Microsoft

PowerShell Jobs

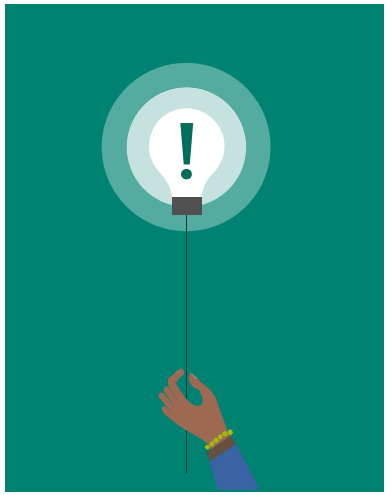
Microsoft Services

Working with PowerShell Jobs

Objectives

After completing Working with PowerShell Jobs, you will be able to:

- Understand what Jobs are
- Learn different Job types
- Learn to use Job cmdlets
- Understand a basic Job flow
- Understand methods of starting Remote Jobs
- Work with execution remoting of Jobs
- Review key differences between scheduled tasks
- View how to create and manage a scheduled Job
- Understand scheduled job options
- Understand modifying Scheduled Task Options



Overview

What are Jobs

PowerShell feature to perform **Asynchronous** operation(s)

- Can be **scheduled**
- Supports **remoting**
- Can be started as a parameter (**-AsJob**) of
  - Workflows
  - WMI/CIM cmdLet
- Results (**objects**) can be retrieved after the operation is fully complete or ongoing

```
PS C:\> Start-Job { Get-ChildItem -Recurse | Sort-Object Length }

Id Name PSJobTypeName State   HasMoreData Location  Command
-- ---- ------------- -----   ----------- --------  -------
1  Job1 BackgroundJob Running True        localhost Get-ChildItem -Recurs...

PS C:\>
```

When to use Jobs

- Scheduling PowerShell scripts
- Multitasking / Multithreading Scenarios
- Long running background operations (backup, copy, create VM's, etc.)
- Log collection / Inventory / Operating multiple servers
- Divide a huge operation into smaller pieces (Ex: file script on a file server)

Job Types

BackgroundJob – Basic Job that is not scheduled

RemoteJob – Job on remote computer

ScheduledTask – Windows Task Scheduler, ClusterAware, better for non-PowerShell tasks

PSScheduledJob – Hybrid of BackgroundJob and ScheduledTask

CIMJob/WMIJob – CIM/WMI CmdLets those run as a job using (-AsJob)

RunSpace – Supports Multi-Threading, better for scaling

PSWorkflowJob –Workflow that runs as a job (-AsJob)

CIMJob/WMIJob – CIM/WMI CmdLets those run as a job using (-AsJob)

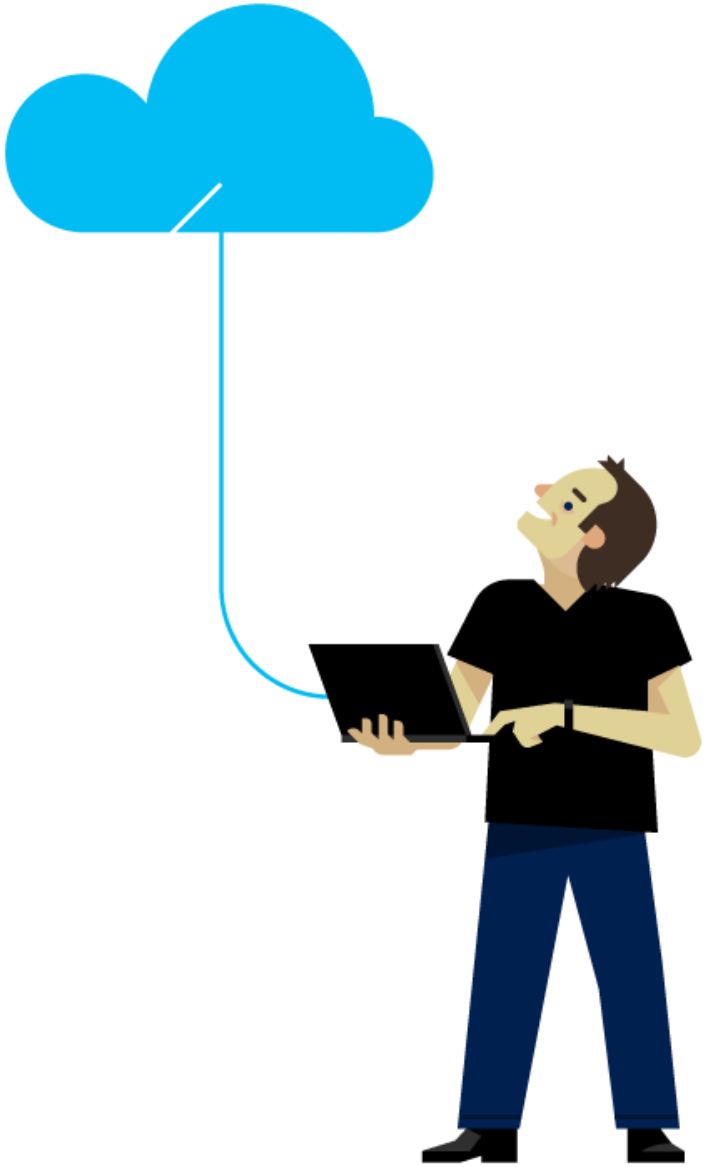
PSEventJob – Created by Register-ObjectEvent to watch and act on Object change

```
PS C:> Get-Job
Id  Name          PSJobTypeName   State       HasMoreData  Location   Command
--  ----          -------------   -----       -----------  --------   -------
2   LocalData     BackgroundJob   Completed   True         localhost  Get-Process
4   RemoteData    RemoteJob       Completed   True         Server01   Get-Process
6   TestWFJob     PSWorkflowJob   Completed   True         localhost  WorkflowJob
8   ScheduledJob  PSScheduledJob  Completed   True         localhost  Get-Process
```

Demonstration

Background Jobs

Questions?

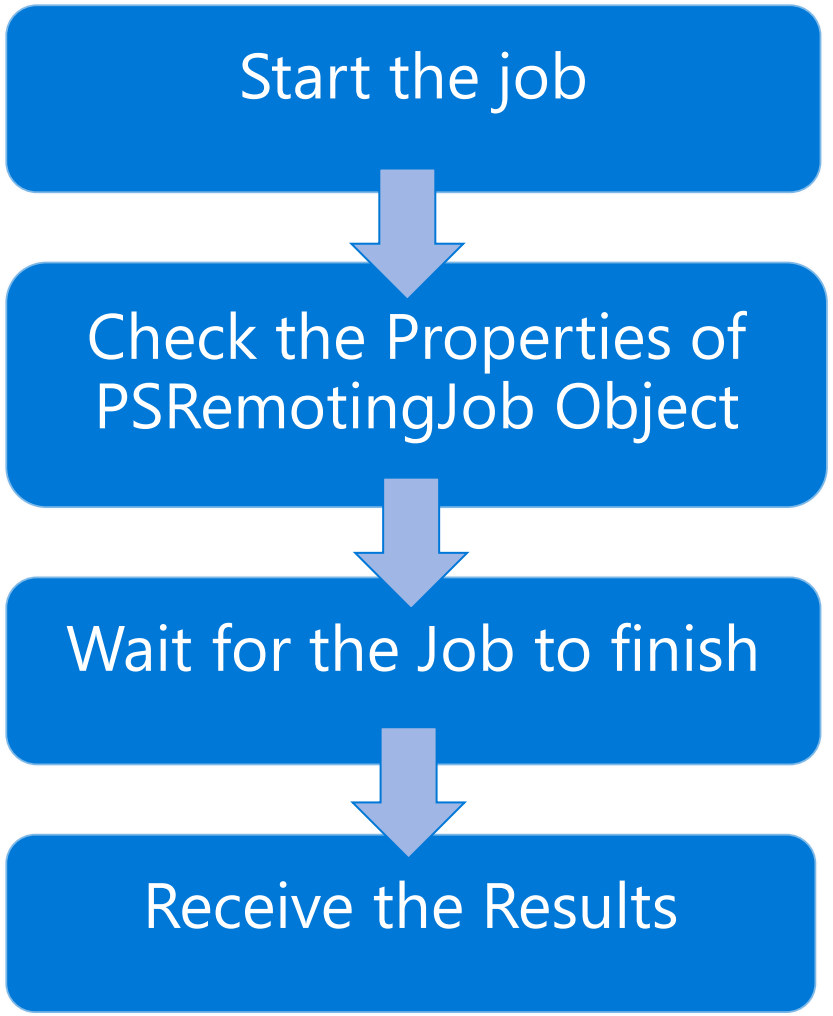


Starting a Background Job

Background Jobs

- Runs a command or expression asynchronously
  - Also used for multithreading purposes
- Used for local scenarios
- Managed by Job Following Cmdlets:
  - **Start-Job**
  - **Get-Job**
  - **Stop-Job**
  - **Wait-Job**
  - **Remove-Job**
- Job cmdlets returns or accepts PSRemotingJob Objects
- Results received by Receive-Job
- Consists of a parent job and one or more child jobs

Basic Job Flow



Starting a Background Job

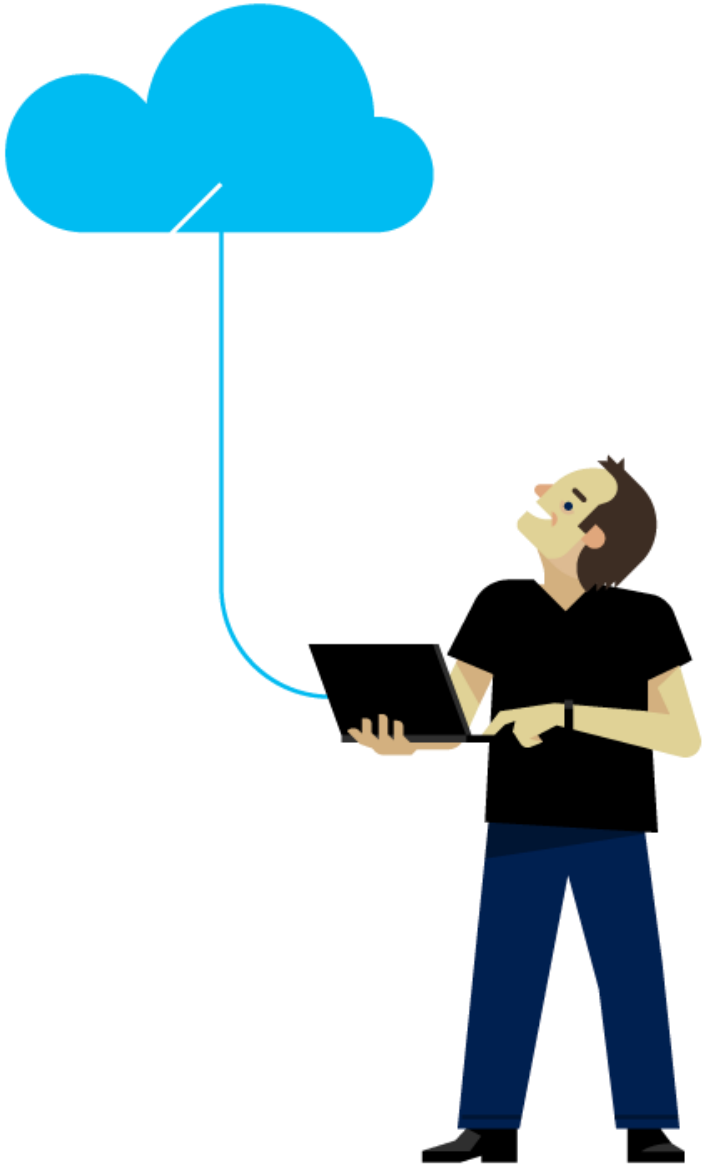
- Start-Job cmdlet starts the job with:
  - ScriptBlock
  - .ps1 script
- Returns PSRemotingJob Object
- To start a script before the job use –InitializationScript parameter

```
 # Starting a job with -ScriptBlock parameter
Start-Job –ScriptBlock { Get-ChildItem –Recurse –Path C:\ }

# Jobs can be also started using scripts but are converted to ScriptBlocks
Start-Job –FilePath C:\Scripts\GetFilesRecursively.ps1

# It is possible to start a script before the job starts
Start-Job -Name GetMapFiles -InitializationScript {Import-Module MapFunctions}`
 -ScriptBlock {Get-Map 0 -Name * | Set-Content D:\Maps.tif} -RunAs32
```

Demonstration

Background Jobs

Questions?



Working with Job Objects

Access, Monitor, and Manage Jobs

- To access, monitor, and manage jobs, use **Get-Job** which returns:
  - Background Jobs
  - Child Jobs (-includechildjobs)
- Has filter options
  - -Filter
  - -Before / -after
  - -HasMoreData

```
PS C:\>  Start-Job { Get-Service Spooler, FakeService }

PS C:\> Get-Job # For a full list of properties used to manage o job get-job | get-
member

Id  Name  PSJobTypeName   State      HasMoreData     Location        Command
--  ----  -------------   -----      -----------     --------        -------
1   Job1  BackgroundJob   Completed  True            localhost       Get-Service..
```

PSRemotingJob Object

- The executive which manages Child Jobs doing the actual work
- Created by **Start-Job** and accessed by **Get-Job**

Property	Description
Error	Gets or sets the error buffer. Errors of job are written into this buffer.
HasMoreData	Indicates that more data is available in this result object for reading.
ID	Short identifier for this result which will be recycled and used within a process.
JobStateInfo	StateInformation and reason is kept on this
Location	Indicates the location of the job object (in a local background job it will be localhost).
Name	Name for identifying this job object.
PSBeginTime	Time job was started.
PSEndTime	Time job stopped.
PSJobTypeName	Job type name.
State	Indicates the state of the job.

PSRemotingChildJob

- Each Background job consists of one or more ChildJobs
- Error and Output is stored in ChildJobs
- Parent Job is an executive to store overall state of ChildJobs
- Can be accessed by –IncludeChildJob (as of Powershell 3.0)

```
PS C:\> # Each Job spawns a child Job to do the Actual Work
PS C:\> Get-Job -IncludeChildJob

Id   Name   PSJobTypeName   State       HasMoreData   Location    Command
--   ----   -------------   -----       -----------   --------    -------
1    Job1   BackgroundJob   Completed   True          localhost   Get-Service...
2    Job2                   Completed   True          localhost   Get-Service...
```

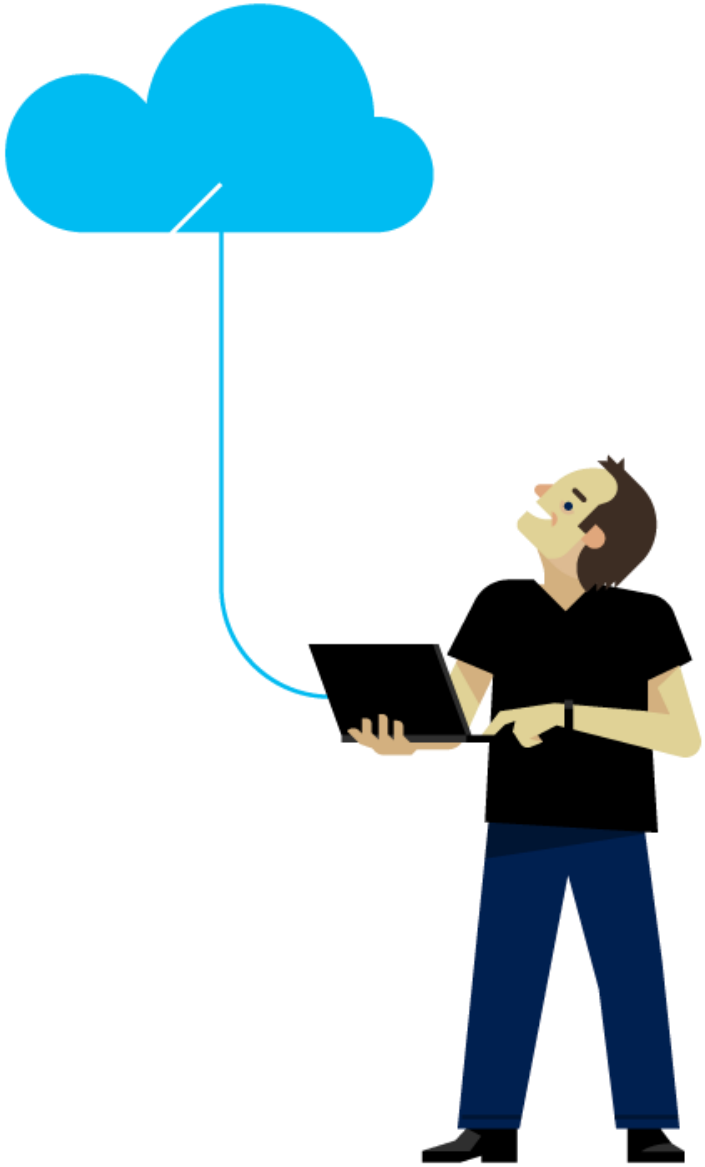
Child Jobs Collection

```
PS C:\> $job = Get-Job

PS C:\> # ChildJobs is a collection, so index notation can be used.

PS C:\> $job.ChildJobs[0].Output

Status    Name                DisplayName
------    ----                -----------
Running   Spooler             Print Spooler

PS C:\> # The Child Job(s) will also store all of the Stream Data.
PS C:\> $job.ChildJobs[0].Error
Cannot find any service with service name 'FakeService'.
.
.
```

Demonstration

Working with Job Objects

Questions?



Managing Background Jobs

Managing Jobs

Cmdlet	Description	Example
Stop-Job	Used to Stop a current running Job.	`# Stop the Job with an Id of 3.` `Stop-Job -Id 3`
Remove-Job	Remove a Job. Use –Force parameter to remove running Job.	`# Remove All Jobs regardless of State.` `Get-Job | Remove-Job -Force`
Suspend-Job	Suspend Execution of a Running Workflow Job.	`# Suspends all Jobs regardless of State.` `Get-Job | Suspend-Job -Force`
Resume-Job	Resume a Suspended Workflow Job.	`# Resume All Suspended Jobs.` `Resume-Job -State Suspended`

Wait-Job	Bring a Background Job to the Foreground.
	Suspends Console interaction until Job is complete.
		`# Wait on Completion of All Jobs.`
		`Get-Job | Wait-Job`
		…
		…

Job States

- All Jobs have a state property

State	Description
NotStarted	Execution of command in job not started.
Running	Execution of command in progress.
Stopped/Stopping	Execution is Canceled or Canceling.
Suspended/Suspending	Execution is Suspended or Suspending.
Completed	Execution is Completed
AtBreakPoint	Script execution is halted in a debugger stop.
Blocked	Execution is blocked (on user input, host calls, etc).
Disconnected	The job is a remote job and has been disconnected from the server.
Failed	An error was encountered.

Receiving Job Results

- Receive-Job
  - Job has results ready to be received as long as HasMoreData is $true
  - Does not keep the received objects unless –keep
  - Does not wait for the job to complete unless –wait

```
PS C:\>  Start-Job { Get-ChildItem -Recurse -Filter *.vhdx }
# Wait until Job is Complete
PS C:\> Get-Job | Receive-Job

 Directory: C:\Users\All Users\Microsoft\Windows\Hyper-V\DC1\Virtual Hard Disks

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        9/11/2017   6:03 AM     9634316288 DC1.vhdx

# Output is not kept because -Keep is not used.
# Console is not frozen at any point because –Wait or Wait-Job was not used.
```

Job Output

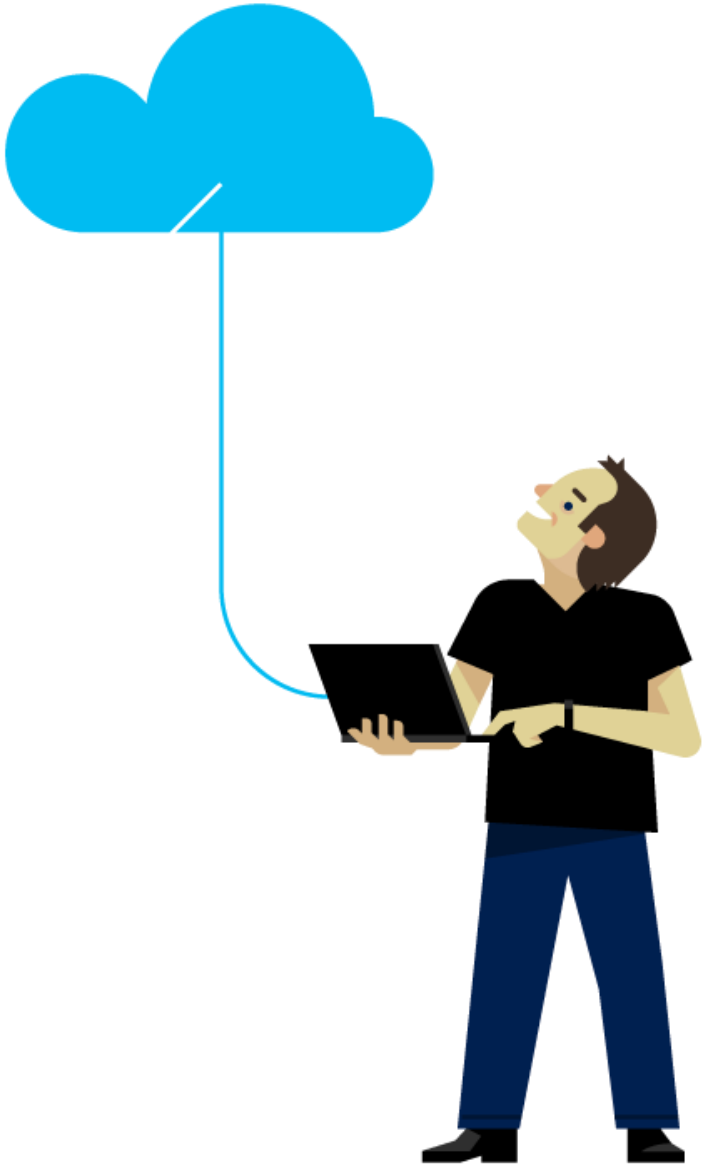
- Result is deserialized

```
PS C:\>  Start-Job { Get-Service Spooler }

PS C:\> Get-Job | Receive-Job | Get-Member

   TypeName: Deserialized.System.ServiceProcess.ServiceController

Name                      MemberType   Definition
----                      ----------   ----------
Name                      NoteProperty string Name=spooler
PSComputerName            NoteProperty string PSComputerName=localhost
RequiredServices          NoteProperty Deserialized.System......
RunspaceId                NoteProperty guid RunspaceId=f5f9b4ce-3d54-43dc-8ed
CanPauseAndContinue       Property     System.Boolean {get;set;}
CanShutdown               Property     System.Boolean {get;set;}
CanStop                   Property     System.Boolean {get;set;}
```

Demonstration

Managing Background Jobs

Questions?



Remote Jobs

Remote Jobs

- Leverages **Invoke-Command -AsJob**
- Objects are serialized and return to your local machine
- Objects are not lost if the session ends on remote machine
- Local Job Object can monitor and manage the job that is running on the remote machine
  - Local job receives the serialized output from the remote machine

Starting Jobs using -AsJob

```
 PS C:\> Invoke-Command -ComputerName dc01 -ScriptBlock {Get-EventLog `
-LogName System } -AsJob

Id Name PSJobTypeName State   HasMoreData Location Command
-- ---- ------------- -----   ----------- -------- -------
3  Job3 RemoteJob     Running True        dc01      Get-Ev...

 PS C:\> Get-Job

Id Name PSJobTypeName State     HasMoreData Location  Command
-- ---- ------------- -----     ----------- --------  -------
1  Job1 BackgroundJob Completed True        localhost ...
3  Job3 RemoteJob     Completed True        dc01      ...
```

Receiving Results using -AsJob

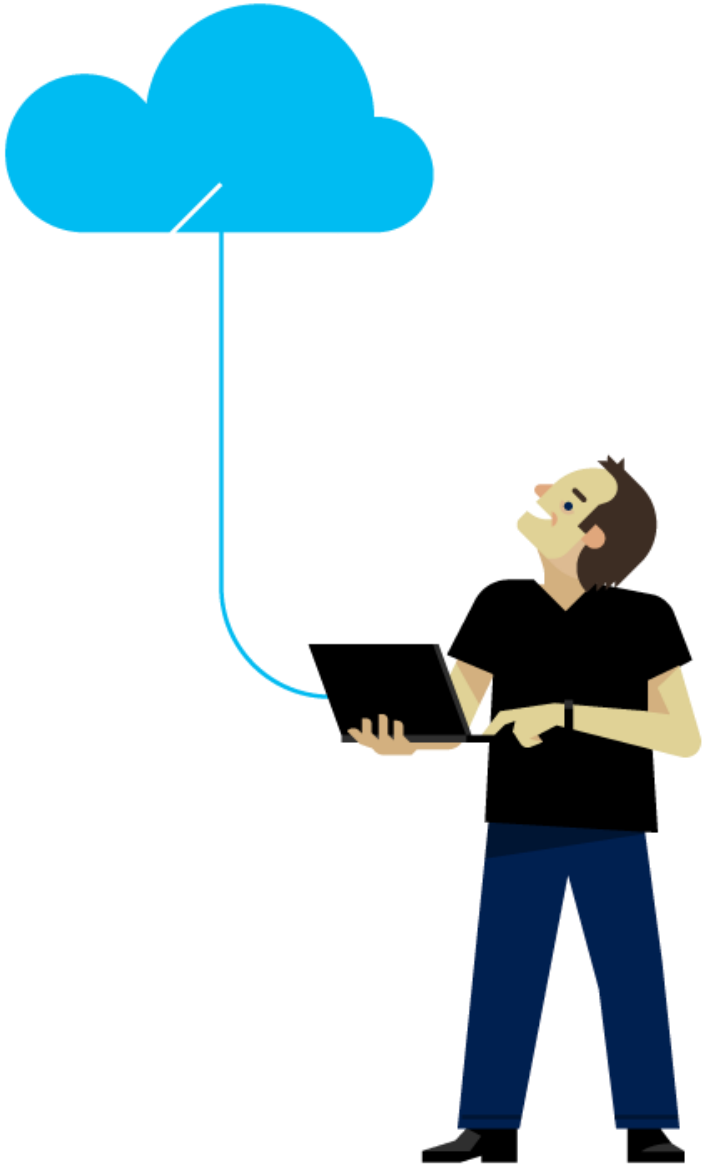
- Job objects are local results are attached to local jobs serialized

```
 PS C:\>  Receive-Job -Id 3  | Select-Object -Property EventID,`
TimeGenerated, Source, PsComputerName


 EventID  TimeGenerated           Source                         PSComputerName
-------  -------------           ------                         --------------
    7036  12/10/2018 4:44:37 PM  Service Control Manager         dc01
    7036  12/10/2018 4:38:13 PM  Service Control Manager         dc01
    7036  12/10/2018 4:32:32 PM  Service Control Manager         dc01
    7036  12/10/2018 4:32:31 PM  Service Control Manager         dc01
```

Demonstration

Remote Background Jobs

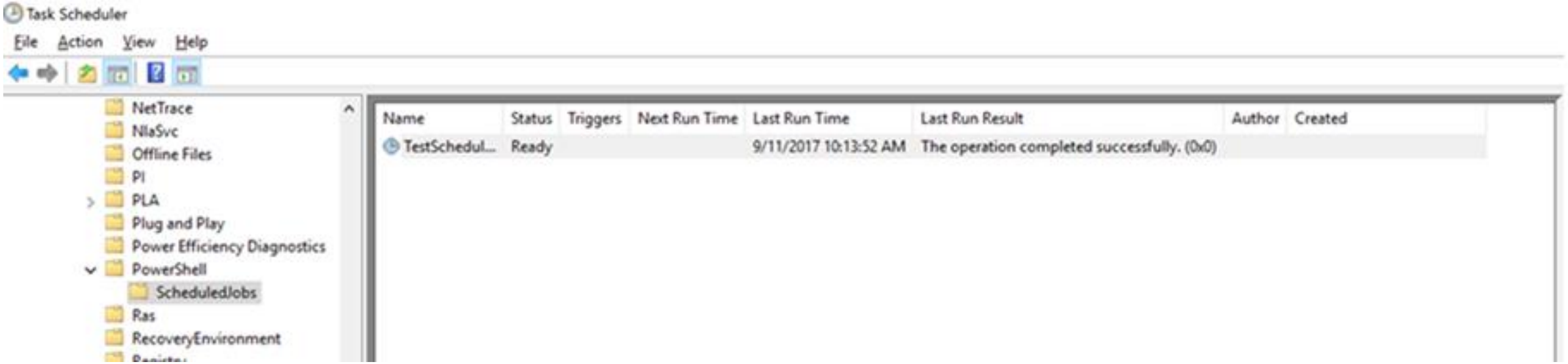
Questions?



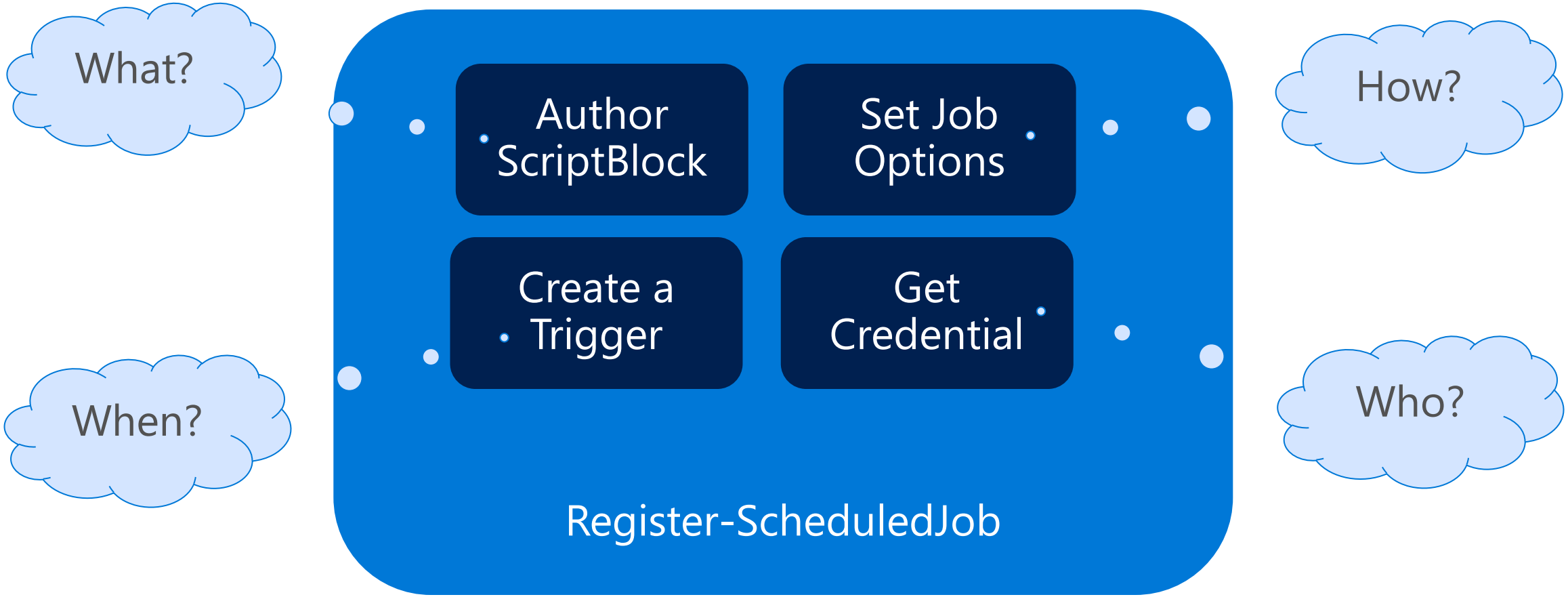
Scheduled Jobs

What are Scheduled Jobs?

- Hybrid of background jobs and Task Scheduler tasks
- Managed by Job cmdlets included in PSScheduledJob Module
- Results and execution history are saved to disk
  - Can be received by **Receive-Job**



Preparing for Scheduled Jobs



Job Triggers – When?

Repetition	StartTime	Interval of Repetition	EndTime
-Once	-At <datetime>	-RepetitionInterval <timespan>	-RepetitionDuration <timespan> -RepeatIndefinitely
-Weekly	-At <datetime>	-WeeksInterval <int> -DaysOfWeek <DayOfWeek[]>	Repeats Indefinitely
-Daily	-At <datetime>	-DaysInterval <int>	Repeats Indefinitely
-AtStartup	On Event	At Event Occurrence	Repeats Indefinitely
-AtLogon	On Event	At Event Occurrence	Repeats Indefinitely

Creating a Trigger

Job triggers define the schedule

Cmdlet	Description
New-JobTrigger	Creates a job trigger.
Get-JobTrigger	Gets a job trigger.
Add-JobTrigger	Adds a job trigger to a scheduled job.
Set-JobTrigger	Changes a job trigger.
Disable-JobTrigger	Temporarily disables a job trigger.
Enable-JobTrigger	Re-enables a job trigger.
Remove-JobTrigger	Deletes a job trigger.

Trigger Examples

```
PS C:\> $trigger = New-JobTrigger -Daily -At “04:45:00PM”

PS C:\> $trigger

Id   Frequency  Time                   DaysOfWeek     Enabled
--   ---------  ----                   ----------     -------
0    Daily      9/11/2017 4:45:00 PM                  True

PS C:\> $trigger = New-JobTrigger -Weekly -DaysOfWeek Monday, Wednesday,
Friday -At "23:00" -WeeksInterval 4 # Repeats for 4weeks

PS C:\> $trigger

Id   Frequency  Time                   DaysOfWeek                Enabled
--   ---------  ----                   ----------                -------
0    Weekly     1/1/2019 11:00:00 PM   {Monday, Wednesday, ... True
```

Setting Job Options – How?

- Are the same found in TaskScheduler dialog boxes

- The actual options are stored as properties of a [ScheduledJobOptions] object **typeManaged** by **ScheduledJobOption** cmdlets

Cmdlet	Description
New-ScheduledJobOption	Creates a job options object.
Get-ScheduledJobOption	Gets the job options of a scheduled job.
Set-ScheduledJobOption	Changes the job options of a scheduled job.

Working with ScheduledJobOptions Object

```
PS C:\> Get-ScheduledJob | Get-ScheduledJobOption

StartIfOnBatteries      : False
StopIfGoingOnBatteries  : True
WakeToRun               : False
StartIfNotIdle          : True
StopIfGoingOffIdle      : False
RestartOnIdleResume     : False
IdleDuration            : 00:10:00
IdleTimeout             : 01:00:00
ShowInTaskScheduler     : True
RunElevated             : True
RunWithoutNetwork       : True
DoNotAllowDemandStart   : False
MultipleInstancePolicy  : IgnoreNew
JobDefinition           : Microsoft.PowerShell.ScheduledJob.ScheduledJobDefinition
```

Job Credentials – Who?

- ScheduledJobs are invoked as current user by default
- For alternate credentials, pass in PSCredential object as -Credential parameter to Start-Job

```
PS C:\> $credential = Get-Credential
```

Registering/Creating a Scheduled Job

- Use parameters to set: Options, Credentials, ScriptBlob, and Trigger
- Creates a ScheduledJobDefinition Object in:
  - TaskSchedulerLibrary\Microsoft\Windows\PowerShell

```
# The Register-ScheduledJob returns a ScheduledJobDefinition Object.
PS C:\>  Register-ScheduledJob
-Name Test `
-Trigger $trigger `
-ScheduledJobOption $options `
-Credential $credential
-ScriptBlock { Restart-Service DHCP -Force -Verbose}

Id   Name    JobTriggers   Command                        Enabled
--   ----    -----------   -------                        -------
6    Test    1             Restart-Service DHCP           True
```

Modifying an Existing Scheduled Job

```
# Disable a Job Trigger
PS C:\>  Get-ScheduledJob | Get-JobTrigger | Disable-JobTrigger

# Add a Job Trigger. Note that the Date should fit to your regional settings
PS C:\>  Get-ScheduledJob | Add-JobTrigger -Trigger (New-JobTrigger -At "04/17/2017
04:00:00AM" -Once)

# Turn off Process Elevation.
PS C:\>  (Get-ScheduledJob).Options | Set-ScheduledJobOption -RunElevated:$false

# Prevent the Job from being manually run.
PS C:\>  (Get-ScheduledJob).Options | Set-ScheduledJobOption -DoNotAllowDemandStart

# Change RunAs account.
PS C:\>  Set-ScheduledJob -Name "Test" -Credential (Get-Credential)

# Change the Action
PS C:\>  Get-ScheduledJob | Set-ScheduledJob -FilePath "C:\Scripts\RunTask.ps1"
```

Receiving Results

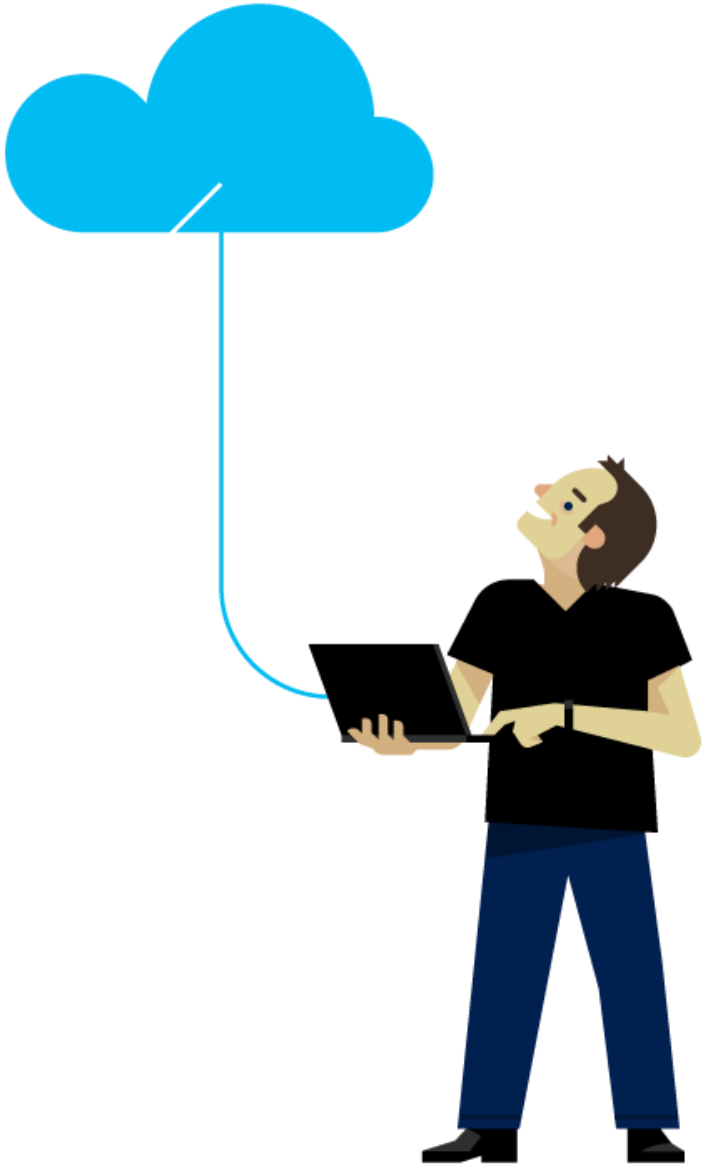
- Job results are received using **Receive-Job**
  - Key advantage over Scheduled Tasks which cannot return data with **Receive-Job**
- PSScheduledJob module is loaded when **Get-Job** is ran first.
- If not, it needs to be imported to be able to use **Receive-Job**
- Results are saved to disk
- **-keep** parameter not required, open a new PowerShell session and re-receive results

```
# Using the ScheduledJob object, output can now be retrieved as well.
PS C:\>  Get-Job -Name Test | Receive-Job
VERBOSE: Performing the operation "Restart-Service" on target "DHCP Client...
WARNING: Waiting for service 'DHCP Client (DHCP)' to stop...
WARNING: Waiting for service 'DHCP Client (DHCP)' to stop...
WARNING: Waiting for service 'DHCP Client (DHCP)' to stop...
```

Demonstration

Scheduled Jobs

Questions?



Scheduled Tasks

Scheduled Tasks vs. Scheduled Jobs

Differences:

- Cannot be queried in real-time like ScheduledJobs (**Receive-Job**)
- Lack of native ScriptBlock support
- Ability to run as built-in accounts (ScheduledJobs cannot)

Similarities:

- Both are viewable in **schtasks.exe** GUI
- Both involve creation of: Job Options, Triggers, Credentials, Actions before Creation
- Both have methods for immediate invocation
- Both have variety of cmdlets for management

Registering a Scheduled Task

- Register-ScheduledTask is the primary cmdlet for task creation

Parameter	Source Data Example
-TaskName	”Test”
-Principal	New-ScheduledTaskPrincipal -UserId ‘System’ -LogonType ServiceAccount
-Settings	New-ScheduledTaskSettingsSet -RestartCount 3
-Trigger	New-ScheduledTaskTrigger -Daily -At ‘9am’
-Action	New-ScheduledTaskAction -Execute ‘PowerShell.exe’ -Argument ` ’-NoProfile -WindowStyle Hidden -Command "&{Restart-Service DHCP}”’

Scheduled Task Triggers

```
PS C:\> $trigger = New-ScheduledTaskTrigger -Daily -At "04:45:00PM"
```

Method	StartTime	Interval of Repetition	EndTime
-Once	-At <datetime>	-RepetitionInterval <timespan>	-RepetitionDuration <timespan> -RepeatIndefinitely <switch>
-Weekly	-At <datetime>	-WeeksInterval <int> -DaysOfWeek <DayOfWeek[]>	# Repeats Indefinitely
-Daily	-At <datetime>	-DaysInterval <int>	# Repeats Indefinitely
-AtStartup	# On Event	# At Event Occurrence	# Repeats Indefinitely
-AtLogon	# On Event	# At Event Occurrence	# Repeats Indefinitely

Scheduled Task Settings

- Setting Objects are created by New-ScheduledTaskSettingsSet

```
PS C:\>  $settings = New-ScheduledTaskSettingsSet –WakeToRun
```

- The resultant objects are then passed to –Setting parameter of Register-ScheduledTask

```
 Register-ScheduledTask -TaskName "MyTask" `
-Trigger $Time -Settings $settings -Action $RunThisAction
```

Commonly used Settings

Setting	Description
`-WakeToRun`	Wakes the computer to run the task.
`-AllowStartIfOnBatteries` `-DontStopIfGoingOnBatteries`	Control Scheduled Job Execution on battery power.
`-IdleWaitTimeout` and `-IdleDuration` `-RunOnlyIfIdle` and `–DontStopOnIdleEnd` and `–RestartOnIdle`	Control Idle Settings and Timeouts.
`-Hidden` or `-Disabled`	Disables or Hides the ScheduledTask
`-RestartCount -RestartInterval`	Indicates whether the Task should retry on failure.

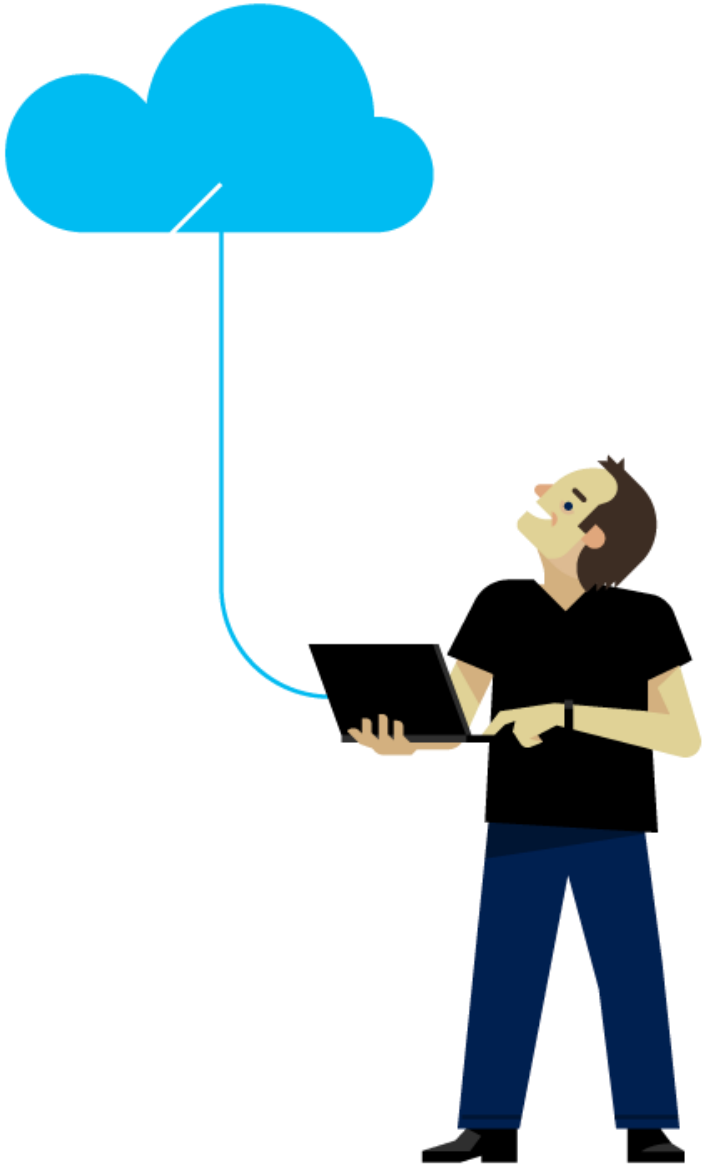
- 20+ settings, for a full list;

```
New-ScheduledTaskSettingsSet | Get-Member -MemberType Property
```

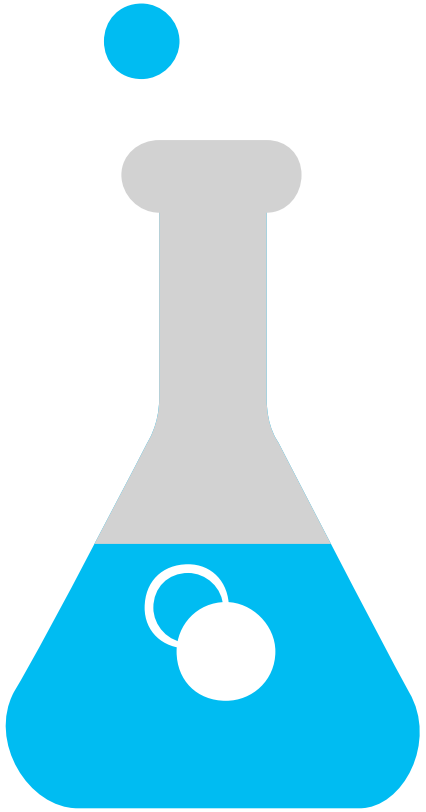
Demonstration

Scheduled Tasks

Questions?



PowerShell Jobs



LAB