Advanced Live Training:

Power on VMs for X Hours

Revision 1

June 20, 2024

Lesson 4

Power on VMs for X hours:

- ✓ Uses runtime parameters
- ✓ Uses Az module to modify Azure

Lab: Azure Runbook Script

Power on VMs for X Hours

Sometimes it is necessary to have VMs powered on for an extended period, for example when running a centralized patching service once a week.

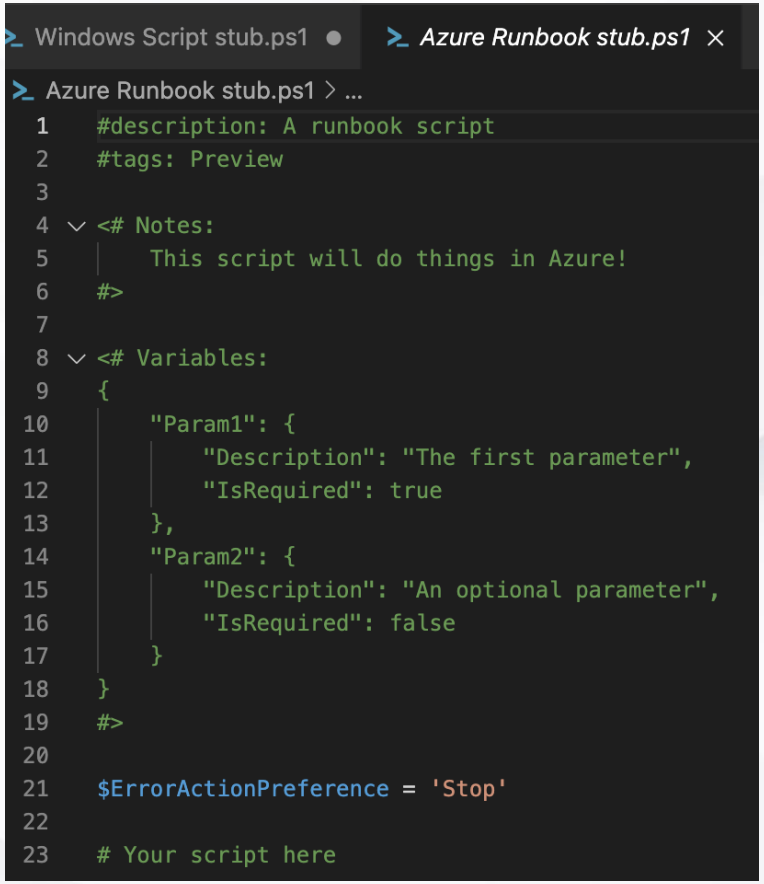
This script will ensure all hosts in a host pool are powered on for the specified number of hours, regardless of the auto-scale settings.

Lab: Azure Runbook Script

Power on VMs for X Hours

Begin by copying the Azure Runbook “Stub” file

Note: The `Variables` section is json-formatted text that defines the parameters you can provide when running the script. NMM parses this section and provides a UI for entering these values at runtime.



Lab: Azure Runbook Script

Power on VMs for X Hours

This script should accept the following parameters:

- ✓ **`HostPoolName`**: The host pool containing the VMs.
- ✓ **`HostPoolResourceGroup`**: The RG containing the host pool.
- ✓ **`Hours`**: An integer indicating how long the hosts should remain on.

Hint: The stub file illustrates how to define the above parameters.

Lab: Azure Runbook Script

Power on VMs for X Hours

This script will:

1. Set a tag on all VMs to exclude hosts from scale-in activity for X hours.
2. Power on all hosts.

Hint: Both of these tasks can be accomplished using the Az module commands.

Lab: Azure Runbook Script

Power on VMs for X Hours

Tip:

- ✓ When developing an Azure runbook script, you can connect to Azure using your own credentials, if your account has permissions to perform the actions required by the script.
- ✓ Do not add the authentication line to the script itself as Nerdio Manager will take care of authentication for the runbook.

```
PS C:\Users\NWagner> Connect-AzAccount -SubscriptionId xxxxxxxx-xxxx-xxxx-xxxxxxxxxxxx
```

Lab: Azure Runbook Script

Power on VMs for X Hours

Tip:

- ✓ If you defined **HostPoolName**, **HostPoolResourceGroup**, and **Hours** in the **variables** json section of the script, you should have access to the variables **$HostPoolName**, **$HostPoolResourceGroup,** and **$Hours** in your script.
- ✓ For testing and development, you can define these variables manually in the console:

```
PS C:\Users\NWagner> $HostPoolName = 'MyHostPool'
PS C:\Users\NWagner> $HostPoolResourceGroup = 'MyRG'
PS C:\Users\NWagner> $Hours = 8
```

Lab: Azure Runbook Script

Power on VMs for X Hours

Retrieving the VMs:

You can find all hosts in the host pool using:

```
Get-AzWvdSessionHost –HostPoolName $HostPoolName -ResourceGroupName $HostPoolResourceGroup
```

However, the session host's name and VM name are not the same. You'll need to derive a list of VM names from the session host names, then retrieve the VMs from Azure using **`Get-AzVM`**.

Lab: Azure Runbook Script

Power on VMs for X Hours

Setting a VM tag to exclude from scale-in:

The tag name is ‘WAP_SCALE_IN_RESTRICTION’, and the value needs to be in a specific format:

`2024-06-15T08:00;Central Standard Time`

```
$RestrictUntil = (Get-Date).AddHours([int]$Hours)
$TimeZoneId = (Get-TimeZone).id

Write-output "Setting VM Tags"
foreach ($VM in $VMs) {
    $tags = $vm.tags

    # Add the scale in restriction tag to prevent Nerdio from turning the VMs off
    $tags["WAP_SCALE_IN_RESTRICTION"] = $RestrictUntil.ToString("yyyy-MM-ddTHH") + ";$TimeZoneId"
    Set-AzResource -ResourceGroupName $vm.ResourceGroupName -Name $vm.name -ResourceType "Microsoft.Compute/VirtualMachines" -Tag $tags -Force
}
```

Lab: Azure Runbook Script

Power on VMs for X Hours

Testing in NMM:

When ready to test your script in NMM, commit the changes to your repository and sync to GitHub. Then, initiate a sync in NMM, and your latest changes will be available in the **Scripted Actions** section.