POWERSHELL IN HANDS-ON LABS

Modules Available in Hands-on Labs Azure PowerShell

The following table lists the modules currently installed for use by the Azure Automation accounts when executing pre and post-deployment scripts in Azure PowerShell during provisioning. If the content package you are developing has a dependency on a different version of a module, please let us know and we will work with you to ensure it is made available. Likewise, if you do not see a module you need, please let us know.

Module Name	Version(s) Available	Last Updated in HOL Platform
Azure	4.3.1	08/23/2017
Azure.Storage	3.3.1	08/23/2017
AzureAD	2.0.0.33	08/22/2017
AzureMLPS	1.0.0.0	08/22/2017
AzureRM.ApiManagement	4.3.1	08/23/2017
AzureRM.Automation	3.3.1	08/23/2017
AzureRM.CognitiveServices	0.8.4	08/23/2017
AzureRM.Compute	3.3.1	08/23/2017
AzureRM.DataLakeAnalytics	3.3.1	08/23/2017
AzureRM.DataLakeStore	4.3.1	08/23/2017
AzureRM.HDInsight	3.3.1	08/23/2017
AzureRM.Insights	3.3.1	08/23/2017
AzureRM.IoTHub	2.3.1	09/07/2017
AzureRM.KeyVault	1.1.2	08/23/2017
AzureRM.Network	4.2.0	08/23/2017
AzureRM.PowerBIEmbedded	3.3.1	08/23/2017
AzureRM.profile	3.3.1	08/23/2017
AzureRM.Resources	4.3.1	08/23/2017
AzureRM.Sql	3.3.1	08/23/2017
AzureRM.Storage	3.3.1	08/23/2017
AzureRM.StreamAnalytics	3.3.1	08/23/2017
AzureRM.Websites	3.3.1	08/23/2017
Microsoft.PowerShell.Core		8/3/2017
Microsoft.PowerShell.Diagnostics		8/3/2017
Microsoft.PowerShell.Management		8/3/2017

Microsoft.PowerShell.Security		8/3/2017
Microsoft.PowerShell.Utility		8/3/2017
Microsoft.WSMan.Management		8/3/2017
MSCorp.Immersion.Cmdlets	1.0.6401.41470	5/31/2017
MSOnline	1.0	08/22/2017
Orchstrator.AssetManagement.Cmdlets	1.0	8/3/2017
SqlServer	21.0.17152	08/23/2017

Hands-on Labs Immersion PowerShell Module

The Immersion PowerShell module is downloaded to all Hands-on Labs jump host environments in order to provision the Guide application, download payload files, and execute long running scripts. There are a few public functions exported for use in other scripts invoked during the provisioning process. These functions are available for all content vendors.

Import the Module

Template Deploy Usage

When running scripts in conjunction with the ARM template deployment, the Immersion module will be in context at the execution of your script. Thus, usage of the functions is available without an Import statement.s

Post-Deploy Usage

When running scripts post-deployment, using the content.json property “post_deploy_script”, the Immersion module must be imported before you can use the exported functions.

Use the following in command in your PowerShell script to import the module if your script is not executed from the ARM template:

```
# Find and import Immersion helper module
Get-ChildItem -Recurse -Path 'C:\Packages' -Include 'Immersion.psm1' | Select -
First 1 | Import-Module
```

Exported Functions

Get-CustomScriptConfig

This function reads the properties from the Microsoft Custom Script Extension’s runtime settings and protectedSettings; then, generates a container for both labelled $config.public and $config.private respectively. Private settings are made available in a decrypted state.

Invoke-CustomScript

This function uses properties from the Microsoft Custom Script Extension’s runtime settings to download payload files; download and run additional scripts; and install the Guide app on the jumphost.

The ‘fileUris’ property must include a reference to the Immersion PowerShell module. The ‘otherFileUris’ property can be used to download additional payload data to the jumphost from Azure storage. When the ‘installGuide’ property is true, this function will invoke the installation for the Guide app. The ‘storageSentinelName’ property can be used to manage long running processes. Finally, this function invokes the script referenced by the ‘scriptFileUri’ property of the public settings with a unique hash to ensure the referenced script file is run only once. The ‘storageAccountName’ and ‘storageAccountKey’ are required private properties for access to Azure storage resources.

All activity performed in the Invoke-CustomScript function is recorded in a time-stamped log file on the jumphost’s temporary disk. All log files are copied to the lab’s storage account at script completion.

Read-AzureBlobFile

Taking parameters for StorageAccountName, StorageAccountKey, BlobUrl and OutFile (destination file name) this function provides a mechanism for copying data from an Azure Storage location to the jumphost. If no OutFile parameter is specified, the content from the response is expanded onto standard output in the cmdlet.

StorageAccountName and StorageAccountKey can be injected from the private runtime settings of the template.

Write-AzureBlobFile

This function provides a mechanism for writing files to a specified storage account location from the jumphost. There are two parameter sets available: FromSourceFile which expects the StorageAccountName, StorageAccountKey, BlobPath, and SourceFile (a path on the jumphost); and FromSourceBytes which expects the StorageAccountName, StorageAccountKey, BlobPath, and SourceBytes (an array of bytes representing the file to upload.)

StorageAccountName and StorageAccountKey can be injected from the private runtime settings of the template.