WorkshopPLUS

Microsoft Azure Infrastructure as a Service (IaaS)

Desired State Configuration (Classic)

Student Lab Manual

V2.1, November 19, 2015

Information in this document is subject to change without notice. The example companies, organizations, products, people, and events depicted herein are fictitious. No association with any real company, organization, product, person or event is intended or should be inferred. Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarked, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

© 2010 Microsoft Corporation. All rights reserved.

Microsoft, MS-DOS, MS, Windows, Windows NT, MSDN, Active Directory, BizTalk, SQL Server, SharePoint, Outlook, PowerPoint, FrontPage, Visual Basic, Visual C++, Visual J++, Visual InterDev, Visual SourceSafe, Visual C#, Visual J#,  and Visual Studio are either registered trademarks or trademarks of Microsoft Corporation in the U.S.A. and/or other countries.

Other product and company names herein may be the trademarks of their respective owners.

Contents

[Desired State Configuration 4](#_Toc435694786)

[Overview 4](#_Toc435694787)

[Objectives 4](#_Toc435694788)

[Prerequisites 4](#_Toc435694789)

[Exercises 4](#_Toc435694790)

[Exercise 1: Configuration Management with PowerShell DSC 5](#_Toc435694791)

[Task 1 – Setting up your environment 5](#_Toc435694792)

[Task 2 – Creating Your Script Files 8](#_Toc435694793)

[Task 3 – Running the Scripts 13](#_Toc435694794)

[Task 4 – Confirming VM Configuration 14](#_Toc435694795)

[Task 5 – Remote Desktop into the VM 15](#_Toc435694796)

# Desired State Configuration

## Overview

During this lab, you will run several exercises that will help you achieve a better understanding of the capabilities of PowerShell Desired State Configuration (DSC).

### Objectives

In this hands-on lab, you will learn how to:

* Set, verify and enforce a desired state of configuration using PowerShell

### Prerequisites

The following is required to complete this hands-on lab:

* [Microsoft Azure PowerShell](http://msdn.microsoft.com/en-us/library/windowsazure/jj156055)
* A Microsoft Azure subscription - [sign up for a free trial](http://aka.ms/WATK-FreeTrial)
* If you are running an operating system older than Windows Server 2012 R2 or Windows 8.1, you will need to install PowerShell 4.0. You can download that from here <http://www.microsoft.com/en-us/download/details.aspx?id=40855> . Note that you may need to re-install Microsoft Azure PowerShell after installing this KB.
* A pre-existing Microsoft Azure Storage account (classic)

## Exercises

This hands-on lab includes the following exercises:

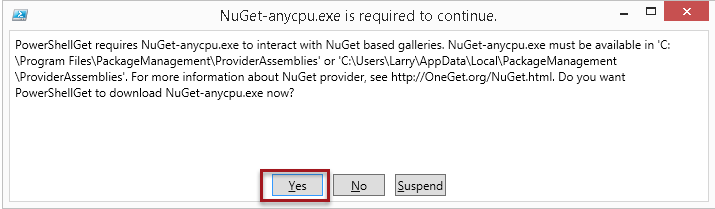
* Configuration management using PowerShell DSC

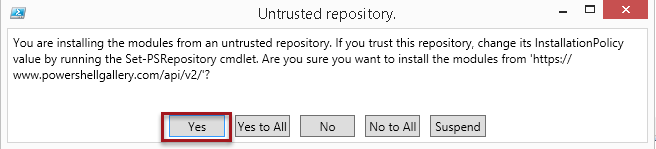
# Exercise 1: Configuration Management with PowerShell DSC

The purpose of this lab exercise is to demonstrate how to use PowerShell and Desired State Configuration (DSC) to deploy a virtual machine out into your Azure subscription. This machine will be configured with IIS, ASP.Net 4.5 and you will deploy a simple web application to the virtual machine.

Since we will be configuring IIS, we will need to use a module from the PowerShell Gallery (<https://www.powershellgallery.com>) in order to obtain resource files that will assist DSC in the configuration tasks.

## Task 1 – Setting up your environment

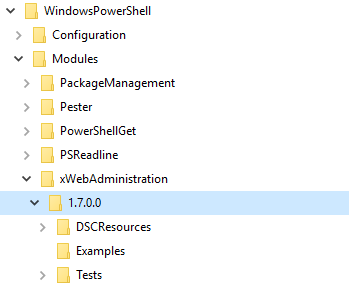
1. Make sure that you have installed the latest Microsoft Azure PowerShell by using the Web Platform installer.
2. Browse to the folder **.\M5-Management\Labs\DesiredStateConfig\Apps** and install the PackageManagement\_x64.msi package. This will allow you to install packages from NuGet.
3. Open PowerShell ISE as an administrator.
4. In the ISE command prompt window, type in:  
   Install-Module xWebAdministration
5. You will be prompted with the following prompt. Select Yes.  
     
   
6. You will be prompted regarding the install. Select Yes.



1. Once the install has completed, you can confirm that the location of the PowerShell Gallery module by browsing to C:\Program Files\WindowsPowerShell\Modules. Depending on what other DSC functionality you may need outside of this lab, you may see other items installed.

Also, depending on which OS you install the package on, the structure under the Modules directory could be slightly different. The view below is from a Windows 10 client machine.

Record the full path, ie, C:\Program Files\WindowsPowerShell\Modules\xWebAdministration\1.7.0.0, you will need this in your script.



1. Within the PowerShell ISE command window, type the following command, separately, to see a list of all of the DSC resource modules. NOTE: You can also do this within PowerShell ISE.

PowerShell

Get-DSCResource

1. In the command prompt window, change the directory to the location where you will create the scripts for the hands-on labs **.\M5-Management\Labs\DesiredStateConfig\Scripts**.
2. You first need to make a connection out to Azure for your account. In the PowerShell ISE command window, type in:

PowerShell

Add-AzureAccount

NOTE: Sometimes PowerShell ISE has trouble finding the Azure module(s) and you will get an error stating that Add-AzureAccount is not a known cmdlet. **If** this happens, you will first need to type in:  
PowerShell

Import-Module Azure

1. Download your Azure publishsettings file. You need to save this file into the **.\M5-Management\Labs\DesiredStateConfig\Scripts** directory. This information will be used by our PowerShell script when creating and configuring the VM in Azure. Type this into the PowerShell ISE command/output window:

PowerShell

Get-AzurePublishSettingsFile

NOTE: In a production scenario, you will typically not use the publishsettings file. In the case of this lab exercise, we want to pull information out of the publishsettings file to use in our script so that our DSC script can use the values at runtime. When you type in Add-AzureAccount and sign in to Azure, that loads all the information you need in to the running PowerShell session if you are manually testing a script.

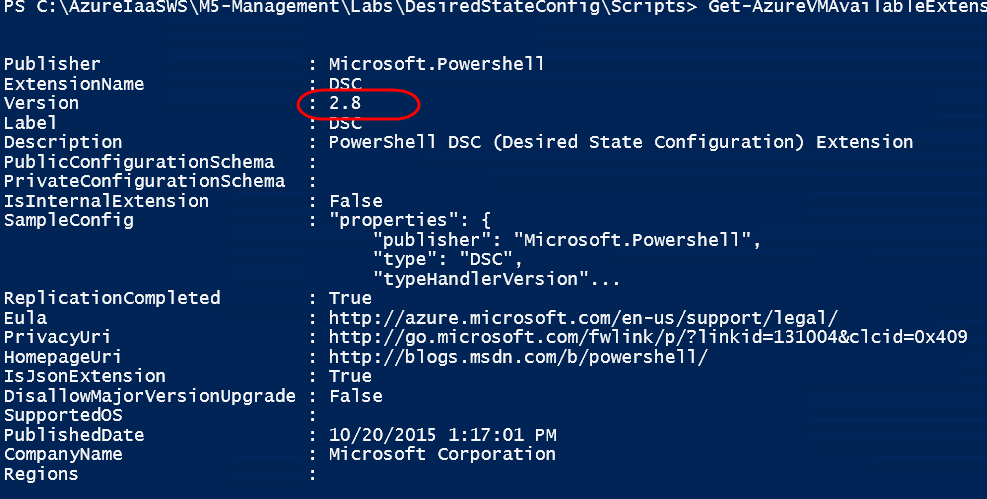
1. Next, we need to make sure that we have the DSC extension available. In the PowerShell ISE command/output window, type the following:

PowerShell

Select-AzureSubscription –SubscriptionName ‘<Your subscription name’>

Get-AzureVMAvailableExtension –Publisher Microsoft.PowerShell

You should see something similar to this (the Version number you see may be greater):



## Task 2 – Creating Your Script Files

1. In the PowerShell ISE **script** window, paste in the following code:

PowerShell

@{

AllNodes =

@(

@{

NodeName = "localhost";

Role = "WebServer"

}

);

}

This file will represent our environment configuration data. The environment configuration data is the file you would make changes to when you are moving the configuration from Dev to Staging then Production because this is where we keep the names of the servers (nodes).

In the case above, we are saying that whenever we apply this configuration to a machine (who’s default name is ‘localhost’), we are naming it ‘WebServer’. In this configuration file we could have settings for web servers, SQL servers etc.

1. Save this file as **.\M5-Management\Labs\DesiredStateConfig\Scripts\ConfigurationData.psd1**.

NOTE: Why are we using the .psd1 extension? This is known as a module manifest file <http://msdn.microsoft.com/en-us/library/dd878337(v=vs.85).aspx> and a manifest file (although not required) can be used for environment configuration, like we see here.

1. Create a new script file by clicking on the **New** icon on the toolbar and save this file as **.\M5-Management\Labs\DesiredStateConfig\Scripts\SimpleWebServerConfiguration.ps1**.
2. Copy and paste the following code into the script file you just created:

PowerShell

# This is the structural configuration file that creates the structure and components that will be on the

# server being setup

Configuration InstallWebServer

{

# We will use a parameter to determine where our source code is located on our

# local disk

param (

[Parameter(Mandatory=$True,Position=1)]

[string]$websiteSourceLocation

)

Import-DscResource -Module xWebAdministration

# It is possible that in our ConfigurationData.psd1 file we could specify

# multiple nodes of various names. In this case, if one or more of these servers

# we named 'WebServer' it would fall into this node detail area

node $AllNodes.Where{$\_.Role -eq "WebServer"}.NodeName

{

LocalConfigurationManager

{

ConfigurationID ="43d4995d-3199-4e0d-aef5-d52d3b681ac4";

RebootNodeIfNeeded = $true;

}

# Install the IIS role

WindowsFeature IIS

{

Ensure = "Present"

Name = "Web-Server"

}

# Install the ASP .NET 4.5 role

WindowsFeature AspNet45

{

Ensure = "Present"

Name = "Web-Asp-Net45"

}

# Stop the default website

xWebsite DefaultSite

{

Ensure = "Present"

Name = "Default Web Site"

State = "Stopped"

PhysicalPath = "C:\inetpub\wwwroot"

DependsOn = "[WindowsFeature]IIS"

}

# Copy the website content

File WebContent

{

Ensure = "Present"

SourcePath = $websiteSourceLocation

DestinationPath = "C:\inetpub\OurBakery"

Recurse = $true

Type = "Directory"

DependsOn = "[WindowsFeature]AspNet45"

}

# Create a new website

xWebsite BakeryWebSite

{

Ensure = "Present"

Name = "OurBakery"

State = "Started"

PhysicalPath = "C:\inetpub\OurBakery"

DependsOn = "[File]WebContent"

}

}

}

There are a few points that need to be considered when looking at the code that you pasted in:

* We are using a parameter that will be declared in the script (**Install.ps1**) that calls *this* script. This parameter is used to define where, on the virtual machine, our Bakery web site source code will reside. When we create our configuration .zip file to be uploaded to Azure, we will be placing the source code for the Bakery web site under the **xWebAdministration** directory.
* We are importing the xWebAdministration resource module that we installed earlier:  
  Import-DscResource -Module xWebAdministration
* We are shutting down the DefaultWebSite. We will be replacing the default web site with the Bakery web site.
* We install the Bakery website into the OurBakery directory.

1. Create a new script file by clicking on the **New** icon on the toolbar and save this file as **.\M5-Management\Labs\DesiredStateConfig\Scripts\Install.ps1.**
2. Copy and paste the following code into the file.

PowerShell

#region Load dependent modules

Import-Module Azure -Force

#endregion

#region Initialize variables

$currentDirectory = "[Full path to directory where scripts are located]"

$subscriptionName = "[Name of your Azure subscription]”

$publishSettingsFileName = "[Full name and extension of Azure publishsettings file, NOT including full path]"

$location = "[Data center location]"

$contentInstallPath = "[Location where you xWebAdministration source folder exists]"

$storageAccountName = "[Name of storage account]"

$configurationScript = Join-Path $currentDirectory 'SimpleWebServerConfiguration.ps1'

$configurationName = "InstallWebServer"

$vmName = "[Name of virtual machine]"

$vmServiceName = "${vmName}-Service"

$vmFQDN = "$vmServiceName.cloudapp.net"

# Local Credential for the Azure VM

$userName = "[RDP login name]"

$password = "[RDP password]"

$dscExtensionVersion = "2.8" #you may need to change this depending on which version of DSC is installed

$configurationArchive = [IO.Path]::GetFileName($configurationScript) + ".zip"

$configurationDataPath = Join-Path $currentDirectory "ConfigurationData.psd1"

# This is the location, on the new VM, where the Bakery web site code will be pulled from

$websiteSourceLocation = "C:\Program Files\WindowsPowerShell\Modules\xWebAdministration\BakeryWebsite"

#endregion

#region Initialize Azure Subscription settings in current context

Get-AzureSubscription | Remove-AzureSubscription -Force

# Set the full path to the publishsettingsfile, then import the file and select the subscription and storage

# account to use

$myConfigPath = Join-Path $currentDirectory $publishSettingsFileName

Import-AzurePublishSettingsFile $myConfigPath

Select-AzureSubscription -SubscriptionName $subscriptionName

Set-AzureSubscription -CurrentStorageAccountName $storageAccountName -SubscriptionName $subscriptionName -Verbose

#endregion

#region Obtain Azure Storage Context

$storageAccountKey = (Get-AzureStorageKey $storageAccountName).Primary

$storageContext = New-AzureStorageContext -StorageAccountName $storageAccountName -StorageAccountKey $storageAccountKey -Protocol https

#endregion

# Copy the BakeryWebsite folder into the xWebAdministration folder so it is zipped up as a resource to be deployed to Azure

Copy-Item "..\Source\BakeryWebsite" $contentInstallPath -Recurse

#region Publish the VM DSC configuration

# Typically, what you would do is just zip up the contents of the package and place it immediately into Storage. In

# our case, we need to build a zip file that contains our web Source code.

Publish-AzureVMDscConfiguration $configurationScript -ConfigurationArchivePath $configurationArchive -Force -Verbose

#call the publish cmdlet again, this time a new storage container

#is created called ‘windows-powershell-dsc’ and the .ps1.zip

#file is copied there

Publish-AzureVMDscConfiguration $configurationArchive -Force

#endregion

#region Initialize arguments for Set-AzureVMDscExtension

$configurationArgument = @{

websiteSourceLocation = $websiteSourceLocation;

}

$configurationDataPath = Join-Path $currentDirectory "ConfigurationData.psd1"

# Arguments for Set-AzureVMDscExtension

$arguments = @{

Version = $dscExtensionVersion;

StorageContext = $storageContext;

ConfigurationArchive = $configurationArchive;

ConfigurationName = $configurationName;

ConfigurationArgument = $configurationArgument

ConfigurationDataPath = $configurationDataPath

}

#endregion

#region Apply the Extension to the VM

# Get the instance of the VM

$vm = Get-AzureVM -ServiceName $vmServiceName -Name $vmName -ErrorAction SilentlyContinue

if ($vm)

{

# VM already exists, so apply the configuration using the Extension

$vm | Set-AzureVMDSCExtension @arguments -Verbose | Update-AzureVM

}

else

{

# VM does not exist. Initialize the provisioning config

$vm2012R2images = Get-AzureVMImage | Where-Object {$\_.ImageName -match "2012-R2"}

$vm = New-AzureVMConfig -Name $vmName -InstanceSize Small -ImageName $vm2012R2images[0].ImageName | Add-AzureEndpoint –Name ‘web’ –PublicPort 80 –LocalPort 80 –Protocol tcp

Add-AzureProvisioningConfig -Windows -Password $password -AdminUsername $userName -VM $vm

# Set the DSC Extension properties on the VM object, Create a new VM

$vm | Set-AzureVMDSCExtension @arguments -Verbose | New-AzureVM -Location $location -ServiceName $vmServiceName -WaitForBoot

}

# now that the script has completed, remove the Bakery website from the

# DSC resources

Remove-Item (Join-Path $contentInstallPath "\BakeryWebsite") –Recurse

#endregion

Make sure you read the comments in the script files to understand the various sections in the code.

Pay careful attention to this setting:

$contentInstallPath = “[Location where your xWebAdministration source folder exists]”

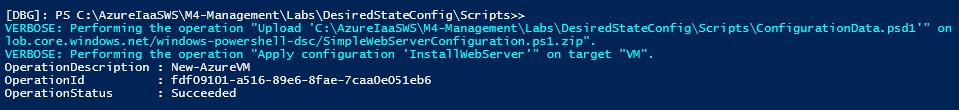
Depending on which operating system you have installed xWebAdministration on, the structure of this folder may be a bit different. On a Windows 10 machine, the xWebAdministration has a sub-directory of ‘1.7.0.0’ and then under that, you see the source files and directories for the xWebAdministration functionality. You need to record the path to where the source files and directories are, because the script will copy the BakeryWebsite code here.

For example, on a Windows 10 machine, the path is:  
C:\Program Files\WindowsPowerShell\Modules\xWebAdministration\1.7.0.0

1. Save the **Install.ps1** script file.

## Task 3 – Running the Scripts

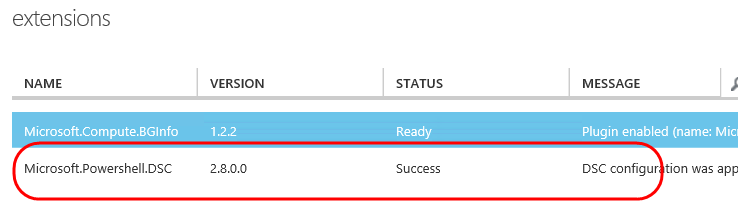
1. Make sure you have filled in the values of your subscription information, etc. in the **Install.ps1** as instructed in the previous steps. With Install.ps1 still open in PowerShell ISE, select the **Debug->Run/Continue** menu item.
2. Creating the VM in Azure and having the configuration script run is going to take several minutes (maybe as long as 10-15 minutes). At this point, you must wait for step to complete before continuing other steps in the lab. Remember, you are not only just creating a VM, but you are also installing IIS and other components, which takes more time.
3. When the script has complete running, in the PowerShell ISE command\output window, it will look similar to this:

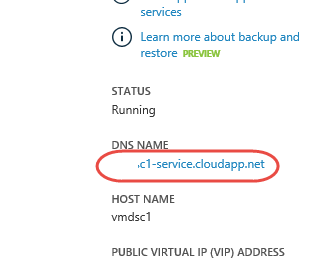


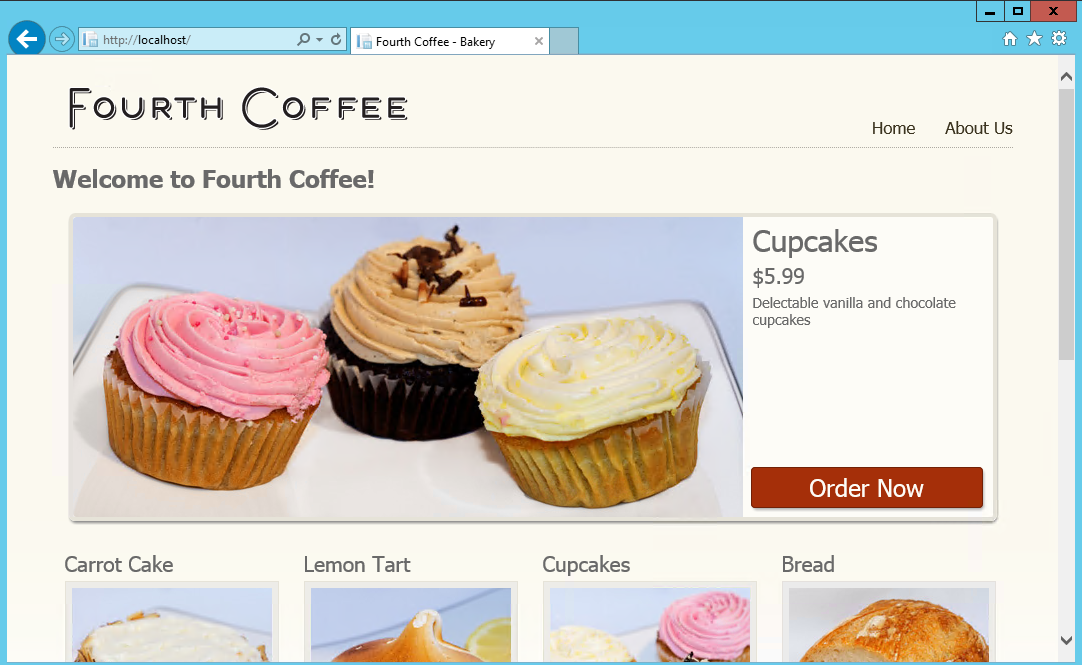
This does not mean the process in Azure has completed, this only means that the script has finished handing off commands for Azure to perform, now Azure will do the rest of the work.

## Task 4 – Confirming VM Configuration

1. Log in to the Azure Management portal.
2. Go to the Virtual Machines menu item and click on the name of the virtual machine that the script has created in order to open the dashboard.
3. Once the dashboard opens, scroll down to the bottom of the dashboard screen. You should be able to view the successful completion of the installation of the DSC configuration. It may take a few minutes for this to appear. (The version you see may be different than the one you see below).

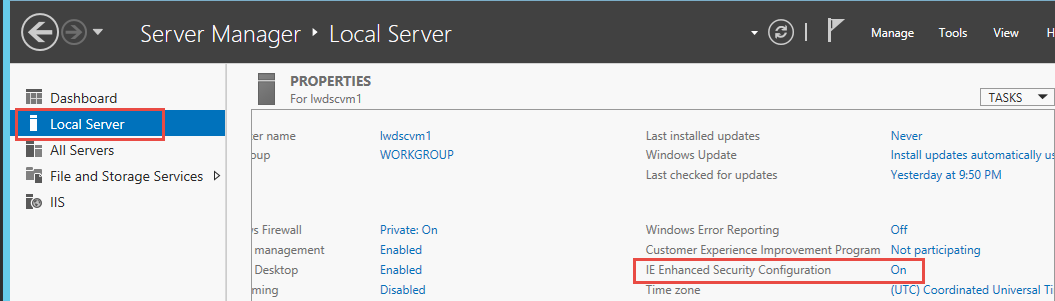


1. For a quick test to see if everything has been deployed correctly, you can type in http://<your-cloud-servicename>.cloudapp.net. You can get the name of your cloud service if you go into the dashboard of your virtual machine. All you need to do is open your browser and type http: in from of the value you get from the dashboard:  
     
   
2. The Fourth Coffee (bakery) site should appear. **Note: This site is not fully functional because it requires a database, connection strings etc and the purpose of this lab was just to demonstrate how to distribute the web site to the virtual machine**.

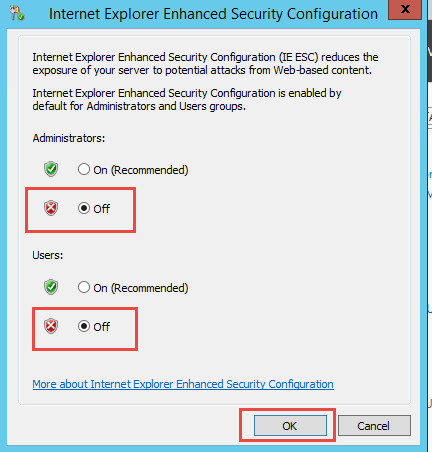


## Task 5 – Remote Desktop into the VM

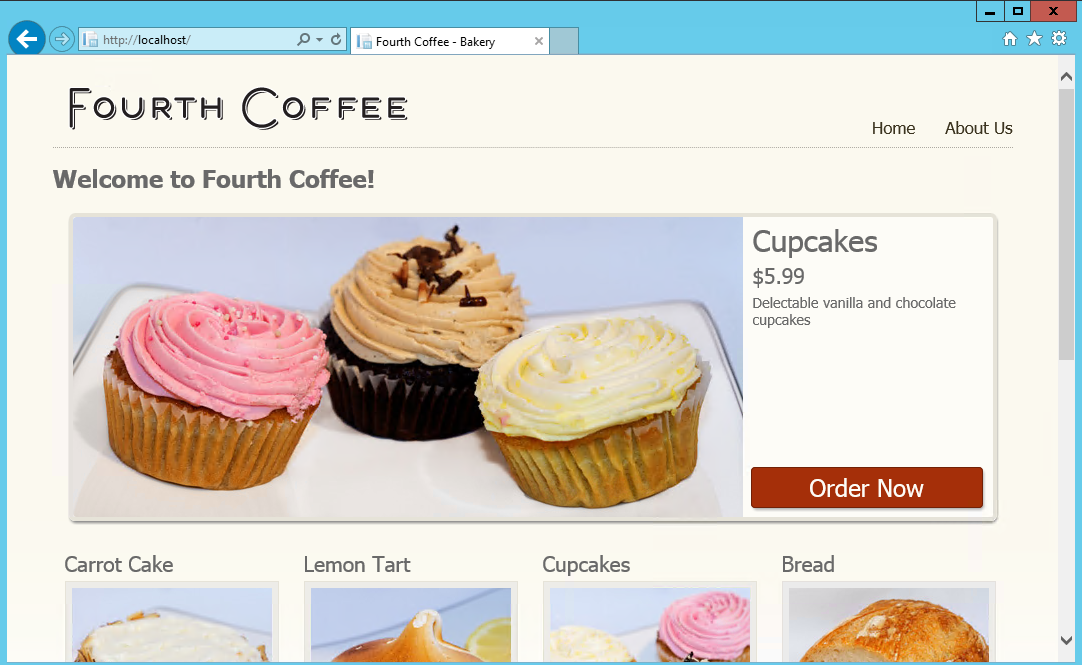
1. Click on the **Connect** button at the bottom of the virtual machine dashboard window and RDP into the virtual machine.
2. Once the virtual machines RDP window opens, open **Server Manager**.
3. Select **Local Server** and then **IE Enhanced Security Configuration**.



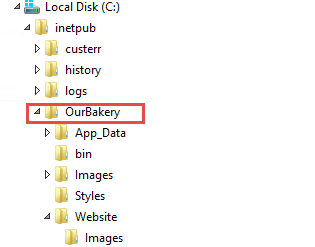
1. Turn security off and select **OK**. Close the Server Manager.



1. Open Internet Explorer and type in <http://localhost>.
2. The Fourth Coffee (bakery) site should appear. **Note: This site is not fully functional because it requires a database, connection strings etc and the purpose of this lab was just to demonstrate how to distribute the web site to the virtual machine**.



1. You can also browse to the C:\inetpub directory to see the location of the OurBakery website.



1. If you needed to debug the DSC process and read the log files, you could browse to the C:\WindowsAzure\Logs\Plugins\Microsoft.Powershell.DSC\<dscversionnumber> directory and read the log files.

