Corso di Sistemi Distribuiti

Corso di Laurea Magistrale in Ingegneria Informatica A.A. 2014/2015

DIMES - Università degli Studi della Calabria

DEVELOPMENT SERVICES ON WINDOWS AZURE

Ing. Loris Belcastro

Microsoft Azure emulator

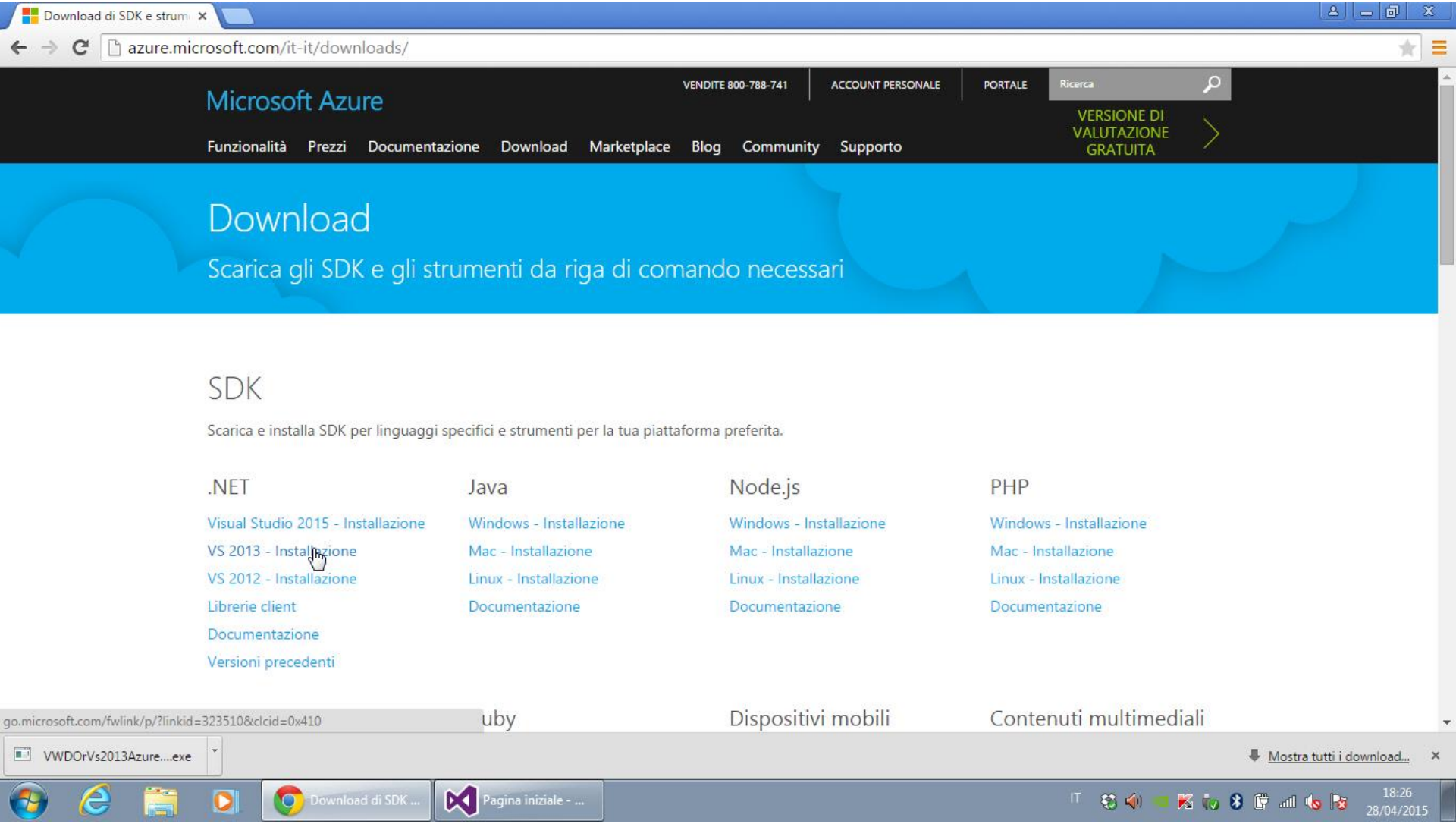
- The Azure SDK includes the Microsoft Azure compute emulator.
- The compute emulator is a local emulator of Azure, so that you can build and test your application before you deploy it.
- To use the compute emulator, you must download and install the Azure SDK.
  - Download it at https://www.microsoft.com/windowsazure/sdk/
- Microsoft .NET Framework 3.5 SP1 or .NET Framework 4.
- Microsoft SQL Server™ Express 2005 or SQL Server 2008 R2 Express for running applications that use the storage emulator.
- If you are intending to use a web role in your application, you must enable Internet Information Services (IIS) 7.0 with ASP.NET.

Useful link:
https://msdn.microsoft.com/en-us/library/azure/ff687127.aspx
https://msdn.microsoft.com/en-us/library/azure/hh403990.aspx

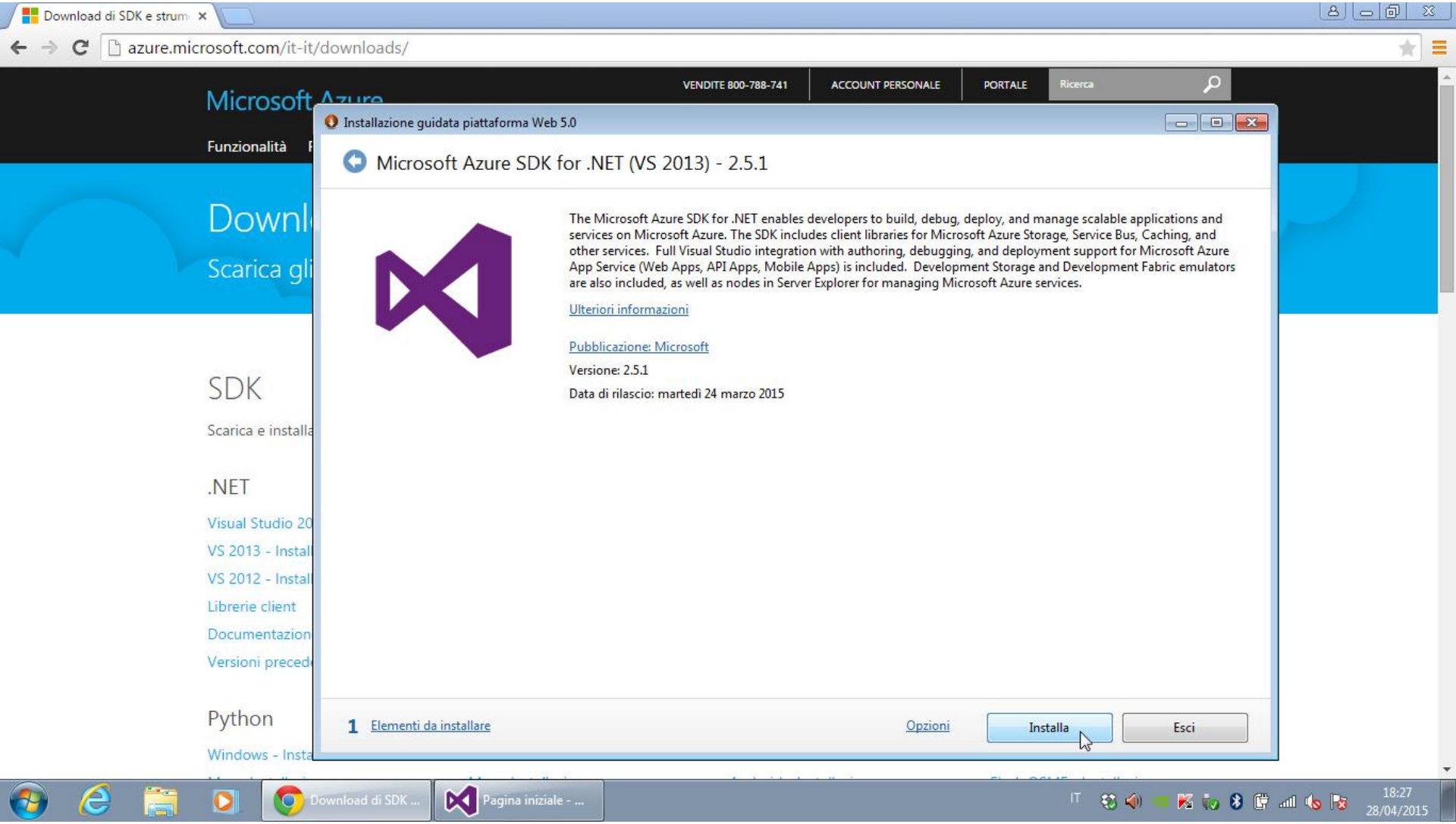
Step by step

- Install Visual Studio 2013 or 2015
- Install Azure SDK
- Launch Azure and install additional packages (if required).
- Create an Azure cloud service
- Build, run and debug your application locally.
- Publish your cloud code to Azure (if required).

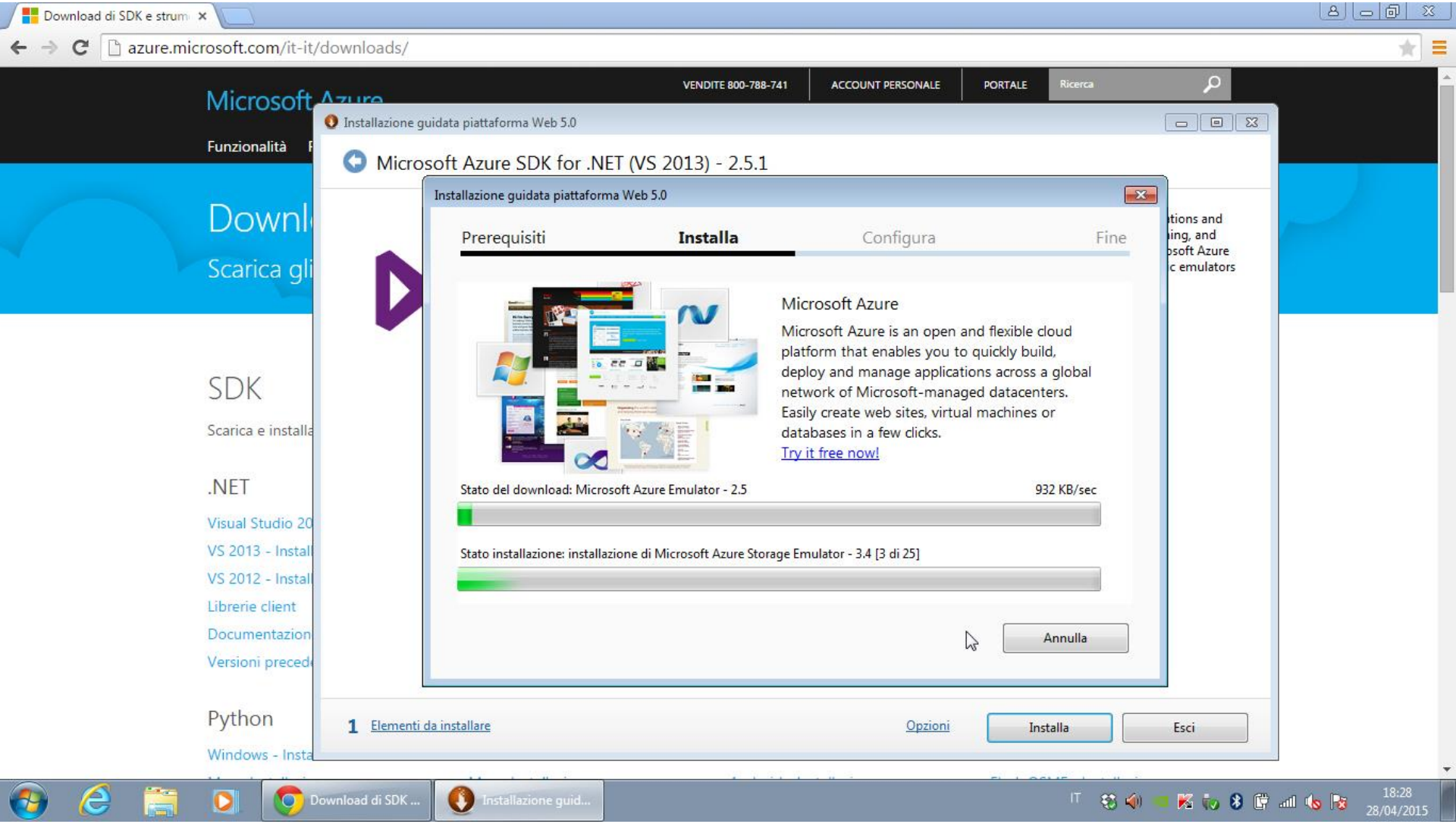
Azure SDK installation



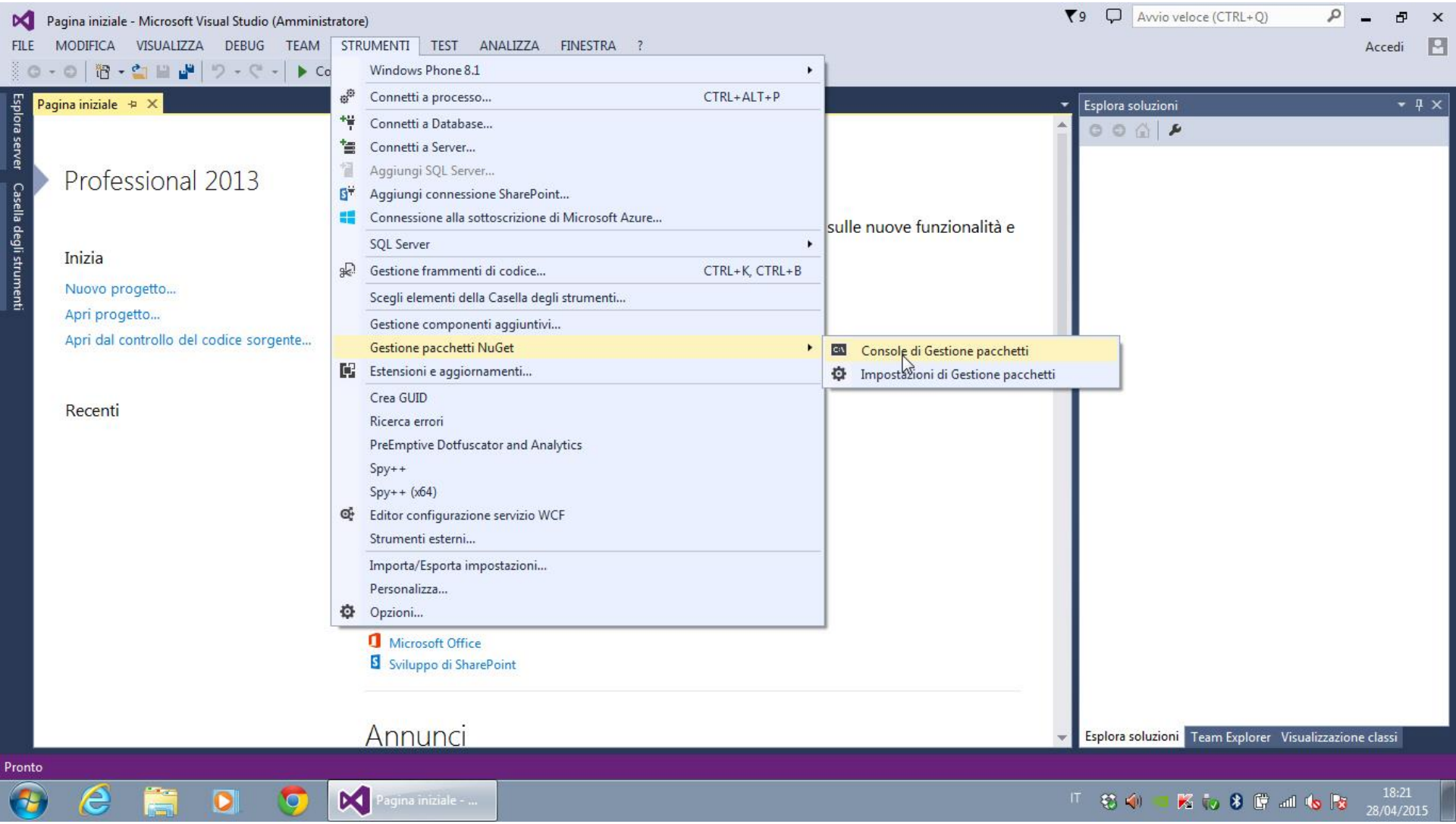
Azure SDK installation



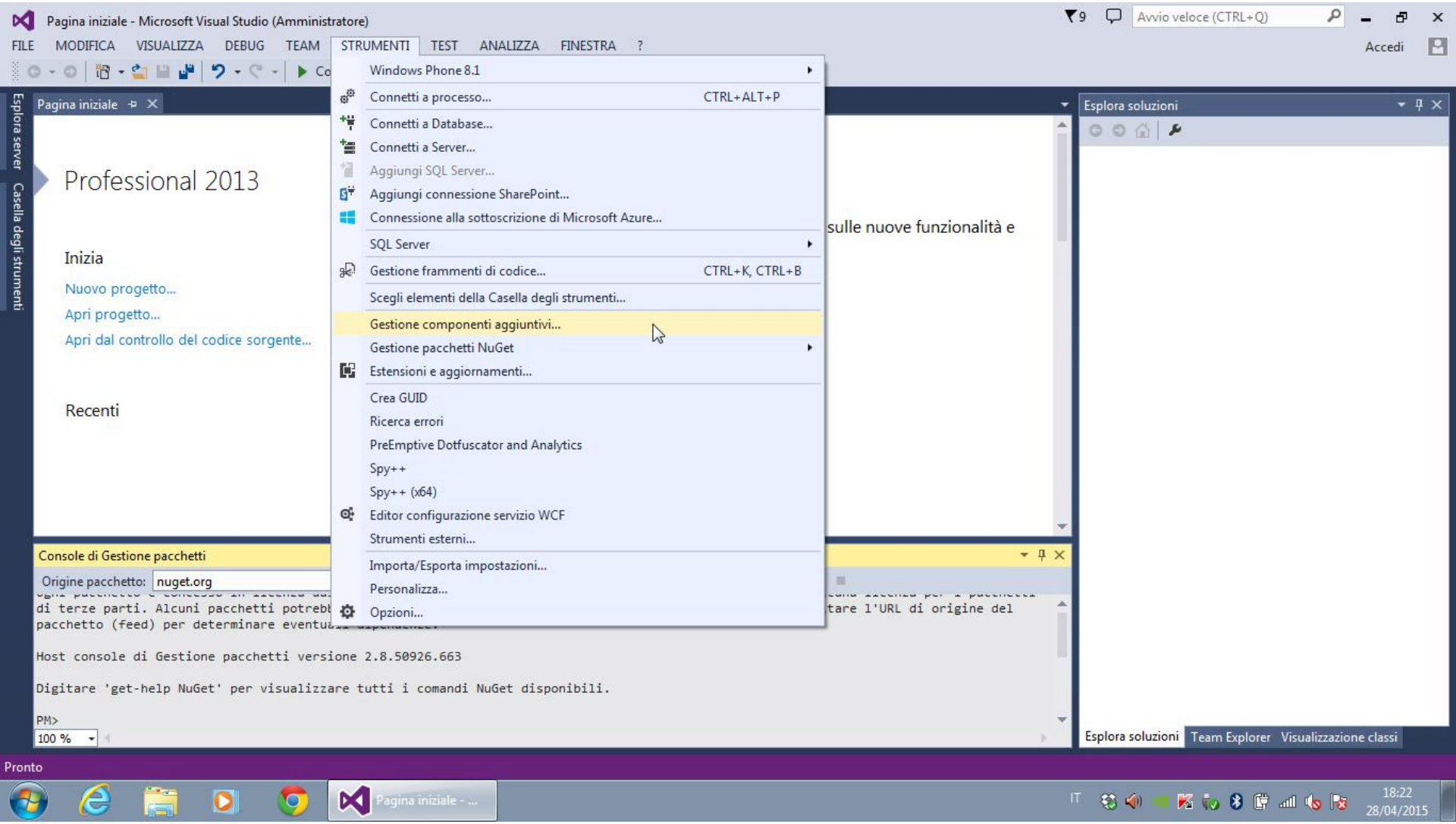
Azure SDK installation



Visual Studio additional components



Visual Studio additional components



Summary

- Hello World example
- Working with Queues (RdStorage)
- Working with Queues (RdStorage-MultipleMessages)

All the code has been tested by exploiting:

- MS Visual Studio 2013
- Windows Azure Compute Emulator – Version 3.4.0.0

Summary

- Hello World example
- Working with Queues (RdStorage)
- Working with Queues (RdStorage-MultipleMessages)

Hello World example

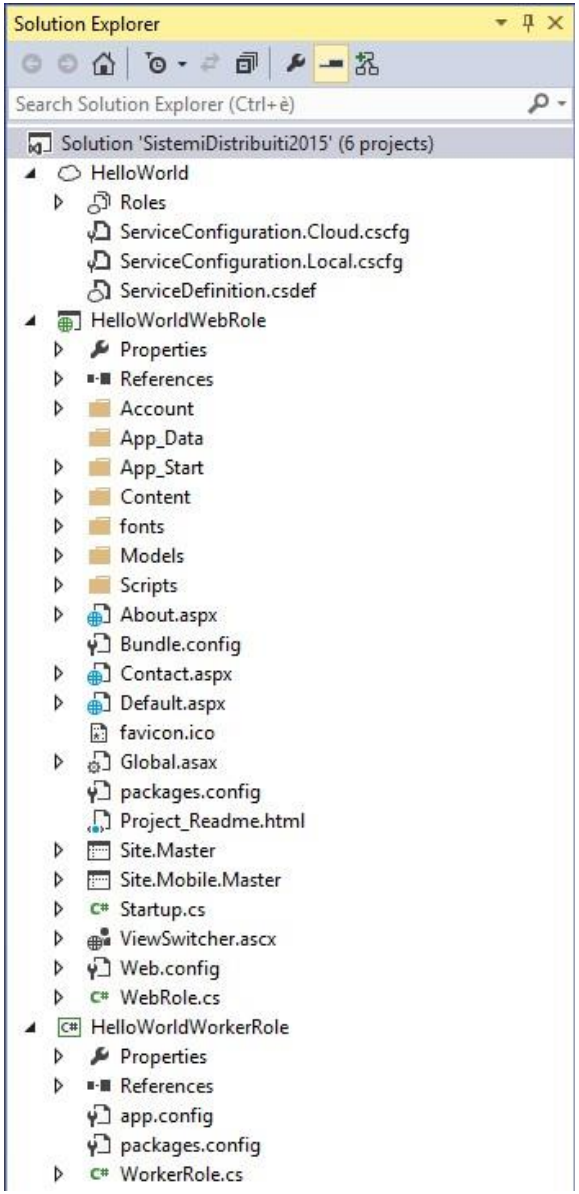
Create a simple Web ‘HelloWorld’ application

- Step 1: from Visual Studio create a new project Azure Cloud Service
- Step 2: add a Web Role ASP.NET and a Worker Role
- Step 3: run the application

Code snippets

The solution created has 3 projects related to the Hello World example:

- **HelloWorld:** it includes the service definition file, ServiceDefinition.csdef, which contains metadata (i.e., which roles are used, their trust level, etc.).
- **HelloWorld_WebRole:** is a standard ASP.NET Web Application project template, with additional class to manage the initialization, starting, and stopping.
- **HelloWorld_WorkerRole:** it contains classes for the worker logic.



ASP.NET vs ASP.NET MVC

based on ASP.NET (Active Server Page), ASP.NET MVC allows software developers to build a web application as a composition of three roles: Model, View and Controller.

- A model represents the state of a particular aspect of the application.
- A controller handles interactions and updates the model to reflect a change in state of the application, and then passes information to the view.
- A view accepts necessary information from the controller and renders a user interface to display that information.

Hello World example

- Demo in classroom
- Question: any interaction with the worker ?

Summary

- Hello World example
- Working with Queues (RdStorage)
- Working with Queues (RdStorage-MultipleMessages)

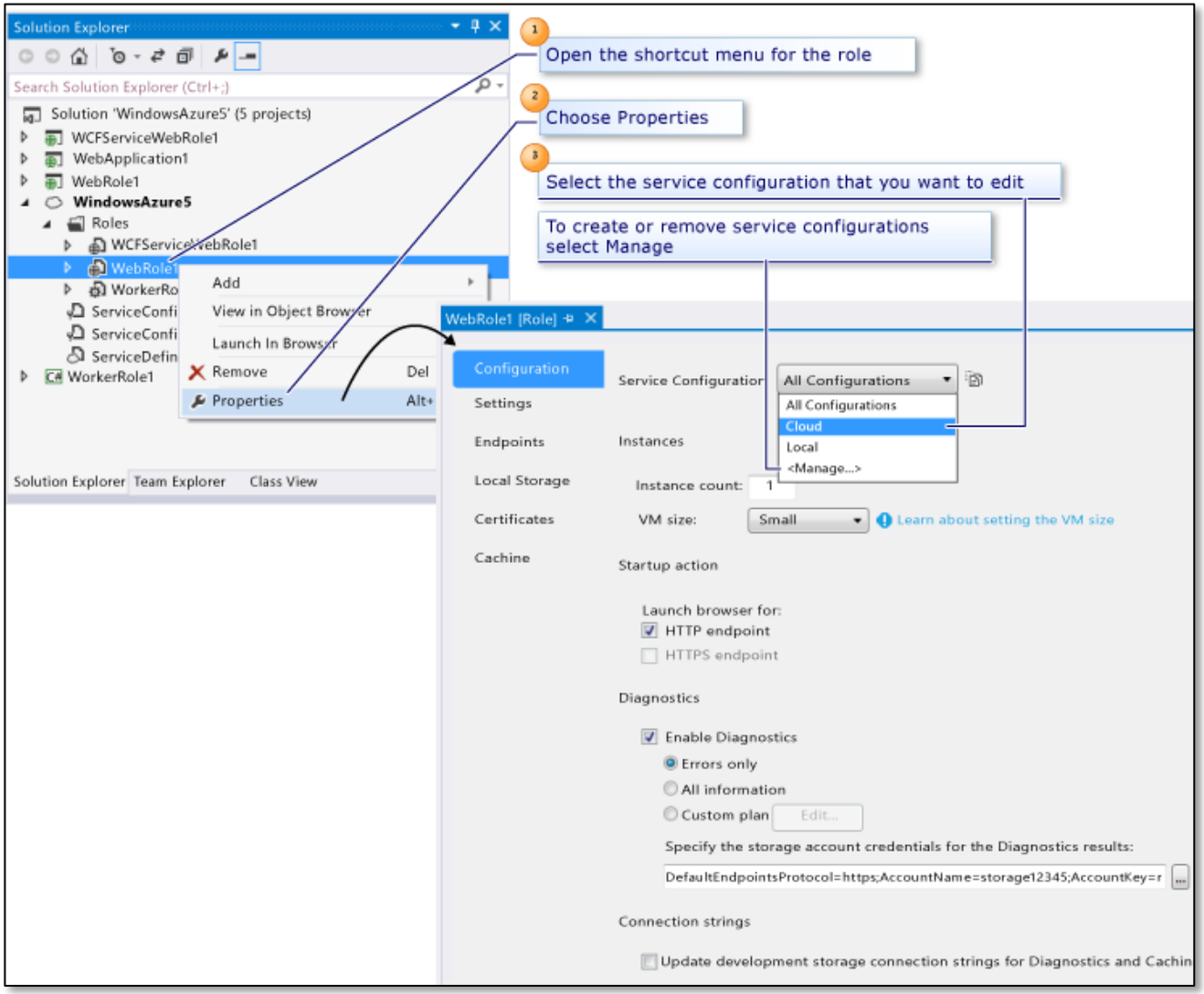
Working with Queues (Storage)

Create a simple Web application to send messages to a Windows Azure queue

- A Worker role in the solution retrieves the messages and writes them to the compute emulator log.

- Queue service is a way to send messages between front-end roles and worker roles.
  - A queue can contain an unlimited number of messages, each of which can be up to 64 KB in size
  - Messages are pushed to the end of the queue and popped from the front of the queue.

How to configure Roles for an Azure Cloud Service



More info at https://msdn.microsoft.com/en-us/library/azure/hh369931.aspx

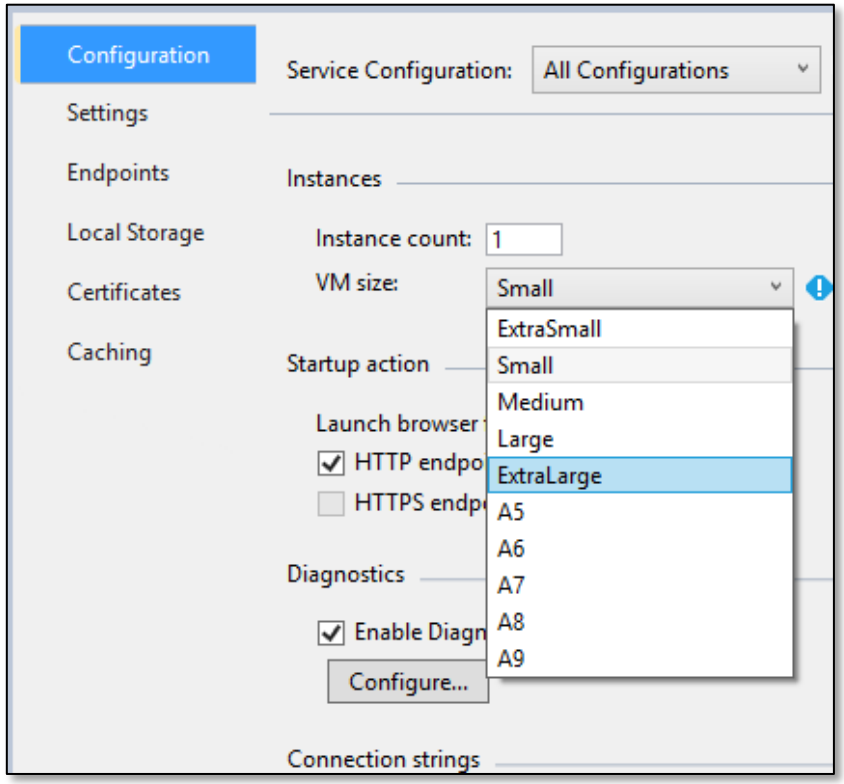
How to change the number of instances for a role

1. Choose the Configuration tab.

2. In the Service Configuration list, choose the service configuration that you want to update.
   - ✓ Note: you can set the instance count for a specific service configuration or for all service configurations.

3. In the Instance count text box, type the number of instances that you want to start for this role.
   - ✓ Note: each instance is run on a separate virtual machine when you publish your cloud service to Azure.

4. Choose the Save icon in the toolbar to save these changes to the service configuration file.



More info at https://msdn.microsoft.com/en-us/library/azure/hh369931.aspx

How to manage connection strings for storage accounts

- You can add, remove or modify connection strings for your service configurations.
- For example, a local connection string for a local service configuration has a value of *UseDevelopmentStorage=true*.
- You might also want to configure a cloud service configuration that uses a storage account in Azure.

1. Choose the Settings tab.
2. In the Service Configuration list, choose the service configuration that you want to update.
   - ✓ Note: you can update connection strings for a specific service configuration, but if you need to add or delete a connection string you must select All Configurations.
3. To add a connection string, choose the Add Setting button -> A new entry is added to the list.
4. In the field Name text box, type the name for the connection string (e.g. *DataConnectionString*).
5. In the Type drop-down list, choose Connection String.
6. To change the value for the connection string, choose the ellipsis button. -> The Create Storage Connection String dialog box is displayed.
   - ✓ To use the local storage account emulator, choose Microsoft Azure storage emulator.
   - ✓ To use a storage account in Azure, choose the Your subscription radio button, and select the desired storage account.
7. Choose the Save icon.

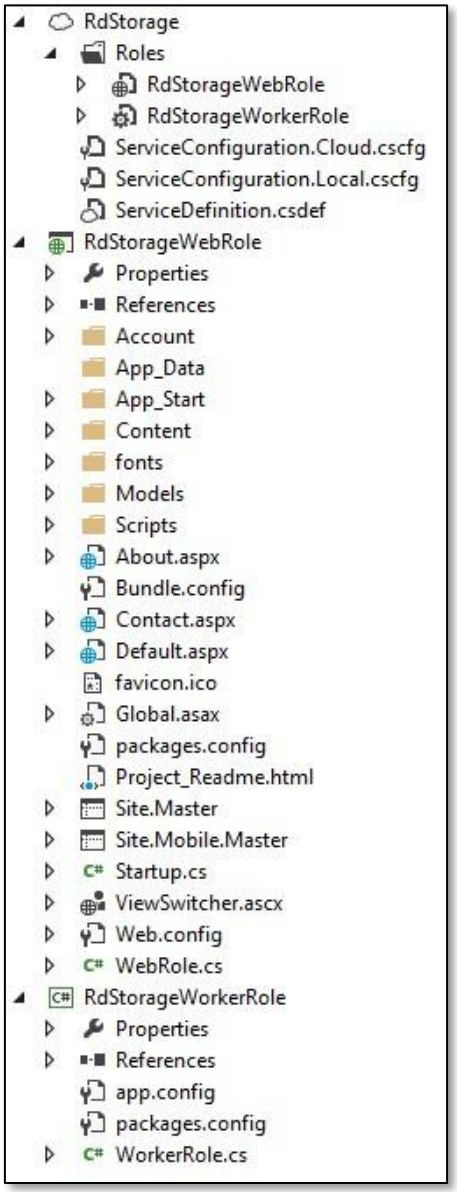
More info at https://msdn.microsoft.com/en-us/library/azure/hh369931.aspx

Task 1: Creating the initial solution

- Create a new Windows Azure Project (RdStorage)
- Create a new ASP.NET Web Role (RdStorage_WebRole)
- Create a new Worker Role (RdStorage_WorkerRole)
- Create a Storage ConnectionString
    - Since the Web and Worker Roles use Windows Azure storage services, the user has to specify the storage account settings for each role
    - A “DataConnectionString” is specified for each role in the project, each one using the Storage Local Emulator

The solution has 3 projects related to RdStorage:

- **RdStorage:** it includes the service definition file, ServiceDefinition.csdef, which contains metadata (i.e., which roles are used, their trust level, etc.).
- **RdStorage_WebRole:** is a standard ASP.NET Web Application project template, with additional class to manage the initialization, starting, and stopping.
- **RdStorage_WorkerRole:** it contains classes for the worker



Task 2: Sending messages to the Queue

- The RdStorage_WebRole web application is developed, to send messages to the queue.
- Create the ASP Web Page of the Web Role (Default.aspx file of the RdStorage_WebRole)

```
…
<asp:TextBox ID="txtMessage" runat="server"></asp:TextBox>
<asp:Button ID="btnSend" runat="server" Text="Send message"
OnClick="btnSend_Click"/>
…
```

- Implement the btnSend_Click event (Default.aspx.cs file of the RdStorage_WebRole)

```
…
 protected void btnSend_Click(object sender, EventArgs e)
      {
        // initialize the account information
        …
        // retrieve a reference to the messages queue
        …
         // Create a message and add it to the queue.
        CloudQueueMessage message = new CloudQueueMessage(txtMessage.Text);
        queue.AddMessage(message);
        //Clear text box value
        txtMessage.Text = string.Empty;
      }//btnSend_Click
…
```

For a better solution, these parts can be moved to another method of Default.aspx.cs (see next slide for the full code)

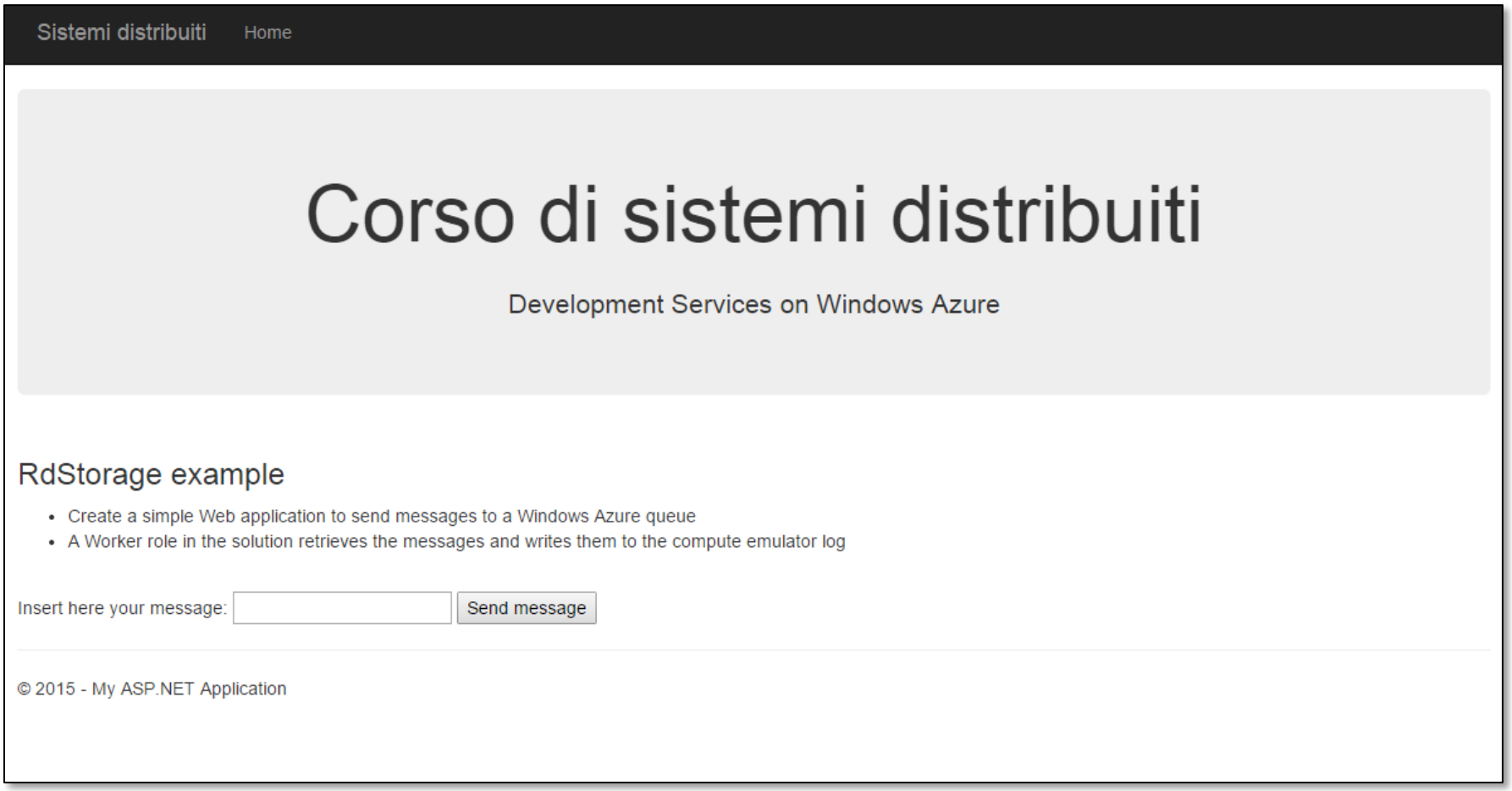
Full code 1/2 – Web Role

```
using Microsoft.WindowsAzure;
using Microsoft.WindowsAzure.Storage;
using Microsoft.WindowsAzure.Storage.Queue;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace RdStorageWebRole
{
    public partial class _Default : Page
    {
        CloudQueue queue = null;

        protected void Page_Load(object sender, EventArgs e)
        {
            // initialize the account information
            CloudStorageAccount storageAccount =
            CloudStorageAccount.Parse(CloudConfigurationManager.GetSetting("DataConnectionString"));
            // Create the queue client
            CloudQueueClient queueClient = storageAccount.CreateCloudQueueClient();
            // Retrieve a reference to a queue
            queue = queueClient.GetQueueReference("messagequeue");
            // Create the queue if it doesn't already exist
            queue.CreateIfNotExists();
        }
…
```

Full code 2/2 – Web Role

```
…
 protected void btnSend_Click(object sender, EventArgs e)
        {
            // Create a message and add it to the queue.
            CloudQueueMessage message = new CloudQueueMessage(txtMessage.Text);
            queue.AddMessage(message);

            //Clear text box value
            txtMessage.Text = string.Empty;
        }//btnSend_Click
```

The Web interface



Code snippets – Worker Role

Task 3: Retrieving messages from the Queue (who? … the worker!!!)

- The RdStorage_WorkerRole is implemented, to retrieve messages from the queue and show them in the compute emulator log.

- Implement the logic of the Worker Role (WorkerRole.cs file of the RdStorage_WorkerRole project).

1. Initialize some account information (in **onStart** or **Run** method of the WorkerRole.cs file).

2. Create and retrieve a reference to the queue (in **onStart** or **Run** method of the WorkerRole.cs file.

3. Retrieve messages from the queue and write them to the compute emulator log (**Run** method of the WorkerRole.cs file).

Full code 1/2 – WorkerRole.cs

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Net;
using System.Threading;
using System.Threading.Tasks;
using Microsoft.WindowsAzure;
using Microsoft.WindowsAzure.Diagnostics;
using Microsoft.WindowsAzure.ServiceRuntime;
using Microsoft.WindowsAzure.Storage;
using Microsoft.WindowsAzure.Storage.Queue;

namespace RdStorageWorkerRole
{
    public class WorkerRole : RoleEntryPoint
    {

      public override bool OnStart()
        {
            // Set the maximum number of concurrent connections
            ServicePointManager.DefaultConnectionLimit = 12;
            bool result = base.OnStart();
            Trace.TraceInformation("RdStorageWorkerRole has been started");
            return result;
        }
…
```

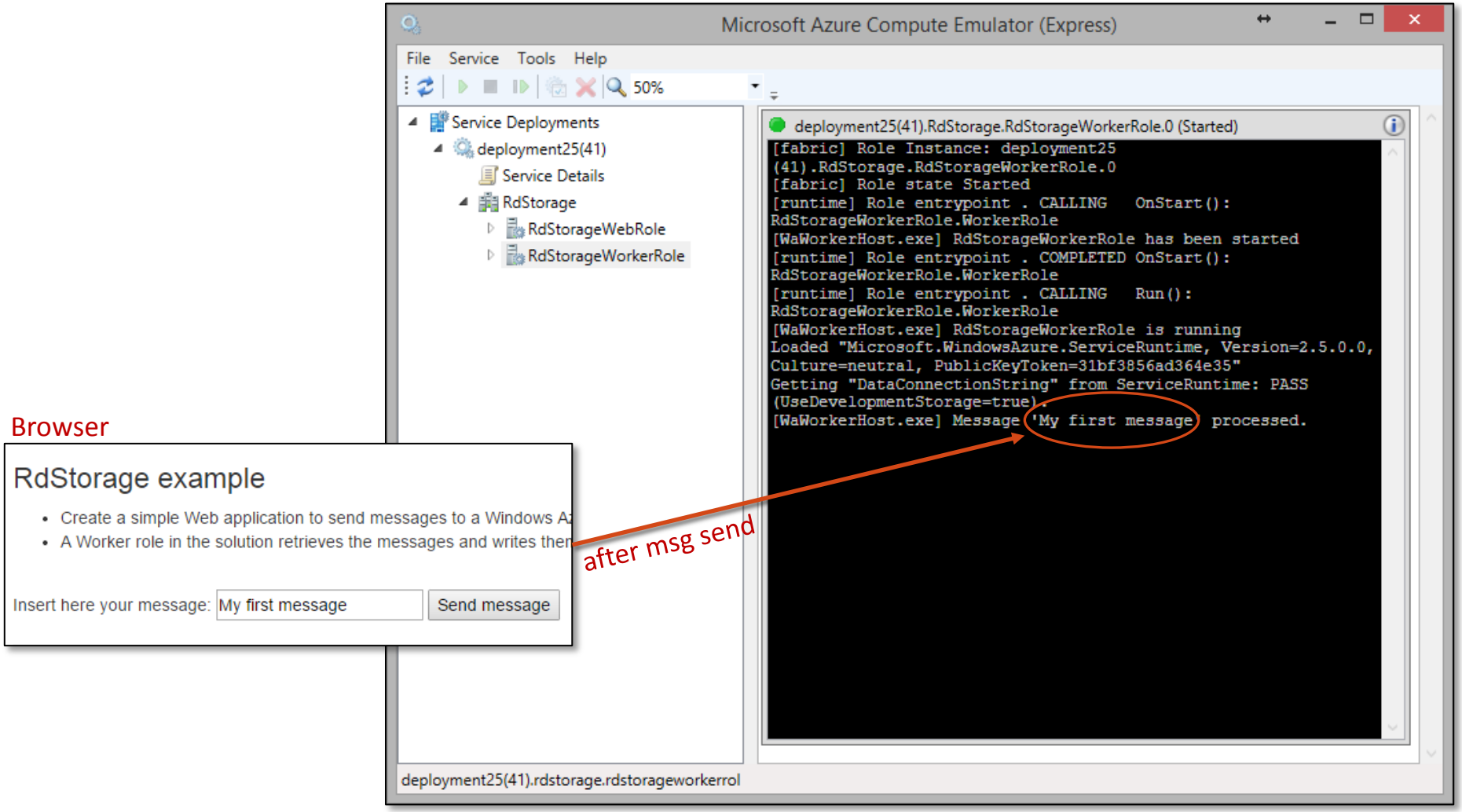
Full code 2/2 – WorkerRole.cs

```
…
      public override void Run()
      {
            Trace.TraceInformation("RdStorageWorkerRole is running");
            // initialize the account information
            CloudStorageAccount storageAccount =
            CloudStorageAccount.Parse(CloudConfigurationManager.GetSetting("DataConnectionString"));
            // Create the queue client
            CloudQueueClient queueClient = storageAccount.CreateCloudQueueClient();
            // Retrieve a reference to a queue
            CloudQueue queue = queueClient.GetQueueReference("messagequeue");
            // retrieve messages and write them to the compute emulator log
            while (true)
            {
                Thread.Sleep(2000);
                if (queue.Exists())
                {
                    var msg = queue.GetMessage();
                    if (msg != null)
                    {
                        Trace.TraceInformation(string.Format("Message '{0}' processed.",
msg.AsString));
                        queue.DeleteMessage(msg);
                    }
                }
            }//while
       } //Run end

    } //class
} //namespace
```

The worker process will try to get a message from the queue every 2 seconds using the **GetMessage** method. If there are messages in the queue, it will show them in the Compute Emulator log.

Demo

Task 4: Execution of the application and watch by the Emulator UI

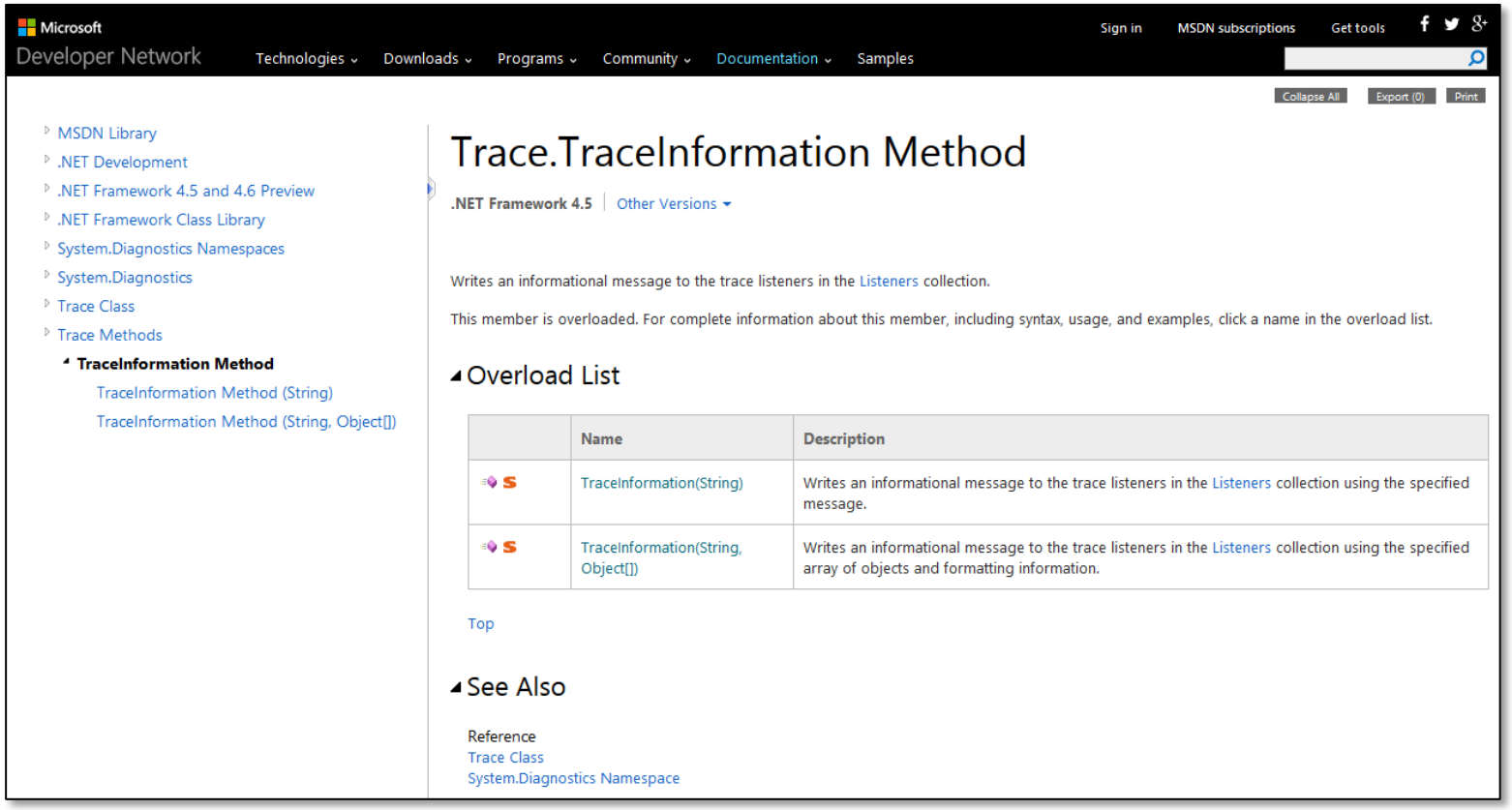


Trace Information

- You can fix the Logging Level (**TraceInformation (…)** write at Information level)
  - Example:

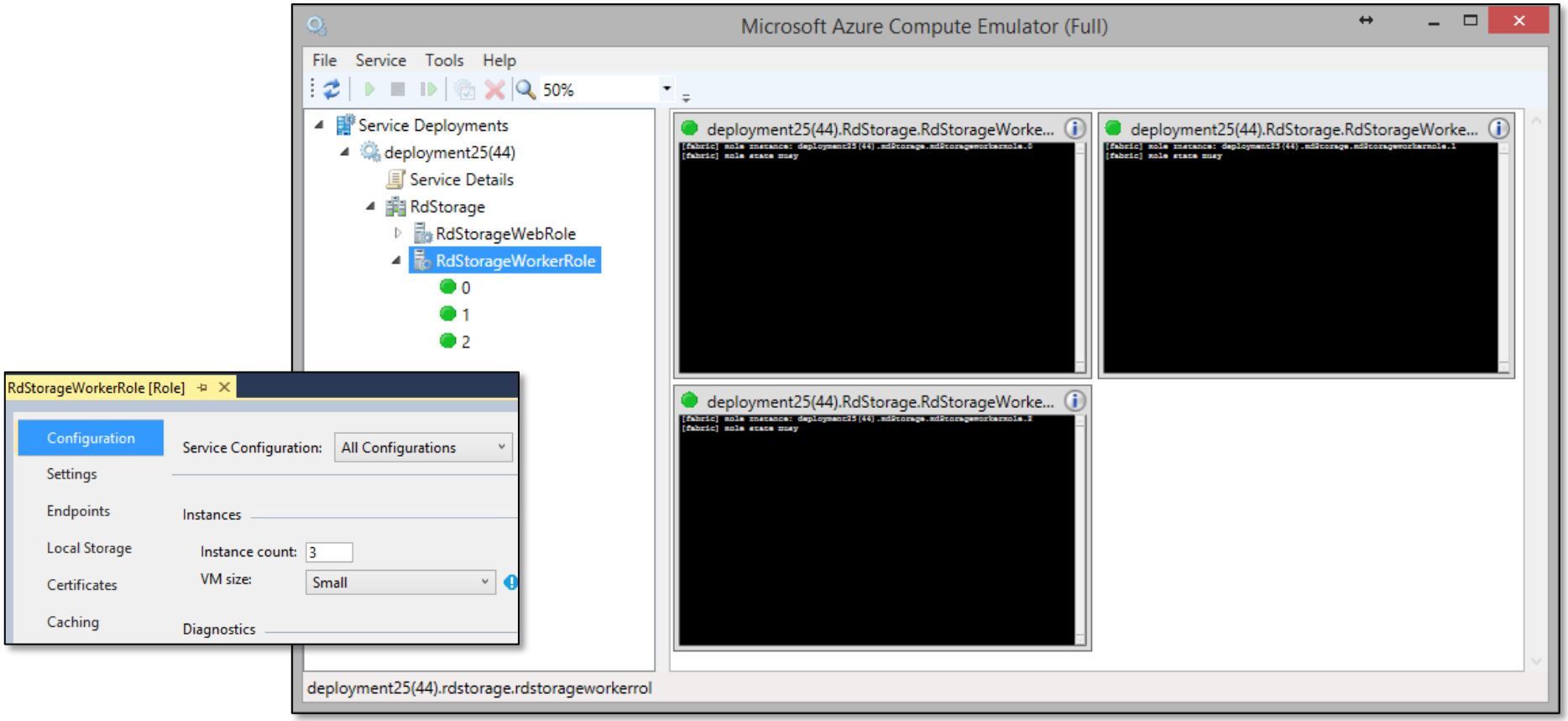
```
Trace.TraceInformation(string.Format(" Message '{0}''{1}' processed.", msg.AsString, msg.AsString));
```

More info at https://msdn.microsoft.com/en-us/library/system.diagnostics.trace.traceinformation.aspx



Demo

- RdStorage example works with 1 worker.
- You can change the number of Worker instances by «RdStorage->Roles»
- More info at https://msdn.microsoft.com/en-us/library/azure/dn339018.aspx



Summary

- Hello World example
- Working with Queues (RdStorage)
- Working with Queues (RdStorage-MultipleMessages)

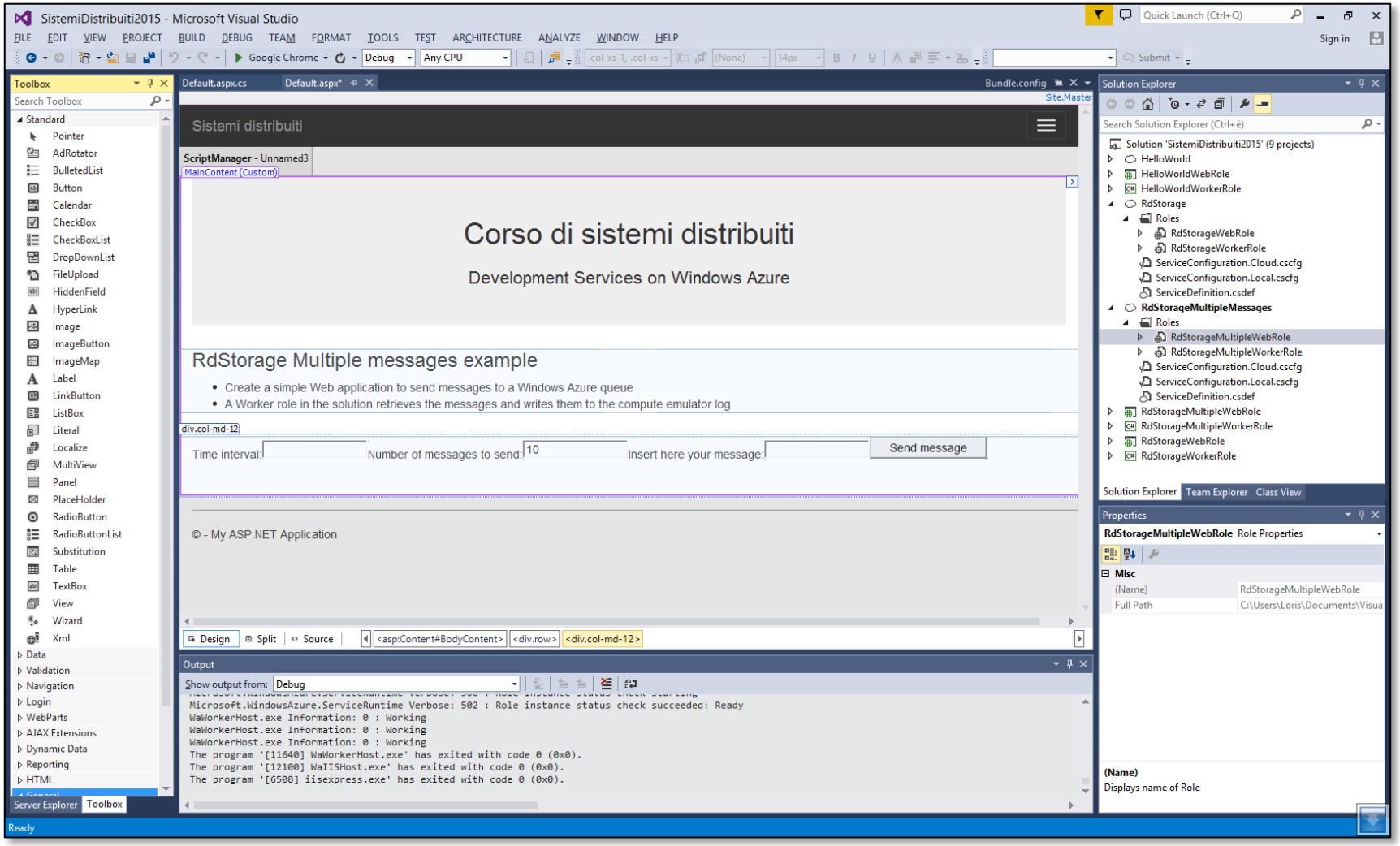
RdStorage-MultipleMessages

- Let us modify the previous example, by automatically sending more messages in sequence.

- The user can fix:
  - String msg to be sent to the worker(s)
  - Period of repetition
  - Number N of messages to be sent (msg<i>, for i=1…N)

RdStorage-MultipleMessages

The ASP Web Page can be designed by graphical tools



RdStorage-MultipleMessages

Modified the Default.aspx file of the Web Role

```
<div class="col-md-12">
     <asp:Label ID="Label2" runat="server" Text="Time interval:"></asp:Label>

     <asp:TextBox ID="txtPeriod" runat="server"></asp:TextBox>

     <asp:Label ID="Label3" runat="server" Text="Number of messages to send:"></asp:Label>

     <asp:TextBox ID="numMessages" runat="server" Text="10"></asp:TextBox>

     <asp:Label ID="Label1" runat="server" Text="Insert here your message:"></asp:Label>
     <asp:TextBox ID="txtMessage" runat="server"></asp:TextBox>
     <asp:Button ID="btnSend" runat="server" Text="Send message" OnClick="btnSend_Click" />
</div>
```

RdStorage-MultipleMessages

Modified the 'btnSend_Click' method in Default.aspx.cs of the Web Role

```
protected void btnSend_Click_Multiple(object sender, EventArgs e)
        {
            // initialize the account information
            CloudStorageAccount storageAccount =
            CloudStorageAccount.Parse(CloudConfigurationManager.GetSetting("DataConnectionString"));
            // Create the queue client
            CloudQueueClient queueClient = storageAccount.CreateCloudQueueClient();
            // Retrieve a reference to a queue
            CloudQueue queue = queueClient.GetQueueReference("messagequeue");
            // insert multiple messages into the queue
            int n, sleepTime;
            string txtMsg = txtMessage.Text;
            string txtP = txtPeriod.Text;
            int.TryParse(txtP, out sleepTime);
            string numMsg = numMessages.Text;
            int.TryParse(numMsg, out n);
            for (int i = 0; i < n; i++)
            {
                Thread.Sleep(sleepTime * 1000);
                string currMsg = txtMsg + i;
                var msg = new CloudQueueMessage(currMsg);
                queue.AddMessage(msg);
            }
            txtMessage.Text = string.Empty;
        }//btnSend_Click
```

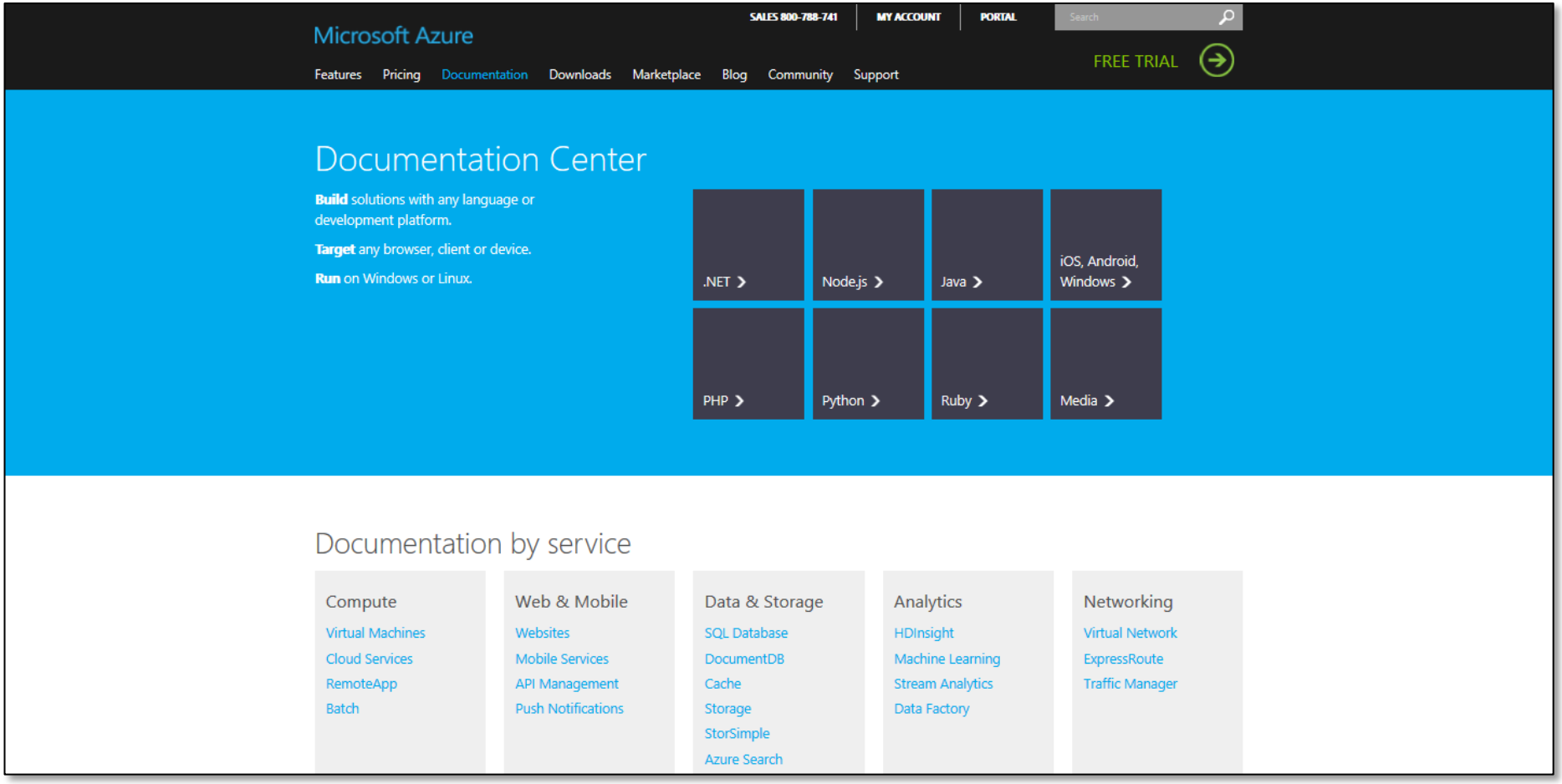
append an 'id' to the string automatically built

RdStorage-MultipleMessages

- Demo in classroom

Documentation Azure

http://azure.microsoft.com/en-us/documentation/



Documentation .NET

http://azure.microsoft.com/en-us/develop/net/