# Module 04 - Azure Automation - State Configuration

## Exercise 1: Create a DSC Configuration

### Introduction

In this lab, we will look at how to create Desired State Configuration (DSC), to configure a Windows Server with IIS, download website source code then deploy the website to the server. Once the configuration is applied, it will result in a ready-to-go web server.

### Scenario

In this lab, we will create a DSC Configuration File to:

- Download Website source files from Azure Storage
- Unzip Website source files on the web server
- Install IIS and ASP.Net on the Web Server
- Disable the default IIS Website
- Create a New Website based on the downloaded source files

### Estimated Time to Complete This Lab

20 minutes

## Task Description

1. [] On your lab machine, open **Visual Studio Code** from the Start menu or on the taskbar.
2. [] Click on **Open Folder** and then click **Select Folder** to select the current folder.
3. [] Click on the new file button and give the file the name **DeployWebsite.ps1**. This will tell Visual Studio Code that you are writing a PowerShell script and allow Intellisense and formatting features to work.
4. [] Enter the following code into the Script Pane:

```
Configuration DeployWebsite {

    Node localhost {

    }
}
```

5. [] This code means we are creating a PowerShell DSC configuration, called DeployWebsite, to be set to the `localhost` where DSC is applied.
6. [] The first thing we want the DSC configuration to do, is download the website source files from an Azure Storage Account (you will upload this later ). To do this, we are going to make use of the **cAzureStorage** DSC resource. You can read about this resource at: https://www.powershellgallery.com/packages/cAzureStorage.

7. [] Inside of the curly bracket of **DeployWebsite** {} on the line above `Node localhost` in your script, type in the following to import the cAzureStorage module:

```
    Import-DscResource -ModuleName cAzureStorage
```

8. [] After the curly bracket following `Node localhost`, enter the following code:

```
        cAzureStorage DownloadWebsiteZip {
             Path                    = $outputPath
             StorageAccountName      = $StorageAccountName
             StorageAccountContainer = $StorageAccountContainer
             StorageAccountKey       = $StorageAccountKey
             Blob = $downloadFileName
        }
```

9. [] This block of code called **DownloadWebsiteZip** will use the cAzureStorage resource to download a file. Notice, we have not told it which file to download or where from. We have just created some variables, which will be set later.

10. [] Let's check your code. Now, it should look like this:

```
Configuration DeployWebsite {

    Import-DSCResource -ModuleName cAzureStorage

    Node localhost {

        cAzureStorage DownloadWebsiteZip {
             Path                    = $outputPath
             StorageAccountName      = $StorageAccountName
             StorageAccountContainer = $StorageAccountContainer
             StorageAccountKey       = $StorageAccountKey
             Blob = $downloadFileName
        }

    }
}
```

11. [] Next, we want DSC to unzip the source files, which are downloaded by DownloadWebsiteZip. To do this, write in the following code, making sure you put this **below** the closing curly bracket which closes the DownloadWebsiteZip block, called **UnzipWebsiteFiles:**

```
        Archive UnzipWebsiteFiles
        {
            Ensure = "Present"
            Path = "$($outputPath)\$($downloadFileName)"
            Destination = $outputPath
            DependsOn = "[cAzureStorage]DownloadWebsiteZip"
        }
```

12. [] Notice how this contains the **DependsOn** property. This means that this action cannot happen until DownloadWebsiteZip has completed. This makes sense, as we need to download the zip file, before it is

unzipped.

13. [] Your code should now look like this:

```
Configuration DeployWebsite {

    Import-DSCResource -ModuleName cAzureStorage

    Node localhost {

        cAzureStorage DownloadWebsiteZip {
             Path                    = $outputPath
             StorageAccountName      = $StorageAccountName
             StorageAccountContainer = $StorageAccountContainer
             StorageAccountKey       = $StorageAccountKey
             Blob = $downloadFileName
        }

        Archive UnzipWebsiteFiles
        {
            Ensure = "Present"
            Path = "$($outputPath)\$($downloadFileName)"
            Destination = $outputPath
            DependsOn = "[cAzureStorage]DownloadWebsiteZip"
        }

    }
}
```

14. [] Next, enter in the following two blocks of DSC configuration, which will Install IIS and ASP.Net on the Server. Be sure to enter these after the closing curly bracket from the UnzipWebsiteFiles block:

```
        WindowsFeature IIS
        {
            Ensure          = "Present"
            Name            = "Web-Server"
            DependsOn = "[Archive]UnzipWebsiteFiles"
        }

        WindowsFeature AspNet45
        {
            Ensure          = "Present"
            Name            = "Web-Asp-Net45"
            DependsOn = "[Archive]UnzipWebsiteFiles"
        }
```

15. [] Notice how both these actions use **DependsOn** to make sure they happen after the website files have been unzipped by the UnzipWebsiteFiles action.

16. [] At this point we have told DSC to download and unzip the website files, and install IIS and ASP. Next, we want to use DSE to create a new website based on the unzipped website source files. To configure websites, we need to make use of a DSC resource **xWebAdministration**. You can read information about this module at **https://www.powershellgallery.com/packages/xWebAdministration/**.

17. [] In your DSC file, write in the following statement to import the **xWebAdministration** resource. Write

this under the existing Import-DscResource statement which you already have:

```
    Import-DscResource -ModuleName xWebAdministration
```

18. [] Next, enter in the following configuration blocks to create a new site and stop the default IIS website. Put these blocks below the closing curly bracket of the **AspNet45** block:

```
        xWebsite StopDefaultSite
        {
            Ensure          = "Present"
            Name            = "Default Web Site"
            State           = "Stopped"
            PhysicalPath    = "C:\inetpub\wwwroot"
            DependsOn       = "[WindowsFeature]IIS"
        }

        xWebsite DeploySimpleWebsite
        {
            Ensure          = "Present"
            Name            = "DSCDemo"
            State           = "Started"
            PhysicalPath    = $outputPath
            DependsOn       = "[xWebsite]StopDefaultSite"
        }
```

19. [] Let's check your code to see what it should look like now:

```
Configuration DeployWebsite {

    Import-DSCResource -ModuleName cAzureStorage
    Import-DSCResource -ModuleName xWebAdministration

    Node localhost {

        cAzureStorage DownloadWebsiteZip {
             Path                    = $outputPath
             StorageAccountName      = $StorageAccountName
             StorageAccountContainer = $StorageAccountContainer
             StorageAccountKey       = $StorageAccountKey
             Blob = $downloadFileName
        }

        Archive UnzipWebsiteFiles
        {
            Ensure = "Present"
            Path = "$($outputPath)\$($downloadFileName)"
            Destination = $outputPath
            DependsOn = "[cAzureStorage]DownloadWebsiteZip"
        }

        WindowsFeature IIS
        {
            Ensure          = "Present"
            Name            = "Web-Server"
            DependsOn = "[Archive]UnzipWebsiteFiles"
```

```
        }

        WindowsFeature AspNet45
        {
            Ensure          = "Present"
            Name            = "Web-Asp-Net45"
            DependsOn = "[Archive]UnzipWebsiteFiles"
        }

        xWebsite StopDefaultSite
        {
            Ensure          = "Present"
            Name            = "Default Web Site"
            State           = "Stopped"
            PhysicalPath    = "C:\inetpub\wwwroot"
            DependsOn       = "[WindowsFeature]IIS"
        }

        xWebsite DeploySimpleWebsite
        {
            Ensure          = "Present"
            Name            = "DSCDemo"
            State           = "Started"
            PhysicalPath    = $outputPath
            DependsOn       = "[xWebsite]StopDefaultSite"
        }



    }
}
```

20. []**IMPORTANT!** You may have noticed that there is a variable in the configuration named $outputPath. This is the directory where the Website source files are unzipped to and used to create the new IIS site. We need to set a location in the script. To do this, write in the following line **above** the **DownloadWebsiteZip** configuration block (but under Node `localhost`):

```
        $outputPath = "C:\Website"
```

21. [] As we discussed earlier, the **DownloadWebsiteZip** configuration block requires some input, to configure where to download the website files from. To set these, we will configure input parameters to the DSC configuration. We want to enter in this code on line 3, which should be after the opening curly bracket for **Configuration DeployWebsite**

```
    param(
        [Parameter(Mandatory=$true)]
        [string]$downloadFileName,

        [Parameter(Mandatory=$true)]
        [string]$StorageAccountName,

        [Parameter(Mandatory=$true)]
        [string]$StorageAccountContainer,

        [Parameter(Mandatory=$true)]
        [string]$StorageAccountKey
    )
```

22. [] It's highly recommended that you check your code to ensure it's correct. Visual Studio Code has an excellent compare feature if you would like to try it. You will find a complete example in your lab files at **c:\Labs\module4\DeployWebsite.ps1**

23. [] Now we can save your DSC configuration. Save it on your desktop as **DeployWebsite.ps1**

# Exercise 2: Upload Website Files to Azure Storage

## Introduction

The DSC Configuration you created in the previous exercise will securely download the Contoso website source code and install it on a Web Server. In this Exercise, we need to create an Azure Storage account and upload the website source code to it. This will be where DSC deploys the website from.

## Scenario

In this lab, we will:

- Create an Azure Storage Account
- Upload the website source code
- Make a note of the storage account key for secure access

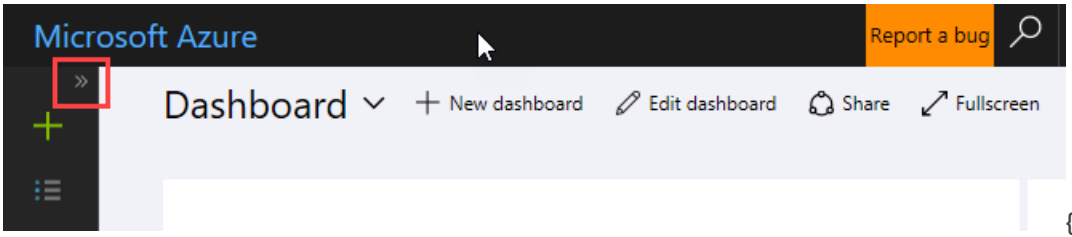
## Estimated Time to Complete This Lab

15 minutes

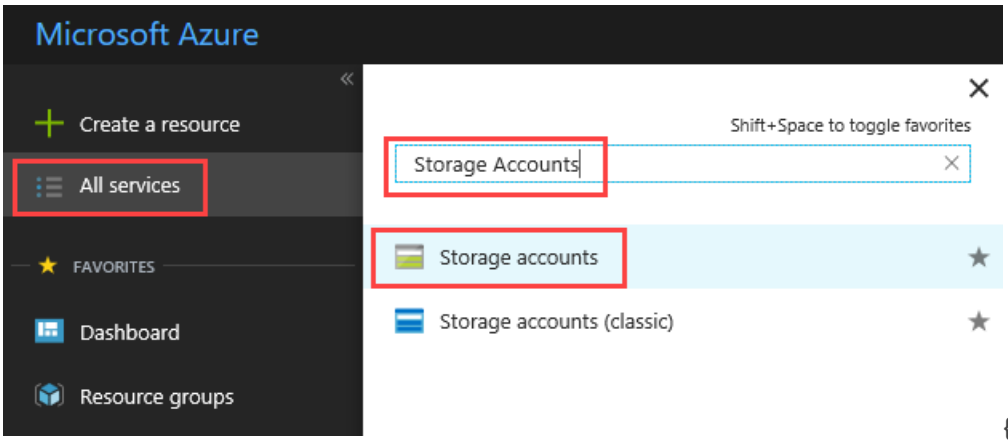
# Task Description

1. [] From your workstation, open the Azure Portal at **https://portal.azure.com**

2. [] In the top left corner of the Azure portal, click this `>>` to expand out the left panel. From the panel click **All Services**. Search for **Storage Accounts**. Click **Storage Accounts**.

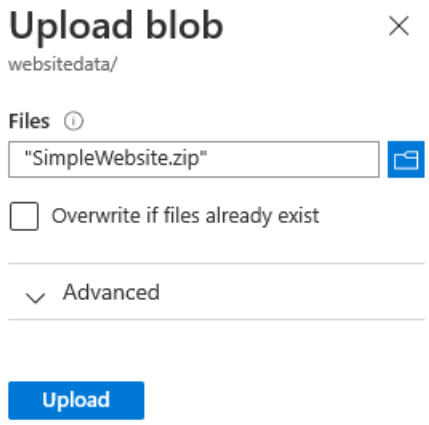



{400}




{400}

3. [] In the Storage accounts pane, click **+Add**
4. [] In the **Create storage account** panel, fill in the details of your new storage account. Leave all options as their default apart from the following:
    - **Resource Group**. Select **Create new** and give the name: **WebsiteStorage**
    - **Storage Account Name** must be *GLOBALLY* unique, can contain only *lower case a-z and numbers* and be a maximum of *24 characters* in length. Try coming up with a unique name by putting your name first, then some random characters.
    - **Region** pick out a location from the allowed Azure Pass locations. This does not have to be in the same location as your VMs or Automation Account
    - **Performance**. Select Standard
    - **Redundancy**. Select Locally-redundant storage (LRS)
5. [] Once filled in, click **Review** and then **Create** to create the storage account:
6. [] After a minute or two, your new storage account will be created. From the **Storage accounts** pane, click Refresh. You should now be able to see the new storage account you created and click it.
7. [] Once you have clicked on the Storage Account, from the many options on the left, under **Data storage** click **Containers**. Click the **+ Container** button to create a new container. (A container is similar to a folder)
8. [] Name your new container **websitedata** and set the Access type as **Private** then click **OK**
9. [] Once it is created, click the new *websitedata* container and click the **Upload** button
10. [] In the **Upload blob** pane, upload the **SimpleWebsite.zip**, which is in your lab files **c:\Labs\Module4** folder. Click the **Upload** button.




{400}

11. [] After a moment, you should see that the SimpleWebsite.zip file is uploaded successfully.

12. [] Once your file is uploaded, go back to the Storage Account and from the menu on the left, click **Access Keys** under **Security + Networking**:

13. [] We need to make a note of your storage account name and the access key. It will be fed into the DSC configuration in the next lab. Open Notepad on your lab machine. From the Access Keys, copy your **Storage account name** and the value of either **key1** or **key2** (You have to click on **Show keys**). Paste each of these into notepad and save it for the next exercise:


{400}

# Exercise 3: Create DSC Node Configuration

## Introduction

Now that you have created your DSC Configuration, we will upload and compile this into a DSC Node Configuration in DSC. This needs to be completed before the DSC configuration can be applied to a server.

## Scenario

In this lab, we will:

- Upload DSC Configuration to Azure Automation
- Import DSC Resources into Azure Automation Assets
- Compile DSC Configuration into DSC node Configuration

## Estimated Time to Complete This Lab

20 minutes

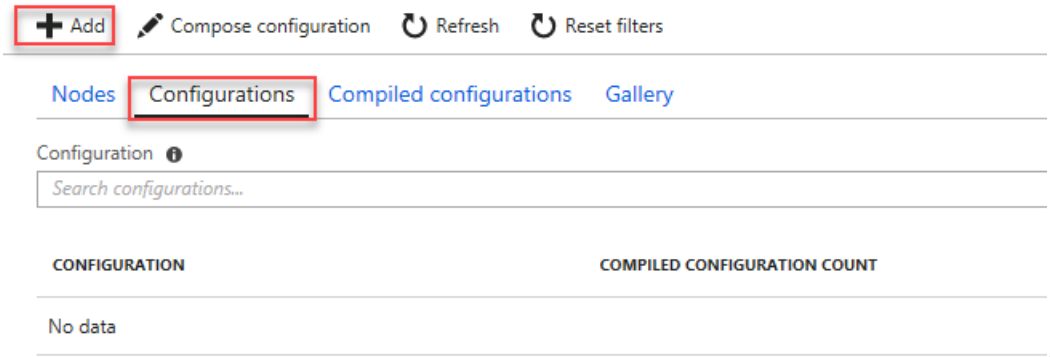
# Task Description

1. [] From your workstation, open the Azure Portal at **https://portal.azure.com/**

2. [] In the Azure Portal, navigate to your Automation Account **ContosoAutomationAccount**

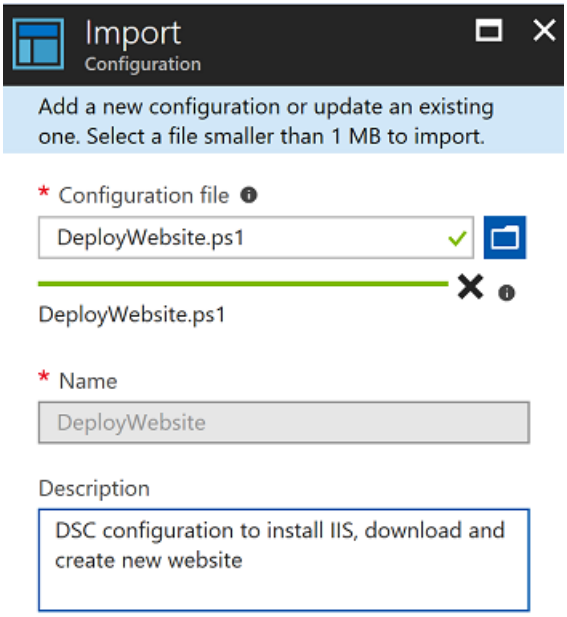
3. [] From the menu pane in your ContosoAutomationAccount, click **State Configuration (DSC)** under **Configuration Management**

4. [] From DSC Configurations, click on the **Configurations** tab and click **Add**




{400}

5. [] Upload the **DeployWebsite.ps1** file, which you saved in the previous Exercise. Feel free to enter a Description and then click **OK**.

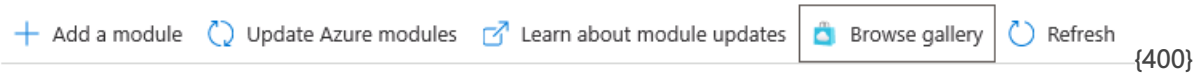



{400}

6. [] You should now see that your **DeployWebsite** DSC configuration has a Compiled Configuration Count of 0.

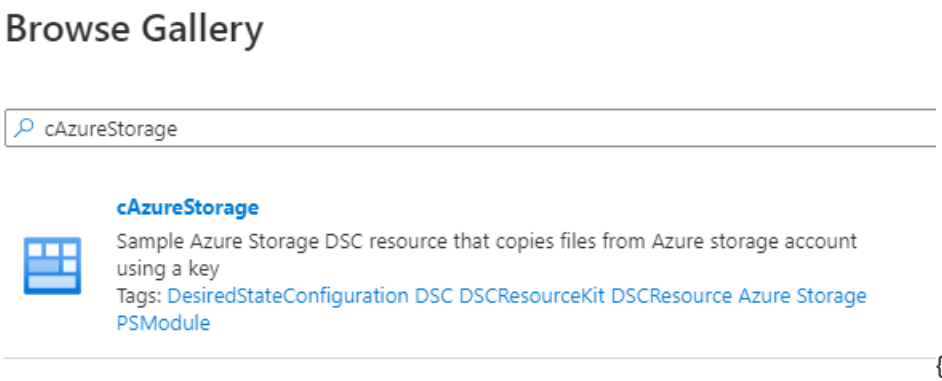
7. [] Now that we have uploaded the DSC Configuration, we can compile it. However, there is another step first - Remember we used 2 DSC Resources in the DSC Configuration you built in Exercise 1? Those Resources now need to be imported into your Automation Account. From the menu on the left of the screen, click **Modules** under **Shared Resources**

8. [] Azure Automation allows us to directly import resources from the PowerShell Gallery. To do this, click the **Browse gallery** button:



{400}

9. [] In the **Browse Gallery** pane, search for **cAzureStorage** and click it:

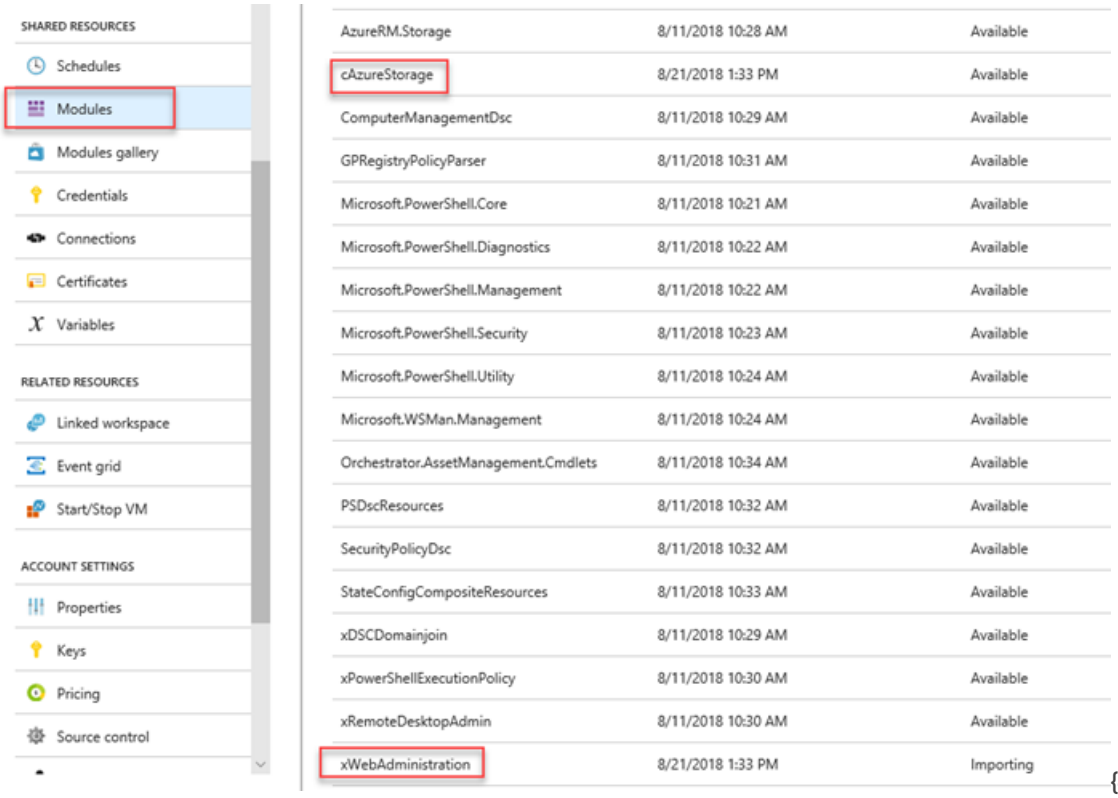


{400}

10. [] In the **cAzureStorage** window click the **Import** button. Click **OK** when prompted to import the resource. This will import the resource into your Azure Automation account.

11. [] Go back to the **Browse Gallery** pane. Repeat the previous few steps to search for and import the module called: **xWebAdministration**

12. [] When you import a Resource to your Automation Account, it can take several minutes for the activities to be extracted. Go back to the **Modules** section of your Automation Account and see the module status:



{400}

13. [] Click refresh every couple of minutes and wait until both modules you imported are in status **Available**

14. [] Now that the resources are imported, we can compile your DSC Configuration. In your Automation Account, go back to the **State Configuration (DSC)** section.

15. [] Click on **Configurations** and the **DeployWebsite** configuration you imported earlier:

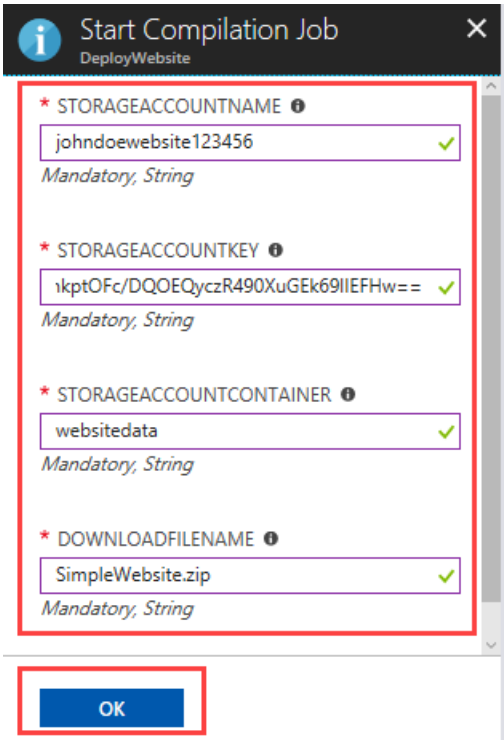
16. [] Click **Compile.**

{400}

17. [] Remember in Exercise 1, we defined that the DSC configuration will require some input parameters? Now that we are compiling the configuration, we must supply those values. These parameters are to define the file to download, which we uploaded to Azure Storage in the previous lab. Fill in the following values:

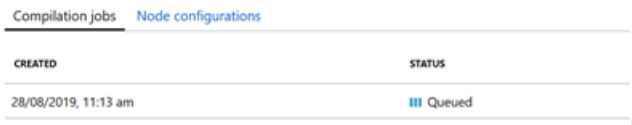
    - **STORAGEACCOUNTNAME**: This is the name of the storage account you created in the previous lab. You should have made a note of this in notepad. **NOTE: This must be in lower case**
    - **STORAGEACCOUNTCONTAINER:** websitedata
    - **STORAGEACCOUNTKEY**: This is the key to access your storage account. You should have made a note of this in notepad.
    - **DOWNLOADFILENAME**: SimpleWebsite.zip

18. [] Once you have filled in each field, click **OK**



{400}

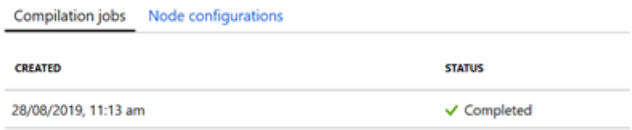
19. [] You should now see that a Compilation job is Queued.



{400}

20. [] It will take several minutes to process this. Close the pane, and reopen after a few minutes and you

should see that the DSC Configuration is now successfully **Completed**:



{400}

# Exercise 4: Deploy Azure VM and Apply DSC

## Introduction

Now that you have created your DSC Configuration and compiled it into a DSC node Configuration, it can be applied to a Virtual Machine. Applying the DSC Node Configuration to a VM will automatically complete the configuration.

## Scenario

In this lab, we will:

- Deploy a new Azure Virtual Machine
- Apply DSC Configuration to the Virtual Machine
- Verify DSC applied successfully

## Estimated Time to Complete This Lab

45 minutes

# Task Description

1. [] From your workstation, open the Azure Portal at **https://portal.azure.com**
2. [] Click on the Cloud Shell icon in the top right corner.


{400}

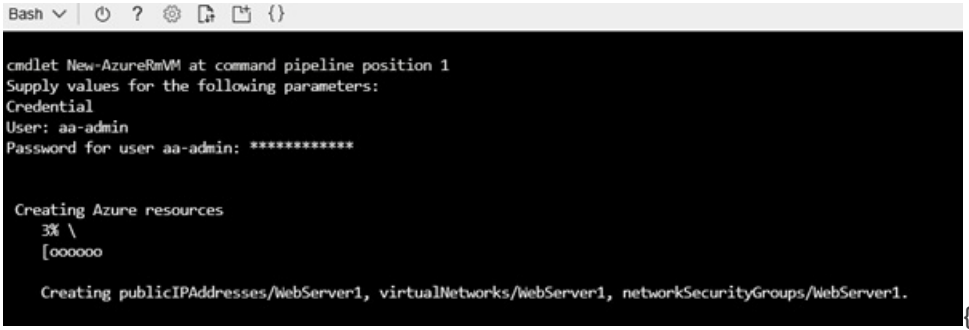
3. [] Select **Bash**
4. [] If a prompt appears saying you have no storage - click on **Create Storage**
5. [] Enter *pwsh* at the prompt and press **Enter** to start PowerShell.
6. [] Enter the code below to generate a new virtual machine an press **Enter**

```
Get-AzLocation | Select Location
$location = ""
New-AzVM -ResourceGroupName ContosoWebServers -Name WebServer1 -Location $location  -
PublicIpAddressName webserver1
```

> [!note] You will have to enter a location and press Enter before the virtual machine is created.

7. [] When prompted enter *aa-admin* for the username, and *R3dDwarf2017* for the password.

8. [] Monitor the output as the virtual machine is created and wait until it returns back to the PowerShell prompt.



{400}

9. [] Minimise the Cloud Shell.

10. [] Now that your VM is running, we can go ahead and apply the DSC Node Configuration to it. Start by navigating back to your **ContosoAutomationAccount** Automation Account in the Azure Portal.

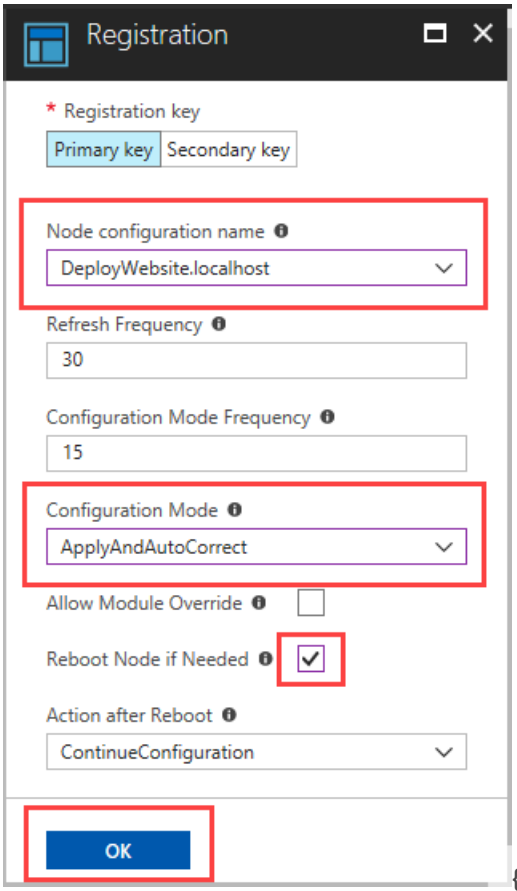
11. [] From your Automation Account, click **State configuration (DSC)**.

12. [] Ensure the **Nodes** tab is selected and click the **Add** button.

13. [] Click the **WebServer1**, Click **+Connect**.

14. [] Set the following values in the Registration pane: (leave all other values at default):

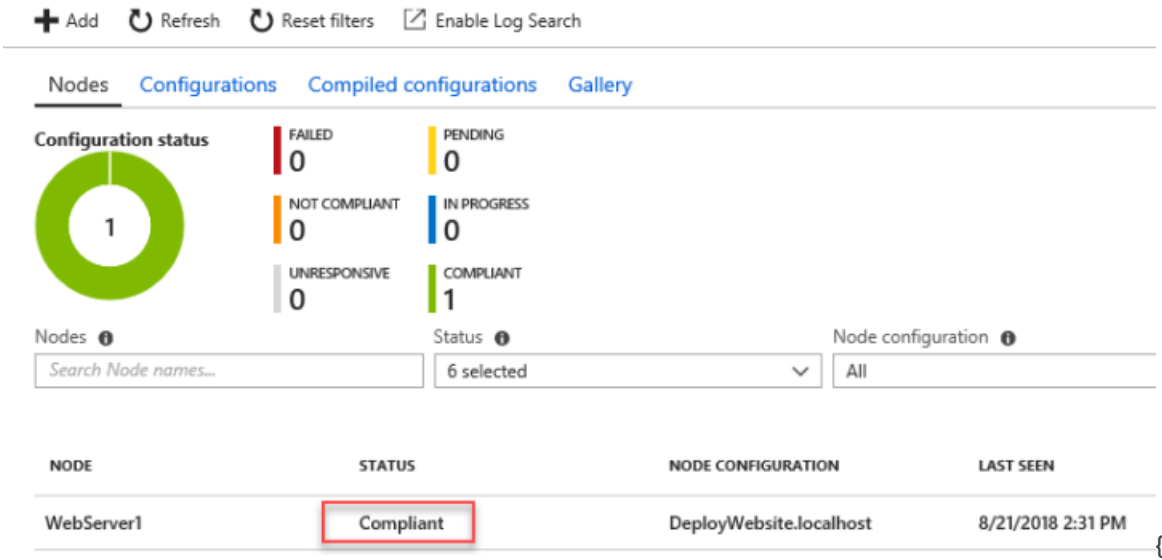
    - **Node Configuration Name**: DeployWebsite.localhost
    - **Configuration Mode:** ApplyAndAutoCorrect
    - **Reboot Node if Needed:** Check this box



{400}

15. [] At this point, the Azure DSC Extension is getting installed on your VM, then synchronizing with your DSC Automation Account. This will take several minutes to complete.

16. [] In the **State configuration (DSC)** blade, click the **Refresh** button. After a few minutes, you should see **WebServer1** with a STATUS of **Compliant**.

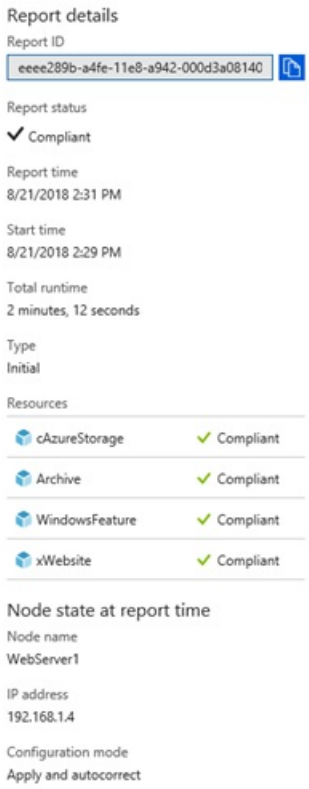


{400}

17. [] From the DSC Nodes pane, click the **WebServer1** node. You may notice that the Initial STATUS is **Compliant**, but the Consistency has as a STATUS of **In progress**. This is because we must wait for the DSC client to receive and apply the configuration. This may take around 15 minutes. You can check the status by going back to the **DSC Nodes** pane, clicking **Refresh** then clicking on **WebServer1**

18. [] From the Report, you should be able to see each of the configurations which were applied to the node

successfully. You can click each to see detailed information. You may also click the **View raw report** button at the top, to see the verbose data from when the configuration was applied. This is useful data for troubleshooting.




{400}

19. [] Finally, let's navigate to the website, to make sure it works. Before we do, we need to open the Network Security Group on the VM, to allow HTTP traffic in. Open the Cloud shell as before.

> [!note] You may have to enter *pwsh* at the prompt again to launch PowerShell.

20. [] Enter the code below and press **Enter** to add the rule.

```
$nsg = Get-AzNetworkSecurityGroup -ResourceGroupName ContosoWebServers
$nsg | Add-AzNetworkSecurityRuleConfig -Name "HTTP_In" -Priority 1100 -Protocol TCP -Access
Allow -SourceAddressPrefix * -SourcePortRange * -DestinationAddressPrefix * -
DestinationPortRange 80 -Direction Inbound | Set-AzureRmNetworkSecurityGroup
Get-AzPublicIpAddress -ResourceGroupName ContosoWebServers | select Name,IpAddress
```

21. [] Copy the Ip address returned into a browser - you should see the website displayed.


DSC Demo Website
13.65.207.246
Congratulations!
You've sucessfully deployed this website via DSC

{400}