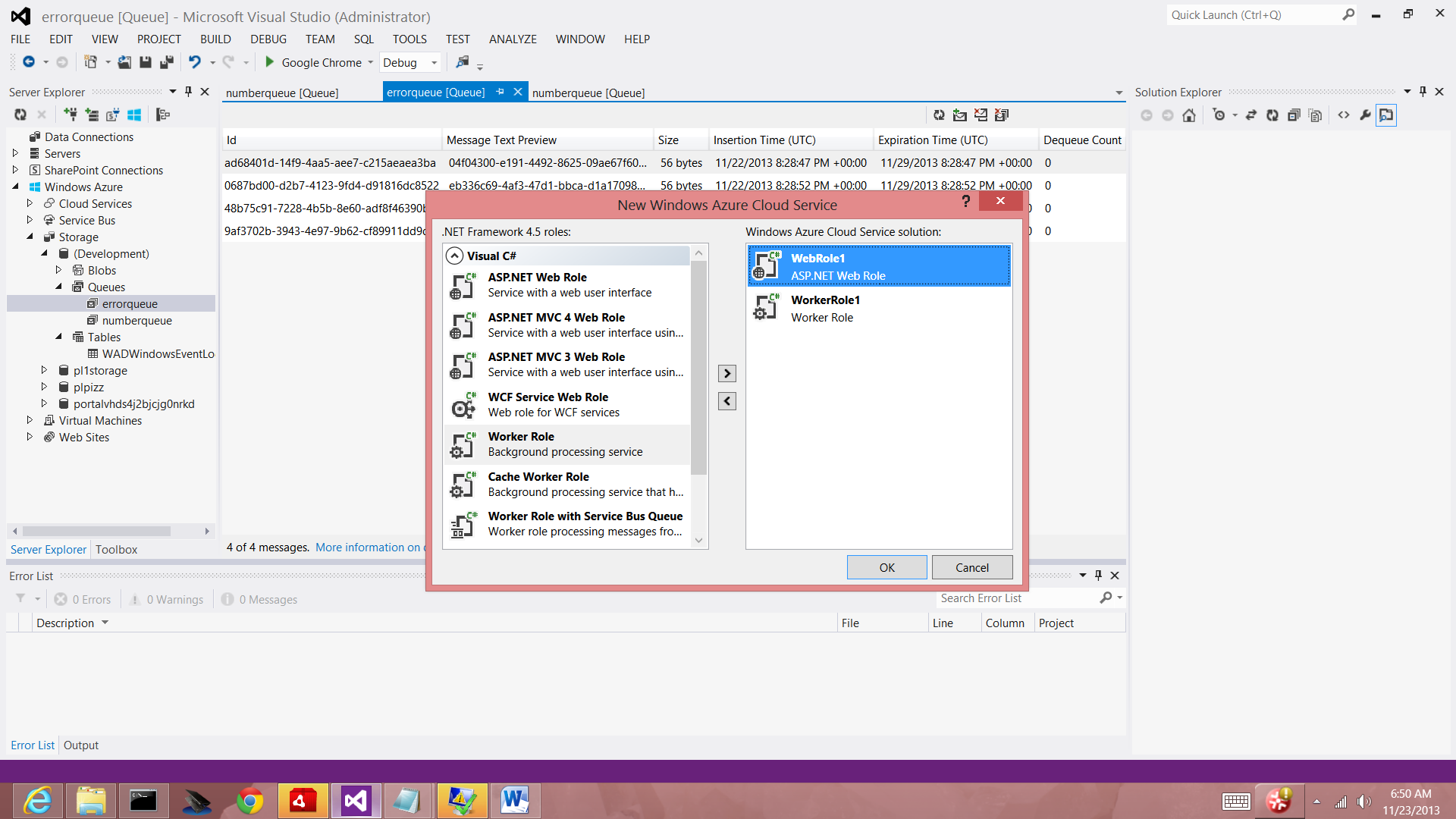
A few notes:

1. The web, worker roles and queues can be replicated and load balanced to scale in accordance to peak periods.
2. When there are more than one worker role used, the request message is hidden from other worker roles so that not more than one worker role in working on a single request.
3. Time consuming working such as accessing the database is done asynchronously, hence reducing the waiting time for users.
4. The design ensures idempotent operation, meaning end result is the same no matter how many times the request is done.
5. The above design also expects poison messages, handling error for non-transient reason. It uses a dequeue count property to ensure the worker role is not ‘overprocessing’ invalid messages. Such poison messages are put on the error or dead letter queue for review.

Problem 2

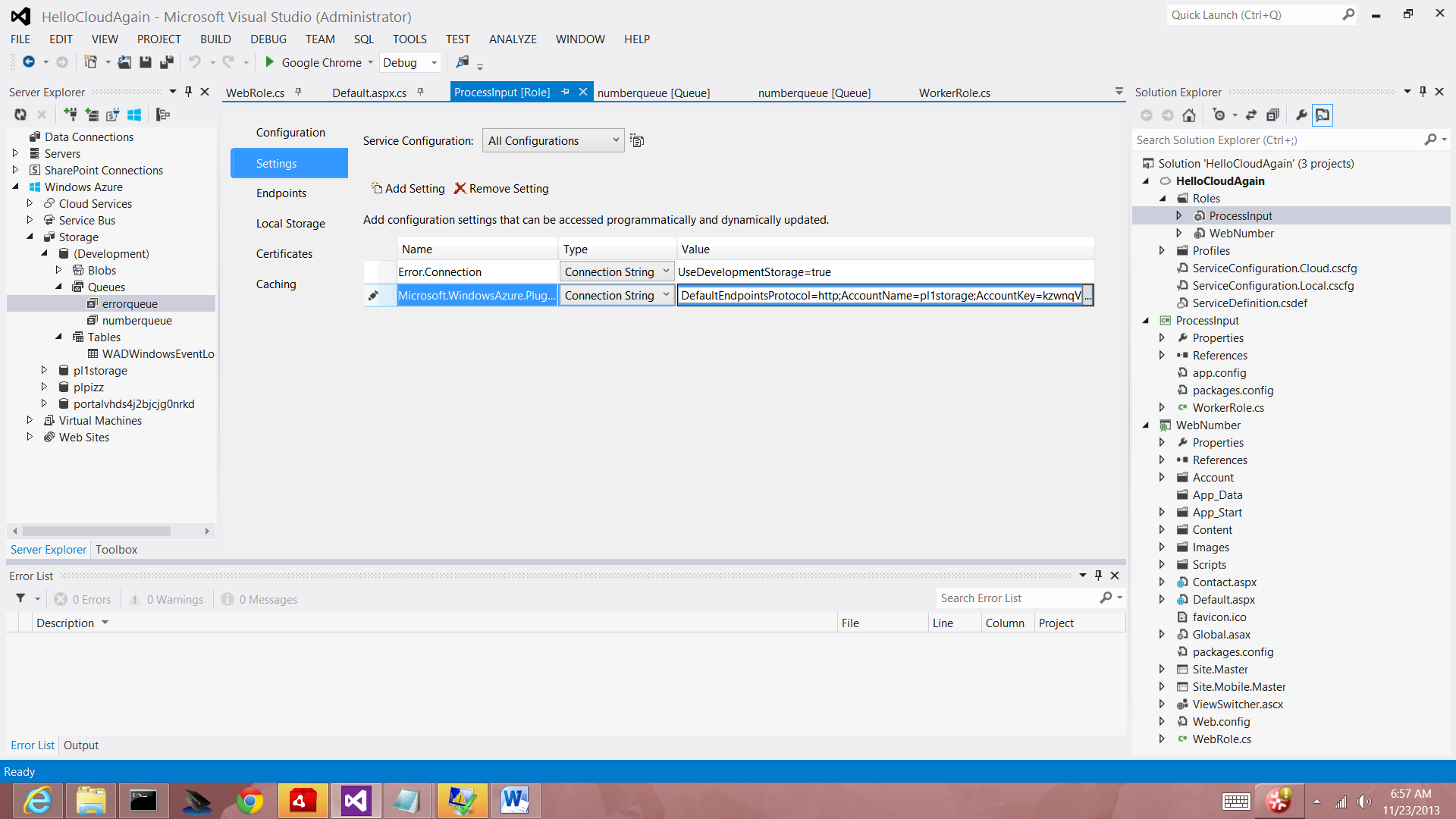
I used Visual Studio with the SDK to do the problem.

Create a new project within Visual Studio with a web role and a worker role.



Setup a Windows Azure Storage Connection String

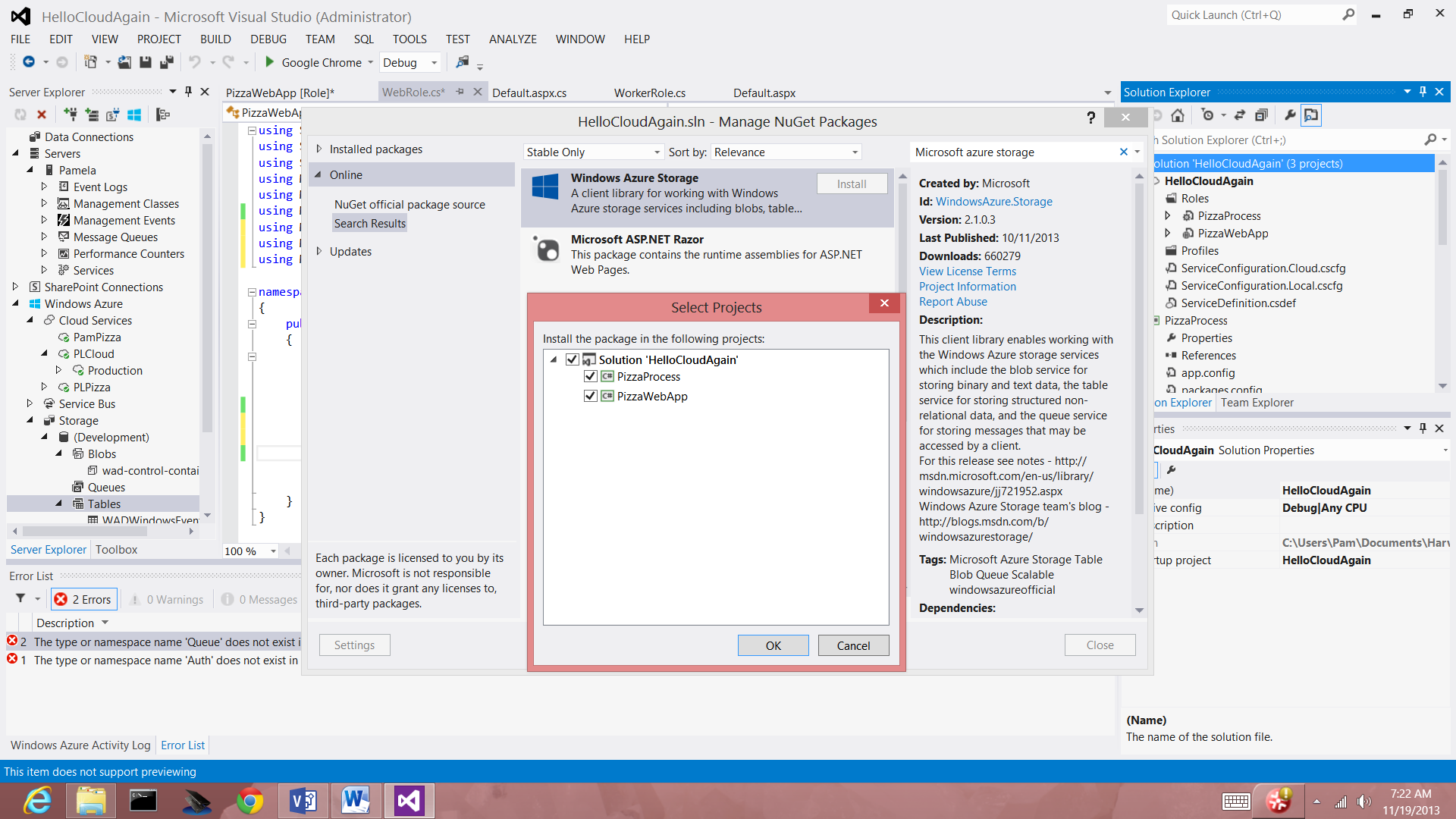
1. Within the Solution Explorer of Visual Studio, in the **Roles** folder of the, right-click on the web role or worker role and click **Properties**.
2. Click the **Settings** tab and press the **Add Setting** button.
3. I modified the setting on the current connection, to link to my Windows Azure account:



Obtaining the assembly to use the queues in the programs

I used NuGet to obtain the Microsoft.WindowsAzure.Storage.dll assembly. Right-click on **Solution Explorer** and choose **Manage NuGet Packages**. Search online for "WindowsAzure.Storage" and click **Install** to install the Windows Azure Storage package and dependencies.

Add the right services required for queue management at Manage Nuget Packages for Solution (right click on Solution on Solution Server, and then search for ‘Microsoft Azure Storage’ and install that into both the **Web Role** and the **Worker Role**:



Namespace declarations

I added the following code namespace declarations to the top of both the workerrole.cs and the webrole.cs C# file so as to programmatically access Windows Azure Storage:

using Microsoft.WindowsAzure.Storage;

using Microsoft.WindowsAzure.Storage.Auth;

using Microsoft.WindowsAzure.Storage.Queue;

**WEBROLE.CS**

The codes for creating a queue and populating it with the integer as required by the problem, is done in the webrole.cs. I created a transaction id before inserting the integer from the user input:

// Retrieve storage account from connection string.

CloudStorageAccount storageAccount =CloudStorageAccount.Parse(

CloudConfigurationManager.GetSetting("Microsoft.WindowsAzure.Plugins.Diagnostics.ConnectionString"));

// Create the queue client.

CloudQueueClient queueClient = storageAccount.CreateCloudQueueClient();

// Retrieve a reference to a queue.

CloudQueue queue = queueClient.GetQueueReference("numberqueue");

// Create the queue if it doesn't already exist.

queue.CreateIfNotExists();

// Create a message and add it to the queue.

CloudQueueMessage message =new CloudQueueMessage(string.Format("{0} : {1}", Guid.NewGuid(), number));

queue.AddMessage(message);

// Retrieve storage account from connection string.

CloudStorageAccount storageAccount =CloudStorageAccount.Parse(

CloudConfigurationManager.GetSetting("StorageConnectionString"));

// Create the queue client.

CloudQueueClient queueClient = storageAccount.CreateCloudQueueClient();

// Retrieve a reference to a queue.

CloudQueue queue = queueClient.GetQueueReference("myqueue");

// Get the message from the queue and update the message contents.

CloudQueueMessage message = queue.GetMessage();

message.SetMessageContent("Updated contents.");

queue.UpdateMessage(message,

TimeSpan.FromSeconds(0.0),// Make it visible immediately.

MessageUpdateFields.Content|MessageUpdateFields.Visibility);

**WORKERROLE.CS**

The worker role accesses the queue and reads the message, and processes it. While processing it, the message will be hidden for 10 seconds. If it is successful, the message is written to an output file and deleted. Otherwise, it will reappear on the queue. If the message has been dequeued five times, it will be treated as a poison message and deleted from the queue. A new error is written on the error queue.

The codes are as follows:

// Retrieve a reference to a queue

var queue = queueClient.GetQueueReference("numberqueue");

// Read the next message

while (true)

{

CloudQueueMessage qMessage = null;

do

{

qMessage = queue.GetMessage(TimeSpan.FromSeconds(10));

if (qMessage != null)

{

string details = qMessage.AsString;

if (processmessage(details))

queue.DeleteMessage(qMessage); //delete successful message

else

{

if (qMessage.DequeueCount >= 5) //try 5 times b4 classifying poison message

{

// Retrieve a reference to a error queue

CloudQueue q = queueClient.GetQueueReference("errorqueue");

// Create the queue if it doesn't already exist

q.CreateIfNotExists();

// Create a message and add it to the error queue.

q.AddMessage(qMessage); // add error message

queue.DeleteMessage(qMessage); //delete poison message

}

else

{

queue.UpdateMessage(qMessage,

TimeSpan.FromSeconds(0.0), // Make it visible immediately.

MessageUpdateFields.Content | MessageUpdateFields.Visibility);

}

}

}

}

while (qMessage != null);

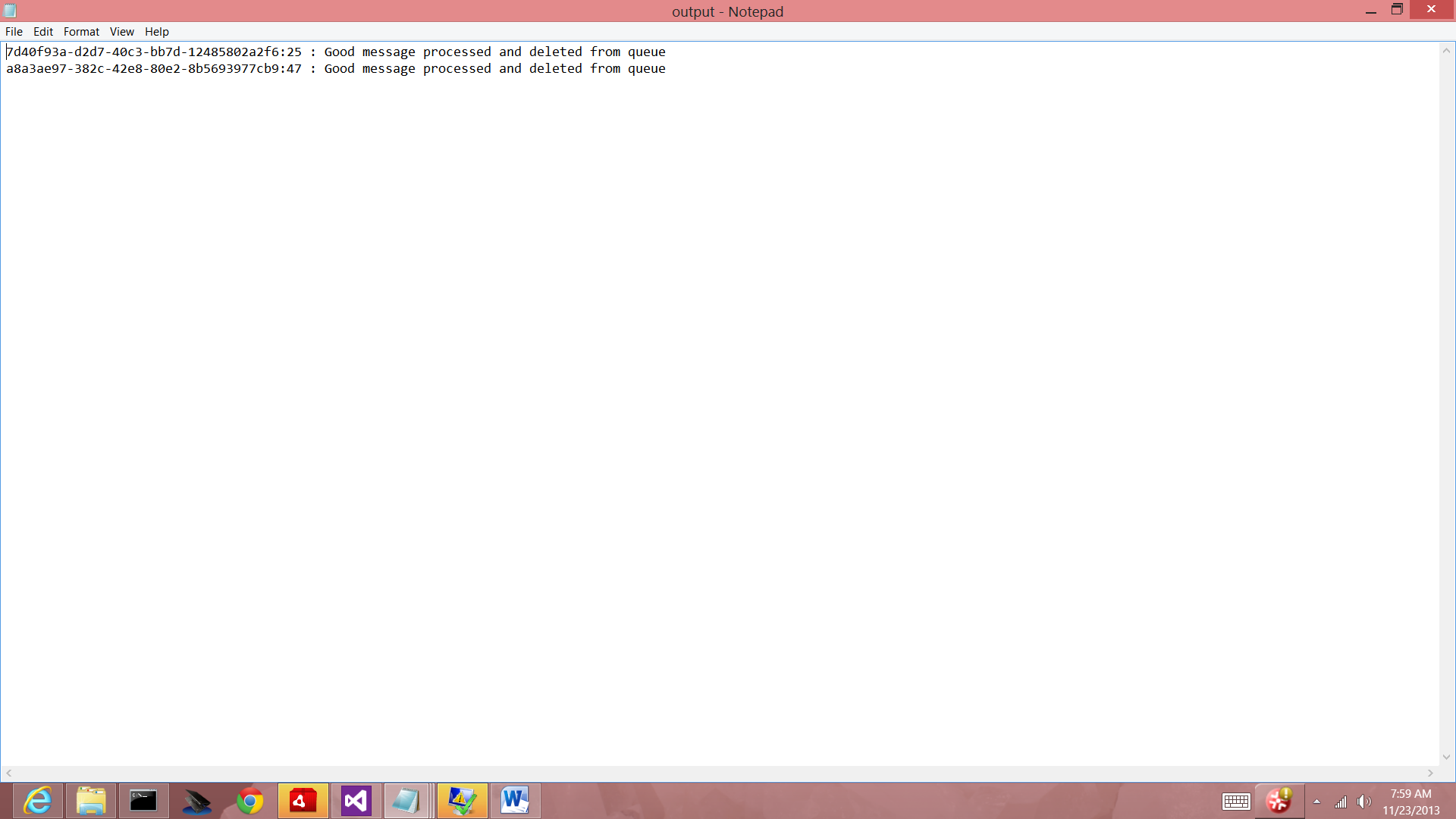
Thread.Sleep(1000);

Trace.TraceInformation("Working", "Information");

}

}

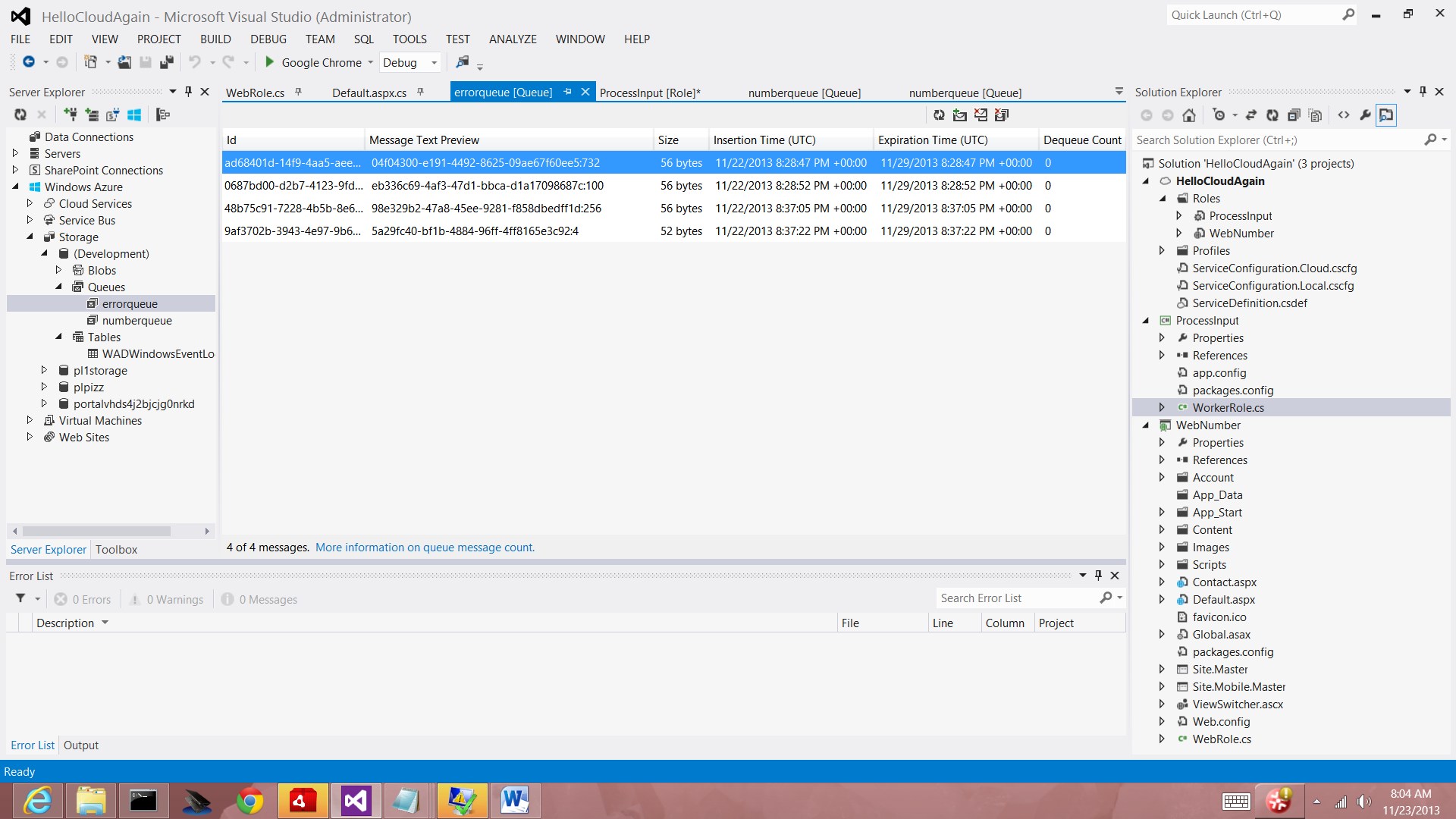
The output from the output file is as follows:



User Input:> 1 and <100

Transaction ID

Error queue:



Messages with input not within 10 and 99.

Problem 3

1. The Cloud Service encapsulates the VMs as a container for the VMs.  You can put multiple VMs into a single cloud service, so they can communicate on their internal IP Addresses instead of the public IPs.
2. Yes, according to documentation, two VMs can be created behind the same DNS name. We can then load balance them or put them into an availability set, as documented in

<http://www.windowsazure.com/en-us/manage/windows/common-tasks/how-to-load-balance-virtual-machines/>

<http://www.windowsazure.com/en-us/manage/windows/common-tasks/manage-vm-availability/>

1. I compared the Virtual Machines, Cloud Service, Storage Account, Instances and Images to AWS and there are as follows:<http://blogs.technet.com/b/matthewms/archive/2013/06/13/technet-radio-part-4-infrastructure-as-a-service-windows-azure-vs-amazon-amp-rackspace.aspx>