










Out-of-Hours Start of Services in Azure


Last updated by | Paul Kelleher | 17 Jun 2020 at 07:34 GMT


Introduction


				
Microsoft Forms 	Microsoft Flows	Azure DevOps 	Find Change Requests 	Submit Start Request 

Submit

Flows button to call automation

 Manually trigger a flow ...

 Add an input

 Create job ...

* Subscription

MS-DataEngineering-Dev-Test

▼

* Resource Group

rg-datadev-services

▼

* Automation Account

datadev-automation

▼

Runbook Name

Set-AzureRMVMSynchronizedStartStop

▼

Wait for Job

No

▼

Runbook Parameter Action

start

Runbook Parameter TagKey

AutomatedStopStart

Runbook Parameter TestRun

No

▼

Runbook Parameter GroupTag

Group

Runbook Parameter TargetGroup

windows

Runbook Parameter ConnCredName

AzureRunAsConnection

Runbook Parameter UseAutomationConne

Yes

▼

Runbook Parameter ReverseOrder

No

▼

[Show advanced options](#) ▼

Automation to Launch VMs

Launch VMs with Tag

```

<#
.Synopsis
  This Azure Runbook Workflow helps to Start/Stop VMs in Azure ARM portal in a defined sequence using Tags
  To onboard servers, simply add a Tag named 'AutoStart' with a Json formatted value in the form {"Group":"1","Seque
  You can target individual GROUPS or ALL VMs in the subscription and Start/Stop them in defined sequence.
  START is performed in Ascending order of sequence 1.2.3.. and during STOP sequence is reversed 3..2..1
  Parameters allows you to customize Tags and Values further.

  CREATED BY:
  Sumesh P. <maxsumesh@hotmail.com>
    <Version 1.21> <Dec 2017>
    <Version 1.2> <June 2017>
.DESCRIPTION
  Start or Stop an Azure ARM VM using this PowerShell Runbook Automation in a specified sequence.
  You can use the integrated runbook scheduler to run the automated start/stop at a given time of day.

  This requires the latest PowerShell modules to be updated in the Automation account.
  Requires AzureRM.Compute to be at Version 3.1.0 or higher
  Go to Automation Account -> Modules -> Update Azure Modules

.PARAMETER TargetGroup
  Default: '*'
  Defines which Server Group to target. By default the runbook targets all groups/VMs with the AutoStart Tag.
  You can set it to 1,2 or whatever the valid group name you want to target.
.PARAMETER UseAutomationConnection
  Default: $true
  Tells the runbook to use Automation connection as the default method of login.
  Works with parameter 'ConnCredName'
  If 'UseAutomationConnection' is $true it treats 'ConnCredName' as Connection name.
  If 'UseAutomationConnection' is $false it treats 'ConnCredName' as Credential name.
.PARAMETER ConnCredName
  Default: 'AzureRunAsConnection'
  Name of the CONNECTION OR CREDENTIAL ASSET used for automated login to Azure.
  Works with parameter 'UseAutomationConnection'. By default the script expects the login to be based on Automation C
  If 'UseAutomationConnection' is $true it treats 'ConnCredName' as Connection name.
  If 'UseAutomationConnection' is $false it treats 'ConnCredName' as Credential name.
.PARAMETER Action
  Default: 'Start'
  Currently it is set as the only Mandatory parameter.
  Defines if the runbook will start or stop the VMs.
.PARAMETER TestRun
  Default: $true
  When $true the runbook will only simulate actions and runs a WhatIf analysis on the tagged schedules.
  No actual START/STOP is performed. Use this to test the runbook flow; to run normally set to $false.
.PARAMETER TagKey
  Default:'AutoStart'
  Allows you to change the Name used for the Tag. The tag is used to identify VMs eligible for this automation.
  May not be frequently required, but just in case! Also allows you to easily reprogram to use Double Tags if you so
.PARAMETER GroupTag
  Default:'Group'
  Allows you to change the name of the Tag used to identify server groups inside the 'AutoStart'/'TagKey' Json string
  May not be frequently required, but just in case!

.EXAMPLE
  Set-AzureRMVMSynchronizedStartStop -Action "Stop" -TestRun 1
  Initiate a TEST run of the runbook to test Stopping of VMs
.EXAMPLE
  Set-AzureRMVMSynchronizedStartStop -Action "Start" -TestRun $false -TargetGroup 1
  Initiates a Prod run of the runbook to START a server group 1
.EXAMPLE
  Set-AzureRMVMSynchronizedStartStop -Action "Start" -TestRun $false -UseAutomationConnection $false -ConnCredName "I
  Start ALL VMs using PS credential login, it used the credential saved in the automantion account named 'MyCred'
.EXAMPLE
  Set-AzureRMVMSynchronizedStartStop -Action "Start" -TestRun $true -ConnCredName "MyConnection"
  Start ALL VMs using Connection based login, it used the connection saved in the automantion account named 'MyConne
.INPUTS
  Only mandatory parameter is 'Action' to define "Start" or "Stop"
.OUTPUTS
  Currently you can see the output in PowerShell window or based on requirement EMAIL functionality can be added.

.NOTES
  CREATED BY:
  Sumesh P. <maxsumesh@hotmail.com>
  When you modify or use the script, do retain the Creator Credits and contact while adding yours.

  WARNING: Malformed Tag Values/Json Strings can cause the runbook to target ALL VMs.
  Test thoroughly before production use. Limited testing has been performed on Non-Prod environment.
  The Script is provided AS IS without any warranties or liabilities, use at your OWN RISK
.COMPONENT
  Requires AzureRM.Compute to be at Version 3.1.0 or higher
  Go to Automation Account -> Modules -> Update Azure Modules

```

```
#>

param (
    [Parameter(Mandatory=$false)]
    [ValidateNotNull()]
    [ValidateNotNullOrEmpty()]
    [String] $TargetGroup = '*',

    [Parameter(Mandatory=$false)]
    [ValidateNotNull()]
    [ValidateNotNullOrEmpty()]
    [bool] $UseAutomationConnection = $true,

    [Parameter(Mandatory=$false)]
    [ValidateNotNull()]
    [ValidateNotNullOrEmpty()]
    [String] $ConnCredName = "AzureRunAsConnection",

    [Parameter(Mandatory=$true)]
    [ValidateNotNull()]
    [ValidateNotNullOrEmpty()]
    [ValidateSet("Start", "Stop")]
    [string] $Action,

    [Parameter(Mandatory=$false)]
    [ValidateNotNull()]
    [ValidateNotNullOrEmpty()]
    [bool] $TestRun = $true,

    [Parameter(Mandatory=$false)]
    [ValidateNotNull()]
    [ValidateNotNullOrEmpty()]
    [bool] $ReverseOrder = $false,

    [Parameter(Mandatory=$false)]
    [ValidateNotNull()]
    [ValidateNotNullOrEmpty()]
    [String] $TagKey = 'AutomatedStopStart',

    [Parameter(Mandatory=$false)]
    [ValidateNotNull()]
    [ValidateNotNullOrEmpty()]
    [String] $GroupTag = 'Group'
)

#Json format
#{"Group":"1","Sequence":"2"}

#$TagKey = 'AutoStart'
#$TargetGroup = '1'

#Consider whether to use a single tag using Json or two tags for Group and sequence respectively.
#Single - Less tags, overhead; easy using powershell
#Double - Easy to track servers in a group using portal

if($UseAutomationConnection){
try
{
    # Get the connection "AzureRunAsConnection "
    $AzureRunAsConnection = Get-AutomationConnection -Name $ConnCredName

    Write-Output "Logging in to Azure using AzureRunAsConnection..."
    Add-AzureRmAccount `
        -ServicePrincipal `
        -TenantId $AzureRunAsConnection.TenantId `
        -ApplicationId $AzureRunAsConnection.ApplicationId `
        -CertificateThumbprint $AzureRunAsConnection.CertificateThumbprint

    if($?){Write-output "Login Successful!"}else{Write-output "Error Logging in??"}
}
catch {
Write-Output "An Error was encountered using Automation Connection..."
    if (!$AzureRunAsConnection)
    {
        $ErrorMessage = "Connection $ConnCredName not found. Create a RunAs connection or check Connection Name"
        Write-Error $ErrorMessage
        throw $ErrorMessage
    } else{

```

```

        Write-Error "Error logging in to Azure using AzureRunAsConnection"
        Write-Error -Message $($_.Exception)
        throw $($_.Exception)
    }
}
#>
}
else{
try{
    Write-Output "Logging in to Azure using AutomationPSCredential..."
    $Credential = Get-AutomationPSCredential -name $ConnCredName
    Login-AzureRmAccount -credential $Credential | fl

    if($?){Write-output "Login Successful!"}
}
catch {
Write-Output "An Error was encountered using AutomationPSCredential..."
    if (!$AzureRunAsConnection)
    {
        $ErrorMessage = "Connection $ConnCredName not found. Create an Automation Credential or check Name"
        throw $ErrorMessage
        Write-Output $ErrorMessage
    } else{
        Write-Error "Error logging in to Azure using AutomationPSCredential"
        Write-Error -Message $($_.Exception)
        throw $($_.Exception)
    }
}
}

#region Code
$ReverseOrder = $true # For Ascending sort order in Sequence
if($Action -eq "Start"){ $ReverseOrder = $False}else{}

Write-output "Getting Azure ARM VM Status"
#Get-AzureRmResourceGroup | %{Get-azurermmvm -Status -wa SilentlyContinue -ResourceGroupName $($_.ResourceGroupName) |
#Get-AzureRmResource | where {$_.ResourceType -like "Microsoft.*virtualMachines"} | Get-AzureRmVM -status | Select R

$AllAzureRMVMs = get-azurermmvm -Status -WarningAction SilentlyContinue

$VMsTaggedAutoStart = $AllAzureRMVMs | where { $_.Tags.Keys -eq $TagKey }

$VMListData = @()
$VMsTaggedAutoStart | %{
    $JsonValue = $null
    $JsonValue = $_.Tags.$TagKey | ConvertFrom-Json

    $prop = [ordered] @{
        Name = $_.Name
        ResourceGroupName = $_.ResourceGroupName
        ServerGroup = $JsonValue.$GroupTag
        Sequence = $JsonValue.Sequence
        Size = ($_ | Select -exp HardwareProfile).VMSize
        Status = 'NA'
        PreviousPowerState = $_.PowerState
        PowerState = $_.PowerState
        StartTime = ''
        EndTime = ''
        Error = ''
    }

    $VM2Start = New-Object -TypeName psobject -Property $prop

    $VMListData += $VM2Start
}
$VMListData | Select ServerGroup, Name, ResourceGroupName, Sequence, PowerState, Size | Sort ServerGroup, Sequence -I

#$VMListData | Group-Object Group

# Old logic for all server group start/stop at once
#$ServerGroups = $VMListData | Group-Object ServerGroup

#New logic to target specific server group
write "# Grouping maintained for compatibility with old method - having to target all servers"
$VMListData | ?{$_.ServerGroup -like $TargetGroup} | Group-Object ServerGroup | ft

$ServerGroups = $VMListData | ?{$_.ServerGroup -like $TargetGroup} | Group-Object ServerGroup | Sort Name -Descending

Write-Output "Found $($ServerGroups.Count) Server Groups"

```

```

$ServerGroups | % {

$ServerGroup = $_.Group | Sort-Object ServerGroup,Sequence -Descending:$ReverseOrder
Write-Output "Start/Stop ORDER"

Write-Output "`nFound $($ServerGroup.Count) Servers in Group. $($Action)ing VMs in following defined sequence`n"
$ServerGroup | ft Name, ResourceGroupName, ServerGroup, Sequence, PowerState, Size

$ServerGroup | % {
$VM = $_
Write-output "`nAttempting to $($Action) VM: $($VM.Name)"

if($Action -eq "Start"){
if($($_.powerstate) -like "*running*"){Write-output "VM is already in Running State!!! `n"}
#try{
$OpsStatus = $VM | Start-AzureRmVM -Verbose -WhatIf:$TestRun
$OpsStatus | ft
#}catch{
#Write-Output "Error encountered in VM START Operation.."
#continue
#}
}else{
if($($_.powerstate) -like "*deallocated*"){Write-output "VM is already in Deallocated State!!! `n"}
#try{
$OpsStatus = $VM | Stop-AzureRmVM -Verbose -WhatIf:$TestRun -Force
$OpsStatus | ft
#}catch{
#Write-Output "Error encountered in VM STOP Operation.."
#continue
#}
}

$updateVM = ($VMListData | where {$_.name -eq ($VM.name) })
$updateVM.Error = $OpsStatus.Error
$updateVM.StartTime = $OpsStatus.StartTime
$updateVM.EndTime = $OpsStatus.EndTime
$updateVM.Status = $OpsStatus.Status
}

}
# Update New Status
#improve perf, to further improve, put it outside outer loop
$newVMstatus = Get-AzureRmVM -Status -WarningAction SilentlyContinue
$VMListData | % {
$VMName = $_.Name

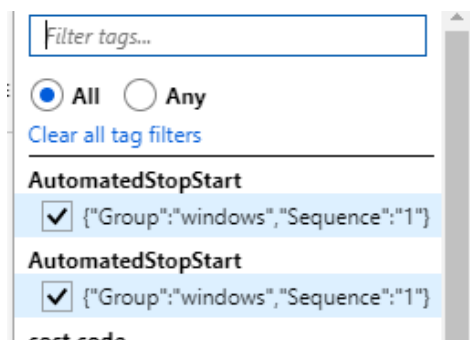
$x = ($newVMstatus | where {$_.name -eq ($VMname) })
$_.PowerState = ($newVMstatus | where {$_.name -eq ($VMname) } | select -exp PowerState)
}

$VMListData | Sort ServerGroup, Sequence -Descending:$ReverseOrder | ft

if($TestRun){Write-Output "This is a SIMULATED Test run, no changes were made. Turn off TestRun parameter to make changes"}
#endregion code

```

see PK



Contents

- Introduction
 - [Flows button to call automation](#)
 - [Automation to Launch VMs](#)