# Suscripción Nueva en Azure

Cloud Computing

Table of Contents

[Suscripción Nueva en Azure 1](#_Toc85466895)

[Revisiones 3](#_Toc85466896)

[Pasos 4](#_Toc85466897)

[Grupo Arquitectura (tareas) 4](#_Toc85466898)

[Seguridad (tareas) 5](#_Toc85466899)

[Anejos 6](#_Toc85466900)

Revisiones

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| Sept/20/2021 | 1.0 | Initial version | Enterprise Architecture |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Pasos

## Grupo Arquitectura (tareas)

1. **Usar un Private Session del browser** y entrar a suscripción en Partner Portal como usuario llc
2. Ir a Cost and Managements, escoger opción de Cost Management a la izq.
3. **Verificar que el scope diga customer for commerce root y termina en 4bbd-8d4f-883718c2bf9b para Evertec, si no es así cambia el scope a Evertec Inc., no Evertec LLC, este debe ser el scope:**
4. **Graphical user interface, text, application

   Description automatically generated**
5. Buscar Azure subscriptions en la izq. Y dar clic en +Add
6. Crear suscripción bajo EvertecLLC. Si esto no te trae la siguiente pantalla entonces debes ir a través de un Search en Subscription y de allí darle +Add
7. Application, table

   Description automatically generated
8. Crear Automation Account aut-[account name]-runbook1 en un RG nuevo rg-[account name-autoacautomation[##], es necesario hacerlo con la cuenta llc para crear el Run as Account
9. Dar logout y volver a entrar como llc para que puedas tener el scope de Evertec LLC, esta vez escoger la subs nueva
10. Crear usuario (tu usuario como Owner, crear a Lila Pagan y Melvin Mora)
11. Desde VSCode correr el script de registrar providers ([anejo](#_Anejos)) a nivel de la suscripción, al final verificar que el provider de Blueprints este registrado
12. En tu pc vas a necesitar el modulo llamado AZ.Automation , usa este comando si necesitas instalarlo install-module -name az.automation. Igual puedes instalar cualquier otro modulo que te pida el proceso
13. Actualiza los tags y el subscription ID de los archivos CreateTagToAllResourceGroups.ps1 y CreateBackupTagToAllResources.ps1 dentro del folder runbooks en tu home path. Si no tienes los archivos puedes verificar la sección de [anejo](#_Script_de_tag) al final de este documento para que los copies en el folder runbooks de tu home path. Recuerda actualizar los tags y el subscription ID de estos scripts con la información de los Tags de la suscripción creada
14. Correr el script SetTagsToAllCurrentResources desde VSCode para crear los runbooks, publicar los runbooks, crear los schedules y atachar los schedules a los runbooks. El proceso toma su tiempo cuando finalice proseguir con la verificación
15. **Verificar que los schedules que tengan horario de PR, los modulos no estén en failed y código de los scripts de power shell**
16. FIN

De nuestra parte solo falta solo el crear la cuenta en Cloudcheckr una vez aparezca en el portal.

## Seguridad (tareas)

Ahora pasas la suscripción a Seguridad de Información para que ellos finalicen la toma de control de la suscripción. Avisar por correo y por Teams a Lila Pagan, Melvin Mora y Jorge Torres Vargas. Esto lo trabaja Lila Pagan o Melvin Mora:

1. Crear Management Groups bajo EvertecLLC y mover la suscripción al Management Groups nuevo
2. Eliminar los Owners que no sean de Seguridad de Información
3. Correr los blueprints CommonPolicies, CommonResourceGroups, CreateEventHubName, CreateLogAnalyticsWorkspace, CreateStorageAccountforEventHub
4. Crear los Custom Roles ASG Team Monitoring [account name], ASGNetworkWatcher [account name], Network Watcher Managements [account name], Stop Restart Start VM [account name], Virtual Machine Update Manager [account name] (en la cuenta plot crt se unificaron en un solo role Stop Restart Start VM+Virtual Machine Update Manager), hay otros customs que fueron creados en varias suscripciones Lila y Melvin ya conocen cuales son y de ser necesarios los incorporan
5. Asignar los permisos a los grupos de cada custom role a nivel de la suscripción

# Anejos

Script de configurar Providers en suscripción (Powershell):

############################################

# Enterprise Architecture #

# Walter Gonzalez #

# June 2020 #

############################################

#Registrar providers mas comunes

#######################################################

# Must be run from the PowerShell IDE or Cloud Shell #

#######################################################

# Variables

$providers = @("Microsoft.Advisor", "Microsoft.Security", "Microsoft.AlertsManagement", "Microsoft.PolicyInsights", "Microsoft.Network", "Microsoft.EventHub", "Microsoft.Storage", `

"Microsoft.Compute", "Microsoft.ResourceHealth", "Microsoft.SqlVirtualMachine", "Microsoft.RecoveryServices", "Microsoft.insights", `

"Microsoft.Automation", "Microsoft.Sql", "Microsoft.OperationalInsights", "Microsoft.OperationsManagement", "Microsoft.SecurityInsights", `

"Microsoft.ManagedIdentity", "Microsoft.Capacity","Microsoft.Blueprint")

function setProviders {

foreach ($list in $providers) {

Register-AzResourceProvider -ProviderNamespace $list

write-host $list

}

}

function listSubscriptions {

Clear-Host

$c = 0

foreach ($list in $subscriptionlist) {

write-host $c "- " $list.Name

$c ++

}

$subscriptionID = Read-Host -Prompt 'Please enter the number of the SubscriptionId'

return $subscriptionlist[$subscriptionID]

}

function selectSubscription {

Select-AzSubscription -SubscriptionId $mySubscription.Id #-TenantId $mySubscription.TenantId

}

#Start

Clear-Host

$login = Read-Host -Prompt 'Do you want or need to login?(y/n)'

if ($login -eq "Y" -or $login -eq "y") {

Connect-AzAccount

}

Set-ExecutionPolicy -Scope Process - ExecutionPolicy Bypass

write-host "Getting Azure Information, please wait..."

$subscriptionlist = @(Get-AzSubscription)

$mySubscription = listSubscriptions

write-host "Connecting to subscription..."

selectSubscription

write-host $mySubscription.Name $mySubscription.Id

setProviders

## Script de tag de recursos para Runbook CreateTagtoAllResourceGroups:

Estos proximos dos scripts de PowerShell son de los dos Runbooks y deben estar creados en los path:

Win:

c:\runbooks\

MacOS:

~/runbooks/

Runbook scripts Recuerda actualizer el subscription ID y los tags:

############################################################################

# Walter Gonzalez #

# Mayo 2020 #

# #

############################################################################

# Update Tags - Applied to all the resources in a subscription

############################################################################

# CHANGE THIS PER SUBSCRIPTION #

############################################################################

# RunBook for Update Tags - Applied at the RG level

#Parameters -----------------------------------------------------------------------------------------------------

$mySubscription = " TBD "

#Code of the application the resource is associated with in the CMDB.

$applicationCode = " TBD "

#Name of the application, service, or workload the resource is associated with in the CMDB

$applicationName = " TBD "

#Name of the module or component associated with the application.

$applicationModule = " TBD "

#Evertec group responsible for the implementation of the project

$group = "Digital Solutions"

#Owner of the application, workload, or service.(email)

$ownerName = " TBD "

#Accounting cost center associated with this resource

$costCenter = " TBD "

#Deployment environment of this application, workload, or service

$env = " TBD "

#Person responsible for approving costs related to this resource.(email)

$approver = " TBD "

#Top-level division of your company that owns the workload the resource belongs to.

#In smaller organizations, this may represent a single corporate or shared top-level organizational element

$businessUnit = " TBD "

#User that requested the creation of this application.(email)

$requestor = " TBD "

#Service Level Agreement level of this application, workload, or service

$serviceClass = " TBD "

#Date when this application, workload, or service was first deployed

$startDate = " TBD "

#End of parameters----------------------------------------------------------------------------------------------------

#Start

# Ensures you do not inherit an AzContext in your runbook

Disable-AzContextAutosave –Scope Process

#Authenticate using the account configured when the automation account was created

$connection = Get-AutomationConnection -Name AzureRunAsConnection

while(!($connectionResult) -And ($logonAttempt -le 10))

{

$LogonAttempt++

# Logging in to Azure...

$connectionResult = Connect-AzAccount `

-ServicePrincipal `

-Tenant $connection.TenantID `

-ApplicationId $connection.ApplicationID `

-CertificateThumbprint $connection.CertificateThumbprint

Start-Sleep -Seconds 30

}

Select-AzSubscription -SubscriptionId $mySubscription

#Update tags for all services

$allResourcesGroup = Get-AzResourceGroup

$tags = @{"applicationCode" = $applicationCode; "applicationName" = $applicationName;"applicationModule" = $applicationModule; "group" = $group; "ownerName" = $ownerName;"costCenter" = $costCenter; "env" = $env; "approver"= $approver; "businessUnit"= $businessUnit; "requestor" = $requestor; "serviceClass" = $serviceClass; "startDate" = $startDate}

foreach($resourceList in $allResourcesGroup){

write-output $resourceList.ResourceId

write-output $resourceList.Location

write-output $resourceList.Tags

if ($resourceList.Tags.applicationCode -eq $null){

$resourceGroup = Get-AzResourceGroup -Name $resourceList.ResourceGroupName

Update-AzTag -ResourceId $resourceGroup.ResourceId -tag $tags -Operation Merge

}

}

## Script de etiquetas de recursos para Runbook CreateBackupTagtoAllResources:

############################################################################

# Walter Gonzalez #

# Junio 2020 #

# #

############################################################################

#Just for VMs

# Update Tags - Applied to all the VMs in a subscription

############################################################################

# CHANGE THIS PER SUBSCRIPTION #

############################################################################

# RunBook for Update Tags - Applied at the Subscription level

#Parameters -----------------------------------------------------------------------------------------------------

$mySubscription = "TBA"

#The resource needs a backup plan? The value will be set in the VSP or the Backup Team will set the value in accordance with the owner

#End of parameters----------------------------------------------------------------------------------------------------

$backup = ""

function doLogin{

#Start

# Ensures you do not inherit an AzContext in your runbook

Disable-AzContextAutosave –Scope Process

#Authenticate using the account configured when the automation account was created

$connection = Get-AutomationConnection -Name AzureRunAsConnection

while(!($connectionResult) -And ($logonAttempt -le 10))

{

$LogonAttempt++

# Logging in to Azure...

$connectionResult = Connect-AzAccount `

-ServicePrincipal `

-Tenant $connection.TenantID `

-ApplicationId $connection.ApplicationID `

-CertificateThumbprint $connection.CertificateThumbprint

Start-Sleep -Seconds 30

}

}# end login func

function doTagging { param( [string]$myTag )

Select-AzSubscription -SubscriptionId $mySubscription

#Update tags for all services

$allResources = Get-AzVM | Select Id,Name,Tags

$tags = @{"backup" = $myTag}

foreach($resourceList in $allResources){

write-output $resourceList.id

write-output $resourceList.Name

write-output $resourceList.tags

if ($resourceList.Tags.backup -eq $null){

Update-AzTag -ResourceId $resourceList.id -Tag $tags -Operation Merge

write-output $resourceList.tags

}

}

}#end doTagging func

#Func calls

doLogin

doTagging $backup

Este script debes correrlo desde VSCode. Recuerda actualizer los tags y el ID de la suscripción que se encuentran dentro de este script al igual que los scripts que almacenaste en el path c:\runbooks\ para Windows o ~/runbooks/ para Mac OS

Script:

############################################################################

# Walter Gonzalez #

# Mayo 2020 #

# #

############################################################################

# Update Tags - Applied to all the resources in a subscription

############################################################################

# CHANGE THIS PER SUBSCRIPTION #

############################################################################

# RunBook for Update Tags - Applied at the Subscription level

#

#

#

#

# RUN THIS VSCODE

# RUN THIS VSCODE

# RUN THIS VSCODE

# RUN THIS VSCODE

# RUN THIS VSCODE

#

#

#

#

#

#

$modules = @("az.accounts", "az.compute", "az.automation", "az.storage", "az.resources", "az.profile")

#Parameters -----------------------------------------------------------------------------------------------------

$mySubscription = "TBD"

#Code of the application the resource is associated with in the CMDB.

$applicationCode = "TBD"

#Name of the application, service, or workload the resource is associated with in the CMDB

$applicationName = "TBD"

#Name of the module or component associated with the application.

$applicationModule = "TBD"

#Evertec group responsible for the implementation of the project

$group = "TBD"

#Owner of the application, workload, or service.(email)

$ownerName = "TBD"

#Accounting cost center associated with this resource

$costCenter = "TBD"

#Deployment environment of this application, workload, or service

$env = "TBD"

#Person responsible for approving costs related to this resource.(email)

$approver = "TBD"

#Top-level division of your company that owns the workload the resource belongs to.

#In smaller organizations, this may represent a single corporate or shared top-level organizational element

$businessUnit = "TBD"

#User that requested the creation of this application.(email)

$requestor = "TBD"

#Service Level Agreement level of this application, workload, or service

$serviceClass = "TBD"

#Date when this application, workload, or service was first deployed

$startDate = "TBD"

#End of parameters----------------------------------------------------------------------------------------------------

#Update tags for all CURRENT services

function createRunbooks {

write-host "Installing Automation Account Runbooks..."

New-AzAutomationRunbook -Name CreateTagToAllResourceGroups -Type PowerShell -ResourceGroupName $rg -AutomationAccountName $autoacc

New-AzAutomationRunbook -Name CreateBackupTagToAllResources -Type PowerShell -ResourceGroupName $rg -AutomationAccountName $autoacc

}

function putCodeintoScript {

#Put code into runbook

##########################################

#

#

# Important

#

#

##########################################

write-host "Installing Automation Account Powershell script code..."

$osType = Read-host "What OS are you using (Win = 1, Mac = 2), (Ctrl/c to quit)"

if ($osType -eq "1") {

#Win

$path1 = "c:\runbooks\CreateTagToAllResourceGroups.ps1"

$path1 = "c:\runbooks\CreateBackupTagToAllResources.ps1"

}elseif ($osType -eq "2") {

#Mac

$path1 = "~/runbooks/CreateTagToAllResourceGroups.ps1"

$path2 = "~/runbooks/CreateBackupTagToAllResources.ps1"

}else {

exit

}

Import-AzAutomationRunbook -Path $path1 -Name CreateTagToAllResourceGroups -Type PowerShell -AutomationAccountName $autoacc -ResourceGroupName $rg -Force

Import-AzAutomationRunbook -Path $path2 -Name CreateBackupTagToAllResources -Type PowerShell -AutomationAccountName $autoacc -ResourceGroupName $rg -Force

Publish-AzAutomationRunbook -Name CreateTagToAllResourceGroups -AutomationAccountName $autoacc -ResourceGroupName $rg

Publish-AzAutomationRunbook -Name CreateBackupTagToAllResources -AutomationAccountName $autoacc -ResourceGroupName $rg

}

function createSchedules {

write-host "Installing Automation Account Schedules..."

#Set schedule time

$StartTime1 = Get-Date "3:00:00"

$StartTime2 = Get-Date "3:30:00"

#Create schedules

New-AzAutomationSchedule -AutomationAccountName $autoacc -Name "sch-RGtag1" -TimeZone "PR" -StartTime $StartTime1 -DayInterval 1 -ResourceGroupName $rg

New-AzAutomationSchedule -AutomationAccountName $autoacc -Name "sch-BKPtag1" -TimeZone "PR" -StartTime $StartTime2 -DayInterval 1 -ResourceGroupName $rg

Start-Sleep -s 30 # wait for all resources to be deployed from the above command

#Publish runbook

Publish-AzAutomationRunbook -AutomationAccountName $autoacc -Name CreateTagToAllResourceGroups -ResourceGroupName $rg

Publish-AzAutomationRunbook -AutomationAccountName $autoacc -Name CreateBackupTagToAllResources -ResourceGroupName $rg

Start-Sleep -s 60 # wait for all resources to be deployed from the above command

#Link schedule to runbook

Register-AzAutomationScheduledRunbook -Name CreateTagToAllResourceGroups -ResourceGroupName $rg -AutomationAccountName $autoacc -ScheduleName "sch-RGtag1"

Register-AzAutomationScheduledRunbook -Name CreateBackupTagToAllResources -ResourceGroupName $rg -AutomationAccountName $autoacc -ScheduleName "sch-BKPtag1"

}

function setModules{

write-host "Installing Automation Account Modules..."

foreach($dep in $modules){

$module = Find-Module -Name $dep

$link = $module.RepositorySourceLocation + "/package/" + $module.Name + "/" + $module.Version

New-AzAutomationModule -AutomationAccountName $autoacc -Name $module.Name -ContentLinkUri $link -ResourceGroupName $rg

Start-Sleep -s 300

}

}

function listSubscriptions {

Clear-Host

$c = 0

foreach ($list in $subscriptionlist) {

write-host $c "- " $list.Name

$c ++

}

$subscriptionID = Read-Host -Prompt 'Please enter the number of the SubscriptionId (Ctrl/c to quit)'

return $subscriptionlist[$subscriptionID]

}

function selectSubscription {

Select-AzSubscription -SubscriptionId $mySubscription.Id #-TenantId $mySubscription.TenantId

}

#Start

Clear-Host

$login = Read-Host -Prompt 'Do you want or need to login?(y/n), (Ctrl/c to quit)"'

if ($login -eq "Y" -or $login -eq "y") {

Connect-AzAccount

}

Set-ExecutionPolicy -Scope Process - ExecutionPolicy Bypass

write-host "Getting Azure Information, please wait..."

$subscriptionlist = @(Get-AzSubscription)

$mySubscription = listSubscriptions

write-host "Connecting to subscription..."

#function call:

selectSubscription

write-host $mySubscription.Name $mySubscription.Id

$allResources = Get-AzResource | Select-Object Id,Tags

$tags = @{"applicationCode" = $applicationCode; "applicationName" = $applicationName;"applicationModule" = $applicationModule; "group" = $group; "ownerName" = $ownerName;"costCenter" = $costCenter; "env" = $env; "approver"= $approver; "businessUnit"= $businessUnit; "requestor" = $requestor; "serviceClass" = $serviceClass; "startDate" = $startDate}

foreach($resourceList in $allResources){

write-output $resourceList.id

write-output $resourceList.tags

Update-AzTag -ResourceId $resourceList.id -Tag $tags -Operation Merge

}

#function call:

$rg = Read-Host "Please enter Automation Account Resource Group (Ctrl/c to quit)"

$autoacc = Read-Host "Please enter Automation Account name (Ctrl/c to quit)"

setModules

createRunbooks

createSchedules

putCodeintoScript