PowerShell for Sysadmins

Workflow Automation Made Easy

by Adam Bertram

errata updated to print 7

Page	Error	Correction	Print corrected
15	For more information about strict mode, run `Get Help Set-StrictMode Examples`.	For more information about strict mode, run `Get-Help Set-StrictMode -Examples`.	Print 2

31	`PS> $users = @{`
	`abertram = 'Adam Bertram';`
	`raquelcer = 'Raquel Cerillo'`
		`PS> $users = @{`
		`abertram = 'Adam Bertram';`
		`raquelcer = 'Raquel Cerillo';`
			Print 3

41	Because parameters passed via`ByValue` depend on the type of input, each parameter passed via `ByValue` can be **one type only.**	Because parameters passed via`ByValue` depend on the type of input, each parameter passed via `ByValue` can be **passed by `ByValue`.**	Print 4
49	Returns `True` if the second value is "in" the **second**. You can use this to determine whether a value is inside an array.	Returns `True` if the second value is "in" the **first**. You can use this to determine whether a value is inside an array.	Print 4

52	`if (-not (Test-Connection -ComputerName $servers[0] -Quiet -Count 1)) { ❶`
	`Write-Error -Message "The server $servers[0] is not responding!"`
	`} elseif ($servers[0] -eq $problemServer) ❷`
	`Write-Error -Message "The server $servers[0] does not have the right file!"`
		`if (-not (Test-Connection -ComputerName $servers[0] -Quiet -Count 1)) { ❶`
		`Write-Error -Message "The server $servers[0] is not responding!"`
		`} elseif ($servers[0] -eq $problemServer) { ❷`
		`Write-Error -Message "The server $servers[0] does not have the right file!"`
			Print 3

57	`$servers = @('SERVER1','SERVER2','SERVER3','SERVER4','SERVER5')`
	`for ($i = 0; $i -lt $servers.Length; $i++) {`
	`$servers[$i] = "new $server"`
	`}`
	`$servers`
		`$servers = @('SERVER1','SERVER2','SERVER3','SERVER4','SERVER5')`
		`for ($i = 0; $i -lt $servers.Length; $i++) {`
		`$servers[$i] = "new $($servers[$i])"`
		`}`
		`$servers`
			Print 3

74	In that case, the function will fail to find the **appropriate** folder because it doesn't exist.	In that case, the function will fail to find the **version's** folder because it doesn't exist.	Print 4

Page	Error	Correction	Print corrected

76	`function Install-Software {`
	`  param(`
	`  [Parameter(Mandatory)]`
	`  [string]$Version`
	`  [ValidateSet('1','2')],`
	`  )`
		`function Install-Software {`
		`  param(`
		`  [Parameter(Mandatory)]`
		`  [ValidateSet('1','2')]`
		`  [string]$Version,`
		`  )`
			Print 4

99	To properly disconnect from a session, you pass your remote **computer's** name to the `Session` parameter by either calling it explicitly via `Disconnect-PSSession -Session` *`session object`*	To properly disconnect from a session, you pass your remote **session** name to the `Session` parameter by either calling it explicitly via `Disconnect-PSSession -Session` *`session name`*	Print 4
103	`PS> Enable-WSManCredSSP ❶ -Role ❷ Client ❸ -DelegateComputer WEBSRV1`	**NOTE** **To get CredSSP to work, you may have to relax a local policy. If you receive a permission error when trying to enable CredSSP, be sure you enable the Allow Delegating Saved Credentials with NTLM-only Server Authentication setting by running gpedit.msc and looking under Computer Configuration ▶ Administrative Templates ▶ System ▶ Credentials Delegation. While in the policy, click on the Show button and enter WSMAN/* to allow delegation from any endpoint.** `PS> Enable-WSManCredSSP ❶ -Role ❷ Client ❸ -DelegateComputer WEBSRV1 -Force`	Print 4
134	Insertion	**In Windows PowerShell, `Invoke-WebRequest` relies on Internet Explorer. If you don't have Internet Explorer on your computer, you may have to use the `-UseBasicParsing` parameter to remove the dependency. "Advanced" parsing breaks down the resulting HTML output a bit more but it's not needed in all cases.**	Print 4
155	`## Find all of the CSV <--> AD user account matches` `$positiveMatches = (Find-UserMatch).where({ $_.CSVProperties -ne 'NoMatch' })`	`## Find all of the CSV <--> AD user account matches` `$positiveMatches = (Find-UserMatch -SyncFieldMap $syncFieldMap -FieldMatchIds $fieldMatchIds).where({ $_.CSVProperties -ne 'NoMatch' })`	Print 4
166	`PS> $vm = Set-AzVMOSDisk -Name OSDisk` `-CreateOption 'fromImage' -VM $vm -VhdUri $osDiskUri`	`PS> $vm = Set-AzVMOSDisk -Name $osDiskName` `-CreateOption 'fromImage' -VM $vm -VhdUri $osDiskUri`	Print 4
169	`ServerName = 'PowerShellForSysAdmins-SQLSrv'`	`ServerName = 'powershellforsysadmins-sqlsrv'`	Print 4

Page	Error	Correction	Print corrected
173	You can install `AWSPowerShell` from the PowerShell Gallery the same way you installed the `AzureRM` module, by calling **`Install-Module AWSPowerShell`**. Once this module is downloaded and installed, you're ready to go.	You can install `AWSPowerShell` from the PowerShell Gallery the same way you installed the `Az` module, by calling **`Install-Module AWSPowerShell`**. Once this module is downloaded and installed, you're ready to go.	Print 4
175	Insertion	**Notice the `Arn` property in Listing 13-2. You'll need this value coming up when you create the IAM role.**	Print 4

176	Listing 13-3 is an example of a trust relationship policy document.
	`{`
	`"Version": "2019-10-17",`
	`"Statement": [`
	`{`
	`"Effect": "Allow",`
	`"Principal" : { "AWS": "arn:aws:iam::013223035658:user/Automator" },`
		Listing 13-3 is an example of a trust relationship policy document. **Important: Notice the *XXXXXX* on the Principal line. Be sure to replace the ARN of the IAM user you just created there.**
		`{`
		`"Version": "2012-10-17",`
		`"Statement": [`
		`{`
		`"Effect": "Allow",`
		`"Principal" : { "AWS": "arn:aws:iam::XXXXXX:user/Automator" },`
			Print 4

176	`PS> New-IAMRole -AssumeRolePolicyDocument $json -RoleName 'AllAccess'`
	`Path RoleName RoleId CreateDate`
	`---- -------- ------ ----------`
	`/ AllAccess AROAJ2B7YC3HH6M6F2WOM 9/16/2019 6:05:37 PM`
		`PS> New-IAMRole -AssumeRolePolicyDocument $json -RoleName 'AllAccess'`
		`Path RoleName RoleId CreateDate`
		`---- -------- ------ ----------`
		`/ AllAccess <Your Specific Role ID> <Date created>`
			Print 4

184	`PS> $ebApp = New-EBApplication -ApplicationName 'AutomateWorkflow'` `PS> $ebSApp`	`PS> $ebApp = New-EBApplication -ApplicationName 'AutomateWorkflow'` `PS> $ebSApp`	Print 4

Page	Error	Correction	Print corrected

185–186	`>> $parameters = @{`
	`>>  ApplicationName = 'AutomateWorkflow'`
	`>>  EnvironmentName = 'Testing'`
	`>>  SolutionStackName = '64bit Windows Server Core 2012 R2 running IIS 8.5'`
	`>>  Tier_Type = 'Standard'`
	`>>  Tier_Name = 'WebServer'`
	`>>  }`
	`PS> New-EBEnvironment @parameters`
	
	`AbortableOperationInProgress : False`
	`ApplicationName : AutomateWorkflow`
	`CNAME :`
	`DateCreated : 9/19/2019 12:19:36 PM`
	`DateUpdated : 9/19/2019 12:19:36 PM`
	`Description :`
	`EndpointURL :`
	`EnvironmentArn : arn:aws:elasticbeanstalk:...`
	`EnvironmentId : e-wkba2k4kcf`
	`EnvironmentLinks : {}`
	`EnvironmentName : Testing`
	`Health : Grey`
	`HealthStatus :`
	`PlatformArn : arn:aws:elasticbeanstalk:...`
	`Resources :`
	`SolutionStackName : 64bit Windows Server Core 2012 R2 running IIS 8.5`
	`Status : Launching`
	`TemplateName :`
	`Tier : Amazon.ElasticBeanstalk.Model.EnvironmentTier`
	`VersionLabel :`
		`PS> $instanceProfileOptionSetting = New-Object Amazon.ElasticBeanstalk.Model`
		`.ConfigurationOptionSetting -ArgumentList aws:autoscaling:launchconfiguration,`
		`IamInstanceProfile,'aws-elasticbeanstalk-ec2-role'`
		
		`>> $parameters = @{`
		`>>  ApplicationName = 'AutomateWorkflow'`
		`>>  EnvironmentName = 'Testing'`
		`>>  SolutionStackName = '64bit Windows Server Core 2019 v2.5.9 running IIS 10.0'`
		`>>  Tier_Type = 'Standard'`
		`>>  Tier_Name = 'WebServer'`
		`>>  OptionSetting = $instanceProfileOptionSetting`
		`>>  }`
		`PS> New-EBEnvironment @parameters`
		
		`AbortableOperationInProgress : False`
		`ApplicationName : AutomateWorkflow`
		`CNAME :`
		`DateCreated : 10/3/2020 9:31:49 AM`
		`DateUpdated : 10/3/2020 9:31:49 AM`
		`Description :`
		`EndpointURL :`
		`EnvironmentArn : arn:aws:elasticbeanstalk:us-east-1:054715970076:`
		`environment/AutomateWorkflow/Testing`
		`EnvironmentId : e-f3pfgxhrzf`
		`EnvironmentLinks : {}`
		`EnvironmentName : Testing`
		`Health : Grey`
		`HealthStatus :`
		`OperationsRole :`
		`PlatformArn : arn:aws:elasticbeanstalk:useast-1::platform/IIS`
		`10.0 running on 64bit Windows Server Core`
		`Resources :`
		`SolutionStackName : 64bit Windows Server Core 2019 v2.5.9 running`
		`IIS 10.0`
		`Status : Launching`
		`TemplateName :`
		`Tier : Amazon.ElasticBeanstalk.Model.EnvironmentTier`
		`VersionLabel :`
			Print 4

200	`PS> Get-ChildItem -Path '\\WEBSRV1\c$\Users\' -File | Measure-Object` `-Property Length -Sum`	`PS> Get-ChildItem -Path '\\WEBSRV1\c$\Users\' -File -Recurse | Measure-Object` `-Property Length -Sum`	Print 4

Page	Error	Correction	Print corrected
200	`$output.'UserProfilesSize (MB)' = (Get-ChildItem -Path "\\$server\c$\Users\" -File | Measure-Object -Property Length -Sum).Sum`	`$output.'UserProfileSize (MB)' = (Get-ChildItem -Path '\\WEBSRV1\c$\Users\' -File -Recurse | Measure-Object -Property Length -Sum).Sum`	Print 4
201	`$userProfileSize = (Get-ChildItem -Path "\\$server\c$\Users\" -File | Measure-Object -Property Length -Sum).Sum`	`$output.'UserProfileSize (MB)' = (Get-ChildItem -Path "\\$server\c$\Users\" -File -Recurse | Measure-Object -Property Length -Sum).Sum`	Print 5
202	Insertion	**NOTE** *The above command will only work if a computer only has a single disk. In my test environment, sqlsrv1 only has a C drive. If your server has more than one drive, you can combine the free space of all mounted drives by using the Measure-Object command like so: (Get-CimInstance -ComputerName sqlsrv1 -ClassName Win32_LogicalDisk | Measure-Object -Property FreeSpace -Sum).Sum. Subsequent free space code listings will assume your remote server has a single drive.*	Print 5
203, 205, 208, 209, 211, 212	`$output.'UserProfilesSize (MB)' = (Get-ChildItem -Path "\\$server\c$\Users\" -File | Measure-Object -Property Length -Sum).Sum / 1MB`	`$userProfileSize = (Get-ChildItem -Path "\\$server\c$\Users\" -File -Recurse \| Measure-Object -Property Length -Sum).Sum` `$output.'User ProfileSize (MB)' = [int]($userProfileSize / 1MB)`	Print 5

203, 205, 208, 209, 210	`ServerName UserProfilesSize (MB)`
	`---------- --------------------`
	`SQLSRV1 636245`
	`WEBSRV1 603942`
		`ServerName UserProfilesSize (MB)`
		`---------- --------------------`
		`SQLSRV1 1`
		`WEBSRV1 1`
			Print 5

210	`PS> C:\Get-ServerInformation.ps1 | Format-Table -AutoSize`	`PS> C:\Get-ServerInformation.ps1`	Print 4
212	`Remove-CimSession -CimSession $cimSession`	`Remove-CimSession -CimSession $getCimInstParams.CimSession`	Print 4
219	Since you're logged in via the local administrator account and may one day allow others to use your PowerLab module, create the module in the ***All Users* location of *C:\Files***.	Since you're logged in via the local administrator account and may one day allow others to use your PowerLab module, create the module in ***C:\Program Files\WindowsPowerShell\Modules***.	Print 4
222	Deletion	~~Because you previously imported the module, PowerShell hadn't loaded any functions into the session.~~	Print

Page	Error	Correction	Print corrected
225–226	`-Path 'C:\PowerLab\VHDs\LABDC.vhdx'`	`-Path 'C:\PowerLab\VHDs\MYVM.vhdx'`	Print 4
229	`[+] created a virtual machine called LABDC 62ms`	`[+] created a virtual machine called MYVM 62ms`	Print 4
232	And finally, you need the unattended XML answer file (also available via the chapter's downloadable resources) called ***unattend***.*xml* in the PowerLab module folder.	And finally, you need the unattended XML answer file (also available via the chapter's downloadable resources) called ***LABDC***.*xml* in the PowerLab module folder.	Print 4
242	`Get-Credential | Export-CliXml -Path C:\Files.xml`	`Get-Credential | Export-CliXml -Path C:\PowerLab\VMCredential.xml`	Print 4
248	`New-ADGroup -Name GroupName -GroupScope GroupScope -Path "OU=OUName,DC=powerlab,DC=local"` ❹	`New-ADGroup -Name $group.GroupName -GroupScope GroupScope -Path "OU=$group.OUName,DC=powerlab,DC=local"` ❹	Print 4