PSAutoLab Manual v4.21.0

Table of Contents

Introduction 1
PSAutoLab 2
- Overview 3
- Requirements 4
  - Pester Requirement 4
- Installation 5
  - Hyper-V 5
  - Previous Versions 7
  - Note for VMware Users 7
- Aliases and Language 8
- Setup Host 9
  - Lab Summary 9
- Creating a Lab 11
  - A Note on Pluralsight Labs 11
  - Manual Setup 11
  - Unattended Setup 12
  - Stopping a Lab 12
  - Starting a Lab 12
  - Lab Checkpoints 12
  - To Remove a Lab 13
  - Customizing a Lab 13
- Windows 10 Remote Server Administration Tools (RSAT) 15
- Windows Updates 16
- Updating PSAutoLab 17
- Removing PSAutoLab 18
- Pester 19
- Troubleshooting 20
  - Package Provider or Module Installation 20
  - Autolab Configurations 20
- Known Issues 22
  - *I get an error when importing the module* 22
  - *I get an error trying to update Lability* 22
  - Multiple DSC Resources 22
- Acknowledgments 23
- Road Map 24

Detailed Setup Instructions 25
- Pre-Check 26
  - Operating System and Memory 26
  - PowerShell Remoting 26
  - Disk Space 27
  - Virtualization 27

Pester . . . 27
Installation and Configuration . . . 28
Install the Module . . . 28
Setup the Host . . . 28
Setup a Configuration Unattended . . . 29
Manual Configuration Setup . . . 30
Help . . . 31
Using the Environment Prefix . . . 32
Troubleshooting Tips . . . 33
Manually Apply the Configuration . . . 33
Getting Help . . . 35
Updating From PSAutoLab v3.x . . . 36
Before You Upgrade . . . 37
Wipe Labs . . . 37
Remove LabNet . . . 37
Remove Module . . . 37
Delete AutoLab Folder . . . 37
Update and Reboot . . . 38
Verify . . . 39
Install . . . 40
Setup a Configuration . . . 41
Usage FAQ . . . 42
Authoring a Custom Configuration . . . 45
Specify a Different Operating System . . . 45
I'm getting an error when my VM is booting for the first time . . . 45
I'm Still Stuck . . . 46
Module Commands . . . 47
Enable-Internet . . . 48
Synopsis . . . 48
Syntax . . . 48
Description . . . 48
Examples . . . 48
Parameters . . . 48
Inputs . . . 49
Outputs . . . 49
Notes . . . 49
Related Links . . . 49
Get-LabSnapshot . . . 50
Synopsis . . . 50
Syntax . . . 50
Description . . . 50
Examples . . . 50
Parameters . . . 50

Inputs . . . 51
Outputs . . . 51
Notes . . . 51
Related Links . . . 51
Get-LabSummary . . . 52
Synopsis . . . 52
Syntax . . . 52
Description . . . 52
Examples . . . 52
Parameters . . . 53
Inputs . . . 53
Outputs . . . 53
Notes . . . 53
Related Links . . . 53
Get-PSAutoLabSetting . . . 54
Synopsis . . . 54
Syntax . . . 54
Description . . . 54
Examples . . . 54
Parameters . . . 55
Inputs . . . 55
Outputs . . . 55
Notes . . . 55
Related Links . . . 55
Invoke-RefreshHost . . . 56
Synopsis . . . 56
Syntax . . . 56
Description . . . 56
Examples . . . 56
Parameters . . . 56
Inputs . . . 58
Outputs . . . 58
Notes . . . 58
Related Links . . . 58
Invoke-RefreshLab . . . 59
Synopsis . . . 59
Syntax . . . 59
Description . . . 59
Examples . . . 59
Parameters . . . 59
Inputs . . . 60
Outputs . . . 61
Notes . . . 61

Related Links . . . 61
Invoke-RunLab . . . 62
Synopsis . . . 62
Syntax . . . 62
Description . . . 62
Examples . . . 62
Parameters . . . 62
Inputs . . . 63
Outputs . . . 63
Notes . . . 63
Related Links . . . 63
Invoke-SetupHost . . . 64
Synopsis . . . 64
Syntax . . . 64
Description . . . 64
Examples . . . 64
Parameters . . . 64
Inputs . . . 65
Outputs . . . 66
Notes . . . 66
Related Links . . . 66
Invoke-SetupLab . . . 67
Synopsis . . . 67
Syntax . . . 67
Description . . . 67
Examples . . . 67
Parameters . . . 67
Inputs . . . 69
Outputs . . . 69
Notes . . . 69
Related Links . . . 69
Invoke-ShutdownLab . . . 70
Synopsis . . . 70
Syntax . . . 70
Description . . . 70
Examples . . . 70
Parameters . . . 70
Inputs . . . 71
Outputs . . . 71
Notes . . . 71
Related Links . . . 71
Invoke-SnapshotLab . . . 72
Synopsis . . . 72

Syntax . . . 72
Description . . . 72
Examples . . . 72
Parameters . . . 72
Inputs . . . 73
Outputs . . . 74
Notes . . . 74
Related Links . . . 74
Invoke-UnattendLab . . . 75
Synopsis . . . 75
Syntax . . . 75
Description . . . 75
Examples . . . 75
Parameters . . . 76
Inputs . . . 77
Outputs . . . 77
Notes . . . 77
Related Links . . . 77
Invoke-ValidateLab . . . 78
Synopsis . . . 78
Syntax . . . 78
Description . . . 78
Examples . . . 78
Parameters . . . 78
Inputs . . . 79
Outputs . . . 79
Notes . . . 79
Related Links . . . 79
Invoke-WipeLab . . . 80
Synopsis . . . 80
Syntax . . . 80
Description . . . 80
Examples . . . 80
Parameters . . . 80
Inputs . . . 82
Outputs . . . 82
Notes . . . 82
Related Links . . . 82
Open-PSAutoLabHelp . . . 83
Synopsis . . . 83
Syntax . . . 83
Description . . . 83
Examples . . . 83

Parameters . . . 83
Inputs . . . 83
Outputs . . . 83
Notes . . . 84
Related Links . . . 84
Test-ISOImage . . . 85
Synopsis . . . 85
Syntax . . . 85
Description . . . 85
Examples . . . 85
Parameters . . . 85
Inputs . . . 86
Outputs . . . 86
Notes . . . 86
Related Links . . . 86
Test-LabDSCResource . . . 87
Synopsis . . . 87
Syntax . . . 87
Description . . . 87
Examples . . . 87
Parameters . . . 87
Inputs . . . 88
Outputs . . . 88
Notes . . . 88
Related Links . . . 88
Update-Lab . . . 89
Synopsis . . . 89
Syntax . . . 89
Description . . . 89
Examples . . . 89
Parameters . . . 89
Inputs . . . 90
Outputs . . . 90
Notes . . . 90
Related Links . . . 91

Introduction

This manual is a PDF document compiled from several reference files as well as help documentation for the PSAutoLab module's commands. The goal is to provide a single source for all documentation. Many of the source files contain internal cross-references. Best efforts have been made to port those links in this PDF file. External links should work as expected.

Note that the terms *Autolab* and *PSAutoLab* are used interchangeably but generally refer to the same thing. Technically, *PSAutoLab* is the name of the PowerShell module that manages the *Autolab* configuration which in turn is managed by the Lability module "under the hood".

PSAutoLab

Overview

This project serves as a set of "wrapper" commands that utilize the Lability module which is a terrific tool for creating a lab environment of Windows-based systems. The downside is that it is a difficult module for less experienced PowerShell users. The configurations and control commands for the Hyper-V virtual machines in this module are written in PowerShell using `Desired State Configuration (DSC)` and deployed via Lability commands. If you feel sufficiently skilled, you can skip using this project and use the Lability module on your own. Note that the Lability module is not owned or managed by Pluralsight. This project and all files are released under an MIT License - meaning you can copy and use as your own, modify, borrow, steal - whatever you want.

While this project is under the Pluralsight banner, it is offered AS-IS as a free tool with no official support from Pluralsight. Pluralsight makes no guarantees or warranties. This project is intended to be used for educational purposes only.

Beginning with module version 4.17.0, you can run Open-PSAutoLabHelp to view a local PDF version of the module's documentation.

Requirements

This module is designed and intended to be run on a **Windows 10** client that supports virtualization. Windows 10 Pro or Enterprise should be sufficient. It is assumed you will be installing this on a Windows 10 desktop running Windows PowerShell 5.1. This module will **not** work and is unsupported on Windows 10 Home or any Student edition. Although there are reports of the module working on Windows 10 Education. The module *might* run on Windows Server (2016 or 2019) platforms, but this capability has not been fully tested nor is it supported.

Using this in a nested virtual environment *may* work, but don't be surprised if there are problems, especially related to networking and NAT.

The host computer, where you are installing, must meet the following requirements:

- Windows PowerShell 5.1.
- A high-speed internet connection.
- Minimum 16GB of RAM (32GB is recommended).
- Minimum 100GB free disk space preferably on a fast SSD device or equivalent.
- An Intel i5 processor or equivalent. An i7 is recommended for the best performance.
- Windows PowerShell Remoting enabled.
- You should be logged in with a local or domain user account. The setup process may not work properly if using an O365 or Microsoft account to logon to Windows.

You must have administrator access and be able to update the TrustedHosts setting for PowerShell remoting. If you are in a corporate environment, these settings may be locked down or restricted. If this applies to you, this module may not work properly, if at all.

This module and configurations have NOT been tested running from PowerShell Core or PowerShell 7 and is not supported at this time.

Pester Requirement

The module uses a standard PowerShell tool called Pester to validate lab configurations. Without getting into technical details, if you are running the out-of-the-box version of Pester on Windows 10, you need to manually update Pester before attempting to install this module. In an elevated Windows PowerShell session run this command:

```
Get-Module Pester -ListAvailable
```

If the *only* result you get is for version `3.4.0`, then you must run:

```
Install-Module pester -RequiredVersion 4.10.1 -Force -SkipPublisherCheck
```

Re-run the `Get-Module` to verify version `4.10.1` is installed. If you have newer versions installed, that will have no effect on this module. Once you have verified Pester version 4.10.1 you can install the PSAutoLab module.

Installation

This module has been published to the PowerShell Gallery. It is recommended that you have at least version 2.2 of the `PowerShellGet` module which handles module installations.

Open an elevated PowerShell prompt and run:

```
Install-Module PSAutoLab -Force -SkipPublisherCheck
```

The installation should install required dependencies which is why you need the additional parameters.

If prompted, answer yes to update the nuget version and to install from an untrusted repository, unless you've already marked the PSGallery as trusted. If you have an old copy of this module from before Pluralsight took ownership, you will get an error. Manually remove the old module files and try again.

Do not download or use any of the release packages from this Github repository. You must install this module from the PowerShell Gallery.

See the Changelog for update details.

DO NOT run this module on any mission-critical or production system.

If you encounter issues with a basic installation and setup, you should read and use the detailed setup instructions.

You can verify the module with these commands:

```
PS C:\> Import-Module PSAutoLab -force
PS C:\> Get-Module PSAutoLab

ModuleType Version    Name           ExportedCommands
---------- -------    ----           ----------------
Script     4.18.0      PSAutoLab     {Enable-Internet, Get-LabSnapshot,...}
```

Your version number may differ.

Hyper-V

This module and its configurations should not conflict with any existing Hyper-V virtual machines or networking. But you should be aware that the module will create a new., internal Hyper-V switch called `LabNet`. This switch will use a NAT configuration called `LabNat`.

```
PS C:\> Get-NetNat LabNat


Name                             : LabNat
ExternalIPInterfaceAddressPrefix :
InternalIPInterfaceAddressPrefix : 192.168.3.0/24
IcmpQueryTimeout                 : 30
TcpEstablishedConnectionTimeout  : 1800
TcpTransientConnectionTimeout    : 120
TcpFilteringBehavior             : AddressDependentFiltering
UdpFilteringBehavior             : AddressDependentFiltering
UdpIdleSessionTimeout            : 120
UdpInboundRefresh                : False
Store                            : Local
Active                           : True
```

The `Instructions.md` file in each configuration folder should provide an indication of what VMs will be created. You can also check the `VMConfigurationData.psd1` file.

```
PS C:\Autolab\Configurations\MultiRole\> (Import-PowerShellDataFile .\VMConfigurationData.psd1).allnodes.Nodename
*
DC1
S1
Cli1
```

Current configurations will use these names for the virtual machine and computer name:

- DC1
- S1
- S2
- Cli1
- Cli2
- PullServer
- DOM1
- SRV1
- SRV2
- SRV3
- WIN10
- Win10Ent
- S12R2
- S12R2GUI

Nano Server images have been removed from configurations. These configurations were using the now deprecated version of Nano. Microsoft has changed direction with regards to Nano Server and none of the existing configurations use this new version.

Previous Versions

If you installed previous versions of this module (v3.x) and struggled, hopefully, this version will be an improvement. To avoid any other complications, it is STRONGLY recommended that you manually remove the old version which is most likely under `C:\Program Files\WindowsPowerShell\Modules\PSAutoLab`. You can run a command like:

```
Get-Module PSAutoLab -ListAvailable | Select-Object Path
```

To identify the module location. Use this information to delete the PSAutoLab folder.

The previous version was not installed using PowerShell's module cmdlets so it can't be updated or removed except manually.

Note for VMware Users

This project is designed to work with Hyper-V. If you are going to build a Host VM of Server 2016 or Windows 10, In the general settings for your VM, you must change the OS type to `Hyper-V(Unsupported)` or the Host Hyper-V will not work! This module and its configurations have **not** been tested for compatibility with VMware.

Aliases and Language

While this module follows proper naming conventions, the commands you will typically use employ aliases that use non-standard verbs such as Run-Lab. This is to avoid conflicts with commands in the Lability module and to maintain backward compatibility. You can use the aliases or the full function name. All references in this document use the aliases. You do not need to run any commands from the Lability module.

Setup Host

The first time you use this module, you will need to configure the local machine or host. Open an elevated PowerShell session and run:

```
Setup-Host
```

This will install and configure the Lability module and install the Hyper-V feature if it is missing. By default, all AutoLab files will be stored under `C:\AutoLab`, which the setup process will create. If you prefer to use a different drive, you can specify it during setup.

```
Setup-Host -DestinationPath D:\AutoLab
```

You will be prompted to reboot, which you should do especially if the setup had to add the Hyper-V feature. To verify your configuration open an elevated PowerShell session and run this command:

```
PS C:\> Get-PSAutoLabSetting


AutoLab                     : C:\Autolab
PSVersion                   : 5.1.19041.1
PSEdition                   : Desktop
OS                          : Microsoft Windows 10 Pro
FreeSpaceGB                 : 172.49
MemoryGB                    : 32
PctFreeMemory               : 44.66
Processor                   : Intel(R) Core(TM) i7-7700T CPU @ 2.90GHz
IsElevated                  : True
RemotingEnabled             : True
NetConnectionProfile        : Private
HyperV                      : 10.0.19041.1
PSAutoLab                   : {4.18.0, 4.17.0}
Lability                    : {0.19.1, 0.19.0, 0.18.0}
Pester                      : {4.10.1, 4.10.0, 4.9.0, 4.4.4...}
PowerShellGet               : 2.2.3
PSDesiredStateConfiguration : 1.1
```

Some of your values may be different. Please include this information when reporting any problems or issues.

Lab Summary

Once the host setup is complete, you can use the module's Get-LabSummary command to better understand what the lab configuration will setup. Run the command in the configuration folder.

```
PS C:\Autolab\Configurations\SingleServer-GUI-2016\> Get-LabSummary


   Computername: S1 VMName: S1

Lab                       IPAddress       MemGB Procs Role    Description
---                       ---------       ----- ----- ----    -----------
SingleServer-GUI-2016     192.168.3.75        4     1 RDP     Windows Server
                                                              2016 Standard
                                                              64bit English
                                                              Evaluation
```

The module includes a custom format file. You might also run the command like this:

```
PS C:\Autolab\Configurations\SingleServer-GUI-2016> Get-LabSummary |
Select-Object *


Computername : S1
VMName       : S1
InstallMedia : 2016_x64_Standard_EN_Eval
Description  : Windows Server 2016 Standard 64bit English Evaluation
Role         : RDP
IPAddress    : 192.168.3.75
MemoryGB     : 4
Processors   : 1
Lab          : SingleServer-GUI-2016
```

The `Computername` and `VMName` properties might differ if you are using an environmental prefix. The `Computername` is how the virtual machines refer to each themselves and each other. The `VMName` is how you reference them in Hyper-V.

Creating a Lab

Lab information is stored under the AutoLab Configurations folder, which is `C:\AutoLab\Configurations` by default. Open an elevated PowerShell prompt and change location to the desired configuration folder. View the `Instructions.md` and/or `README.md` files in the folder to learn more about the configuration. Where possible information about what course goes with a particular Pluralsight course will be indicated.

A Note on Pluralsight Labs

This module started several years ago and there are several Pluralsight courses that rely on configurations that may no longer exist. Configurations that were named as `Test` or `POC` were not assumed to be used in any courses. But that is turning out to not be the case. If you are trying to setup a lab for a specific course, and can't find the configuration the instructor calls for, please post an issue indicating the configuration you are looking for and the title of the Pluralsight course. Hopefully, there is an existing configuration you can use. Or the module can be updated with an appropriate lab configuration. In some cases, the course may assume a different password. All configurations use `P@ssw0rd` for all passwords.

The first time you set up a lab, Lability will download evaluation versions of required operating systems in ISO format. This may take some time depending on your Internet connection. These downloads only happen when the required ISO is not found locally. When you wipe and rebuild a lab it won't download files a second time.

Once the lab is created, you can use the PSAutoLab commands for managing it. If you have additional PowerShell experience, you can manage individual virtual machines using the Hyper-V manager or cmdlets.

It is assumed that you will only have one lab configuration created at a time.

Please be aware that all configurations were created for an `EN-US` culture and keyboard.

Manual Setup

Most, if not all, configurations should follow the same manual process. Run each command after the previous one has completed.

- Setup-Lab
- Run-Lab
- Enable-Internet

To verify that all virtual machines are properly configured you can run Validate-Lab. This will invoke a set of tests and keep looping until everything passes. Due to the nature of DSC and the complexity of some configurations, this could take up to 60 minutes. You can use `Ctrl+C` to break out of the testing loop at any time. You can manually run the test one time to see the current state of the configuration.

```
PS C:\Autolab\Configurations\SingleServer\> Invoke-Pester VMValidate.test.ps1
```

This can be useful for troubleshooting.

Unattended Setup

As an alternative, you can setup a lab environment with minimal prompting.

```
PS C:\Autolab\Configurations\SingleServer\> Unattend-Lab
```

Assuming you don't need to install a newer version of `nuget`, you can leave the setup alone. It will run all of the manual steps for you. Beginning in version `4.3.0` you also have the option to run the unattend process in a PowerShell background job.

```
PS C:\Autolab\Configurations\SingleServer\> Unattend-Lab -asjob
```

Use the PowerShell job cmdlets to manage.

Stopping a Lab

To stop the lab VMs, change to the configuration folder in an elevated Windows PowerShell session and run:

```
PS C:\Autolab\Configurations\SingleServer\> Shutdown-Lab
```

You can also use the Hyper-V manager or cmdlets to manually shut down virtual machines. If your lab contains a domain controller such as `DOM1` or `DC1`, that should be the last virtual machine to shut down.

Starting a Lab

The setup process will leave the virtual machines running. If you have stopped the lab and need to start it, change to the configuration folder in an elevated Windows PowerShell session and run:

```
PS C:\Autolab\Configurations\SingleServer\> Run-Lab
```

You can also use the Hyper-V manager or cmdlets to manually start virtual machines. If your lab contains a domain controller such as `DOM1` or `DC1`, that should be the first virtual machine to start up.

Lab Checkpoints

You can snapshot the entire lab very easily. Change to the configuration folder in an elevated Windows PowerShell session and run:

```
PS C:\Autolab\Configurations\SingleServer\> Snapshot-Lab
```

To quickly rebuild the labs from the checkpoint, run:

```
PS C:\Autolab\Configurations\SingleServer\> Refresh-Lab
```

Or you can use the Hyper-V cmdlets to create and manage VM snapshots.

To Remove a Lab

To destroy the lab completely, change to the configuration folder in an elevated Windows PowerShell session and run:

```
PS C:\Autolab\Configurations\SingleServer\> Wipe-Lab
```

This will remove the virtual machines and DSC configuration files. If you intend to rebuild the lab or another configuration, you can keep the `LabNat` virtual switch. This is the default behavior. If you want to remove everything you would need to run a command like this:

```
PS C:\Autolab\Configurations\SingleServer\> Wipe-Lab -force -removeswitch
```

Customizing a Lab

It is possible to customize a lab configuration by editing the `VMConfigurationData.psd1` file that is in each configuration folder. You must modify the file before creating the lab. For example, the configuration my use Server Core and you want the Desktop Experience on the server. Open the file in your scripting editor and scroll down to find the Node definitions.

```
@{
    NodeName                = 'DOM1'
    IPAddress               = '192.168.3.10'
    Role                    = @('DC', 'DHCP', 'ADCS')
    Lability_BootOrder      = 10
    Lability_BootDelay      = 60
    Lability_timeZone       = 'US Mountain Standard Time'
    Lability_Media          = '2016_x64_Standard_Core_EN_Eval'
    Lability_MinimumMemory  = 2GB
    Lability_ProcessorCount = 2
    CustomBootStrap         = @'
            # This must be set to handle larger .mof files
            Set-Item -path wsman:\localhost\maxenvelopesize -value 1000
'@
},

@{
    NodeName           = 'SRV1'
    IPAddress          = '192.168.3.50'
    Role               = @('DomainJoin')
    Lability_BootOrder = 20
    Lability_timeZone  = 'US Mountain Standard Time'
    Lability_Media     = '2016_x64_Standard_Core_EN_Eval'
},
```

You can edit the `Lability_Media` setting. Change the setting using one of these ID values.

```
Id                                    Arch Media Description
--                                    ---- ----- -----------
2019_x64_Standard_EN_Eval              x64   ISO Windows Server 2019 Standard 64bit English Evaluation with Desktop Experience
2019_x64_Standard_EN_Core_Eval         x64   ISO Windows Server 2019 Standard 64bit English Evaluation
2019_x64_Datacenter_EN_Eval            x64   ISO Windows Server 2019 Datacenter 64bit English Evaluation with Desktop Experience
2019_x64_Datacenter_EN_Core_Eval       x64   ISO Windows Server 2019 Datacenter Evaluation in Core mode
2016_x64_Standard_EN_Eval              x64   ISO Windows Server 2016 Standard 64bit English Evaluation
2016_x64_Standard_Core_EN_Eval         x64   ISO Windows Server 2016 Standard Core 64bit English Evaluation
2016_x64_Datacenter_EN_Eval            x64   ISO Windows Server 2016 Datacenter 64bit English Evaluation
2016_x64_Datacenter_Core_EN_Eval       x64   ISO Windows Server 2016 Datacenter Core 64bit English Evaluation
2016_x64_Standard_Nano_EN_Eval         x64   ISO Windows Server 2016 Standard Nano 64bit English Evaluation
2016_x64_Datacenter_Nano_EN_Eval       x64   ISO Windows Server 2016 Datacenter Nano 64bit English Evaluation
2012R2_x64_Standard_EN_Eval            x64   ISO Windows Server 2012 R2 Standard 64bit English Evaluation
2012R2_x64_Standard_EN_V5_Eval         x64   ISO Windows Server 2012 R2 Standard 64bit English Evaluation with WMF 5
2012R2_x64_Standard_EN_V5_1_Eval       x64   ISO Windows Server 2012 R2 Standard 64bit English Evaluation with WMF 5.1
2012R2_x64_Standard_Core_EN_Eval       x64   ISO Windows Server 2012 R2 Standard Core 64bit English Evaluation
2012R2_x64_Standard_Core_EN_V5_Eval    x64   ISO Windows Server 2012 R2 Standard Core 64bit English Evaluation with WMF 5
2012R2_x64_Standard_Core_EN_V5_1_Eval  x64   ISO Windows Server 2012 R2 Standard Core 64bit English Evaluation with WMF 5.1
2012R2_x64_Datacenter_EN_Eval          x64   ISO Windows Server 2012 R2 Datacenter 64bit English Evaluation
2012R2_x64_Datacenter_EN_V5_Eval       x64   ISO Windows Server 2012 R2 Datacenter 64bit English Evaluation with WMF 5
2012R2_x64_Datacenter_EN_V5_1_Eval     x64   ISO Windows Server 2012 R2 Datacenter 64bit English Evaluation with WMF 5.1
2012R2_x64_Datacenter_Core_EN_Eval     x64   ISO Windows Server 2012 R2 Datacenter Core 64bit English Evaluation
2012R2_x64_Datacenter_Core_EN_V5_Eval  x64   ISO Windows Server 2012 R2 Datacenter Core 64bit English Evaluation with WMF 5
2012R2_x64_Datacenter_Core_EN_V5_1_Eval x64   ISO Windows Server 2012 R2 Datacenter Core 64bit English Evaluation with WMF 5.1
WIN81_x64_Enterprise_EN_Eval           x64   ISO Windows 8.1 64bit Enterprise English Evaluation
WIN81_x64_Enterprise_EN_V5_Eval        x64   ISO Windows 8.1 64bit Enterprise English Evaluation with WMF 5
WIN81_x64_Enterprise_EN_V5_1_Eval      x64   ISO Windows 8.1 64bit Enterprise English Evaluation with WMF 5.1
WIN81_x86_Enterprise_EN_Eval           x86   ISO Windows 8.1 32bit Enterprise English Evaluation
WIN81_x86_Enterprise_EN_V5_Eval        x86   ISO Windows 8.1 32bit Enterprise English Evaluation with WMF 5
WIN81_x86_Enterprise_EN_V5_1_Eval      x86   ISO Windows 8.1 32bit Enterprise English Evaluation with WMF 5.1
WIN10_x64_Enterprise_20H2_EN_Eval      x64   ISO Windows 10 64bit Enterprise 2009 English Evaluation (20H2)
WIN10_x86_Enterprise_20H2_EN_Eval      x86   ISO Windows 10 32bit Enterprise 2009 English Evaluation
WIN10_x64_Enterprise_LTSC_EN_Eval      x64   ISO Windows 10 64bit Enterprise LTSC 2019 English Evaluation
WIN10_x86_Enterprise_LTSC_EN_Eval      x86   ISO Windows 10 32bit Enterprise LTSC 2019 English Evaluation
```

You can also make changes to values such as minimum memory and processor count. When you run `Unattend-Lab` or `Setup-Lab` you can use the `-UseLocalTimeZone` to set all virtual machines to use your time zone. You could make *minor* changes to the IP address such as changing the address from `192.168.3.50` to `192.168.3.60`. To change the entire subnet will require modifying the virtual switch and should not be attempted unless you are very proficient with PowerShell and Hyper-V.

Note that if you make changes, the validation test may fail unless you modify it. But you can always try to run the lab without validating it.

If you make a mistake or want to restore the original configurations, run the `Refresh-Host` command.

Windows 10 Remote Server Administration Tools (RSAT)

A number of lab configurations that include a Windows 10 client will also install RSAT. In the past, this has meant trying to install *all* RSAT features. This takes a long time and has caused validation issues. Beginning with v4.21.0 of the PSAutlab module, the RSAT configuration and testing will only use a subset of features.

- Rsat.ActiveDirectory.DS-LDS.Tools~~0.0.1.0
- Rsat.BitLocker.Recovery.Tools~~0.0.1.0
- Rsat.CertificateServices.Tools~~0.0.1.0
- Rsat.DHCP.Tools~~0.0.1.0'
- Rsat.Dns.Tools~~0.0.1.0'
- Rsat.FailoverCluster.Management.Tools~~0.0.1.0'
- Rsat.FileServices.Tools~~0.0.1.0'
- Rsat.GroupPolicy.Management.Tools~~0.0.1.0'
- Rsat.IPAM.Client.Tools~~0.0.1.0'
- Rsat.ServerManager.Tools~~0.0.1.0'

If you require any other tool, you will need to use `Add-WindowsCapability` in the Windows 10 client to add it.

This change has improved setup-performance and module stability.

Windows Updates

When you build a lab, you are creating Windows virtual machines based on evaluation software. You might still want to make sure the virtual machines are up to date with security patches and updates. You can use Update-Lab to invoke Windows update on all lab members. This can be a time-consuming process, so you have an option to run the updates as a background job. Just be sure not to close your PowerShell session before the jobs complete.

```
PS C:\Autolab\Configurations\PowerShellLab\> update-lab -AsJob

Id   Name        PSJobTypeName   State      HasMoreData   Location    Command
--   ----        -------------   -----      -----------   --------    -------
18   WUUpdate    RemoteJob       Running    True          DOM1        WUUpdate
21   WUUpdate    RemoteJob       Running    True          SRV1        WUUpdate
24   WUUpdate    RemoteJob       Running    True          SRV2        WUUpdate
27   WUUpdate    RemoteJob       Running    True          SRV3        WUUpdate
30   WUUpdate    RemoteJob       Running    True          WIN10       WUUpdate

PS C:\Autolab\Configurations\PowerShellLab> receive-job -id 27 -Keep
[11/22/2019 12:05:43] Found 5 updates to install on SRV3
[11/22/2019 12:25:13] Update process complete on SRV3
WARNING: SRV3 requires a reboot
```

Run the update process as a background job. Use the PowerShell job cmdlets to manage.

Updating PSAutoLab

As this module is updated over time, new configurations may be added, or bugs fixed in existing configurations. There may also be new Lability updates. Use PowerShell to check for new versions:

```
Find-Module PSAutoLab
```

And update:

```
Update-Module PSAutoLab -Force
```

If you update, it is recommended that you update the AutoLab configuration.

```
Refresh-Host
```

This will update the Lability and Pester modules if required and copy all-new configuration files to your AutoLab\Configurations folder. It will NOT delete any files.

Removing PSAutoLab

If you want to completely remove the PSAutoLab module, first use `Wipe-Lab` to remove any existing lab configurations including the Hyper-V switch. Run this command to uninstall the module and its dependencies

```
Uninstall-Module PSAutoLab,Lability
```

You may need to manually delete the `C:\Autolab` folder. If you want to remove the NAT configuration"

```
Remove-NetNat LabNat
```

If you want to remove Hyper-V you can use the Control Panel to manually remove the optional feature. Or you can try using PowerShell.

```
Get-WindowsOptionalFeature -FeatureName *Hyper* -online |
Disable-WindowsOptionalFeature -Online
```

You will almost certainly need to reboot to complete the removal process.

Pester

If you are running Pester v5.x you need to be running at least version 4.11.0 of this module.

The validation tests for each configuration are written for the Pester module. This is a widely adopted testing tool. In June of 2020 version 5 was released. This version of Pester introduced several breaking changes to how tests are written. The tests in this module are **incompatible** with Pester 5.0 and will need to be re-written. As an interim step, this module will test for Pester v 4.10.1. If you don't have that version it will be installed when you run `Setup-Host`. Or if you've already setup Autolab you can run Refresh-Host. If you have Pester 5.x, it will not be uninstalled, but it will be removed from the current PowerShell session.

Troubleshooting

Package Provider or Module Installation

If you try to install a module or update the nuget provider, you might see warnings like these:

```
WARNING: Unable to download from URI 'https://go.microsoft.com/fwlink/?LinkID=627338&clcid=0x409' to ''.
WARNING: Unable to download the list of available providers. Check your internet connection.
PackageManagement\Install-PackageProvider : No match was found for the specified search criteria for the provider 'NuGet'. The package provider requires
'PackageManagement' and
'Provider' tags.
```

The first thing to check is to make sure you are using valid TLS settings. You can try running this command in PowerShell:

```
[Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls12
```

Beginning with v4.17.0 of this module, this change is made when the module is imported. It will only last for as long as your PowerShell session is running.

You could also modify the registry in an elevated PowerShell session for a more permanent solution.

```
Set-ItemProperty -Path 'HKLM:\SOFTWARE\Microsoft\.NetFramework\v4.0.30319' -Name 'SchUseStrongCrypto' -Value 1
```

Modifying the registry will require a reboot for the changes to take effect.

If the problem is the nuget provider, after making the TLS changes try:

```
Install-PackageProvider nuget -force -forcebootstrap
```

You might also need to update the `PackageManagement` and/or `PowerShellGet` modules.

```
Update-Module powershellget,packagemanagement -force
```

Autolab Configurations

The commands and configurations in this module are not foolproof. During testing a lab configuration will run quickly and without error on one Windows 10 desktop but fail or take much longer on a different Windows 10 desktop. Most setups should be complete in under an hour. If validation is failing, manually run the validation test in the configuration folder.

```
PS C:\Autolab\Configurations\SingleServer\> Invoke-Pester VMValidate.test.ps1
```

Take note of which virtual machines are generating errors. Verify the virtual machine is running in Hyper-V. On occasion for reasons still undetermined, sometimes a virtual machine will shutdown and not reboot. This often happens with the client nodes of the lab configuration. Verify that all virtual machines are running and manually start those that have stopped using the Hyper-V manager or cmdlets.

Sometimes even if the virtual machine is running, manually shutting it down and restarting it can resolve the problem. Remember to wait at least 5 minutes before manually running the validation test again when restarting any virtual machine.

As a last resort, manually break out of any testing loop, wipe the lab and start all-over.

If you *still* are having problems, wipe the lab and try a different configuration. This will help determine if the problem is with the configuration or a larger compatibility problem.

At this point, you can open an issue in this repository. Open an elevated PowerShell prompt and run Get-PSAutoLabSetting which will provide useful information. Copy and paste the results into a new issue along with any error messages you are seeing.

Known Issues

I get an error when importing the module

Starting with version 4.12.0 of this module, you might see this error when you import the module.

```
Import-Module : Assertion operator name 'Be' has been added multiple times.
```

This is most likely due to a conflict in Pester versions. The solution is to remove the Pester module from your current session.

```
Get-Module Pester | Remove-Module
```

Then import this module again.

I get an error trying to update Lability

If you try to run `Refresh-Host` you might see an error about a certificate mismatch. Between v0.18.0 and v0.19.0 the Lability module changed code signing certificates. If you encounter this problem, run

```
Refresh-Host -SkipPublisherCheck
```

Multiple DSC Resources

Due to what is probably a bug in the current implementation of Desired State Configuration in Windows, if you have multiple versions of the same resource, a previous version might be used instead of the required one. You might especially see this with the `xNetworking` module and the `xIPAddress` resource. If you have any version older than 5.7.0.0 you might encounter problems. Run this command to see what you have installed:

```
Get-DSCResource xIPAddress
```

If you have older versions of the module, uninstall them if you can.

```
Uninstall-Module xNetworking -RequiredVersion 3.0.0.0
```

Beginning with version 4.20.0, you might see the same type of errors with the xDHCP resource. If you encounter errors like `Invalid MOF definition for node 'DC1': Exception calling "ValidateInstanceText" with "1" argument(s): "Undefined property IsSingleInstance` you might have an older version of a DSCResource module installed.

Run `Get-Module xdhcpserver -list` and remove anything older than version 3.0.0.

```
Uninstall-Module xdhcpserver -RequiredVersion 2.0.0.0
```

It is recommended that you restart your PowerShell session and try the lab setup again.

Acknowledgments

This module is a continuation of the work done by Jason Helmick and Melissa (Missy) Januszko, whose efforts are greatly appreciated. Beginning with v4.0.0, this module is unrelated to any projects Jason or Missy may be developing under similar names.

We also appreciate all of the work that has gone into the Lability module. The Lability and PSAutoLab modules are completely separate and independently maintained.

Road Map

These are some of the items that are being considered for future updates:

- While Lability currently is for Windows only, it would be nice to deploy a Linux VM.
- Offer an easy way to customize a lab configuration such as node names and operating systems.

A complete list of enhancements (and bugs) can be found in the Issues section of this module's GitHub repository. Feel free to make a suggestion.

Detailed Setup Instructions

Please refer to this document to assist in installing and setting up the `PSAutolab` module on your computer. Run all commands from an **elevated** Windows PowerShell session. In other words, *run Windows PowerShell as administrator*. You will know you are elevated if you see the word `Administrator` in the title bar of the PowerShell window. **Do NOT run this module in PowerShell 7**. It is assumed you are running this on Windows 10 Professional or Enterprise editions, or the Windows 11 equivalents.

It is also assumed that you have administrator rights to your computer and can make changes. If your computer is controlled by Group Policy, you may encounter problems. You should also be logged in with a **local** or domain user account. The setup process may not work properly if using an O365 or Microsoft account to logon to Windows.

It is *possible* to run this module with nested virtualization inside a Windows 10 Hyper-V virtual machine but it is **not** recommended. Some networking features may not work properly and overall performance will likely be reduced.

Pre-Check

You can run these commands to verify your computer meets the minimum requirements. Run all PowerShell commands in an elevated session. If you can't even open a PowerShell prompt, this module definitely won't work on your computer.

Operating System and Memory

```
PS C:\> Get-CimInstance -ClassName Win32_OperatingSystem |
Select-Object -property Caption,
@{Name="MemoryGB";Expression={$_.TotalVisibleMemorySize/1mb -as [int]}}

Caption                  MemoryGB
-------                  -----
Microsoft Windows 10 Pro    32
```

If the Caption shows anything other than Pro or Enterprise this module may not work. Although it appears that Windows 10 Education might be supported.

The memory size should be at least 12GB. 16GB or greater is recommended. If the number is less than 12, **STOP**. It is unlikely you have enough installed memory. Depending on the configuration you want to run, it *might* be possible to proceed with less memory. Open an Issue and ask for guidance indicating your memory settings from this command:

```
PS C:\> Get-CimInstance Win32_OperatingSystem |
Select-Object -property FreePhysicalMemory,TotalVisibleMemorySize

FreePhysicalMemory TotalVisibleMemorySize
------------------ ----------------------
          14357500               33442716
```

Also indicate what lab configuration you are hoping to run.

PowerShell Remoting

The module relies on Windows PowerShell remoting which should be enabled **before** installing and using this module.

```
PS C:\> test-wsman

wsmid           : http://schemas.dmtf.org/wbem/wsman/identity/1/wsmanidentity.xsd
ProtocolVersion : http://schemas.dmtf.org/wbem/wsman/1/wsman.xsd
ProductVendor   : Microsoft Corporation
ProductVersion  : OS: 0.0.0 SP: 0.0 Stack: 3.0
```

This is what you should see as a result. Any errors mean that PowerShell remoting is disabled. Enable it from your **elevated** PowerShell session. This will fail if your only network connection is over a public network.

```
Enable-PSRemoting -force
```

If this fails, **STOP**. Do not proceed with this module until this is working and `Test-WSMan` gives you a result. If you are running as Administrator and this command fails it is most likely because the related settings are controlled by a Group Policy or your network is public. Run `Get-NetConnectionProfile` and look at the NetworkCategory for the `LabNet` connection. If must be `Private` or `DomainAuthenticated`.

Disk Space

The module requires a lot of disk space for the virtual machines, snapshots and ISO files. Run this command to see how much free space you have.

```
PS C:\> Get-Volume

Drive  SizeGB  FreeGB PercentFree HealthStatus
-----  ------  ------ ----------- ------------
D         477     183       38.41 Healthy
C         237      87       36.71 Healthy
```

You should have close to 100GB of free space on a fixed hard drive such as C or D. The module will setup an Autolab folder on drive C: by default, although you can specify an alternate drive. This module has not been tested running from an externally connected drive.

Virtualization

The module requires the Hyper-V feature on Windows 10. Please refer to the documentation for your computer hardware to determine if it supports virtualization. You may need to configure settings in your BIOS. You don't need to manually enable the Hyper-V feature now, although you are welcome to if you want to verify it is available.

Pester

The module uses a standard PowerShell tool called Pester to validate lab configurations. Without getting into technical details, if you are running the out-of-the-box version of Pester on Windows 10, **you need to manually update Pester** before attempting to install this module. In an elevated Windows PowerShell session run this command:

```
Get-Module Pester -ListAvailable
```

If the *only* result you get is for version `3.4.0`, then you must run:

```
Install-Module pester -RequiredVersion 4.10.1 -Force -SkipPublisherCheck
```

Re-run the `Get-Module` to verify version `4.10.1` is installed. If you have newer versions installed, that will have no effect on this module. Once you have verified Pester version 4.10.1 you can install the PSAutolab module.

Installation and Configuration

Install the Module

If you meet the requirements, you are ready to download and install this module. **Do not download anything from this GitHub repository.** In your PowerShell session run this command:

```
Install-Module PSAutolab -force -SkipPublisherCheck
```

You may be prompted to update to a newer version of `nuget`. Answer "yes". You might also be prompted about installing from an untrusted source. Again, you will need to say "yes". After installation you can verify using `Get-Module`.

```
PS C:\> Get-Module PSAutoLab -list

    Directory: C:\Program Files\WindowsPowerShell\Modules


ModuleType Version    Name                        ExportedCommands
---------- -------    ----                        ----------------
Script     4.19.0      PSAutoLab                  {Enable-Internet, Invoke-RefreshLab, Invoke-Run...
```

You may see a newer version number than what is indicated here. The `README` file in the GitHub repository indicates the current version in the PowerShell Gallery.

Setup the Host

There is a one-time step to setup your computer for the AutoLab environment. In your elevated PowerShell session run this command:

```
Setup-Host
```

This command will create a directory structure for the module and all of its files. The default is `C:\Autolab` which you should be able to accept. If you are low on space or want to use an alternate drive, then you can specify an alternative top-level path.

```
Setup-Host -DestinationPath D:\Autolab
```

If you select a drive other than C:\ it is recommended you still use the `Autolab` folder name. The setup process will install additional modules and files. If necessary, it will enable the Hyper-V feature. If Hyper-V is enabled during the setup, please reboot your computer before proceeding.

To verify your configuration, run `Get-PSAutolabSetting`.

```
PS C:\> Get-PSAutoLabSetting

AutoLab                     : C:\Autolab
PSVersion                   : 5.1.18362.752
PSEdition                   : Desktop
OS                          : Microsoft Windows 10 Pro
FreeSpaceGB                 : 365.45
MemoryGB                    : 32
PctFreeMemory               : 59.71
Processor                   : Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz
IsElevated                  : True
RemotingEnabled             : True
NetConnectionProfile        : Private
HyperV                      : 10.0.18362.1
PSAutolab                   : 4.19.0
Lability                    : {0.19.1, 0.19.0, 0.18.0}
Pester                      : {5.0.2, 4.10.1, 4.9.0, 4.8.1...}
PowerShellGet               : 2.2.4.1
PSDesiredStateConfiguration : 1.1
```

If Hyper-V is not installed you will see errors. Any errors indicate a problem with your setup. Please post this information when reporting an issue.

Setup a Configuration Unattended

In an elevated PowerShell session, **change directory** to the configuration folder that you want to use.

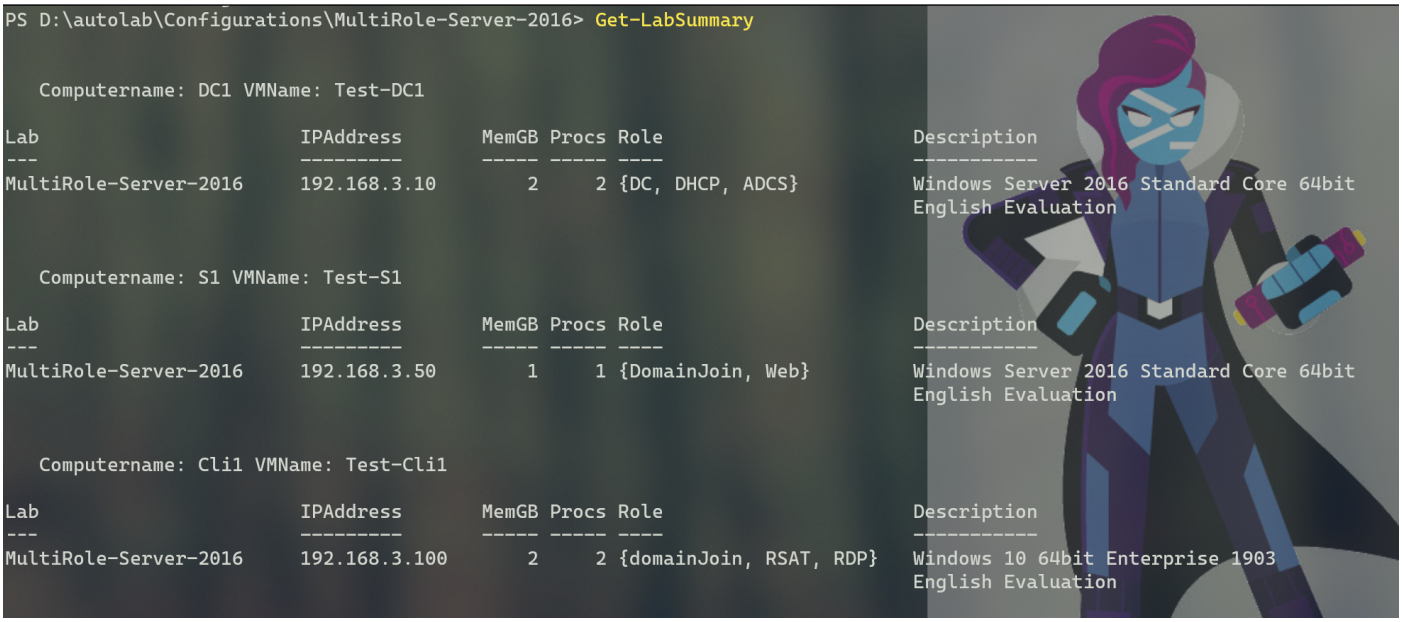
```
PS C:\> cd C:\Autolab\Configurations\SingleServer-GUI-2016
PS C:\Autolab\Configurations\SingleServer-GUI-2016\>
```

You can look at the `instructions.md` file in the folder to get more information about the configuration.

```
PS C:\Autolab\Configurations\SingleServer-GUI-2016\> get-content .\Instructions.md
```

Or open the file with Notepad.

Another option is to use the `Get-LabSummary` command. This will show you what computers will be created and how they will be created. The Computername will also be the virtual machine name.



You can run `Unattend-Lab` for a completely hands-free experience.

```
PS C:\Autolab\Configurations\SingleServer-GUI-2016\> unattend-lab
```

The very first time you setup a lab, the command will download ISO images of evaluation software from Microsoft. These files will be at least 4GB in size. If you are setting up a domain-based configuration, this means you will be downloading ISO images for Windows Server and Windows 10. This download only happens once.

Note that during the validation phase you may see errors. This is to be expected until all of the configurations merge. You can press `Ctrl+C` to break out of the testing. The virtual machines will continue to prepare themselves. Later, you can manually validate the lab:

```
PS C:\Autolab\Configurations\SingleServer-GUI-2016\> Invoke-Pester .\vmvalidate.test.ps1
```

Manual Configuration Setup

If you encounter errors running an unattended setup, you should step through the process manually to identify where exactly an error is occurring. Make sure you are in an elevated PowerShell session and you have *changed location to the configuration folder*. If you have tried to setup the lab before run `Wipe-Lab` to remove previous set-up files. Then run each of these commands individually:

- `Setup-Lab`
- `Enable-Internet`
- `Run-Lab`

Errors that affect setup should happen in one of these steps. If so, open an issue with the configuration name, the command you were working on and the error message. Also include the output from `Get-PSAutolabSetting`.

After about 10 minutes, you can run the validation command:

```
Validate-Lab
```

The `Write-Progress` display will provide feedback on the process.

Run `Validate-Lab -Verbose` to get details on the process

Beginning with v4.21.0, the validation command will restart virtual machines that have stopped and restart-virtual machines that appear to be failing. You will see this as warning messages. Validation will abort after 65 minutes if it hasn't completed. At which point you can manually test to see if the configuration has converged.

```
Invoke-Pester .\vmvalidate.test.ps1
```

Depending on the error, you might simply ignore it or manually attempt to resolve it. Often the best approach is to run `Wipe-Lab -force` and start all over.

Help

All of the commands in this module have help and examples. You are also encouraged to read the about help topic.

```
help about_PSAutoLab
```

Using the Environment Prefix

In the `VMConfigurationData.psd1` file for each lab, you will see a commented out section for an environment prefix value. This value exists for special situations where you might have a virtual machine naming collision or want to be able to identify the virtual machines that belong to the AutoLab module. In a normal setup and for almost all users, the Hyper-V virtual machine name will be the same as the hostname (computer name) in the VM guest. If the lab creates a guest with a computer name of `S1`, the Hyper-V virtual machine will also be called `S1`. If you enable the prefix setting, the Hyper-V virtual machine name will use the prefix, **but the guest computer name will not.** For example, if you enable the default prefix (you can change it to anything you'd like), you will create a Hyper-V virtual machine with a VMName of `Autolab-S1` but the actual computer name will still be `S1`. The validation tests will reference the guest computer name, not the Hyper-V virtual machine name.

If you must use this feature, open the `VMConfigurationData.psd1` file in a text or code editor. Scroll down to the `NonNodeData` section.

```
NonNodeData = @{
    Lability = @{

        # You can uncomment this line to add a prefix to the virtual machine name.
        # It will not change the guest computer name
        # See https://github.com/pluralsight/PS-AutoLab-Env/blob/master/Detailed-Setup-Instructions.md
        # for more information.

        #EnvironmentPrefix = 'AutoLab-'
```

Remove the # character before `EnvironmentPrefix`. If you want to change the value from `Autolab-` to something else go ahead. The prefix will be inserted before the computer name to create the virtual machine name.

This setting should only be used in special situations as it can be confusing. While every effort has been made to ensure compatibility with commands in this module, there is no guarantee of 100% success. Also note that any changes you make to the configuration files could be overwritten in future updates or when you run `Refresh-Host`.

Troubleshooting Tips

Occasionally, things can go wrong for no apparent reason. If you ran through the manual steps to setup a lab but the validation tests are still failing, you may need to stop and restart the virtual machine that is causing problems. For example, *sometimes* the SRV2 member in the `PowerShellLab` configuration simply won't pass validation, often because it can't be connected to. The best solution is to shut down the virtual machine in either the Hyper-V Manager or from PowerShell.

```
Stop-VM srv2 -force
```

Then start it back up.

```
Start-VM srv2
```

Wait about 5 minutes and then test again.

Another good test is to try an setup another simple configuration. Run `Wipe-Lab -force` from the configuration folder, then change to one of the simple configurations like `SingleServer` or `Windows10` and see if that installs. This will help determine if the problem is isolated to a specific lab configuration or a larger issue with your environment.

Manually Apply the Configuration

As a last resort, you can try to manually re-apply a configuration to a virtual machine. If you run `Invoke-Pester .\vmvalidate.test.ps1` you be able to discover what virtual machine is failing. For the sake of demonstration let's say it is a virtual machine called WIN10. To re-apply the configuration, you need to create a remoting session to it.

First, you need the lab password to create a credential object.

```
$data = Import-PowerShellDataFile .\VMConfigurationData.psd1
$pass = ConvertTo-SecureString -AsPlainText -String $data.allnodes.labpassword -force
$cred = new-object PSCredential -ArgumentList administrator, $pass
```

Note that if the lab has a mix of domain and workgroup machines, you will need to look at the VMConfigurationData file to discover the correct password. Almost always they will be same.

Next, create a CIMSession to the virtual machine.

```
$cim = New-CimSession -ComputerName win10 -Credential $cred
```

Now, you can apply the DSC configuration. You need run this from the lab configuration folder.

```
Start-DscConfiguration -CimSession $cim -Path . -wait -Force -verbose
```

You will be able to watch the process. Wait a minute or two for any background processing to finish and then test:

```
Test-DscConfiguration -CimSession $cim -Detailed
```

If all goes well, you should see something like this:

```
PSComputerName  ResourcesInDesiredState        ResourcesNotInDesiredState     InDesiredState
--------------  -----------------------        --------------------------     --------------
win10           {[Registry]TLS, [xIPAddress...                                True
```

If there are resources not in the desired state, wait 10 minutes and test again. If there are still failures, wipe the lab and start all over. At which point, *if you are still having failures*, file an issue and we'll have to determine what is happening.

Getting Help

If encounter problems getting any of this to work, you are welcome to post an Issue. If you get the module installed, please include the results of `Get-PSAutolabSetting`. If your problem is meeting one of the requirements, we will do our best to help. Although if your computer is locked down or otherwise controlled by corporate policies there may not be much that we can do.

Updating From PSAutoLab v3.x

At this point in time, this information should be irrelevant but will be retained just in case.

If you were running older versions of PSAutoLab, most likely v3.x, you might have encountered problems. It is hoped that the updates to 4.x will resolve most if not all of those problems. The plan going forward is to pay closer attention to issues and update the module as needed. This will be easier now that the module is deployed through the PowerShell Gallery.

The terms `AutoLab` and `PSAutoLab` are used interchangeably. PSAutoLab is technically the PowerShell module that manages your AutoLab configuration.

Before You Upgrade

The new module will be installed and updated from the PowerShell Gallery. To avoid conflicts, you should clean up the previous setup before installing the new version. The recommended procedure is to wipe everything and start fresh.

All existing issues from the previous version have been closed as the previous code-base is deprecated.

Wipe Labs

If AutoLab is is in use now, you can wait until you are finished with the lab configuration. Otherwise, use the `Wipe-Lab` command in any configuration folder that has MOF files with virtual machines. On the last configuration you wipe, you can answer yes to remove the LabNet switch.

Remove LabNet

If you didn't remove the NAT switch, you should manually remove it.

```
Remove-VMSwitch LabNet
```

You might also run `Get-VMSwitch` to discover the actual name if it varies from this documentation.

To be on the safe side, you should remove the NAT network configuration if it still exists. If this command gives you a result:

```
Get-NetNat LabNat
```

Then you can run:

```
Remove-NetNat LabNat
```

Remove Module

The previous version was manually copied to your module folder, `C:\Program Files\WindowsPowerShell\Modules`. You can always find the install location with a command like this:

```
Get-Module PSAutoLab -ListAvailable | Select-Object Path
```

Delete AutoLab Folder

Delete your AutoLab folder and all sub-folders which should be `C:\AutoLab` if you accepted the defaults during installation. This will delete all of the ISO files which means you'll need to re-download them when you build a new configuration. But that is OK because the current version of the module contains the correct links to all the relevant evaluation ISO files.

Update and Reboot

It is not necessary to remove Hyper-V. But it is recommended that you install all pending Windows updates and reboot your computer.

Verify

After reboot, open an elevated Windows PowerShell prompt. Type this command to verify you have removed the previous module:

```
Get-Module PSAutoLab -listavailable
```

Install

Assuming you are clean, you can now install the new version.

```
Install-Module PSAutoLab -repository PSGallery
```

Next, setup your local host:

```
Setup-Host
```

If Hyper-V had to be installed, then you should definitely reboot.

Setup a Configuration

From this point, you should be set with the new version. Read the about help topic to learn more.

```
help about_PSAutoLab
```

Or refer to the GitHub repository README file.

Usage FAQ

These are some common questions you might have about this module or errors that you might encounter. Although most of this document is retained merely for archival reference purposes. If you haven't already done so, you should read the README file. And don't forget the About_PSAutolab file.

- **I get an error trying to update Lability**

If you try to run `Refresh-Host` you might see an error about a certificate mismatch. Between v0.18.0 and v0.19.0 the Lability module changed code signing certificates. If you encounter this problem, run `Refresh-Host -SkipPublisherCheck`.

- **I get an error about trying to modify TrustedHosts**

The module commands must be able to use PowerShell remoting to configure and test the virtual machines within a configuration. Because there is no Kerberos authentication between the local host and the virtual machines, you need to configure TrustedHosts. If TrustedHosts can't be configured, you will likely encounter errors. You should make sure remoting is enabled on the localhost. Run this command.

```
Test-WSMan
```

If you get errors, you may need to enable PowerShell remoting.

```
Enable-PSRemoting
```

Ensure that you are running an elevated PowerShell session (Run as Administrator). **If your TrustedHosts configuration is managed by Group Policy, it is unlikely you will be able to use this module.**

- **I get an error about my network connection type being set to Public.**

You might see this error message:

```
Set-WSManQuickConfig : <f:WSManFault xmlns:f="http://schemas.microsoft.com/wbem/wsman/1/wsmanfault" Code="2150859113"
Machine="localhost"><f:Message><f:ProviderFault provider="Config provider" path="%systemroot%\system32\WsmSvc.dll"><f:WSManFault
xmlns:f="http://schemas.microsoft.com/wbem/wsman/1/wsmanfault" Code="2150859113" Machine="tablet"><f:Message>WinRM firewall exception will not work since
one of the network connection types on this machine is set to Public. Change the network connection type to either Domain or Private and try again.
</f:Message></f:WSManFault></f:ProviderFault></f:Message></f:WSManFault>
At line:116 char:17
+                 Set-WSManQuickConfig -force
+                 ~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : InvalidOperation: (:) [Set-WSManQuickConfig], InvalidOperationException
    + FullyQualifiedErrorId : WsManError,Microsoft.WSMan.Management.SetWSManQuickConfigCommand
```

To solve this you can try to modify the connection profile.

```
# Find connections with a NetworkCategory set to Public
Get-NetConnectionProfile

# For each connection, change to Private or Domain
Set-NetConnectionProfile -InterfaceIndex 3 -NetworkCategory Private
```

- **Enable-Internet fails on New-NetNat**

You might get an error like "The parameter is incorrect."

```
New-NetNat : The parameter is incorrect.
At C:\Lability\Configurations\POC-StandAlone-Server-GUI\Enable-Internet.ps1:50 char:9
+         New-NetNat -Name $NatName -InternalIPInterfaceAddressPrefix $ ...
+         ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
+ CategoryInfo          : InvalidArgument: (MSFT_NetNat:root/StandardCimv2/MSFT_NetNat) [New-NetNat], CimException
+ FullyQualifiedErrorId : Windows System Error 87,New-NetNat
```

Currently, Hyper-V only supports a single NAT network, you can read more about this here: https://blogs.technet.microsoft.com/virtualization/2016/05/25/windows-nat-winnat-capabilities-and-limitations/.

Likely, if you receive the error above, you already have a NAT network created. For example, `Docker for Windows` creates a DockerNAT virtual switch and NAT network. You can check if this is the case with the `Get-NetNat` PowerShell cmdlet. If you get back a NAT network object, then you won't be able to create another one for your lab. The solution is to coordinate a single NAT network so it covers all of your NAT networking needs.

- That likely means creating a larger NAT subnet that covers the IP ranges of all of your networks.
- Which also means coordinating IP ranges across apps so they can fall under a single NAT subnet.
- The NAT subnet cannot overlap with the external network that the host is attached to. If a host is attached to 192.168.0.0/24, you can't use 192.168.0.0/16 as a NAT network.

Refer to this article for help on creating NAT networks: https://msdn.microsoft.com/en-us/virtualization/hyperv_on_windows/user_guide/setup_nat_network

Run this command to list NAT networks, take note of the IP range and subnet

```
Get-NetNat
```

If you have an existing that is conflicting, *and that no longer need*, remove it . Here's an example removing an existing NAT network called DockerNAT

```
Remove-NetNat DockerNAT
```

You can then create a NAT network using the corresponding subnet.

```
New-NetNat -Name DockerAndLabilityNAT -InternalIPInterfaceAddressPrefix "192.168.3.0/24"
```

Docker for Windows network settings can be updated from the windows tray icon. Lab network changes require modifying the module's source code. If you are running Docker for Windows *and* the PSAutoLab module, please post an issue on GitHub.

- **How can I change a virtual machine's timezone?**

First, find your desired timezone using one of these techniques:

```
# Filter all timezones, take the Id property from the desired timezone:
[System.TimeZoneInfo]::GetSystemTimeZones()
[System.TimeZoneInfo]::GetSystemTimeZones().Where({$_.Id -like '*Eastern*'})

# Get your current timezone:
(Get-TimeZone).Id
```

Next, open the lab's `VMConfigurationData.psd1` in your script editor and change `Lability_timeZone` per Node.

```
@{
            NodeName                = 'Win10Ent'
            IPAddress               = '192.168.3.101'
            Role                    = @('RSAT','RDP')
            Lability_ProcessorCount = 2
            Lability_MinimumMemory  = 2GB
            Lability_Media          = 'WIN10_x64_Enterprise_EN_Eval'
            Lability_BootOrder      = 20
            Lability_timeZone       = 'Central Standard Time' #[System.TimeZoneInfo]::GetSystemTimeZones()
            Lability_Resource       = @()
        }
```

Or, when you run `Setup-Lab` or `Unattend-Lab` you can use the `UseLocalTimeZone` parameter to set the time zone for all lab members to use the same time zone as the local host.

- **How can I avoid issues when changing VM names?**

Use `Wipe-Lab` before changing names (i.e `NodeName` or `EnvironmentPrefix`), otherwise `Wipe-Lab` won't work and you'll have to manually cleanup previously created VMs.

- **How can I manually clean up a lab?**

Normally, when you run `Wipe-Lab` that should handle everything for you. But if there is a problem you can take these manual steps.

- Open the Hyper-V manager and manually shutdown or turn off the virtual machines in your lab configuration.
- In the Hyper-V manager, manually select each virtual machine and delete it.
- Open Windows Explorer or a PowerShell prompt and change to the configuration directory.
- Manually delete any MOF files.
- Change to C:\Autolab\VMVirtualDisks (or the drive where you have Autolab configured).
- Manually delete any files that are named with virtual machines from your configuration.

Authoring a Custom Configuration

The expectation is that one of the included configurations will meet your needs or has been specified by a Pluralsight author.However, are free to modify or create your own configuration. This process assumes you have experience with writing Desired State Configuration (DSC) scripts, including the use of configuration data files (*.psd1) and Pester. Because configurations might be updated in future versions of the PSAutoLab module, you are encouraged to create a new configuration and not edit existing files.

Find a configuration that is close to your needs and copy it to a new folder under `Autolab\Configurations`. Technically, you can put the configuration folder anywhere but it is easier if all of your configurations are in one location.

Once the files have been copied, use your script editor to modify the files. Don't forget to update the Pester test. You should keep the same file names.

Specify a Different Operating System

First, find the node information in the VMConfigurationData.psd1 file.

```
@{
    NodeName           = 'S1'
    IPAddress          = '192.168.3.50'
    #Role = 'DomainJoin' # example of multiple roles @('DomainJoin', 'Web')
    Role               = @('DomainJoin', 'Web')
    Lability_BootOrder = 20
    Lability_timeZone  = 'US Mountain Standard Time' #[System.TimeZoneInfo]::GetSystemTimeZones()
    Lability_Media     = '2019_x64_Standard_EN_Core_Eval'
}
```

You will need to change the `Lability_Media` section. At a PowerShell prompt, run `Get-LabMedia`. Copy the appropriate ID and replace the `Lability_Media` value. The first time you build the configuration with new media, the corresponding ISO file will be downloaded.

I'm getting an error when my VM is booting for the first time

You might see this error message.

```
Full error: *Windows could not parse or process the unattend answer file [C:\Windows\system32\sysprep\unattend.xml] for pass [specialize]. The answer
file is invalid.*
```

Make sure that the configuration changes you've made are valid. Specifically, this error is known to be caused by an invalid NodeName. NodeName on Windows must match the rules for a Windows Computer Name, notably a 15 character maximum. See details here: https://support.microsoft.com/en-us/kb/909264

I’m Still Stuck

For all other questions, comments or problems, please post an Issue in this repository.

Module Commands

Most of the commands in the module use a domain-specific language (DSL), or a set of aliases, such as `Run-Lab`, to reference the actual commands (e.g. `Invoke-Runlab`). This DSL is intended to make it easier for beginner PowerShell users to use this module.

```
Get-Command -module PSAutolab
```

Experienced users are welcome to use the proper command names if they wish.

Enable-Internet

Synopsis

Configure lab configuration with Internet access.

Syntax

```
Enable-Internet [[-Path] <String>] [-WhatIf] [-Confirm] [<CommonParameters>]
```

Description

This function will enable Internet access for the virtual machines in the current lab configuration using a NAT interface. This command should be run from the configuration directory after the virtual machines have been set up.

Examples

Example 1

```
PS C:\Autolab\Configurations\Windows10> Enable-Internet
```

Parameters

-Confirm

Prompts you for confirmation before running the cmdlet.

```
Type: SwitchParameter
Parameter Sets: (All)
Aliases: cf

Required: False
Position: Named
Default value: None
Accept pipeline input: False
Accept wildcard characters: False
```

-Path

The path to the configuration folder. Normally, you should run all commands from within the configuration folder.

```
Type: String
Parameter Sets: (All)
Aliases:

Required: False
Position: 0
Default value: None
Accept pipeline input: False
Accept wildcard characters: False
```

-WhatIf

Shows what would happen if the cmdlet runs. The cmdlet is not run.

```
Type: SwitchParameter
Parameter Sets: (All)
Aliases: wi

Required: False
Position: Named
Default value: None
Accept pipeline input: False
Accept wildcard characters: False
```

CommonParameters

This cmdlet supports the common parameters: -Debug, -ErrorAction, -ErrorVariable, -InformationAction, -InformationVariable, -OutVariable, -OutBuffer, -PipelineVariable, -Verbose, -WarningAction, and -WarningVariable. For more information, see about_CommonParameters.

Inputs

None

Outputs

System.Object

Notes

Related Links

Get-NetNat

Get-VMSwitch

Get-LabSnapshot

Synopsis

List available snapshots for a lab configuration.

Syntax

```
Get-LabSnapshot [[-Path] <String>] [<CommonParameters>]
```

Description

You can use Snapshot-Lab to create a set of checkpoints for an Autolab configuration. The default snapshot name is "LabConfigured", but you can create a snapshot with your own name. You need to know the snapshot name in order to restore it with Refresh-Lab. This command makes it easier to discover what snapshots you have created.

Note that if you want to remove a snapshot, use the Hyper-V Manager or PowerShell cmdlets as you would any other snapshot.

Examples

Example 1

```
PS C:\Autolab\Configurations\SingleServer> Get-LabSnapshot

All VMs in the configuration should belong to the same snapshot.

VMName Name          SnapshotType  CreationTime          ParentSnapshotName
------ ----          ------------  ------------          ------------------
S1     PreInstall    Standard      9/11/2020 12:06:51 PM
```

You could restore this snapshot by name using Refresh-Lab.

Parameters

-Path

The path to the configuration folder. Normally, you should run all commands from within the configuration folder.

```
Type: String
Parameter Sets: (All)
Aliases:

Required: False
Position: 0
Default value: None
Accept pipeline input: False
Accept wildcard characters: False
```

CommonParameters

This cmdlet supports the common parameters: -Debug, -ErrorAction, -ErrorVariable, -InformationAction, -InformationVariable, -OutVariable, -OutBuffer, -PipelineVariable, -Verbose, -WarningAction, and -WarningVariable. For more information, see about_CommonParameters.

Inputs

None

Outputs

System.Object

Notes

Related Links

Snapshot-Lab

Refresh-Lab

Get-LabSummary

Synopsis

Get a summary of the AutoLab configuration.

Syntax

```
Get-LabSummary [[-Path] <String>] [<CommonParameters>]
```

Description

This command makes it easy to see what the lab will look like when finished. You can see the computer names, what operating system they will be running, and how much memory each will require. Even though dynamic memory will be used in the Hyper-V configuration, for planning purposes you should assume you will need the full amount. This should make it easier to determine if you have enough available memory in your computer. Run the command in the root of the configuration folder.

If you have modified the configuration data file to use the EnvironmentPrefix setting, that value will be included as part of the virtual machine name.

Run Test-LabDSCResource from the configuration directory to see what DSC resources will be required and if they are already installed.

Examples

Example 1

```
PS C:\Autolab\Configurations\Windows10> Get-LabSummary


   Computername: Win10Ent VMName: Win10Ent

Lab           IPAddress       MemGB Procs Role            Description
---           ---------       ----- ----- ----            -----------
Windows10     192.168.3.101       2     2 {RSAT, RDP}     Windows 10 64bit
                                                          Enterprise 1903
                                                          English Evaluation
```

Get the configuration for the Windows 10 lab. The command has an associated formatting file to display the results as you see here. You might also want to pipe the Get-LabSummary command to Format-List to see all properties.

Example 2

```
PS C:\Autolab\Configurations\> Get-Childitem -Directory | Get-LabSummary |
Select-Object * | Out-GridView
```

Go through every active configuration and pipe the folder to Get-LabSummary. The total results are displayed using Out-GridView.

Parameters

-Path

The PATH to the lab configuration folder. Normally, you should run all commands from within the configuration folder. Do NOT include the psd1 file name.

```
Type: String
Parameter Sets: (All)
Aliases:

Required: False
Position: 0
Default value: None
Accept pipeline input: True (ByValue)
Accept wildcard characters: False
```

CommonParameters

This cmdlet supports the common parameters: -Debug, -ErrorAction, -ErrorVariable, -InformationAction, -InformationVariable, -OutVariable, -OutBuffer, -PipelineVariable, -Verbose, -WarningAction, and -WarningVariable. For more information, see about_CommonParameters.

Inputs

System.String

Outputs

System.Object

Notes

Related Links

Test-LabDSCResource

Get-VM

Get-PSAutoLabSetting

Synopsis

Get host and module information related to the PSAutoLab module.

Syntax

```
Get-PSAutoLabSetting [<CommonParameters>]
```

Description

If you need to report a problem with AutoLab, use this command to get relevant configuration and host information. Please include the output in your GitHub issue.

Examples

Example 1

```
PS C:\> Get-PSAutoLabSetting

AutoLab                     : C:\Autolab
PSVersion                   : 5.1.19041.610
PSEdition                   : Desktop
OS                          : Microsoft Windows 10 Pro
FreeSpaceGB                 : 705.95
MemoryGB                    : 32
PctFreeMemory               : 68.27
Processor                   : Intel(R) Core(TM) i9-10900T CPU @ 1.90GHz
IsElevated                  : True
RemotingEnabled             : True
NetConnectionProfile        : Private
HyperV                      : 10.0.19041.1
PSAutolab                   : {4.18.0, 4.17.0}
Lability                    : {0.19.1,0.18.0}
Pester                      : {5.1.0, 4.10.1, 3.4.0}
PowerShellGet               : 2.2.5
PSDesiredStateConfiguration : 1.1
```

The output will also show previously installed versions of the PSAutoLab and Lability modules. Only the latest version of each module will be used. You can remove the older versions if you no longer need them by running a command like `Uninstall-Module -name Lability -requiredversion 0.18.0`. The FreeSpaceGB value is the amount of free space on the drive containing your AutoLab folder.

Copy and paste this information into a GitHub issue along with any relevant error messages.

Note that the command uses a custom formatting file to display key values in color.

Parameters

CommonParameters

This cmdlet supports the common parameters: -Debug, -ErrorAction, -ErrorVariable, -InformationAction, -InformationVariable, -OutVariable, -OutBuffer, -PipelineVariable, -Verbose, -WarningAction, and -WarningVariable. For more information, see about_CommonParameters.

Inputs

None

Outputs

PSAutoLabSetting

Notes

Related Links

Get-Module

Get-Volume

Get-CimInstance

Get-NetConnectionProfile

Invoke-RefreshHost

Synopsis

Refresh the AutoLab configuration on the localhost.

Syntax

```
Invoke-RefreshHost [[-Destination] <String>] [-SkipPublisherCheck] [-WhatIf]
[-Confirm] [<CommonParameters>]
```

Description

If you keep the PSAutoLab module for any length of time, you will most likely update the module from time to time. Part of the update might include fixes or enhancements to current configurations or even entirely new configurations. This command makes it easier to keep your configurations up-to-date. After updating the PSAutoLab module, run this function in a new PowerShell session which will verify you have the correct version of the Lability module and copy configuration files to your Autolab\ConfigurationPath folder. This will not overwrite any MOF files or delete anything.

You will typically use the Refresh-Host alias.

Examples

Example 1

```
PS C:\> Refresh-Host
Version 0.19.1 of Lability is already installed
Updating configuration files from C:\Program Files\WindowsPowerShell\Modules\PSAutoLab\4.19.0\Configurations
This process will not remove any configurations that have been deleted from the module.
```

Parameters

-Confirm

Prompts you for confirmation before running the cmdlet.

```
Type: SwitchParameter
Parameter Sets: (All)
Aliases: cf

Required: False
Position: Named
Default value: None
Accept pipeline input: False
Accept wildcard characters: False
```

-Destination

The path to your configurations folder.

```
Type: String
Parameter Sets: (All)
Aliases:

Required: False
Position: 0
Default value: <drive>:\Autolab\Configurations
Accept pipeline input: False
Accept wildcard characters: False
```

-WhatIf

Shows what would happen if the cmdlet runs. The cmdlet is not run.

```
Type: SwitchParameter
Parameter Sets: (All)
Aliases: wi

Required: False
Position: Named
Default value: None
Accept pipeline input: False
Accept wildcard characters: False
```

-SkipPublisherCheck

If you try to refresh the host and get an error or warning about a certificate mismatch, use this parameter to bypass skipping the code signing certificate.

```
Type: SwitchParameter
Parameter Sets: (All)
Aliases:

Required: False
Position: Named
Default value: None
Accept pipeline input: False
Accept wildcard characters: False
```

CommonParameters

This cmdlet supports the common parameters: -Debug, -ErrorAction, -ErrorVariable, -InformationAction, -InformationVariable, -OutVariable, -OutBuffer, -PipelineVariable, -Verbose, -WarningAction, and -WarningVariable. For more information, see about_CommonParameters.

Inputs

None

Outputs

System.Object

Notes

Related Links

Invoke-RefreshLab

Synopsis

Refresh a PSAutolab configuration.

Syntax

```
Invoke-RefreshLab [[-Path] <String>] [-SnapshotName <String>] [-WhatIf]
[-Confirm] [<CommonParameters>]
```

Description

Use this command to restore your Autolab configuration from the last checkpoint. You will typically use the Refresh-Lab alias.

Examples

Example 1

```
PS C:\Autolab\Configurations\Windows10> Refresh-Lab
```

Restore the lab from a previously created Hyper-V checkpoint or snapshot.

Parameters

-Confirm

Prompts you for confirmation before running the cmdlet.

```
Type: SwitchParameter
Parameter Sets: (All)
Aliases: cf

Required: False
Position: Named
Default value: None
Accept pipeline input: False
Accept wildcard characters: False
```

-Path

The path to the configuration folder. Normally, you should run all commands from within the configuration folder.

```
Type: String
Parameter Sets: (All)
Aliases:

Required: False
Position: 0
Default value: Current Directory
Accept pipeline input: False
Accept wildcard characters: False
```

-WhatIf

Shows what would happen if the cmdlet runs. The cmdlet is not run.

```
Type: SwitchParameter
Parameter Sets: (All)
Aliases: wi

Required: False
Position: Named
Default value: None
Accept pipeline input: False
Accept wildcard characters: False
```

-SnapshotName

Specify a name for the virtual machine checkpoint

```
Type: String
Parameter Sets: (All)
Aliases:

Required: False
Position: Named
Default value: None
Accept pipeline input: False
Accept wildcard characters: False
```

CommonParameters

This cmdlet supports the common parameters: -Debug, -ErrorAction, -ErrorVariable, -InformationAction, -InformationVariable, -OutVariable, -OutBuffer, -PipelineVariable, -Verbose, -WarningAction, and -WarningVariable. For more information, see about_CommonParameters.

Inputs

None

Outputs

System.Object

Notes

Related Links

Snapshot-Lab

Invoke-RunLab

Synopsis

Start a PSAutolab configuration.

Syntax

```
Invoke-RunLab [[-Path] <String>] [-WhatIf] [-Confirm] [<CommonParameters>]
```

Description

Use this command to start a PSAutolab configuration. This command will start all of the virtual machines in the proper order. It is assumed you are running this from within the configuration folder.

You will typically use the Run-Lab alias.

Examples

Example 1

```
PS C:\AutoLab\Configurations\Windows10> Run-Lab
```

Parameters

-Confirm

Prompts you for confirmation before running the cmdlet.

```
Type: SwitchParameter
Parameter Sets: (All)
Aliases: cf

Required: False
Position: Named
Default value: None
Accept pipeline input: False
Accept wildcard characters: False
```

-Path

The path to the configuration folder. Normally, you should run all commands from within the configuration folder.

```
Type: String
Parameter Sets: (All)
Aliases:

Required: False
Position: 0
Default value: Current Directory
Accept pipeline input: False
Accept wildcard characters: False
```

-WhatIf

Shows what would happen if the cmdlet runs. The cmdlet is not run.

```
Type: SwitchParameter
Parameter Sets: (All)
Aliases: wi

Required: False
Position: Named
Default value: None
Accept pipeline input: False
Accept wildcard characters: False
```

CommonParameters

This cmdlet supports the common parameters: -Debug, -ErrorAction, -ErrorVariable, -InformationAction, -InformationVariable, -OutVariable, -OutBuffer, -PipelineVariable, -Verbose, -WarningAction, and -WarningVariable. For more information, see about_CommonParameters.

Inputs

None

Outputs

System.Object

Notes

Related Links

Shutdown-Lab

Invoke-SetupHost

Synopsis

Prepare the localhost for PSAutolab.

Syntax

```
Invoke-SetupHost [[-DestinationPath] <String>] [-WhatIf] [-Confirm]
[<CommonParameters>]
```

Description

The first time you install the PSAutoLab module, you will need to configure the localhost. This configuration will include adding the Hyper-V feature if it is not already installed. It will also install the supported version of the Lability module from the PowerShell Gallery. You only need to run this command once. If you update the PSAutoLab module at some point, it is recommended that you run Refresh-Host.

You will typically use the Setup-Host alias.

Examples

Example 1

```
PS C:\> Setup-Host
```

Follow the on-screen prompts. If you have to install the Hyper-V feature you definitely should reboot before setting up any lab configurations.

Example 2

```
PS C:\> Setup-Host -destination D:\Autolab
```

This will setup the Autolab module but put the necessary files on the D: drive. It is recommended that you use Autolab as the folder name.

Parameters

-Confirm

Prompts you for confirmation before running the cmdlet.

```
Type: SwitchParameter
Parameter Sets: (All)
Aliases: cf

Required: False
Position: Named
Default value: None
Accept pipeline input: False
Accept wildcard characters: False
```

-DestinationPath

Specify the parent path for your Autolab setup. The default is C:\Autolab The command will create the folder.

```
Type: String
Parameter Sets: (All)
Aliases:

Required: False
Position: 0
Default value: C:\Autolab
Accept pipeline input: False
Accept wildcard characters: False
```

-WhatIf

Shows what would happen if the cmdlet runs. The cmdlet is not run.

```
Type: SwitchParameter
Parameter Sets: (All)
Aliases: wi

Required: False
Position: Named
Default value: None
Accept pipeline input: False
Accept wildcard characters: False
```

CommonParameters

This cmdlet supports the common parameters: -Debug, -ErrorAction, -ErrorVariable, -InformationAction, -InformationVariable, -OutVariable, -OutBuffer, -PipelineVariable, -Verbose, -WarningAction, and -WarningVariable. For more information, see about_CommonParameters.

Inputs

None

Outputs

None

Notes

Related Links

Refresh-Host

Invoke-SetupLab

Synopsis

Set up an Autolab configuration.

Syntax

```
Invoke-SetupLab [[-Path] <String>] [-IgnorePendingReboot] [-UseLocalTimeZone]
[-WhatIf] [-Confirm] [<CommonParameters>]
```

Description

Once you have configured the localhost, change to a configuration folder under Autolab\Configurations, and set up a lab configuration. It is recommended that you first review any readme or instruction files. This command will generate the Desired State Configuration (DSC) MOFs, download required DSC resources, and create the virtual machines. Follow the on-screen instructions to continue.

You will typically use the `Setup-Lab` alias.

Examples

Example 1

```
PS C:\Autolab\Configurations\Windows10\> Setup-Lab
```

Follow on-screen instructions and prompts.

Parameters

-Confirm

Prompts you for confirmation before running the cmdlet.

```
Type: SwitchParameter
Parameter Sets: (All)
Aliases: cf

Required: False
Position: Named
Default value: None
Accept pipeline input: False
Accept wildcard characters: False
```

-IgnorePendingReboot

```
Type: SwitchParameter
Parameter Sets: (All)
Aliases:

Required: False
Position: Named
Default value: None
Accept pipeline input: False
Accept wildcard characters: False
```

-Path

The path to the configuration folder. Normally, you should run all commands from within the configuration folder.

```
Type: String
Parameter Sets: (All)
Aliases:

Required: False
Position: 0
Default value: Current Directory
Accept pipeline input: False
Accept wildcard characters: False
```

-WhatIf

Shows what would happen if the cmdlet runs. The cmdlet is not run.

```
Type: SwitchParameter
Parameter Sets: (All)
Aliases: wi

Required: False
Position: Named
Default value: None
Accept pipeline input: False
Accept wildcard characters: False
```

-UseLocalTimeZone

Override any configuration specified time zone and use the local time zone on this computer.

```
Type: SwitchParameter
Parameter Sets: (All)
Aliases:

Required: False
Position: Named
Default value: None
Accept pipeline input: False
Accept wildcard characters: False
```

CommonParameters

This cmdlet supports the common parameters: -Debug, -ErrorAction, -ErrorVariable, -InformationAction, -InformationVariable, -OutVariable, -OutBuffer, -PipelineVariable, -Verbose, -WarningAction, and -WarningVariable. For more information, see about_CommonParameters.

Inputs

None

Outputs

System.Object

Notes

Related Links

Unattend-Lab

Invoke-ShutdownLab

Synopsis

Shutdown an Autolab configuration.

Syntax

```
Invoke-ShutdownLab [[-Path] <String>] [-WhatIf] [-Confirm]
[<CommonParameters>]
```

Description

Use this command to shutdown the virtual machines of an Autolab configuration in the proper order. You can also manually use the Hyper-V management console or cmdlets to do the same thing. It is recommended that you shut down any domain controllers in your configuration last. It is assumed you are running this from within the configuration folder.

You will typically use the Shutdown-Lab alias.

Examples

Example 1

```
PS C:\Autolab\Configurations\PowerShellLab> Shutdown-Lab
```

Parameters

-Confirm

Prompts you for confirmation before running the cmdlet.

```
Type: SwitchParameter
Parameter Sets: (All)
Aliases: cf

Required: False
Position: Named
Default value: None
Accept pipeline input: False
Accept wildcard characters: False
```

-Path

The path to the configuration folder. Normally, you should run all commands from within the configuration

folder.

```
Type: String
Parameter Sets: (All)
Aliases:

Required: False
Position: 0
Default value: Current Directory
Accept pipeline input: False
Accept wildcard characters: False
```

-WhatIf

Shows what would happen if the cmdlet runs. The cmdlet is not run.

```
Type: SwitchParameter
Parameter Sets: (All)
Aliases: wi

Required: False
Position: Named
Default value: None
Accept pipeline input: False
Accept wildcard characters: False
```

CommonParameters

This cmdlet supports the common parameters: -Debug, -ErrorAction, -ErrorVariable, -InformationAction, -InformationVariable, -OutVariable, -OutBuffer, -PipelineVariable, -Verbose, -WarningAction, and -WarningVariable. For more information, see about_CommonParameters.

Inputs

None

Outputs

System.Object

Notes

Related Links

Run-Lab

Invoke-SnapshotLab

Synopsis

Create virtual machine snapshots of your Autolab configuration.

Syntax

```
Invoke-SnapshotLab [[-Path] <String>] [-SnapshotName <String>] [-WhatIf]
[-Confirm] [<CommonParameters>]
```

Description

Use this command to snapshot or checkpoint an entire Autolab configuration. You can later restore your configuration with Refresh-Lab. It is assumed you are running this command from within the configuration folder.

You will typically use the Snapshot-Lab alias.

Examples

Example 1

```
PS C:\Autolab\Configurations\Windows10> Snapshot-Lab
```

Parameters

-Confirm

Prompts you for confirmation before running the cmdlet.

```
Type: SwitchParameter
Parameter Sets: (All)
Aliases: cf

Required: False
Position: Named
Default value: None
Accept pipeline input: False
Accept wildcard characters: False
```

-Path

The path to the configuration folder. Normally, you should run all commands from within the configuration folder.

```
Type: String
Parameter Sets: (All)
Aliases:

Required: False
Position: 0
Default value: Current Directory
Accept pipeline input: False
Accept wildcard characters: False
```

-WhatIf

Shows what would happen if the cmdlet runs. The cmdlet is not run.

```
Type: SwitchParameter
Parameter Sets: (All)
Aliases: wi

Required: False
Position: Named
Default value: None
Accept pipeline input: False
Accept wildcard characters: False
```

-SnapshotName

Specify a name for the virtual machine checkpoint

```
Type: String
Parameter Sets: (All)
Aliases:

Required: False
Position: Named
Default value: LabConfigured
Accept pipeline input: False
Accept wildcard characters: False
```

CommonParameters

This cmdlet supports the common parameters: -Debug, -ErrorAction, -ErrorVariable, -InformationAction, -InformationVariable, -OutVariable, -OutBuffer, -PipelineVariable, -Verbose, -WarningAction, and -WarningVariable. For more information, see about_CommonParameters.

Inputs

None

Outputs

System.Object

Notes

Related Links

Refresh-Lab

Get-LabSnapshot

Invoke-UnattendLab

Synopsis

Create an Autolab configuration unattended.

Syntax

```
Invoke-UnattendLab [[-Path] <String>] [-AsJob] [-UseLocalTimeZone] [-WhatIf]
[-Confirm] [<CommonParameters>]
```

Description

Normally when you set up an Autolab configuration, you can do it manually by running commands in order:

- Setup-Lab
- Run-Lab
- Enable-Internet
- Validate-Lab

Or you can use this command which will string all of these commands together. You may need to answer an initial prompt to update the version of nuget.exe otherwise the installation should run unattended. Note that the validation will loop until the configurations are finalized and converged. You can press Ctrl+C at any time to break out of the test.

You should run this command from within the configuration folder.

You will typically use the Unattend-Lab alias.

Examples

Example 1

```
PS C:\Autolab\Configurations\PowerShellLab> Unattend-Lab
```

Follow any on-screen instructions or prompts.

Example 2

```
PS C:\Autolab\Configurations\MultiRole> Unattend-Lab -asJob
```

Run the setup unattended in a background job.

Parameters

-Confirm

Prompts you for confirmation before running the cmdlet.

```
Type: SwitchParameter
Parameter Sets: (All)
Aliases: cf

Required: False
Position: Named
Default value: None
Accept pipeline input: False
Accept wildcard characters: False
```

-Path

The path to the configuration folder. Normally, you should run all commands from within the configuration folder.

```
Type: String
Parameter Sets: (All)
Aliases:

Required: False
Position: 0
Default value: Current Directory
Accept pipeline input: False
Accept wildcard characters: False
```

-WhatIf

Shows what would happen if the cmdlet runs. The cmdlet is not run.

```
Type: SwitchParameter
Parameter Sets: (All)
Aliases: wi

Required: False
Position: Named
Default value: None
Accept pipeline input: False
Accept wildcard characters: False
```

-AsJob

Run the unattend process in a PowerShell background job.

```
Type: SwitchParameter
Parameter Sets: (All)
Aliases:

Required: False
Position: Named
Default value: None
Accept pipeline input: False
Accept wildcard characters: False
```

-UseLocalTimeZone

Override any configuration specified time zone and use the local time zone on this computer.

```
Type: SwitchParameter
Parameter Sets: (All)
Aliases:

Required: False
Position: Named
Default value: None
Accept pipeline input: False
Accept wildcard characters: False
```

CommonParameters

This cmdlet supports the common parameters: -Debug, -ErrorAction, -ErrorVariable, -InformationAction, -InformationVariable, -OutVariable, -OutBuffer, -PipelineVariable, -Verbose, -WarningAction, and -WarningVariable. For more information, see about_CommonParameters.

Inputs

None

Outputs

System.Object

Notes

Related Links

Setup-Lab

Invoke-ValidateLab

Synopsis

Validate an Autolab configuration.

Syntax

```
Invoke-ValidateLab [[-Path] <String>] [<CommonParameters>]
```

Description

Lab configurations in Autolab use Desired State Configuration. These configurations can take some time to finish and converge. This command will validate that all virtual machines in the configuration are properly configured. It will loop through every 5 minutes running a Pester test suite for the configuration. Once all tests pass, the command will run the test one more time to display the results. You will see errors until all tests have passed. Depending on the configuration, this test could take up to 60 minutes to complete. You can press Ctrl+C at any time to break out of the test. If you prefer, you can also manually run the Pester test.

PS C:\Autolab\Configurations\PowerShellLab> Invoke-Pester .\VMvalidate.test.ps1

You will typically use the Validate-Lab alias.

Note that beginning in v4.21.0, this validation command will keep track of the number of testing loops. After 2 loops it will check for any virtual machine that is failing a test to see if it has stopped. If so, it will be started. After 4 loops and virtual machine that is still failing tests will be restarted. Ideally, Validation should complete in 30 minutes or less. The validation process will abort after 65 minutes.

Run Validate-Lab -Verbose to see details about the number of testing loops and which virtual machines are started or restarted.

Examples

Example 1

```
PS C:\AutoLab\Configurations\Windows10> Validate-Lab -verbose
```

You will see errors until all tests have passed. Press Ctrl+C to break out of the test. Configuration merging will continue in the virtual machines.

Parameters

-Path

The path to the configuration folder. Normally, you should run all commands from within the configuration folder.

```
Type: String
Parameter Sets: (All)
Aliases:

Required: False
Position: 0
Default value: Current Directory
Accept pipeline input: False
Accept wildcard characters: False
```

CommonParameters

This cmdlet supports the common parameters: -Debug, -ErrorAction, -ErrorVariable, -InformationAction, -InformationVariable, -OutVariable, -OutBuffer, -PipelineVariable, -Verbose, -WarningAction, and -WarningVariable. For more information, see about_CommonParameters.

Inputs

None

Outputs

System.Object

Notes

Related Links

Invoke-Pester

Invoke-WipeLab

Synopsis

Remove an Autolab configuration.

Syntax

```
Invoke-WipeLab [[-Path] <String>] [-RemoveSwitch] [-Force] [-WhatIf] [-Confirm] [<CommonParameters>]
```

Description

You can use this command to remove all files and virtual machines related to an Autolab configuration. The command will stop any running virtual machines for you. It is assumed you will be running this command from within a configuration folder.

If you intend to rebuild the lab or create another configuration, you do not need to delete the virtual switch (LabNet).

Use -Force to suppress all prompts.

You will typically use the Wipe-Lab alias.

Examples

Example 1

```
PS C:\AutoLab\Configurations\Windows10> Wipe-Lab
```

Follow any on-screen prompts or instructions.

Example 2

```
PS C:\AutoLab\Configurations\SingleServer> Wipe-Lab -force -RemoveSwitch
```

Forcibly remove all lab elements including the virtual switch.

Parameters

-Path

The path to the configuration folder. Normally, you should run all commands from within the configuration folder.

```
Type: String
Parameter Sets: (All)
Aliases:

Required: False
Position: 0
Default value: Current Directory
Accept pipeline input: False
Accept wildcard characters: False
```

-Confirm

Prompts you for confirmation before running the cmdlet.

```
Type: SwitchParameter
Parameter Sets: (All)
Aliases: cf

Required: False
Position: Named
Default value: None
Accept pipeline input: False
Accept wildcard characters: False
```

-WhatIf

Shows what would happen if the cmdlet runs. The cmdlet is not run.

```
Type: SwitchParameter
Parameter Sets: (All)
Aliases: wi

Required: False
Position: Named
Default value: None
Accept pipeline input: False
Accept wildcard characters: False
```

-Force

Remove lab elements with no prompting.

```
Type: SwitchParameter
Parameter Sets: (All)
Aliases:

Required: False
Position: Named
Default value: None
Accept pipeline input: False
Accept wildcard characters: False
```

-RemoveSwitch

Remove the VM Switch. It is retained by default.

```
Type: SwitchParameter
Parameter Sets: (All)
Aliases:

Required: False
Position: Named
Default value: None
Accept pipeline input: False
Accept wildcard characters: False
```

CommonParameters

This cmdlet supports the common parameters: -Debug, -ErrorAction, -ErrorVariable, -InformationAction, -InformationVariable, -OutVariable, -OutBuffer, -PipelineVariable, -Verbose, -WarningAction, and -WarningVariable. For more information, see about_CommonParameters.

Inputs

None

Outputs

System.Object

Notes

Related Links

Setup-Lab

Open-PSAutoLabHelp

Synopsis

Open the PDF help manual for the PSAutoLab module.

Syntax

```
Open-PSAutoLabHelp [<CommonParameters>]
```

Description

This module ships with a PDF that contains documentation files found in the GitHub repository as well all command help. You might find it easier to read the PDF than to use PowerShell.

This command was introduced in version 4.17.0 of the PSAutoLab module.

Examples

Example 1

```
PS C:\> Open-PSAutoLabHelp
```

Open the PDF help manual with the associated application.

Parameters

CommonParameters

This cmdlet supports the common parameters: -Debug, -ErrorAction, -ErrorVariable, -InformationAction, -InformationVariable, -OutVariable, -OutBuffer, -PipelineVariable, -Verbose, -WarningAction, and -WarningVariable. For more information, see about_CommonParameters.

Inputs

None

Outputs

None

Notes

Related Links

Test-ISOImage

Synopsis

Test PSAutolab ISO images

Syntax

```
Test-ISOImage [<CommonParameters>]
```

Description

This command is designed to test and validate downloaded ISO images used by the PSAutolab module. Each ISO file should have a corresponding checksum file with the valid MD5 hash. Test-ISOImage will test each ISO image file and compare it to the checksum value if found. If you have not setup any lab configurations, then you won't have any ISO image files.

ISO image files are only downloaded once during setup. If any of the ISO images fail to pass validation, you should delete the file along with its associated checksum file. Wipe the lab that is using the image and set it up again.

Examples

Example 1

```
PS C:\> Test-ISOImage

Path                                          Valid Size
----                                          ----- ----
C:\autolab\ISOs\2016_x64_EN_Eval.iso          True  6972221440
C:\autolab\ISOs\2019_x64_EN_Eval.iso          True  4973780992
C:\autolab\ISOs\WIN10_x64_ENT_19H2_EN_Eval.iso True  4892766208
```

You can pipe the results to Select-Object * to see all property values.

Parameters

CommonParameters

This cmdlet supports the common parameters: -Debug, -ErrorAction, -ErrorVariable, -InformationAction, -InformationVariable, -OutVariable, -OutBuffer, -PipelineVariable, -Verbose, -WarningAction, and -WarningVariable. For more information, see about_CommonParameters.

Inputs

None

Outputs

ISOTest

Notes

Related Links

Get-FileHash

Test-LabDSCResource

Synopsis

Test for required DSC resources.

Syntax

```
Test-LabDSCResource [[-Path] <String>] [<CommonParameters>]
```

Description

This is a troubleshooting command for the PSAutoLab module. It is designed to be run in a lab configuration folder. It will report on the required Desired State Configuration (DSC) resources and any versions that may already be installed. This can be used to diagnose potential version conflicts. Resources are installed automatically when you build a configuration so you don't need to take any action unless directed.

Examples

Example 1

```
PS C:\autolab\Configurations\MultiRole> Test-LabDSCResource


   Configuration: MultiRole

ModuleName                   RequiredVersion Installed InstalledVersions
----------                   --------------- --------- -----------------
xActiveDirectory             3.0.0.0         True      3.0.0.0
xComputerManagement          4.1.0.0         True      4.1.0.0
xNetworking                  5.7.0.0         True      5.7.0.0
xDhcpServer                  3.0.0           True      3.0.0
xWindowsUpdate               2.8.0.0         True      2.8.0.0
xPSDesiredStateConfiguration 9.1.0           True      9.1.0
xPendingReboot               0.4.0.0         True      0.4.0.0
xADCSDeployment              1.4.0.0         True      1.4.0.0
xDnsServer                   1.16.0.0        True      1.16.0.0
```

Run this command from the lab configuration folder.

Parameters

-Path

Specify the folder path of an AutoLab configuration or change locations to the folder and run this command.

```
Type: String
Parameter Sets: (All)
Aliases:

Required: False
Position: 0
Default value: .
Accept pipeline input: False
Accept wildcard characters: False
```

CommonParameters

This cmdlet supports the common parameters: -Debug, -ErrorAction, -ErrorVariable, -InformationAction, -InformationVariable, -OutVariable, -OutBuffer, -PipelineVariable, -Verbose, -WarningAction, and -WarningVariable. For more information, see about_CommonParameters.

Inputs

None

Outputs

PSAutoLabResource

Notes

Related Links

Get-LabSummary

Get-DSCResource

Update-Lab

Synopsis

Run Windows update on Autolab virtual machines.

Syntax

```
Update-Lab [[-Path] <String>] [-AsJob] [<CommonParameters>]
```

Description

When you build a lab, you are creating Windows virtual machines based on evaluation software. You might still want to make sure the virtual machines are up to date with security patches and updates. You can use this command to invoke Windows update on all lab members. This can be a time-consuming process, especially for labs with multiple virtual machines. The recommended syntax is to use the -AsJob parameter which runs the update process for each virtual machine in a background job. Use PowerShell's job cmdlets to manage the jobs. Do not close your PowerShell session before the jobs complete.

The virtual machine must be running to update.

It is recommended that you reboot all the lab virtual machines after updating.

Examples

Example 1

```
PS C:\Autolab\Configurations\PowerShellLab> update-lab -AsJob

Id     Name       PSJobTypeName   State     HasMoreData     Location   Command
--     ----       -------------   -----     -----------     --------   -------
18     WUUpdate   RemoteJob       Running   True            DOM1       WUUpdate
21     WUUpdate   RemoteJob       Running   True            SRV1       WUUpdate
24     WUUpdate   RemoteJob       Running   True            SRV2       WUUpdate
27     WUUpdate   RemoteJob       Running   True            SRV3       WUUpdate
30     WUUpdate   RemoteJob       Running   True            WIN10      WUUpdate

PS C:\Autolab\Configurations\PowerShellLab> receive-job -id 27 -Keep
[11/22/2020 12:05:43] Found 5 updates to install on SRV3
[11/22/2020 12:25:13] Update process complete on SRV3
WARNING: SRV3 requires a reboot
```

Run the update process as a background job. Use the PowerShell job cmdlets to manage.

Parameters

-AsJob

Run the update process in a background job.

```
Type: SwitchParameter
Parameter Sets: (All)
Aliases:

Required: False
Position: Named
Default value: None
Accept pipeline input: False
Accept wildcard characters: False
```

-Path

The path to the configuration folder. Normally, you should run all commands from within the configuration folder.

```
Type: String
Parameter Sets: (All)
Aliases:

Required: False
Position: 0
Default value: None
Accept pipeline input: False
Accept wildcard characters: False
```

CommonParameters

This cmdlet supports the common parameters: -Debug, -ErrorAction, -ErrorVariable, -InformationAction, -InformationVariable, -OutVariable, -OutBuffer, -PipelineVariable, -Verbose, -WarningAction, and -WarningVariable. For more information, see about_CommonParameters.

Inputs

None

Outputs

System.Object

Notes

Related Links

Get-Job

Receive-Job