|  |  |
| --- | --- |
|  |  |
| Projectnaam: | Lifecycle Management Amsterdam UMC Research Cloud Platform |
| Projectnummer: | 2018-032 |
| Auteur: | Cédric Laurent / Fons Ullings |
|  |  |
| Gewijzigd door: | Fons Ullings |
| Datum: | 08-05-2020 |
| Versie: | 1.0 |
| Status document: | Final |

Inhoud

[1. Inleiding 3](#_Toc40108596)

[2. Context 4](#_Toc40108597)

[3. Power Shell functieblokken 5](#_Toc40108598)

[4. Azure API observaties 5](#_Toc40108599)

[5. Bijlage 1: Power Shell code 6](#_Toc40108600)

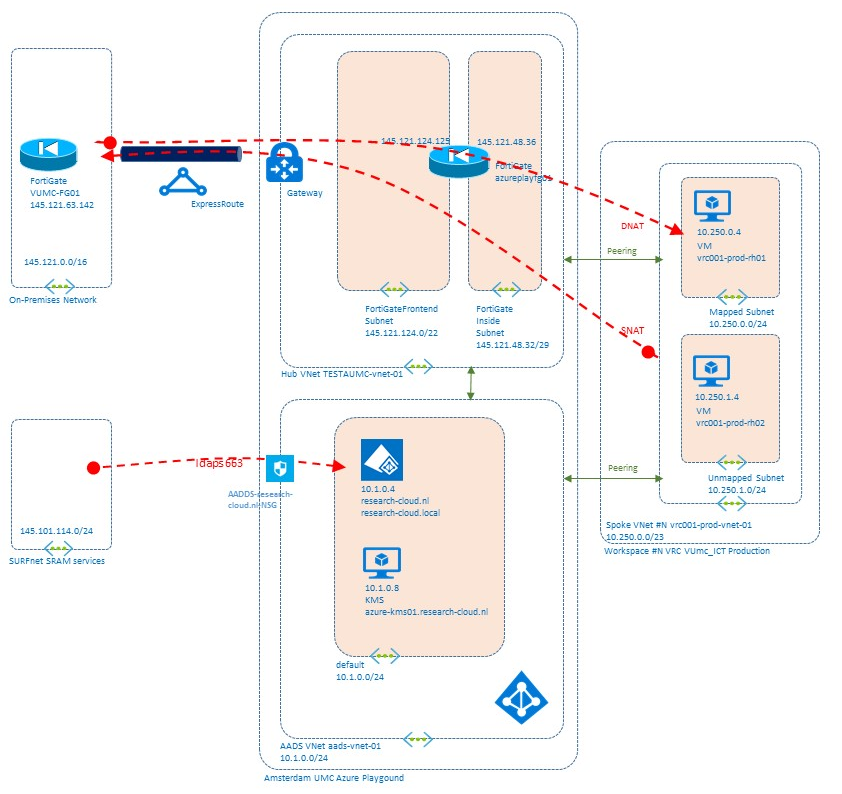
# Inleiding

Bij het eerste ontwerp van de Amsterdam UMC Research Cloud bleek al snel dat het hanteren en vooral handhaven van een heel strakke naamconventies binnen een multi-workspace omgeving een absolute must is. Microsoft en ook Amazon geven dat ook als expliciete richtlijn aan. En die richtlijn is ook in een strakke naamconventie definitie voor de Research Cloud gedocumenteerd.  
Het zou echter onmenselijke discipline zou vragen alle configuratiedetails en alle resource/object namen over 30-100 workspaces strak gelijk te houden. Er is dan ook voor gekozen om de heel (!) de Research Cloud workspace inrichting in Infrastructure as Code (IaC) onder te brengen.   
Dit document geeft een beknopte toelichting bij de daarvoor gerealiseerde Power Shell Automation code die per mei 2020 de nieuwste Azure Power Shell API gebruikt.  
Zie: <https://www.powershellgallery.com/packages/Az/3.8.0>

In de bijlage is de betreffende Power Shell code toegevoegd: updates/revisies zullen bij volgende versies en release in GitHub worden aangeboden en worden onderhouden.

# Context

Zoals gedetailleerd in het LLD aangegeven geeft de volgende figuur de context van de Research Cloud workspaces aan waarbij strikt het Microsoft hub-spoke model is gevolgd. Eén hub voor de koppeling naar de ExpressRoute/SURFnet lichtpad ontsluiting en een hub naar de centrale faciliteiten zoals de DNS services, KMS en SRAM authenticatie/autorisatie Azure AD infrastructuur.



# Power Shell functieblokken

De code is in een aantal functieblokken opgedeeld die hier kort worden genoemd. De code zelf is zoveel mogelijk self-documented met toevoeging van comments opgeschreven. En binnen de code vindt ook de interactieve MFA authenticatie dialoog plaats (TenantAdmin@research-cloud.nl):

* Create workspace attribute list conform the naming convention
* Install, Update and Import the required Azure API modules.....
* Clear authentication cache
* MFA authenticate (TenantAdmin@research-cloud.nl)
* Create workspaces
* Management group assignment, Create Resource Groups, Resource Providers and Storage Account V2/LRS
* Create Backup Vault
* Parameterize Backup Vault, LocalPolicy and LogAnalytics reporting assignments
* Create VNets and associated Subnets
* Create User Defined Routes and DNS reference
* Create Network peerings hubs on spoke
* Create Automation account and associated Event Grid subscription
* Create Storage Account firewalling
* Enable AutoProvsioning MMA and MMA logging to Loganalytics

# Azure API observaties

Bij het ontwikkelen en valideren van deze Power Shell Azure API code hebben in de loop van de tijd zich aantal Azure API “weetjes” gemanifesteerd:

* Er zit heel veel dynamiek in de Azure API ontwikkeling dat deze code met een zekere regelmaat moet worden getest tegen de nieuwste Azure API Az module versie.
* Er moet dan ook steeds de nieuwste API versie worden geïnstalleerd met Install-Module en Update-Module calls.
* Sommige API modules zijn nog zo nieuw, zoals Az.Security dat ze expliciet moeten worden geïnstalleerd.
* Sommige Azure API calls kunnen niet tegen cached authenticatie gegevens: daartoe is expliciet de call Clear-AzContext toegevoegd.
* Sommige Azure API calls triggeren race-condities: je kan bijvoorbeeld wel al een Virtual Network (VNet) hebben aangemaakt, maar dat houdt nog niet zomaar in dat je al per direct attributes of routes in zo een VNet kan wijzigen….. Onder de “motorkap” duurt het langere tijd (zeg een minuut) voordat zo een VNet écht werkt en wijzigingen toelaat, etc. Dit is op twee manieren in de code gemitigeerd:
  + Per functieblok worden eerst alle workspaces doorlopen: dat neemt normaal gesproken zoveel tijd in beslag dat dan de eerste workspace weer ready is voor het aanbrengen avn een volgende functie
  + Per functieblok wordt een <return> gevraagd: zeker als er maar één (test) workspace wordt gecreëerd is het wijs om steeds even per functieblok te pauzeren.

Als er eenmaal een workspace (subscription) is gecreëerd schudt je die niet zomaar af….. het neemt minstens 90 dagen na <delete> van een subscription voordat die echt door Microsoft wordt verwijderd. Microsoft heeft dat als policy om het nog 90 dagen mogelijk te maken om een subscription delete ongedaan te kunnen maken.

# Bijlage 1: Power Shell code

#

<#

.SYNOPSIS

This Infrastructure as Code (IaC) produces 'Spare' additional workspaces using the Azure API within the existing Amsterdam UMC Research Cloud research-cloud.nl context

.DESCRIPTION

.NOTES

AUTHORs: A.H.Ullings

LASTEDIT: 08/05/2020

VERSION : 0.91

Copyright 2020 All Rights Reserved

#>

$MGMTGROUP = 'SURFcumulus-Amsterdam-UMC-VRC-Production'

$ONVMCREATE = 'https://s2events.azure-automation.net/webhooks?token=XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX'

$AZVERSION = '3.8.0'

$DEBUGWORKSPACE = 127

$DEBUGWORKSPACENAME = 'Amsterdam UMC Research Hub'

$HUBWORKSPACENAME = 'Amsterdam UMC Azure Playground'

$HUBRESOURCEGROUP = 'azure-play-rg'

$HUBBACKUPVAULT = 'azure-play-backup'

$HUBLOGANALYTICS = 'azure-play-loganalytics'

$SPOKEWORKSPACENAME = 'SURFcumulus Amsterdam UMC VRC VUmc\_ICT Production'

$SPOKERESOURCEGROUP = 'azure-play-rg'

$PEER1VNET = 'TESTAUMC-vnet-01' # legacy naming

$PEER1RG = 'my\_resources' # legacy naming

$PEER2VNET = 'azure-aads-vnet-01'

$PEER2RG = 'azure-play-rg'

$FIRSTWORKSPACE = 31

$LASTWORKSPACE = 62

#

class workspaceRecord {

[string]$subname

[string]$short

[string]$rg

[string]$sa

[string]$backup

[string]$automate

[string]$event

[string]$vnet

[string]$prefix

[string]$mapped

[string]$unmapped

}

#

function CreateworkspaceList

{

$workspaceList = [System.Collections.Generic.List[workspaceRecord]]::new()

for ($i = $FIRSTWORKSPACE; $i -le $LASTWORKSPACE; $i++)

{

$nums = "{0:00#}" -f $i

$numprefix = $i \* 2 - 2

$short = 'vrc'+$nums+'-prod'

$newRecord = [workspaceRecord] @{ `

subname = $(if($i -ne $DEBUGWORKSPACE) `

{'SURFcumulus Amsterdam UMC VRC Spare'+$nums+' Production'} else {$DEBUGWORKSPACENAME}); `

short = 'vrc'+$nums+'-prod'; `

rg = $short+'-rg'; `

sa = 'vrc'+$nums+'prodsa01'; `

backup = $short+'-backup'; `

automate = $short+'-automate'; `

event = $short+'-event-01'; `

vnet = $short+'-vnet-01'; `

prefix = '10.250.'+("{0:0#}" -f ($numprefix+0))+'.0/23'; `

mapped = '10.250.'+("{0:0#}" -f ($numprefix+0))+'.0/24'; `

unmapped = '10.250.'+("{0:0#}" -f ($numprefix+1))+'.0/24' `

}

$workspaceList.Add($newRecord)

}

return $workspaceList

}

#

# Check the workspace listnames

#

foreach($ws in CreateworkspaceList)

{

Write-Host $ws.subname

}

#

# Import the required modules

#

Write-Host 'import the required Azure API modules.....'

#

# we need the newest stuff....

# Install-Module -Name Az

# Install-Module -Name Az.Security

# Update-Module Az -RequiredVersion $AZVERSION -Force

Import-Module Az -RequiredVersion $AZVERSION

Clear-AzContext -Force

# Authenticate

$res = Connect-AzAccount

#

$res = Select-AzSubscription -Subscription $HUBWORKSPACENAME

#

# Enrollment credentials

#

$enroll = Get-AzEnrollmentAccount

Write-Host 'EnrollmentAccount ObjectId =', $enroll.ObjectId

Read-Host -Prompt "Create workspaces <enter>"

#

foreach($ws in CreateworkspaceList)

{

Write-Host 'create workspace', $ws.subname

# New-AzSubscription -OfferType MS-AZR-0017P -Name $ws.subname `

# -EnrollmentAccountObjectId $enroll.ObjectId -OwnerObjectId $enroll.ObjectId,$enroll.ObjectId

Write-Host 'create workspace done'

}

Read-Host -Prompt "Create Resource Groups and Storage Accounts <enter>"

#

foreach($ws in CreateworkspaceList)

{

$res = Select-AzSubscription -Subscription $ws.subname

$subid = Get-AzSubscription -SubscriptionName $ws.subname

Write-Host $subid.Id

#

# Move new workspace to VRC production management group

#

Write-Host "move the created workspace in the VRC management group"

$res = New-AzManagementGroupSubscription -GroupName $MGMTGROUP -SubscriptionId $subid.Id

#

# Create ResourceGroup

#

Write-Host 'create resource group', $ws.rg

$res = New-AzResourceGroup -Name $ws.rg -Location WestEurope

#

# Register already the required extra Resource Providers

#

Write-Host 'register needed resource providers'

$res = Register-AzResourceProvider -ProviderNamespace Microsoft.EventGrid

$res = Register-AzResourceProvider -ProviderNamespace microsoft.insights

#

# Create Storage Account

#

Write-Host 'create storage account LRS V2', $ws.sa

$res = New-AzStorageAccount -ResourceGroupName $ws.rg -Name $ws.sa -Location WestEurope `

-SkuName Standard\_LRS -Kind StorageV2

}

Read-Host -Prompt "Create Backup Vault <enter>"

#

foreach($ws in CreateworkspaceList)

{

$res = Select-AzSubscription -Subscription $ws.subname

#

# Create Backup (and Migration) vaults and set the InstantBackupandRecovery features

#

Write-Host 'create backup vault', $ws.backup

$res = New-AzRecoveryServicesVault -Name $ws.backup -ResourceGroupName $ws.rg -Location WestEurope

Write-Host 'set InstantBackupandRecovery features'

$res = Register-AzProviderFeature -FeatureName "AllowApplicationSecurityGroups" `

-ProviderNamespace Microsoft.Network

$res = Register-AzProviderFeature -FeatureName "InstantBackupandRecovery" `

–ProviderNamespace Microsoft.RecoveryServices

}

Read-Host -Prompt "Parameterize Backup Vaults <enter>"

#

# Copy 'LocalPolicy' definition

#

Write-Host 'copy azure-play-backup LocalPolicy'

$res = Select-AzSubscription -Subscription $HUBWORKSPACENAME

$vault = Get-AzRecoveryServicesVault -Name $HUBBACKUPVAULT

Set-Item Env:\SuppressAzurePowerShellBreakingChangeWarnings "True"

$res = Set-AzRecoveryServicesVaultContext -Vault $vault

#

# Grab existing Backup Protection Policy and store in a variable

#

$policy = Get-AzRecoveryServicesBackupProtectionPolicy -Name "LocalPolicy"

#

# Grab existing Log Analytics Context en store it in a variable

#

$res = Select-AzSubscription -Subscription $HUBWORKSPACENAME

$loganalytics = Get-AzOperationalInsightsWorkspace -Name $HUBLOGANALYTICS `

-ResourceGroupName $HUBRESOURCEGROUP

#

foreach($ws in CreateworkspaceList)

{

$res = Select-AzSubscription -Subscription $ws.subname

#

# Set Backup and vault LocallyRedundant

#

Write-Host 'set vaults LocallyRedundant'

Set-Item Env:\SuppressAzurePowerShellBreakingChangeWarnings "True"

$vault = Get-AzRecoveryServicesVault -ResourceGroupName $ws.rg -Name $ws.backup

$res = Set-AzRecoveryServicesBackupProperties -Vault $vault `

-BackupStorageRedundancy LocallyRedundant

#

# Create new Backup Protection Policy using above Schedule and Retention Policy Objects

#

Write-Host 'azure-play-backup LocalPolicy'

$vault = Get-AzRecoveryServicesVault -Name $ws.backup -ResourceGroupName $ws.rg

Set-Item Env:\SuppressAzurePowerShellBreakingChangeWarnings "true"

Set-AzRecoveryServicesVaultContext -Vault $vault

$res = New-AzRecoveryServicesBackupProtectionPolicy -Name "LocalPolicy" -WorkloadType "AzureVM" `

-RetentionPolicy $policy.RetentionPolicy -SchedulePolicy $policy.SchedulePolicy

#

# And enable Diagnostic setting to the central LogAnalytics environment

#

$report = $ws.backup.Replace("-backup", "-backup-report")

Write-Host 'enable azure-play-loganalytics backup reporting', $report

Set-Item Env:\SuppressAzurePowerShellBreakingChangeWarnings "true"

$res = Set-AzDiagnosticSetting -Name $report -ResourceId $vault.ID `

-Enabled $True -Category AzureBackupReport -WorkspaceId $loganalytics.ResourceId

}

Read-Host -Prompt "Create VNets and associated Subnets <enter>"

#

foreach($ws in CreateworkspaceList)

{

$res = Select-AzSubscription -Subscription $ws.subname

#

# Create VNet + Subnets

#

Write-Host 'create mapped/unmapped subnets', $ws.mapped, $ws.unmapped

$mappedSubnet = New-AzVirtualNetworkSubnetConfig -Name Mapped -AddressPrefix $ws.mapped

$unmappedSubnet = New-AzVirtualNetworkSubnetConfig -Name UnMapped -AddressPrefix $ws.unmapped

$res = New-AzVirtualNetwork -Name $ws.vnet -ResourceGroupName $ws.rg -Location WestEurope `

-AddressPrefix $ws.prefix -Subnet $mappedSubnet,$unmappedSubnet

}

Read-Host -Prompt "Create User Defined Routes and DNS reference <enter>"

#

foreach($ws in CreateworkspaceList)

{

$res = Select-AzSubscription -Subscription $ws.subname

#

# Create and populate the User Defined Route (UDR)

#

$routename = $ws.short+'-to-Internet'

Write-Host 'create VNet User Defined Route', $routename

$route = New-AzRouteConfig -Name "to-Internet" -AddressPrefix "0.0.0.0/0" `

-NextHopType "VirtualAppliance" -NextHopIpAddress "145.121.48.36"

$routetable = New-AzRouteTable -Name $routename -ResourceGroupName $ws.rg -Location WestEurope -Route $route

$routetable = $routetable | Add-AzRouteConfig -Name "to-KMS" -AddressPrefix 23.102.135.246/32 -NextHopType "Internet"

$routetable = $routetable | Add-AzRouteConfig -Name "to-RedHat01" -AddressPrefix 13.91.47.76/32 -NextHopType "Internet"

$routetable = $routetable | Add-AzRouteConfig -Name "to-RedHat02" -AddressPrefix 40.85.190.91/32 -NextHopType "Internet"

$routetable = $routetable | Add-AzRouteConfig -Name "to-RedHat03" -AddressPrefix 52.187.75.218/32 -NextHopType "Internet"

$routetable = $routetable | Add-AzRouteConfig -Name "to-RedHat04" -AddressPrefix 52.174.163.213/32 -NextHopType "Internet"

$routetable = $routetable | Add-AzRouteConfig -Name "to-RedHat05" -AddressPrefix 52.237.209.198/32 -NextHopType "Internet"

$routetable | Set-AzRouteTable | Out-Null

Write-Host 'associate User Defined Route Mapped/UnMapped', $routename

$vn = Get-AzVirtualNetwork -Name $ws.vnet -ResourceGroupName $ws.rg

$res = Set-AzVirtualNetworkSubnetConfig -VirtualNetwork $vn -Name "Mapped" `

-AddressPrefix $ws.mapped -RouteTable $routetable | Set-AzVirtualNetwork

$res = Set-AzVirtualNetworkSubnetConfig -VirtualNetwork $vn -Name "UnMapped" `

-AddressPrefix $ws.unmapped -RouteTable $routetable | Set-AzVirtualNetwork

#

# Customize DNS

#

Write-Host 'DNS VNet ', $ws.vnet

$vn = Get-AzVirtualNetwork -ResourceGroupName $ws.rg -name $ws.vnet

$vn.DhcpOptions.DnsServers = "10.1.0.4"

# $vn.DhcpOptions.DnsServers += "10.1.0.5"

$res = Set-AzVirtualNetwork -VirtualNetwork $vn

}

Read-Host -Prompt "Create Network peerings hub/spoke <enter>"

#

foreach($ws in CreateworkspaceList)

{

#

# Create peering TESTAUMC-vnet-01

#

$res = Select-AzSubscription -Subscription $ws.subname

$peer = $ws.short+'-peer-01'

Write-Host 'create hub-spoke TESTAUMC-vnet-01 peer01', $peer

$vnet1 = Get-AzVirtualNetwork -Name $ws.vnet -ResourceGroupName $ws.rg

$res = Select-AzSubscription -Subscription $HUBWORKSPACENAME

$vnet2 = Get-AzVirtualNetwork -Name $PEER1VNET -ResourceGroupName $PEER1RG # legacy

$res = Select-AzSubscription -Subscription $ws.subname

# Peer VNet1 to VNet2.

$res = Add-AzVirtualNetworkPeering -AllowForwardedTraffic -AllowGatewayTransit -Name $peer `

-VirtualNetwork $vnet1 -RemoteVirtualNetworkId $vnet2.Id

$res = Select-AzSubscription -Subscription $HUBWORKSPACENAME

# Peer VNet2 to VNet1.

$res = Add-AzVirtualNetworkPeering -AllowForwardedTraffic -AllowGatewayTransit -Name $peer `

-VirtualNetwork $vnet2 -RemoteVirtualNetworkId $vnet1.Id

#

# Create peering azure-aads-vnet-01

#

$res = Select-AzSubscription -Subscription $ws.subname

$peer = $ws.short+'-peer-02'

Write-Host 'create hub-spoke azure-aads-vnet-01 peer02', $peer

$vnet1 = Get-AzVirtualNetwork -Name $ws.vnet -ResourceGroupName $ws.rg

$res = Select-AzSubscription -Subscription $HUBWORKSPACENAME

$vnet2 = Get-AzVirtualNetwork -Name $PEER2VNET -ResourceGroupName $PEER2RG

$res = Select-AzSubscription -Subscription $ws.subname

# Peer VNet1 to VNet2.

$res = Add-AzVirtualNetworkPeering -Name $peer -VirtualNetwork $vnet1 -RemoteVirtualNetworkId $vnet2.Id

$res = Select-AzSubscription -Subscription $HUBWORKSPACENAME

# Peer VNet2 to VNet1.

$res = Add-AzVirtualNetworkPeering -Name $peer -VirtualNetwork $vnet2 -RemoteVirtualNetworkId $vnet1.Id

}

Read-Host -Prompt "Create Automation accounts and associated Event Grid subscriptions <enter>"

#

foreach($ws in CreateworkspaceList)

{

$res = Select-AzSubscription -Subscription $ws.subname

#

# Create automation account facilitating Event subscriptions

#

Write-Host 'create automation account', $ws.automate

$res = New-AzAutomationAccount -Name $ws.automate -Location WestEurope -ResourceGroupName $ws.rg

$res = Get-AzAutomationAccount -ResourceGroupName $ws.rg -Name $ws.automate

#

# Define the Event Subscription parameters to the predefined webhook

#

$includedEventTypes = "Microsoft.Resources.ResourceWriteSuccess"

$advancedFilter = @{operator="StringContains"; key="subject"; values=@("Microsoft.compute/virtualMachines/")}

Write-Host 'create event grid subscription', $event

$res = New-AzEventGridSubscription -EventSubscriptionName $ws.event `

-Endpoint $ONVMCREATE `

-IncludedEventType $includedEventTypes -AdvancedFilter @($advancedFilter)

}

Read-Host -Prompt "Create Storage Account firewalling <enter>"

#

foreach($ws in CreateworkspaceList)

{

$res = Select-AzSubscription -Subscription $ws.subname

Write-Host 'create storage account firewalling', $ws.sa

Set-Item Env:\SuppressAzurePowerShellBreakingChangeWarnings "True"

$vn = Get-AzVirtualNetwork -ResourceGroupName $ws.rg -Name $ws.vnet

$prefix = (Get-AzVirtualNetworkSubnetConfig -VirtualNetwork $vn -Name 'Mapped').AddressPrefix

$res = Set-AzVirtualNetworkSubnetConfig -VirtualNetwork $vn -Name "Mapped" `

-AddressPrefix $prefix -ServiceEndpoint "Microsoft.Storage" | Set-AzVirtualNetwork

Set-Item Env:\SuppressAzurePowerShellBreakingChangeWarnings "True"

$vn = Get-AzVirtualNetwork -ResourceGroupName $ws.rg -Name $ws.vnet

$prefix = (Get-AzVirtualNetworkSubnetConfig -VirtualNetwork $vn -Name 'UnMapped').AddressPrefix

$res = Set-AzVirtualNetworkSubnetConfig -VirtualNetwork $vn -Name "UnMapped" `

-AddressPrefix $prefix -ServiceEndpoint "Microsoft.Storage" | Set-AzVirtualNetwork

$subnet = Get-AzVirtualNetwork -ResourceGroupName $ws.rg -Name $ws.vnet | Get-AzVirtualNetworkSubnetConfig

#

# cleanup any existing rules.....

$res = Update-AzStorageAccountNetworkRuleSet -ResourceGroupName $ws.rg -AccountName $ws.sa `

-IpRule @() -VirtualNetworkRule @()

#

# set the new firewalling ruleset

$res = Update-AzStorageAccountNetworkRuleSet -ResourceGroupName $ws.rg -AccountName $ws.sa `

-Bypass AzureServices -DefaultAction Deny `

-VirtualNetworkRule (@{VirtualNetworkResourceId=$subnet[0].Id;Action="allow"},@{VirtualNetworkResourceId=$subnet[1].Id;Action="allow"})

}

Read-Host -Prompt "Enable AutoProvsioning MMA and logging to azure-play-loganalytics <enter>"

#

# we use vrc001-prod as loganalytics workspace as ID template

#

$res = Select-AzSubscription $SPOKEWORKSPACENAME

$workspaceID = (Get-AzSecurityWorkSpaceSetting).WorkspaceID

#

foreach ($ws in CreateworkspaceList)

{

$sub = Select-AzSubscription -Subscription $ws.subname

Write-Host 'assign MMA auto provisioning setting and logging', $ws.short

$res = Set-AzSecurityAutoProvisioningSetting -Name "default" -EnableAutoProvision

# $res = Get-AzSecurityAutoProvisioningSetting -Name "default"

$subID = (Get-AzSubscription -SubscriptionName $ws.subname).Id

$scope = '/subscriptions/'+$subID

$res = Set-AzSecurityWorkspaceSetting -Name "default" -Scope $scope -WorkspaceId $workspaceID

}

Write-Host 'Done'