

Lesson 4



Lab

Power on VM for X hours

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

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 auto-scale settings.

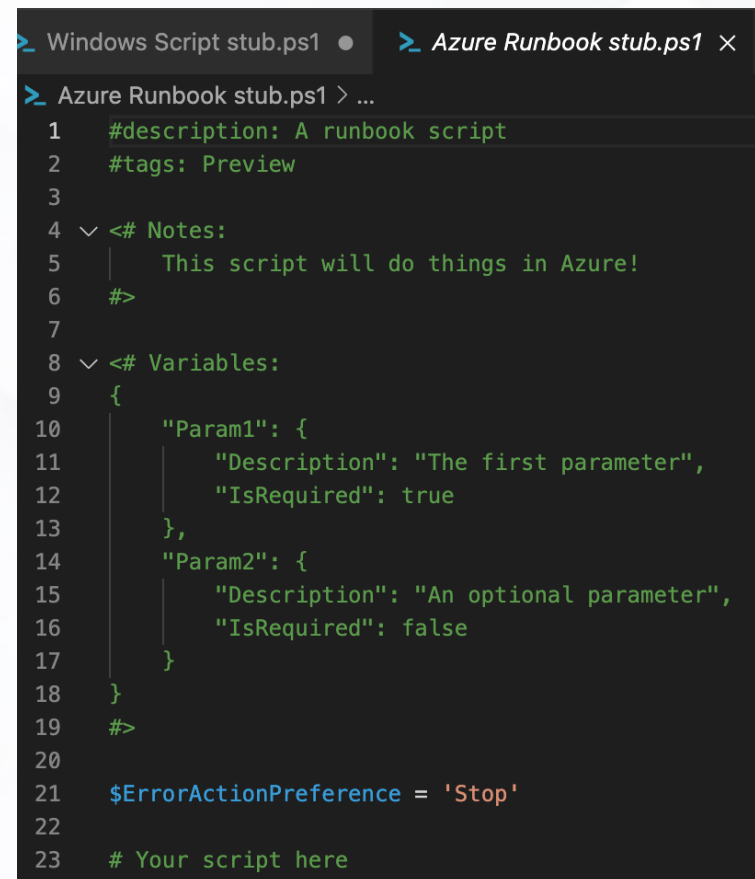
Power on VMs for X hours

Lab | Azure Runbooks Script

Begin by copying the
Azure Runbook Stub file.

Tip:

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 run time.



```
Windows Script stub.ps1 • Azure Runbook stub.ps1 x
Azure Runbook stub.ps1 > ...
1  #description: A runbook script
2  #tags: Preview
3
4  <# Notes:
5  |   This script will do things in Azure!
6  #>
7
8  <# Variables:
9  {
10     "Param1": {
11         "Description": "The first parameter",
12         "IsRequired": true
13     },
14     "Param2": {
15         "Description": "An optional parameter",
16         "IsRequired": false
17     }
18 }
19 #>
20
21 $ErrorActionPreference = 'Stop'
22
23 # Your script here
```

Power on VMs for X hours

Lab | Azure Runbooks Script

This script should accept the following parameters:

HostPoolName – The host pool containing the VMs.

HostPoolResourceGroup – The resource group 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

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 [AZ Module](#) commands.

Power on VMs for X hours

Lab | Azure Runbooks Script

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 will take care of authentication for the runbook.

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

Power on VMs for X hours

Lab | Azure Runbooks Script

Tip:

If you defined **HostPoolName**, **HostPoolResourceGroup**, and **Hours** in the variables JSON section of the script, you will have access to the variables **\$HostPoolName**, **\$HostPoolResourceGroup**, and **\$Hours** in your script.

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

For testing and development, you can define these variables manually in the console.

Retrieving the VMs:

You can find all hosts in the host pool using:

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

However, the session hosts 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**.

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, e.g.:

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

```
26 $RestrictUntil = (Get-Date).AddHours([int]$Hours)
27 $TimeZoneId = (Get-TimeZone).id
28
29 Write-output "Setting VM Tags"
30 foreach ($VM in $VMs) {
31     $tags = $vm.tags
32
33     # Add the scale in restriction tag to prevent Nerdio from turning the VMs off
34     $tags["WAP_SCALE_IN_RESTRICTION"] = $RestrictUntil.ToString("yyyy-MM-ddTHH") + ";$TimeZoneId"
35     Set-AzResource -ResourceGroupName $vm.ResourceGroupName -Name $vm.name -ResourceType "Microsoft.Compute/VirtualMachines" -Tag $tags -Force
36 }
```

Testing in NMM:

When ready to test your script in NMM, commit your 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.