DEPLOYING APPLICATIONS TO AN AZURE VM WITH A CUSTOM SCRIPT EXTENSION
GETTING STARTED WITH CUSTOM SCRIPT EXTENSIONS
THOMAS MITCHELL
www.thomasmitchell.net

DEPLOYING APPLICATIONS TO AN AZURE VIRTUAL MACHINE WITH A CUSTOM SCRIPT EXTENSION

Learn how to leverage Custom Script Extensions to deploy an application to an Azure virtual machine.

Thomas Mitchell

www.thomasmitchell.net

CONTENTS

ABOUT THE AUTHOR

Hello. My name is Tom Mitchell. I am a 20+ year veteran of the IT industry. I own and run thomasmitchell.net and teach courses like this one and this one.

Throughout the course of such a long an interesting career, I have built a rather in-depth skill base. My skillset spans over a dozen different IT disciplines but my specialties include:

- Microsoft Exchange
- Microsoft Office 365
- Microsoft Active Directory
- Microsoft Azure
- Infrastructure Services

I have designed and architected the smallest solutions and some global solutions. I have deployed and implemented said solutions as well as supported them. My expert-level knowledge of these and other disciplines such as virtualization and storage allow me to meticulously design and efficiently implement/troubleshoot all kinds of IT solutions and infrastructures.

In addition to the coveted MCSE: Cloud Platform and Infrastructure certification, I also carry several other industry certifications, including multiple Microsoft MCSA certifications.

My superior ability to see things from a strategic perspective allows me to architect solutions that closely align with business needs. By avoiding "tunnel vision" that focuses on "now" when designing solutions, I can ensure that my solutions stand the test of time and grow with the business.

INTRODUCTION

In a production environment, it will sometimes be necessary to deploy an application to several virtual machines quickly and with an identical configuration. An elegant remote-installation solution in cases such as these is the use of Custom Script Extensions for Windows

Leveraging Custom Script Extensions allows you to, among other things, remotely install applications right from your PowerShell session. By storing the application installer file and an installation script in an Azure storage account, you can easily deploy the application without the need to login to each server and manually install it, making life far easier on yourself.

In this tutorial you will learn how to:

- Deploy an Azure storage account and connect to it remotely
- Provision blob storage and file share storage
- Create an installation script
- Configure a Custom Script Extension to install an application
- Create a VM that uses the Custom Script Extension
- Confirm functionality of the Custom Script Extension

When you finish this tutorial, you will have learned a valuable skill that many IT professionals have overlooked.

CUSTOM SCRIPT EXTENSION OVERVIEW

A Custom Script Extension is a piece of software which downloads pre-configured scripts to a virtual machine and runs those scripts on the VM. Custom Script Extensions are typically used to automate the configuration of server settings after a VM has been deployed. Extensions can also be used to install software automatically and to handle various management tasks.

Scripts used by the extension can be downloaded from Azure Storage, GitHub, or even provided to the Azure portal when the extension runs. This tutorial covers the use of scripts downloaded from Azure Storage.

The Custom Script Extension itself works with both Windows and Linux virtual machines and it can be run via PowerShell, the Azure CLI, and even the Azure Portal. It can also be called from the Azure Virtual Machine REST API. We'll be covering the use of Custom Script Extensions in a Windows environment, using PowerShell, in this tutorial.

REQUIRED INSTALLATION FILES

Deploying an application remotely, using a Custom Script Extension, requires at least two files: the application's installer file (usually an EXE or MSI) and a PowerShell script that is used to call the application installer file.

INSTALLER FILE

In this tutorial, we will be remotely installing MalwareBytes on an Azure virtual machine. As such, the MalwareBytes installer must be acquired. Visit this URL and download the free version of MalwareBytes and to your desktop and note the full name of the file.

INSTALLER SCRIPT

An installer script, called "InstallMWB.ps1" will be used later in this tutorial to run MalwareBytes installer. You can download this installer script here. Don't do anything with it yet. You will edit it with Notepad later so that it reflects your environment.

PROVISION AZURE STORAGE ACCOUNT

Prior to setting up and configuring a Custom Script Extension, the Azure storage account, blob container, and file share must all be setup. The blob container will hold the installer script and the file share will hold the application's installation file (.exe). All resources in this tutorial will be deployed into the EastUS region and in a resource group called VMLab.

NOTE: If you are following along, please be sure to perform the steps in this section in one sitting, as some steps rely on variables that get set in prior steps. Starting the exercises in this section and completing them later will likely not work.

RESOURCE GROUP SETUP

If you don't already have a resource group called VMLab provisioned, connect to your Azure tenant via PowerShell (see instructions) and run the command below:

```
New-AzureRmResourceGroup -Name VMLab -Location EastUS
```

The command above creates a resource group called "VMLab" in the "EastUS" Azure region.

STORAGE ACCOUNT SETUP

Azure storage accounts house blob storage and file storage, among others. Before setting up the blob storage (which will host the installer script) and the file share (which will host the MalwareBytes installer file), a storage account must first be provisioned.

Run the commands below to provision a storage account in your tenant (you need to be connected to your tenant via PowerShell before running the command):

Provision Storage Account

Run the command below to create a storage account in the "VMLab" resource group. Before running this command, replace "mystorageaccount" with something unique (ie. mystorageaccount56356) and make a note of what you call it. This is necessary because storage account names must be unique across the entire Azure landscape.

```
$storageAccount = New-AzureRmStorageAccount -ResourceGroupName VMLab `
```

```
-Name "mystorageaccount" `
-Location EastUS `
-SkuName Standard_LRS `
-Kind Storage
```

After command above completes, you must set a storage account context. The easiest way to do so is to load the current context into a variable and reference the variable moving forward.

Set Storage Account Context

Run the command below to load the current storage context that you are working in, into a variable that can be referenced throughout the rest of these exercises.

```
$ctx = $storageAccount.Context
```

Once the storage account has been provisioned (using the commands above), the blob container and file container can be provisioned.

CONTAINER SETUP

A storage account alone doesn't help us. In order to automate the installation process of an application on an Azure virtual machine via a Custom Script Extension, the installer script (the actual extension) and the application installer executable both have to be uploaded to publicly-accessible storage.

The two containers required for this are a Blob Container and a File Container. The Blob Container will eventually host the installer script (.PS1) and the File Container will eventually host the MalwareBytes installation executable (.EXE).

Over the next few sections, we'll cover the deployment of both the Blob Container and File Container in the Azure storage account.

BLOB CONTAINER SETUP

After provisioning the storage account, a blob container needs to be provisioned so that it can be used to store the installer script that you created earlier. The script will be downloaded and run by the virtual machine from the blob container.

Provision a blob container called "scripts" by running the command below:

```
New-AzureStorageContainer -Name scripts -Context $ctx -Permission blob
```

The command above creates a blob container called “scripts” and sets the permissions to “blob”. Setting permissions to “blob” allows read access to blob data throughout a container through anonymous request without providing access to container data.

FILE SHARE SETUP

After setting up the blob container that will host the installer script, you can setup the file share that will host the MalwareBytes installer executable that you downloaded earlier.

To create the file share in the Azure storage account, run the command below:

```
New-AzureStorageShare `
  -Name fileshare `
  -Context $ctx
```

When the command above completes, a file share called “fileshare” will be provisioned in the Azure storage account.

DIRECTORY SETUP

In keeping with best practices, you will want to create a folder within the newly-provisioned file share so that the MalwareBytes installer executable can be stored in it. You can also store other installer files in it as well for future use.

To create a folder called “software”, run the command below:

```
New-AzureStorageDirectory `
  -Context $ctx `
  -ShareName "fileshare" `
  -Path "software"
```

When the command above completes, a new folder will be created in the file share. This new folder will be called “software”, and it will be where the Malwarebytes installer executable is uploaded in the next section.

UPLOAD FILES

Once the file share has been provisioned, the MalwareBytes installer executable needs to be uploaded to the share. In addition, the installer script needs to be edited and uploaded to the blob container that you provisioned.

UPLOAD THE MALWAREBYTES INSTALLER

To upload the MalwareBytes installer, ensure that you are connected to your Azure tenant via PowerShell and then run the command below. Before running the command, replace "pathtoinstaller" with the actual path to the MalwareBytes installer that you downloaded (ie. "C:\users\tmitchell\downloads\mb3-setup-consumer-3.5.1.2522-1.0.365-1.0.5188.exe"). Replace MBSSetup.exe with the actual name of the installer file that you downloaded (ie. mb3-setup-consumer-3.5.1.2522-1.0.365-1.0.5188.exe).

```
Set-AzureStorageFileContent `
  -Context $ctx `
  -ShareName "fileshare" `
  -Source "pathtoinstaller" `
  -Path "software/MBSSetup.exe"
```

Running the command above uploads the MalwareBytes installer file to the "software" directory within the "fileshare" share in the Azure storage account.

UPLOAD THE INSTALLER SCRIPT

The installer script that will be used to run the installer needs to be modified and uploaded to the "scripts" blob container. To create this script, download this file, edit it with Notepad, and replace all instances of *mystorageaccount* with the name of the storage account that you provisioned earlier. Replace all instances of *password* with the actual pass key (key1) for your storage account.

To retrieve your storage account pass key, run the command below. Be sure to replace "mystorageaccount" with the name of the storage account you created earlier.

```
Get-AzureRmStorageAccountKey -ResourceGroupName "VMLab" -AccountName "mystorageaccount"
```

Replace *mbssetup.exe* with the actual name of the MalwareBytes installer that you downloaded.

After making your edits, save the file as InstallMWB.ps1. Make sure it's saved as .PS1 and not .TXT. The script can be uploaded to the blob container called "scripts" in the Azure storage account, using the command below. Be sure to replace "pathtoscript" with the path to your edited script (ie. c:\users\tmitchell\InstallMWB.ps1) prior to running the command.

```
Set-AzureStorageBlobContent `
```

```
 -File "pathtoscript" `
 -Container scripts `
 -Blob "InstallMWB.ps1" `
 -Context $ctx
```

The command above uploads the installer script to the “scripts” container in the Azure storage account. This script will be downloaded and run by the virtual machine.

CREATE A VIRTUAL MACHINE

To deploy MalwareBytes to a virtual machine with a Custom Script Extension, you will first need a virtual machine deployed in Azure. To do so, connect to your Azure tenant via PowerShell (see instructions) and run the Get-Credential command below to configure a local administrator username and password for the virtual machine:

```
$cred = Get-Credential
```

Once the credentials are created, the virtual machine can be created by running the New-AzureRmVm command. The command below provisions a virtual machine called “cseVM” in the EastUS location. The virtual machine is deployed to a resource group called “VMLab” on a virtual network called “MyVnet”.

```
New-AzureRmVm `
    -ResourceGroupName "VMLab" `
    -Name "cseVM" `
    -Location "EastUS" `
    -VirtualNetworkName "myVnet" `
    -SubnetName "mySubnet" `
    -SecurityGroupName "MyNSG" `
    -PublicIpAddressName "csePublicIP" `
    -Credential $cred
```

It takes a few minutes for the resources and VM to be created. To demonstrate how a Custom Script Extension works, we will ultimately install MalwareBytes remotely.

DEPLOY THE APPLICATION

Once the virtual machine has been provisioned and is running, use Set-AzureRmVmExtension to install the Custom Script Extension on the virtual machine. The extension downloads and runs the PowerShell script, called InstallMWB.ps1, from the Azure blob container called “scripts”. It then runs the script, which installs MalwareBytes using the MalwareBytes installer file that is stored in the file share within the azure storage account.

Before running the command below, replace “mystorageaccount” with the name of your storage account that you provisioned earlier. In addition, replace *mystoragekey* with your own key (key1 from the previous exercise).

```
Set-AzureRmVmCustomScriptExtension `

 -ResourceGroupName VMLab `

-Location EastUS `

-VMName CseVM `

-Name MyMWBInstall `

-TypeHandlerVersion "1.9" `

-StorageAccountName mystorageaccount `

-StorageAccountKey mystoragekey `

-FileName InstallMWB.ps1 `

-ContainerName scripts `

-Run InstallMWB.ps1
```

When specifying the .PS1 script in the *-FileName* and *-Run* switches, it’s important to ensure that the capitalization matches the actual files that you uploaded. The name of the PowerShell script is CASE-SENSITIVE.

The installation of MalwareBytes can take a few minutes to complete. As such, wait about 15 minutes and then login to virtual server and confirm MalwareBytes is installed and running. If you followed the instructions in this tutorial as written, you should see a little MalwareBytes icon in the taskbar, indicating that the software has been installed and is running.

WRAP UP

In this tutorial, you automated the installation of an application (MalwareBytes) on a virtual machine in Azure. You learned how to:

- Deploy an Azure storage account via PowerShell
- Connect to a storage account remotely via PowerShell
- Provision blob storage and file share storage via PowerShell
- Create an installation script
- Configure a Custom Script Extension to install an application
- Create a VM that uses the Custom Script Extension

The skills that you have learned are skills that elude many other IT professionals, making you that more valuable.

For more in-depth technology learning, visit me at any one of my online properties:

- Where I Blog: My Website
- Where I Teach: Udemy & Learn with Tom
- Where I Socialize: LinkedIn & Facebook
- My Free Tutorials: YouTube
- My LinkedIn Group: Tutorials & Training

Thanks for reading and happy learning!