PowerShell Script to Deploy Multiple VM on Azure in Parallel #azure #powershell

[**SQLBI - Marco Russo**](http://sqlblog.com/blogs/marco_russo/default.aspx)

SQLBI is a blog dedicated to building Business Intelligence solutions with SQL Server.  
You can follow me on Twitter: [@marcorus](http://twitter.com/marcorus)

This blog is usually dedicated to Business Intelligence and SQL Server, but I didn’t found easily on the web simple PowerShell scripts to help me deploying a number of virtual machines on Azure that I use for testing and development. Since I need to deploy, start, stop and remove many virtual machines created from a common image I created (you know, Tabular is not part of the standard images provided by Microsoft…), I wanted to minimize the time required to execute every operation from my [Windows Azure PowerShell](http://msdn.microsoft.com/en-us/library/windowsazure/jj156055.aspx) console (but I suggest you using [Windows PowerShell ISE](http://technet.microsoft.com/en-us/library/dd819514.aspx)), so I also wanted to fire the commands as soon as possible in parallel, without losing the result in the console.

In order to execute multiple commands in parallel, I used the [Start-Job](http://technet.microsoft.com/en-us/library/hh849698.aspx) cmdlet, and using [Get-Job](http://technet.microsoft.com/en-us/library/hh849693.aspx) and [Receive-Job](http://technet.microsoft.com/en-us/library/hh849718.aspx) I wait for job completion and display the messages generated during background command execution. This technique allows me to reduce execution time when I have to deploy, start, stop or remove virtual machines. Please note that a few operations on Azure acquire an exclusive lock and cannot be really executed in parallel, but only one part of their execution time is subject to this lock. Thus, you obtain a better response time also in these scenarios (this is the case of the provisioning of a new VM).

Finally, when you remove the VMs you still have the disk containing the virtual machine to remove. This cannot be done just after the VM removal, because you have to wait that the removal operation is completed on Azure. So I wrote a script that you have to run a few minutes after VMs removal and delete disks (and VHD) no longer related to a VM. I just check that the disk were associated to the original image name used to provision the VMs (so I don’t remove other disks deployed by other batches that I might want to preserve).

These examples are specific for my scenario, if you need more complex configurations you have to change and adapt the code. But if your need is to create multiple instances of the same VM running in a workgroup, these scripts should be good enough.

I prepared the following PowerShell scripts:

* **ProvisionVMs**: Provision many VMs in parallel starting from the same image. It creates one service for each VM.
* **RemoveVMs**: Remove all the VMs in parallel – it also remove the service created for the VM
* **StartVMs**: Starts all the VMs in parallel
* **StopVMs**: Stops all the VMs in parallel
* **RemoveOrphanDisks**: Remove all the disks no longer used by any VMs. Run this script a few minutes after RemoveVMs script.

**ProvisionVMs**

# Name of subscription

$SubscriptionName = "Copy the SubscriptionName property you get from Get-AzureSubscription"

# Name of storage account (where VMs will be deployed)

$StorageAccount = "Copy the Label property you get from Get-AzureStorageAccount"

function Provision-VM( [string]$VmName ) {

    Start-Job -ArgumentList $VmName {

        param($VmName)

$Location = "Copy the Location property you get from Get-AzureStorageAccount"

$InstanceSize = "A5" # You can use any other instance, such as Large, A6, and so on

$AdminUsername = "UserName" # Write the name of the administrator account in the new VM

$Password = "Password"      # Write the password of the administrator account in the new VM

$Image = "Copy the ImageName property you get from Get-AzureVMImage"

# You can list your own images using the following command:

# Get-AzureVMImage | Where-Object {$\_.PublisherName -eq "User" }

        New-AzureVMConfig -Name $VmName -ImageName $Image -InstanceSize $InstanceSize |

            Add-AzureProvisioningConfig -Windows -Password $Password -AdminUsername $AdminUsername|

            New-AzureVM -Location $Location -ServiceName "$VmName" -Verbose

    }

}

# Set the proper storage - you might remove this line if you have only one storage in the subscription

Set-AzureSubscription -SubscriptionName $SubscriptionName -CurrentStorageAccount $StorageAccount

# Select the subscription - this line is fundamental if you have access to multiple subscription

# You might remove this line if you have only one subscription

Select-AzureSubscription -SubscriptionName $SubscriptionName

# Every line in the following list provisions one VM using the name specified in the argument

# You can change the number of lines - use a unique name for every VM - don't reuse names

# already used in other VMs already deployed

Provision-VM "test10"

Provision-VM "test11"

Provision-VM "test12"

Provision-VM "test13"

Provision-VM "test14"

Provision-VM "test15"

Provision-VM "test16"

Provision-VM "test17"

Provision-VM "test18"

Provision-VM "test19"

Provision-VM "test20"

# Wait for all to complete

While (Get-Job -State "Running") {

    Get-Job -State "Completed" | Receive-Job

    Start-Sleep 1

}

# Display output from all jobs

Get-Job | Receive-Job

# Cleanup of jobs

Remove-Job \*

# Displays batch completed

echo "Provisioning VM Completed"

**RemoveVMs**

# Name of subscription

$SubscriptionName = "Copy the SubscriptionName property you get from Get-AzureSubscription"

function Remove-VM( [string]$VmName ) {

    Start-Job -ArgumentList $VmName {

        param($VmName)

        Remove-AzureService -ServiceName $VmName -Force -Verbose

    }

}

# Select the subscription - this line is fundamental if you have access to multiple subscription

# You might remove this line if you have only one subscription

Select-AzureSubscription -SubscriptionName $SubscriptionName

# Every line in the following list remove one VM using the name specified in the argument

# You can change the number of lines - use a unique name for every VM - don't reuse names

# already used in other VMs already deployed

Remove-VM "test10"

Remove-VM "test11"

Remove-VM "test12"

Remove-VM "test13"

Remove-VM "test14"

Remove-VM "test15"

Remove-VM "test16"

Remove-VM "test17"

Remove-VM "test18"

Remove-VM "test19"

Remove-VM "test20"

# Wait for all to complete

While (Get-Job -State "Running") {

    Get-Job -State "Completed" | Receive-Job

    Start-Sleep 1

}

# Display output from all jobs

Get-Job | Receive-Job

# Cleanup

Remove-Job \*

# Displays batch completed

echo "Remove VM Completed"

**StartVMs**

# Name of subscription

$SubscriptionName = "Copy the SubscriptionName property you get from Get-AzureSubscription"

function Start-VM( [string]$VmName ) {

    Start-Job -ArgumentList $VmName {

        param($VmName)

        Start-AzureVM -Name $VmName -ServiceName $VmName -Verbose

    }

}

# Select the subscription - this line is fundamental if you have access to multiple subscription

# You might remove this line if you have only one subscription

Select-AzureSubscription -SubscriptionName $SubscriptionName

# Every line in the following list starts one VM using the name specified in the argument

# You can change the number of lines - use a unique name for every VM - don't reuse names

# already used in other VMs already deployed

Start-VM "test10"

Start-VM "test11"

Start-VM "test11"

Start-VM "test12"

Start-VM "test13"

Start-VM "test14"

Start-VM "test15"

Start-VM "test16"

Start-VM "test17"

Start-VM "test18"

Start-VM "test19"

Start-VM "test20"

# Wait for all to complete

While (Get-Job -State "Running") {

    Get-Job -State "Completed" | Receive-Job

    Start-Sleep 1

}

# Display output from all jobs

Get-Job | Receive-Job

# Cleanup

Remove-Job \*

# Displays batch completed

echo "Start VM Completed"

**StopVMs**

# Name of subscription

$SubscriptionName = "Copy the SubscriptionName property you get from Get-AzureSubscription"

function Stop-VM( [string]$VmName ) {

    Start-Job -ArgumentList $VmName {

        param($VmName)

        Stop-AzureVM -Name $VmName -ServiceName $VmName -Verbose -Force

    }

}

# Select the subscription - this line is fundamental if you have access to multiple subscription

# You might remove this line if you have only one subscription

Select-AzureSubscription -SubscriptionName $SubscriptionName

# Every line in the following list stops one VM using the name specified in the argument

# You can change the number of lines - use a unique name for every VM - don't reuse names

# already used in other VMs already deployed

Stop-VM "test10"

Stop-VM "test11"

Stop-VM "test12"

Stop-VM "test13"

Stop-VM "test14"

Stop-VM "test15"

Stop-VM "test16"

Stop-VM "test17"

Stop-VM "test18"

Stop-VM "test19"

Stop-VM "test20"

# Wait for all to complete

While (Get-Job -State "Running") {

    Get-Job -State "Completed" | Receive-Job

    Start-Sleep 1

}

# Display output from all jobs

Get-Job | Receive-Job

# Cleanup

Remove-Job \*

# Displays batch completed

echo "Stop VM Completed"

**RemoveOrphanDisks**

$Image = "Copy the ImageName property you get from Get-AzureVMImage"

# You can list your own images using the following command:

# Get-AzureVMImage | Where-Object {$\_.PublisherName -eq "User" }

# Remove all orphan disks coming from the image specified in $ImageName

Get-AzureDisk |

    Where-Object {$\_.attachedto -eq $null -and $\_.SourceImageName -eq $ImageName} |

    Remove-AzureDisk -DeleteVHD -Verbose

Published Tuesday, October 29, 2013 2:44 PM by [Marco Russo (SQLBI)](http://sqlblog.com/user/Profile.aspx?UserID=13578)

Filed under: [Azure](http://sqlblog.com/blogs/marco_russo/archive/tags/Azure/default.aspx), [PowerShell](http://sqlblog.com/blogs/marco_russo/archive/tags/PowerShell/default.aspx)