WorkshopPLUS

Microsoft Azure Infrastructure as a Service (IaaS)

ARM Template Authoring & Deployment

Student Lab Manual

V2.4, October 9, 2016

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.

© 2014 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

[Prerequisites 4](#_Toc511219125)

[Exercise 1 – Deployment Using a Built in Visual Studio Template 5](#_Toc511219126)

[Task 1 – Create the Azure Resource Group project 5](#_Toc511219127)

[Task 2 – Deploy to Azure 10](#_Toc511219128)

[Task 3 – Confirm Deployment of Virtual Machines with an Azure Load Balancer 13](#_Toc511219129)

[Exercise 2 – Using PowerShell with ARM Templates 16](#_Toc511219130)

[Task 1 – Authoring ARM templates 16](#_Toc511219131)

[Task 2 – Deploying ARM templates using PowerShell 18](#_Toc511219132)

**Introduction to Microsoft Azure Resource Manager**

## Prerequisites

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

* Microsoft V[isual Studio 2015 Professional or Ultimate edition](http://www.microsoft.com/visualstudio/) with Update 3
* [Microsoft Azure SDK for .NET (VS 2015) - 2.](http://www.microsoft.com/windowsazure/sdk/)9.5
* Microsoft Azure PowerShell
* A Microsoft Azure subscription

In this lab, you’ll be deploying an Azure Resource group with two virtual machines behind an **internal** load balancer. Using an internal load balancer means that only traffic from your private network will be load balanced and all other traffic will be blocked. These components will be deployed using a built in Visual Studio template. You’ll also be editing the template to customize your deployment.

You'll learn:

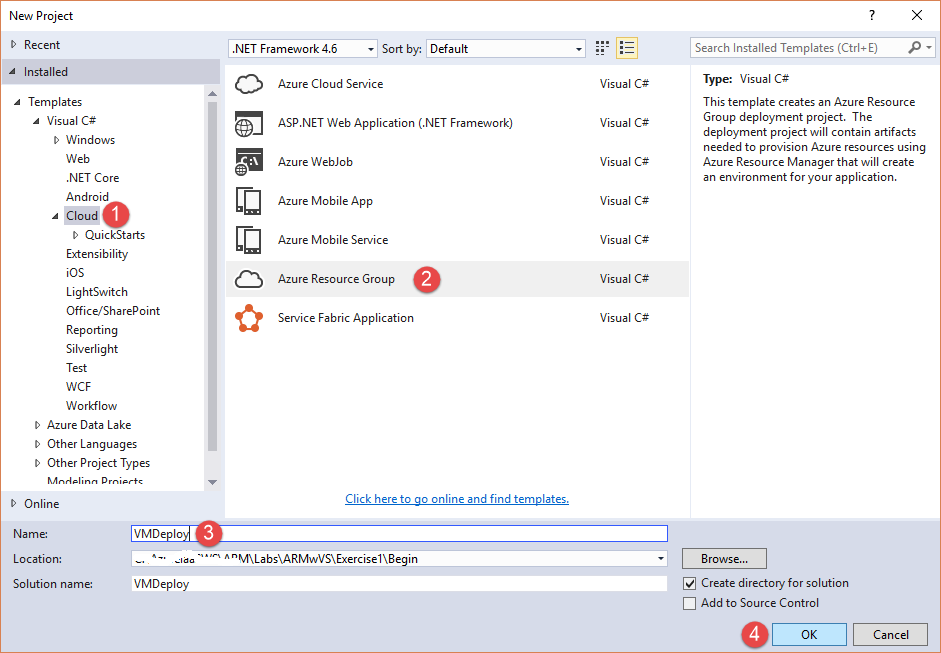
* How to deploy Azure resources using a built in Visual Studio template.
* How to author an ARM template using Visual Studio.

# Exercise 1 – Deployment Using a Built in Visual Studio Template

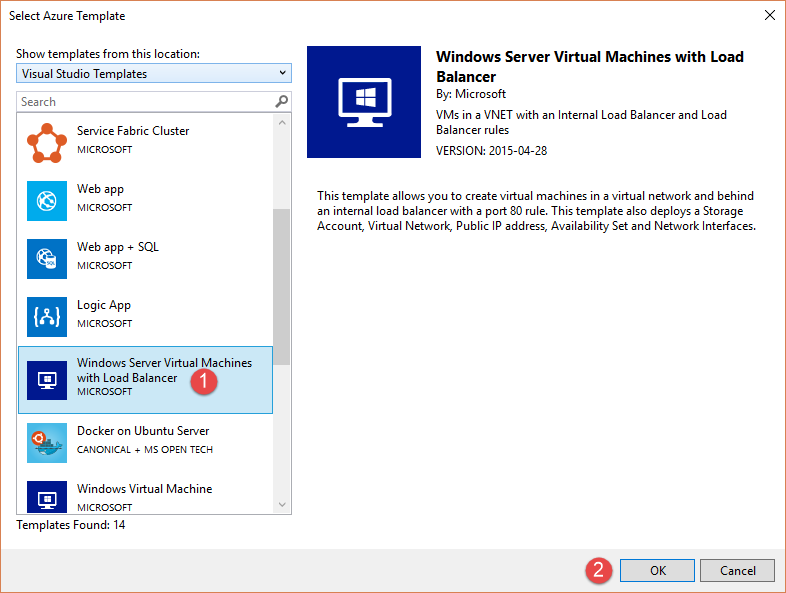
## Task 1 – Create the Azure Resource Group project

Although the template for an Azure Resource Manager (ARM) deployment can be created from scratch, it is quicker and easier to use the built-in templates that are provided by Visual Studio or a GitHub repository.

1. From the Task Bar, open **Visual Studio** as an Administrator.
2. Select **File | New | Project** and then:
   1. Select **Visual C# | Cloud**.
   2. Select **Azure Resource Group**.
   3. Give the project the name **VMDeploy**.
   4. Click the **OK** button.



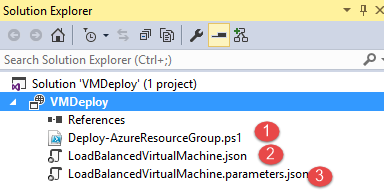
1. In the *Select Azure Template* dialog box:
   1. Select **Windows Server Virtual Machines with Load Balancer**.
   2. Click the **OK** button.



1. View the contents of the project that has been created.

You will see three files:

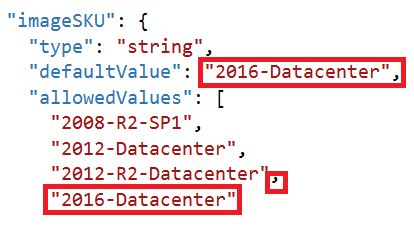
1. **Deploy-AzureResourceGroup.ps1** – this is the PS script that is used to deploy the .json template file.
2. **LoadBalancedVirtualMachine.json** – this is the main template file for the project.
3. **LoadBalancedVirtualMachine.parameters.json** – this is the template parameter file that is used as input by the main template file.



1. Left-click the main template file **LoadBalancedVirtualMachine.json**. This will open the file in the top middle pane of Visual Studio. The contents of this file describe among other things, the resources that you would like to deploy and their configurations.

If you receive a prompt to update Visual Studio, click “**Not Now**”.

1. Scroll down to the **imageSKU** tag and edit the file as follows:
   1. Type in **“2016-Datacenter”** as a value for the **defaultValue** tag.
   2. Add a comma after “2012-R2-Datacenter”
   3. In the following line type in “2016-Datacenter” as an additional value in the allowedValues tag.



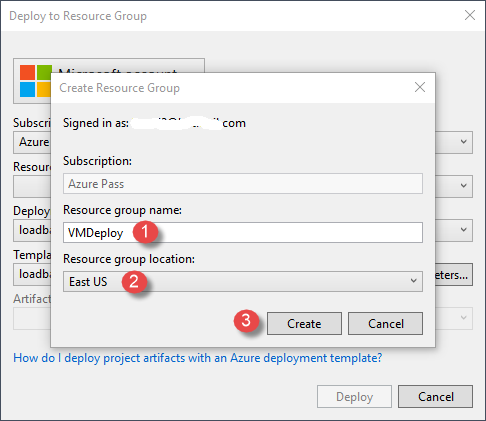
1. Scroll down to the **variables** tag and:
   1. type in **“LBVM”** as a value for the **vmNamePrefix** tag
   2. Save the changes.

Editing this value allows you to specify a custom name prefix for your load balanced virtual machines.



## Task 2 – Deploy to Azure

1. Right-click on the **VMDeploy** project in the **Solution Explorer** and select **Deploy | New…**
2. In the *Deploy to Resource Group* dialog box, if you have not already signed in to Azure, click the **Add an account…** button and sign in.
3. Once you are signed in to Azure, select the dropdown in the *Resource Group* box, click **<Create New…>**, and in the dialog box that appears:
   1. Create a new, unique resource group name.
   2. Select the location closest to you.
   3. Click the **Create** button.

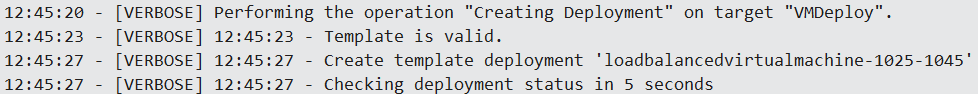


1. Now click the **Edit Parameters** button. The parameters that you enter here will be passed to the deployment, and will also be placed in the LoadBalancedVirtualMachine.parameters.json file in your Visual Studio project.

Edit the parameters that will be required for your deployment.

|  |  |
| --- | --- |
| 1. **adminUsername** – Enter a RDP admin username 2. **adminPassword** – Enter a RDP admin password. *Note the icon at the right-hand side of the adminPassword field, if you click this, you will find links to instructions on how to save your password in Azure KeyVault.* 3. **imageSKU** – the version of Windows VM build to use. *Note the 2016-Datacenter build which you have added in the LoadBalancedVirtualMachine.json file is now available in the drop-down menu.* 4. **vmSize** – the size of the VM. Leave the default value. 5. **numberOfInstances** – the number of VMs that you would like to create which will be placed behind the Azure load balancer. Leave the default value. |  |

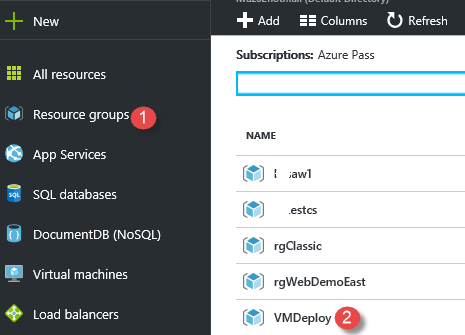
1. Click the **Deploy** button. If the Edit Parameters box pops up again, make sure that you’ve selected the **‘Save passwords as plain text in parameters file’** checkbox and click **Save**.
2. Observe the Visual Studio Output pane for a status update on your deployment. Wait until the deployment is successfully completed before continuing.



1. Close Visual Studio.

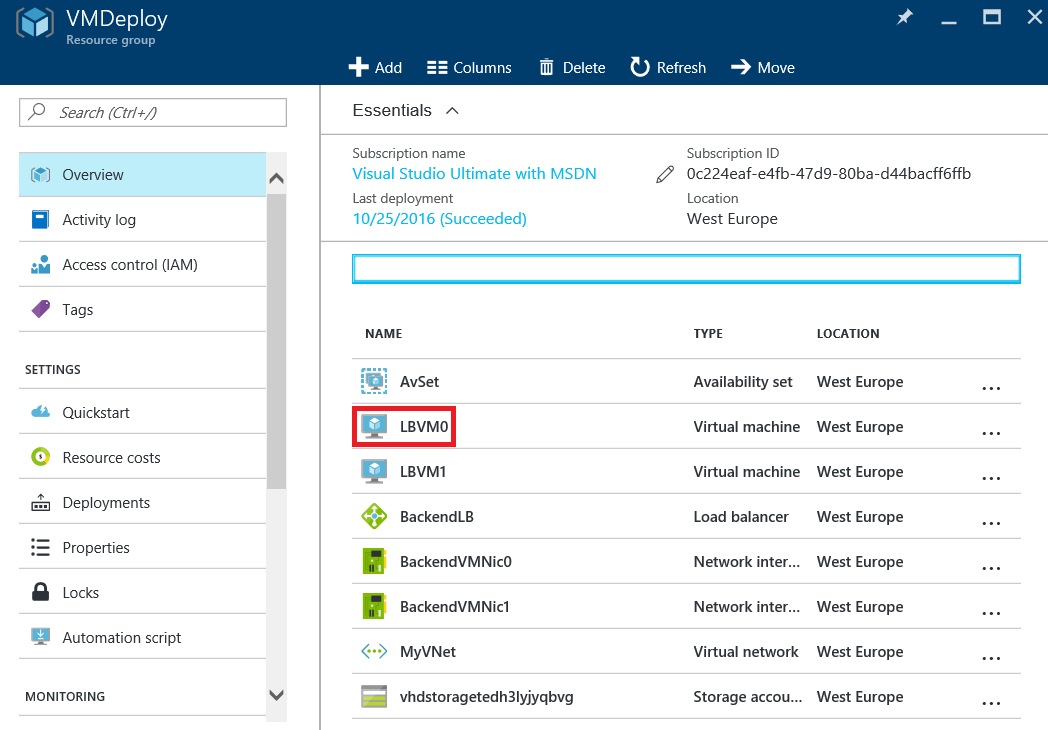
## Task 3 – Confirm Deployment of Virtual Machines with an Azure Load Balancer

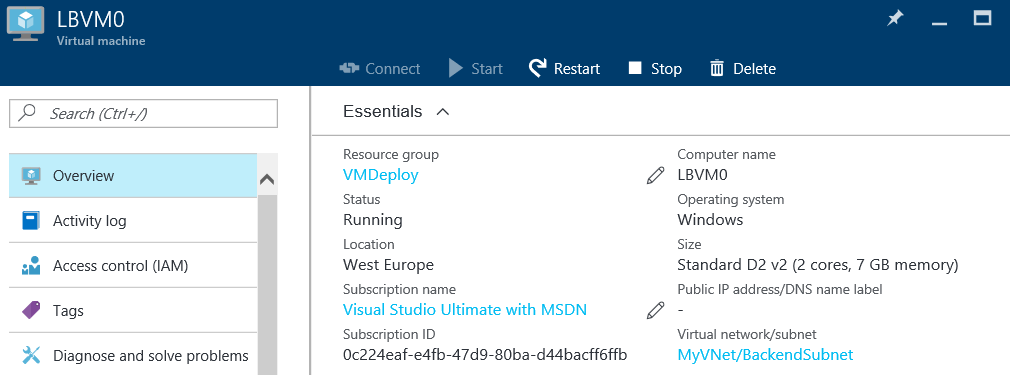
1. Log in to the Azure portal at https://portal.azure.com.
2. On the left-hand side:
   1. Click **Resource groups**.
   2. Click on the name of the resource group you created for the deployment.



1. Your *Resource group* blade will open. It will display all resources that belong to this resource group.

Click on the name of the first VM (ex. LBVM0). The *Virtual machine* blade will appear.





From the VM blade, there are a variety of things that you can do, such as:

1. Add data disks (select **Disks** menu item)
2. Add Role-Based Access Control permissions
3. Stop and Restart the machines
4. Notice in the toolbar on the *Overview* blade that the **Connect** button indicates the use of a **Private IP**. This is because this virtual machine does not have a public IP address. This means that you can only RDP into this machine from another machine that is connected to the same network. You will learn more about this in later labs.
5. Close the *Virtual machine* blade.

# Exercise 2 – Using PowerShell with ARM Templates

In Exercise 1, you used Visual Studio and one of its built-in templates to deploy a set of VMs with an **internal** load balancer.

In a typical IT environment, developers will distribute the JSON files to someone else whose responsibility it is to do the deployments to Azure. From an IT perspective, this deployment is generally performed through the use of PowerShell.

In this exercise, you will be redeploying the previous solution but with a **public** load balancer so that traffic can be load balanced from a public network, such as the Internet and with virtual machines now including a public IP address so that you can RDP into them using the public IP. This solution will be deployed using PowerShell as opposed to Visual Studio. **Note: The main code has already been written for this template. You will be editing common settings.**

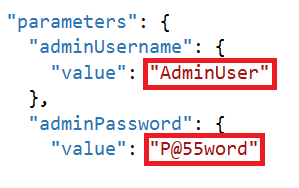
## Task 1 – Authoring ARM templates

Prior to redeploying our solution, we need to author our Main and Parameters templates to configure our desired settings for our solution. In this exercise, we will be adding in a Username and Password for our deployed VM’s, selecting a Windows Server 2016 Datacenter build as their operating systems and configuring a Geo-Redundant storage account.

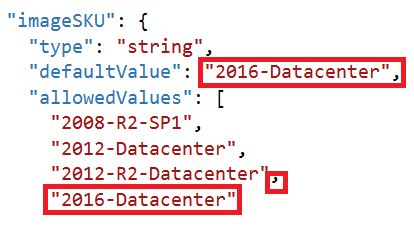
1. Navigate to **C:\AzureIaaSWS\M2 - Azure Resource Manager\Labs\Source\VMDeployPS** and double-click the **PublicLoadBalancedVirtualMachine.parameters.json** file.

This file should be automatically configured to open using Visual Studio. If you are prompted to choose an application to open this file, choose **Visual Studio**.

1. Scroll down to the **adminUsername** tag and add in a value (e.g. AdminUser).
2. Scroll down to the **adminPassword** tag and add in a value (e.g. P@55word).



1. Save the changes (**File |** **Save All**) and close Visual Studio.
2. Return to the **C:\AzureIaaSWS\M2 - Azure Resource Manager\Labs\Source\VMDeployPS** folder and double-click the **PublicLoadBalancedVirtualMachine.json** file.
3. If necessary, make the following changes to the file:
   * Scroll down to the **imageSKU** tag and ensure that **“2016-Datacenter”** is entered as a value for the **defaultValue** tag.
   * Add a comma after **“2012-R2-Datacenter”** and in the following line type in **“2016-Datacenter”** as an additional value in the **allowedValues** tag.



1. Scroll down to the **variables** tag and type in **“Standard\_GRS”** as a value for the **storageAccountType** tag.
2. Save the changes and close Visual Studio.

## Task 2 – Deploying ARM templates using PowerShell

1. Just as it was necessary to To open PowerShell ISE as an Administrator:
   1. On the taskbar, right-click the **PowerShell** icon.
   2. Right-click **Windows PowerShell ISE**.
   3. Click **Run as administrator**.
   4. At the *User Account Control* prompt, click **Yes**.
2. In the PowerShell ISE command prompt window, type in **Add-AzureRmAccount** and press Enter.
3. Enter your credentials into the login dialog box and click **Next** or press Enter.
4. Enter your password and click **Sign in** or press Enter.
5. Once logged in, type **cd “C:\AzureIaaSWS\M2 - Azure Resource Manager\Labs \Source\VMDeployPS”** into the PowerShell ISE command prompt window and press Enter.

This will change the working directory to the location of the ARM templates and PowerShell deployment script.

1. Type **New-AzureRmResourceGroup -Name [YourResourceGroupName] -Location [YourResourceGroupLocation]** e.g. **New-AzureRmResourceGroup -Name VMDeployPS -Location westeurope** and press Enter.

This will create a new resource group which you will deploy your solution into. The built-in deployment script does not include this step so it must be carried out prior to deployment.

Note that it is possible to copy and paste commands from the lab manual to the lab interface. Be certain to edit/add commands as appropriate.

1. Type the following command and press Enter:

**New-AzureRmResourceGroupDeployment -Name [DeploymentName] -ResourceGroupName [ResourceGroupName]**

**-TemplateFile "C:\AzureIaaSWS\M2 - Azure Resource Manager\Labs\Source\VMDeployPS\PublicLoadBalancedVirtualMachine.json"**

**-TemplateParameterFile "C:\AzureIaaSWS\M2 - Azure Resource Manager\Labs\Source\VMDeployPS\PublicLoadBalancedVirtualMachine.parameters.json"**

e.g. New-AzureRmResourceGroupDeployment -Name VMDeployPS -ResourceGroupName VMDeployPS `

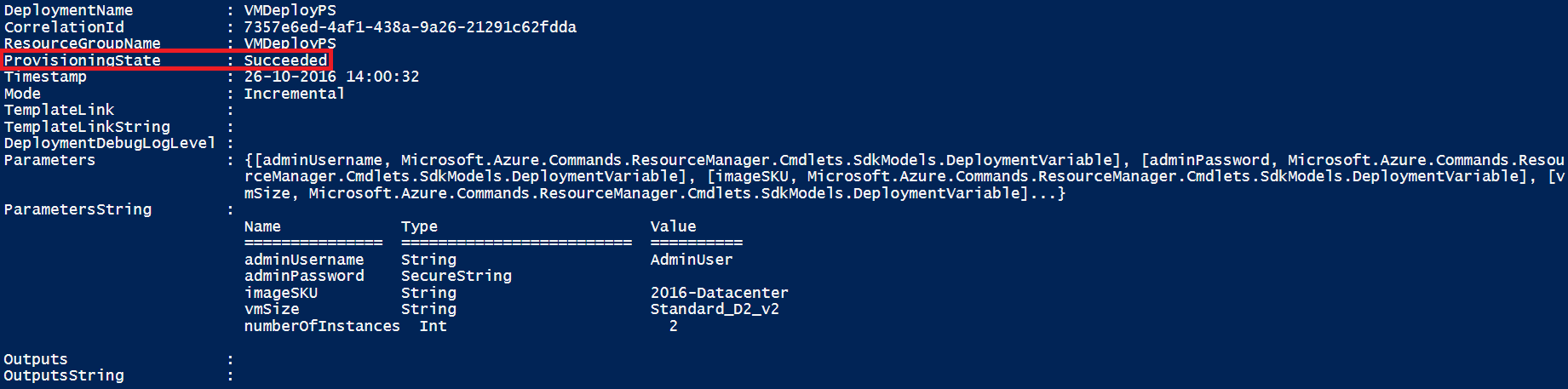
-TemplateFile "C:\AzureIaaSWS\M2 - Azure Resource Manager\Labs\Source\VMDeployPS\PublicLoadBalancedVirtualMachine.json" `

-TemplateParameterFile "C:\AzureIaaSWS\M2 - Azure Resource Manager\Labs\Source\VMDeployPS\PublicLoadBalancedVirtualMachine.parameters.json"

1. When prompted to confirm, click **Yes**.

The deployment of your solution has now been initiated. Refresh the Azure portal and confirm that the newly created resource group is being populated. The deployment should take between 4 – 5 minutes to complete.

1. Confirm that your deployment has successfully completed by reviewing the output of the PowerShell command prompt window.



This is the end of this lab. Do not delete these resources, as they may be used in later labs. However, you should ensure that any existing VMs are in a stopped (deallocated) state, to conserve costs.