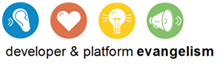
* 1. 

Demo Script

Hello Windows Azure

Lab version: 2.0.0

Last updated: 2/9/2011

* 1. 

Contents

[Overview 3](#_Toc285012811)

[Key Messages 3](#_Toc285012812)

[Key Technologies 3](#_Toc285012813)

[Time Estimates 4](#_Toc285012814)

[Setup and Configuration 4](#_Toc285012815)

[Demo Flow 5](#_Toc285012816)

[Opening Statement 6](#_Toc285012817)

[Step-by-Step Walkthrough 6](#_Toc285012818)

[Creating the Hello Windows Azure application 7](#_Toc285012819)

[Deploying the application 12](#_Toc285012820)

[Summary 20](#_Toc285012821)

[Known Issues 21](#_Toc285012822)



Overview

* 1. This document provides setup documentation, step-by-step instructions, and a written script for showing a demo of Windows Azure. This document can also serve as a tutorial or walkthrough of the technology. In this demo you will build, debug, and deploy a simple “Hello World” ASP.NET Application using Windows Azure and have the application running on the web in less than 5 minutes. For additional demos of the Windows Azure platform, please visit <http://www.azure.com>.
     1. **Note:** In order to run through this complete demo, you must have network connectivity and a Windows Azure account.

### Key Messages

In this demo you will see three key things:

* 1. First, you will see how the Windows Azure SDK and Visual Studio Tools for Windows Azure provide development tools that enable a rich and familiar development experience for cloud applications.
  2. Second, you will see how Windows Azure abstracts you from the underlying infrascture and uses simple models to define the roles and number of instances for your application.
  3. Finally, you will see how to deploy, start, and manage a Windows Azure application through the Management portal.

### Key Technologies

This demo uses the following technologies:

* 1. .NET Framework 4.0
  2. Visual Studio 2010
  3. Windows Azure Tools for Microsoft Visual Studio 1.3

### Time Estimates

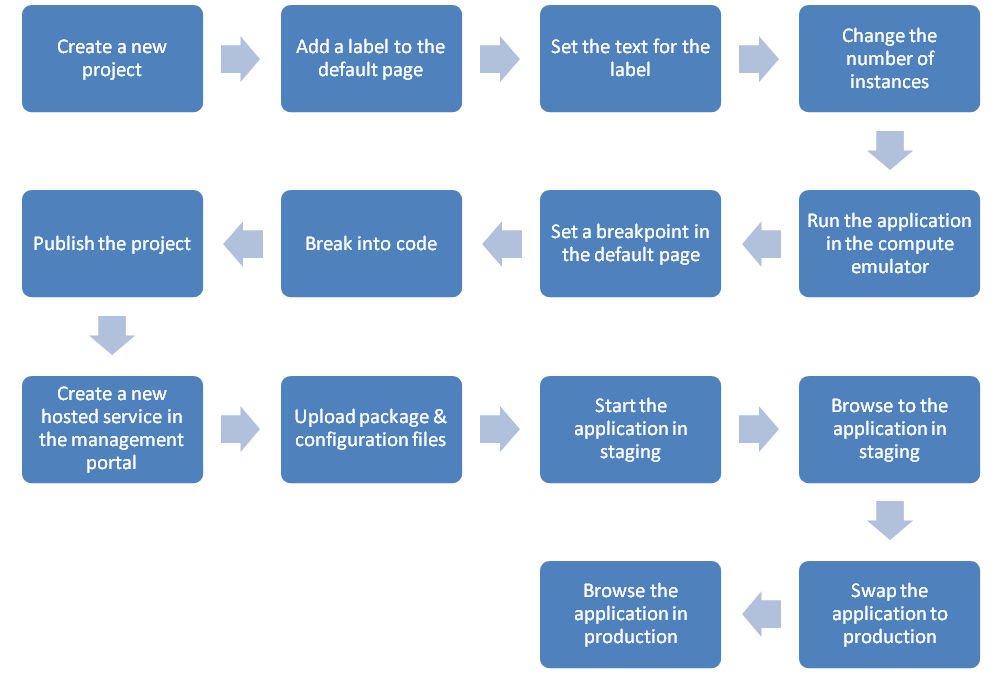
* + Estimated time for setting up and configuring the demo: 10 min
  + Estimated time to complete the demo: 5 min

Setup and Configuration

* 1. This demo does not have any advanced configuration requirements. You simply need to have the Key Technologies (prerequisites) installed and have a Windows Azure account.
  2. The following steps describe how to run the Dependency Checker utility included with the demo to verify that you have the prerequisite components. You can skip this exercise if you are confident that the prerequisites are properly installed and configured.
  3. Open a Windows Explorer window and browse to the demo’s Source\Script folder.
  4. Double-click the Dependencies.dep file in this folder to launch the Dependency Checker tool and install any missing prerequisites.
  5. If the User Account Control dialog is shown, confirm the action to proceed.
  6. **Note:** This process may require elevation. The .dep extension is associated with the Dependency Checker tool during its installation. For additional information about the setup procedure and how to install the Dependency Checker tool, refer to the **Setup.docx** document in the **Assets** folder of the training kit.

1. f

Demo Flow

1. The following diagram illustrates the high-level flow for this demo and the steps involved:
   1. 
   2. Figure 1
   3. Demo Flow

Opening Statement

* 1. In the next 5 minutes we will build, debug, and deploy a simple “Hello World” ASP.NET Application using Windows Azure. I want to show you a simple demo, so you can get a feel for the concepts and steps involved in building a Windows Azure application using the November CTP. In later demos you will see how we can build more complex applications that utilize additional roles, the Windows Azure APIs, and Windows Azure Storage services as well as additional services of the Azure Services Platform.

In this simple “Hello World” demo you will specifically see three key things:

* 1. First, you will see how the Windows Azure SDK and Visual Studio Tools for Windows Azure provide development tools that enable a rich and familiar development experience for cloud applications.
  2. Second, you will see how Windows Azure abstracts you from the underlying infrascture and uses simple models to define the roles and number of instances for your application.
  3. Finally, you will see how to deploy, start, and manage a Windows Azure application through the Management portal.

Step-by-Step Walkthrough

1. This demo is composed of the following segments:
   * Creating the Hello Windows Azure application
   * Deploying the application

### Creating the Hello Windows Azure application

|  |  |  |
| --- | --- | --- |
| Action | Script | Screenshot |
| 1. Start Visual Studio 2010 (as administrator). 2. Select **File 🡪 New 🡪 Project**. 3. Select from the **Visual C#** Installed templates the **Cloud** category. 4. Select the **Windows Azure Project**. 5. Name the solution **HelloAzure**. 6. Click OK to launch a wizard. 7. From the wizard, add the **ASP.NET Web Role** in C#. 8. Select the added Role and click the edit icon. 9. Rename from **WebRole1** to **HelloAzure\_WebRole**. | * Let’s launch Visual Studio 2010 and create a new project. * In this case, I have installed the November CTP of the Windows Azure SDK and the Windows Azure Tools for Visual Studio. * Notice that we now have a new category of projects – Cloud Services. * We have a new template called “Windows Azure Project”. * This template launches a wizard that allows us to select the type and programming language for a role within our project. * Let’s name our new project HelloAzure and rename the WebRole1 to HelloAzure\_WebRole. |  |
| 1. Click **OK** and wait for the new solution to be created. 2. Double click the **Default.aspx** page to open it and select the design view. | * Visual Studio will now create a new solution. * This solution will contain two projects * The first project named HelloAzure\_WebRole is a standard ASP.NET project. * The second project is a new Windows Azure Project. * This Windows Azure Project references to our ASP.NET project. It also contains a configuration file that defines the model for our Windows Azure solution. We’ll take a look at this model in just a few minutes. |  |
| 1. Expand the Toolbox. 2. Drag and drop an ASP.NET Label Control. 3. Change the **Font->Size** property for the new label to **XX-Large** | * First, let’s build our HelloAzure web page. * Let’s open the default web page * Now we’ll add a standard ASP.NET label control to our default page. * Let’s go ahead and adjust the font size on the label. |  |
| 1. Right click the design surface and select View Code from the context menu to view the Default.aspx.cs code behind file. 2. Add code in the Page\_Load event to set the text for the label. | * Now let’s add a bit of code to display the text “Hello Windows Azure” in the label. |  |
| 1. Open the ServiceConfiguration.cscfg file. 2. Updated the settings to run 2 instances of the web role. | * Before we run this application, let’s take a look at the model that defines our Windows Azure application. * Let’s navigate over to the HelloAzure Windows Azure Project and open the ServiceConfiguration.cscfg file * This configuration file defines the roles for our Windows Azure project. As you can see here, it also defines the number of instances per role. * Let’s simply change the number of instances here to 2. |  |
| 1. Select **Debug->Start Debugging** | * Now let’s start the application in debug mode or just press the familiar F5 button. | * 1. C:\Users\ADMINI~1\AppData\Local\Temp\1\SNAGHTMLb43fd93.PNG |
| 1. NOTE: It will take 30-90 seconds until the Web application starts. | * Here we can see our simple ASP.NET web application and the text “Hello Windows Azure” |  |
| 1. Show the compute emulator by right-clicking the Azure icon in the task bar and select “Show Compute Emulator UI”. 2. Expand the HelloAzure application in the compute emulator and expand the WebRole. 3. Select the WebRole and show the two instances that are running. | * This will cause the ASP.NET project to be compiled into a .NET assembly, just as normal. * However, since we have a Windows Azure Project in our solution set as the startup project, this will start the Windows Azure compute emulator. * To see the details of the running projects we need to show the compute emulator UI. * The Windows Azure compute emulator, or simple compute emulator, is a simulated environment for developing and testing Windows Azure applications on your machine. * You can think of this as a Cassini like environment. * Here you can see the compute emulator UI. * Notice that we have a new deployment containing our HelloAzure application. * If we expand the WebRole for the HelloAzure application, we can see that there are two instances for this role. |  |
| 1. Switch back to Visual Studio. 2. Set a breakpoint on the line that sets the label text. | * Let’s quickly switch back over to Visual Studio and set a breakpoint |  |
| 1. Switch back to Internet Explorer and refresh the page. 2. Switch back to Visual Studio to see the breakpoint. 3. Press F5 to continue execution. | * Now let’s refresh the page * Notice that we can drop into Visual Studio to debug our project and step through lines of code – just as we normally do. |  |
| 1. Stop the application by closing Internet Explorer. |  |  |

### Deploying the application

|  |  |  |
| --- | --- | --- |
| Action | Script | Screenshot |
| 1. Switch back to Visual Studio. 2. Right click the **HelloAzure** project and select **Publish** from the context menu. 3. On the Deploy Windows Azure project dialog select “Create Service Package Only” and click OK. | * So you’ve seen how we can build and debug simple web applications for Windows Azure with the new Visual Studio extensions and the SDK * Now let’s deploy our HelloAzure application to the cloud * Let’s switch back to Visual Studio * To deploy our app, first let’s right click the HelloAzure Windows Azure Project and select Publish * We just want to generate the package to deploy it using the management portal, so we choose “Create Service Package Only”. * This will compile and build the solution. It will also create a new service package, essentially a zip file, containing the assemblies and configuration files for the solution. |  |
| 1. Copy the path where the service package was published from the Windows Explorer address bar. | * Once the publish process is complete, Visual Studio will open the directory where the service package was created. * It will also launch the default browser and navigate to the management portal. * Let’s copy the path to this file. |  |
| 1. Switch over to the browser window that was launched during the Publish process. | * Now let’s login to the management portal using our Live ID. * Previously, I have created a subscription in the management portal. |  |
| 1. Click on the New Hosted Service button from the Common Tasks ribbon. | * Once in the management portal, we select the type of project we want to create in the management portal. * In this case, we want to deploy an application to run in Windows Azure, so we click on the “New Hosted Service” button from the Common Tasks ribbon. |  |
| 1. Choose a subscription. 2. Enter the service name (i.e. HelloAzure) 3. Enter the URL prefix (i.e. helloazure) 4. Choose a region (i.e. Anywhere US) | * We need to select the subscription where we want to create the hosted service. * Next, let’s give our service a name. * Now we pick a URL prefix and the dialog will check the availability. * For this demo, we will just choose any region for the deployments, but in practice we can choose the geography for our app as well or use an affinity group. |  |
| 1. Leave the “Deploy to stage environment” option selected and the “Start after successful deployment” check enabled. 2. Set a name for the deployment. 3. Use the Browse Locally button from the Package location and browse for the HelloAzure.cspkg file generated by Visual Studio. 4. Then, use the Browse Locally From the Configuration File and browse for the ServiceConfiguration.cscfg file generated by Visual Studio. You can also paste that address from the notepad opened in step 4. 5. Click OK. | * We’ll leave the deployment options by default to let our service deploy to stage environment and start after the deployment. * Now, let’s give the deployment a name and select the package and configuration files location. * To specify the package from our local filesystem we use the Browse Locally button and browse to the service package that we published from Visual Studio. * We also need to browse to the ServiceConfiguration file that was also published from Visual Studio. You might recall that this file defines the roles and number of instances per role. * In this case, we don’t need to add any certificate for our service, so we click OK to start creating and deploying the hosted service. |  |
| 1. Wait for the package to upload and initialize. | * Now, we have to wait a few minutes to upload the package. * Then, the web role will be initialized. * At this point Windows Azure will start up virtual machine instances for each of the roles. * Once the instances started, our application – the assemblies, ASPX pages, etc. that we uploaded earlier – will be deployed into each of the instances. * We need to wait a while until the service becomes ready. |  |
| 1. Select the node with the deployment name. 2. On the properties pane click the DNS name link. | * Once the application is running in staging, we can now access it using the temporary Website URL generated. * By selecting the deployment in the items list, we can see its properties in the right pane, clicking on the DNS name we can navigate to our Hello Azure application. |  |
| 1. View the HelloAzure web site in the new browser window 2. Close the browser window. | * Here we can see our Hello Windows Azure application running in Windows Azure – it’s publicly exposed on the web! * In fact, you could even access it now if you could enter this long, temporary web site URL. * Let’s close this window and go back to the Management portal. |  |
| 1. Click the **Swap VIP** button in the Deployments ribbon. 2. You will be prompted to confirm that you want to switch to the production environment. Press the **OK** button when prompted. 3. **NOTE:** It can take a few minutes to start the production environment. | * Let’s see how we can move our application from staging to production. * We’ll click on the sync button and confirm that we want to move the switch the staging environment with production. * This can take a few minutes. |  |
| 1. Select the node with the deployment name. 2. On the properties pane click on the DNS name link. | * You can now see that the production environment is running. * Let’s now browse to our application running in production. |  |
| 1. You should see the web page with the text Hello Windows Azure dialog running with the production URL. | * Finally, you can see that our simple “Hello Azure” application is running in the production environment. * Notice that the URL is the sub-domain on cloudapp.net that we specified when creating the project in the Management portal. * If you have internet access, you can browse to this site now. |  |

Summary

* 1. In this demo, you could see how we can easily build and debug web applications using the familiar ASP.NET programming model with the new Windows Azure SDK and extensions for Visual Studio. You could also see how we can use the Windows Azure Management Portal to create projects, deploy applications, and manage the staging and production environments. Then, and in less than five minutes, we created and ran a new application in the cloud.
  2. What you haven’t seen yet is the entire infrastructure provided by Windows Azure. For example, during this demo we haven’t worried about physical machines, rack space, network switches or connectivity. We haven’t also used remote access and physical file copy across machines: Windows Azure abstracted us from the underlying infrastructure, providing to our application automated deployment, isolation, redundancy, and load balancing.
  3. The aim of this demo was to show a simple example that helped you understand the initial concepts. In later demos, you will see how we can build more complex applications that use additional roles, Windows Azure APIs, and Windows Azure Storage services.

Known Issues

* 1. Please note the following known issues with this demo:
  + It can sometimes take several minutes to start a Windows Azure application on the cloud. Consequently, you may want to have an example application already deployed and running in the production environment.