Automation account script to start and stop VM based on their tag value

Task-

how can I use azure tags for stopping servers at 7 pm and starting by 6 am by using azure automation?

Step1-

* Go to the Azure portal and create a new Azure Automation account if you don't already have one.

A screenshot of a computer

Description automatically generated

Automation account created.

A screenshot of a computer

Description automatically generated

Step 2-

Assign required tags to targeted vm’s

Name- AutoShutdownSchedule

Value- as required (7PM-6AM)

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Step 3-

* In your Azure Automation account, create a new PowerShell runbook. This runbook will contain the script to start and stop the VMs based on the tag value.

Go to automation account and create runbook.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Go for review and create.

Step 4-

* Use the PowerShell script as a template for your runbook:

Script-

param (

[string]$TagName = "AutoShutdownSchedule",

[string]$TagValue = "7PM-6AM"

)

# Get the current time

$currentTime = Get-Date

# Define the shutdown and startup times

$shutdownTime = [datetime]::ParseExact("19:00", "HH:mm", $null)

$startupTime = [datetime]::ParseExact("06:00", "HH:mm", $null).AddDays(1)

# Get all VMs with the specified tag

$vms = Get-AzVM -Status | Where-Object { $\_.Tags[$TagName] -eq $TagValue }

foreach ($vm in $vms) {

if ($currentTime -ge $shutdownTime -and $currentTime -lt $startupTime) {

# Stop the VM

Stop-AzVM -ResourceGroupName $vm.ResourceGroupName -Name $vm.Name -Force

} elseif ($currentTime -ge $startupTime) {

# Start the VM

Start-AzVM -ResourceGroupName $vm.ResourceGroupName -Name $vm.Name

}

}

A screenshot of a computer

Description automatically generated

Note- if you want to edit the script on vs code, we need to install “**Azure Automation**” extension for that.

A screenshot of a computer

Description automatically generated

Then we can open and edit it on vs code.

A screenshot of a computer

Description automatically generated

Step 5-

* Create a schedule in your Azure Automation account to run the runbook at regular intervals (e.g., every hour). This ensures that the script checks the current time and starts or stops the VMs as needed.

A screenshot of a computer

Description automatically generated

Also can verify the parameters.

A screenshot of a computer

Description automatically generated

Step 6-

Test-

Scheduler created and is in “on” state.

A screenshot of a computer

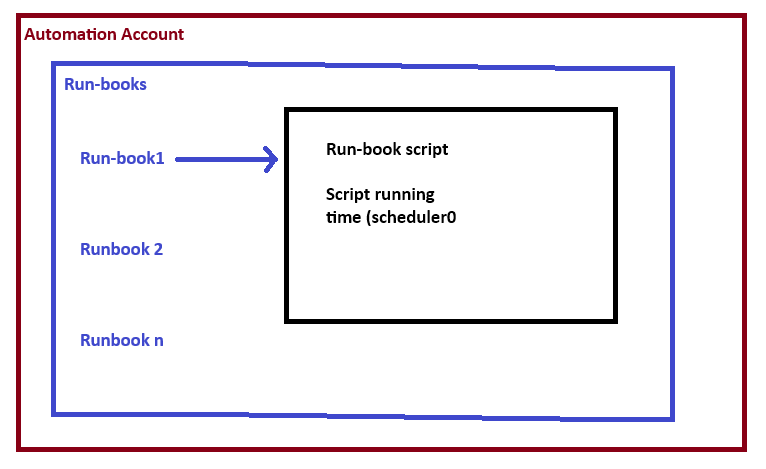
Description automatically generated

To modify the script we can go to view section.

A screenshot of a computer

Description automatically generated

Diagram-



If we target to run the runbook manually. We can start it manually and observe output.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Note-

We can also use start stop by using function app. Azure make it available on market place.

A screenshot of a computer

Description automatically generated

In my case all the targeted vms are in stopped state and the script must start those vms.

A screenshot of a computer

Description automatically generated

Script run automatically at 8:30 as we had set start from 8:30 and run every 1 hr interval.

A screenshot of a computer

Description automatically generated

Now still vms are not running. It should start at next script run by 9:30.

9:30 script runs and completed. But, non of the vm are started.

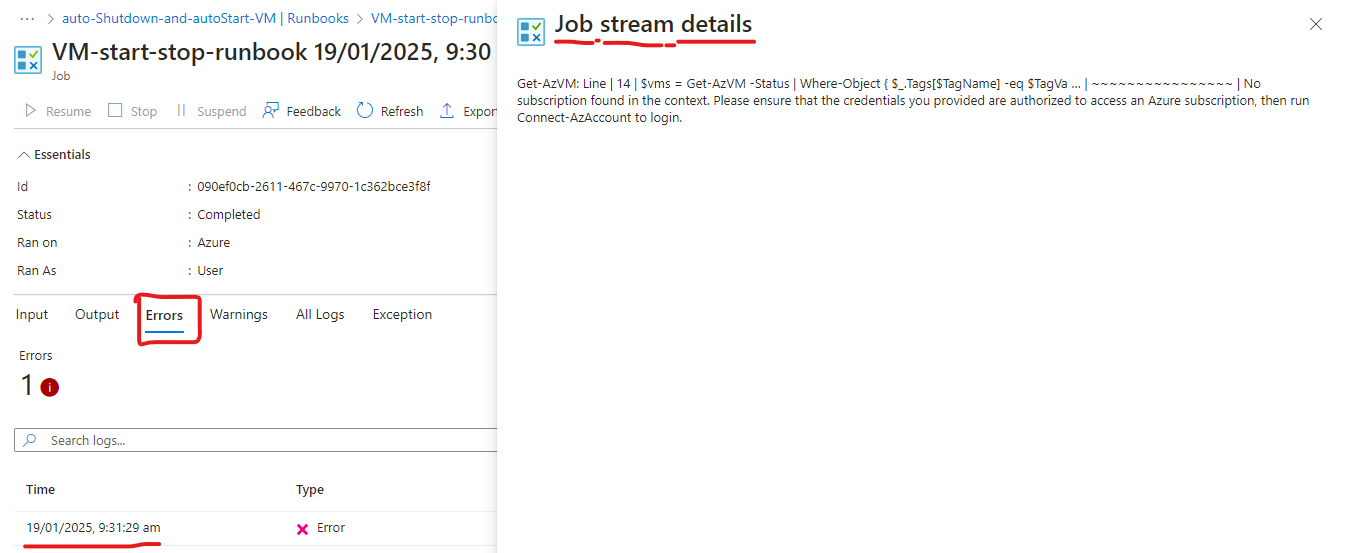
A screenshot of a computer

Description automatically generated

A screenshot of a chat

Description automatically generated

Found 1 error.



Error-

Get-AzVM: Line | 14 | $vms = Get-AzVM -Status | Where-Object { $\_.Tags[$TagName] -eq $TagVa … | ~~~~~~~~~~~~~~~~ | No subscription found in the context. Please ensure that the credentials you provided are authorized to access an Azure subscription, then run Connect-AzAccount to login.

This was due to az login is required on script.

And az-connect we can directly use here for that we need to disable Windows Authentication Manager (WAM) login method:

>> Update-AzConfig -EnableLoginByWam $false

And also as we don’t open it in new browser and do log in to it explicitly. We need to create a cliend id ad secret to use for the same.

>> az ad sp create-for-rbac --name "vm-automation" --role "Contributor" --scopes "/subscriptions/abe10fc9-2089-4a76-8fb7-e9a93870bafc"

A screenshot of a computer program

Description automatically generated

Updated script-

param (

    [string]$TagName = "AutoShutdownSchedule",

    [string]$TagValue = "8AM-9AM"

)

# Define the current time

$currentTime = Get-Date

# Define shutdown and startup times

$shutdownTime = (Get-Date -Hour 8 -Minute 0 -Second 0)

$startupTime = (Get-Date -Hour 9 -Minute 0 -Second 0)

# Authenticate using service principal

$tenantId = "30bf9f37-d550-4878-9494-1041656caf27"

$clientId = "ee256eee-82f7-45c4-b5d7-8931519f66d7"

$clientSecret = "93E8Q~r70KGp2J~MCaZaWvrpC~8lBbZ\_cY.Bnbj~"

$subscriptionId = "abe10fc9-2089-4a76-8fb7-e9a93870bafc"

# Create a secure string for the client secret

$securePassword = ConvertTo-SecureString -String $clientSecret -AsPlainText -Force

# Create a PSCredential object

$credential = New-Object System.Management.Automation.PSCredential ($clientId, $securePassword)

# Connect to Azure

Connect-AzAccount -ServicePrincipal -TenantId $tenantId -Credential $credential

# Set the subscription context

Set-AzContext -SubscriptionId $subscriptionId

# Register resource provider

Register-AzResourceProvider -ProviderNamespace Microsoft.Compute

# Get all VMs with the specified tag

$vms = Get-AzVM -Status | Where-Object { $\_.Tags[$TagName] -eq $TagValue }

# Iterate through the VMs and apply the schedule

foreach ($vm in $vms) {

    if ($currentTime -ge $shutdownTime -and $currentTime -lt $startupTime) {

        # Stop the VM

        Stop-AzVM -ResourceGroupName $vm.ResourceGroupName -Name $vm.Name -Force

        Write-Output "Stopped VM: $($vm.Name)"

    } elseif ($currentTime -ge $startupTime -and $currentTime -lt $shutdownTime.AddHours(24)) {

        # Start the VM

        Start-AzVM -ResourceGroupName $vm.ResourceGroupName -Name $vm.Name

        Write-Output "Started VM: $($vm.Name)"

    }

}

If the script did not worked. Better to run from cloudshell and test.

A screenshot of a computer program

Description automatically generated

Result-

A screenshot of a chat

Description automatically generated

Another way o do az login

Authenticate using Managed Identity

>> Connect-AzAccount -Identity

Make sure the identity is enabled on automation amount.

Final script-

param (

    [string]$TagName = "AutoShutdownSchedule",

    [string]$TagValue = "8AM-9AM"

)

# Authenticate using service principal

$tenantId = "30bf9f37-d550-4878-9494-1041656caf27"

$clientId = "ee256eee-82f7-45c4-b5d7-8931519f66d7"

$clientSecret = "93E8Q~r70KGp2J~MCaZaWvrpC~8lBbZ\_cY.Bnbj~"

$subscriptionId = "abe10fc9-2089-4a76-8fb7-e9a93870bafc"

# Create a secure string for the client secret

$securePassword = ConvertTo-SecureString -String $clientSecret -AsPlainText -Force

# Create a PSCredential object

$credential = New-Object System.Management.Automation.PSCredential ($clientId, $securePassword)

# Connect to Azure

# Authenticate using Managed Identity

# Connect-AzAccount -Credential $credential

Connect-AzAccount -ServicePrincipal -TenantId $tenantId -Credential $credential

# Set the subscription context

Set-AzContext -SubscriptionId $subscriptionId

# Get all VMs with the specified tag

$vms = Get-AzVM -Status | Where-Object { $\_.Tags[$TagName] -eq $TagValue }

foreach ($vm in $vms) {

    Write-Output "VM Name: $($vm.Name)"

}

# Check if any VMs were found

if ($vms.Count -eq 0) {

    Write-Output "No VMs found with tag $TagName = $TagValue"

} else {

    # Iterate through the VMs and start them immediately

    foreach ($vm in $vms) {

        # Start the VM immediately

        Start-AzVM -ResourceGroupName $vm.ResourceGroupName -Name $vm.Name

        Write-Output "Started VM: $($vm.Name) immediately."

    }

}

Result-

A screenshot of a computer

Description automatically generated

Issue and fixing-

A screenshot of a computer

Description automatically generated

There were two variables with same name, so the script was having conflict in this

Issue 2-

# Connect to Azure

# authorized using created service principal

Connect-AzAccount -ServicePrincipal -TenantId $tenantId -Credential $credential

The above is the correct code. Here my miss was I created service principal. ( created above by using cmd on AZ cli >>az ad sp create-for-rbac --name "vm-automation" --role "Contributor" --scopes "/subscriptions/abe10fc9-2089-4a76-8fb7-e9a93870bafc")

But in code I was using managed identity for which I did not assign the permission.

So it was not taking the subscription.

Wrong code which I was doing

>> Connect-AzAccount -identity

Issue 3-

Here in above step I corrected my code and taken created service principal to get access to azure. But if I do want to do it by using identity, I can. In this case I don’t required to create any additional service principal. I can just directly assign the required permission to the identity of the automation account and use it.

How to archive this.

Go to your automation account.

And go for identity.

A screenshot of a computer

Description automatically generated

Go for role assignment and assign role (contributer- to start stop VM/ reader- to only display data)

A screenshot of a computer

Description automatically generated

Alternative way we can go to subscription and assign role for automation account.

Change in code ( do authentication connect to az account by using identity)

# Connect to Azure

# Authenticate using Managed Identity

Connect-AzAccount -identity

A screenshot of a computer

Description automatically generated

Issue 4-

If we are using service principal to have authentication. We are using that as hard coded value in our script.

A computer code with numbers and letters

Description automatically generated

If we want it to use for multiple automation runbook script then we need to place it to all script. To avoid this we can store it as variables on automation account and use it.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Use in script.

We can encrypt those values as well to hide the value.

Use in script.

Get-AutomationVariable -Name “variable-name”

A screenshot of a computer program

Description automatically generated

Run the script to check. It works.

A screenshot of a computer program

Description automatically generated

Note- we can do the authentication from credential as well.

A screenshot of a computer

Description automatically generated