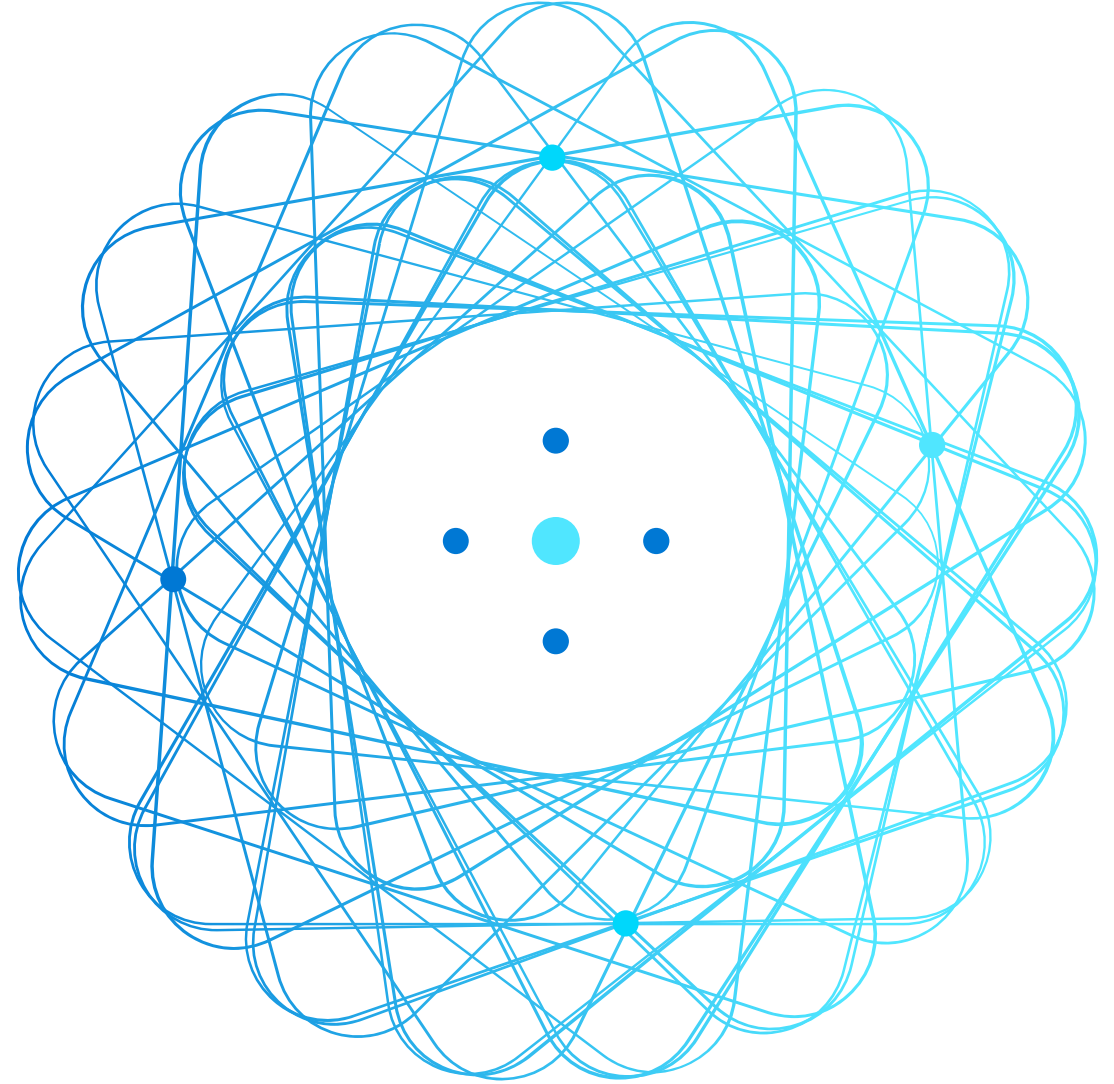


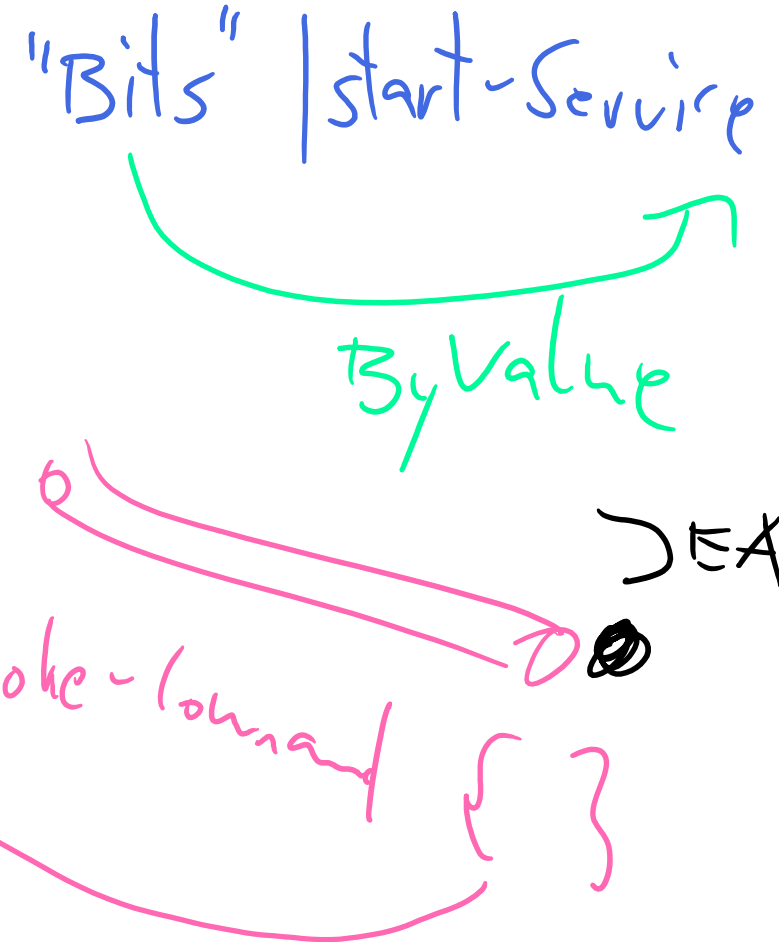
AZ-040

Automating Administration with PowerShell



Course outline

- Module 1: Getting started with Windows PowerShell ✓
- Module 2: Windows PowerShell for local systems administration ✓
- Module 3: Working with the Windows PowerShell pipeline
- Module 4: Using PSProviders and PSDrives HKCU:
- Module 5: Querying management information by using CIM and WMI
- Module 6: Working with variables, arrays, and hash tables
- Module 7: Windows PowerShell scripting
- Module 8: Administering remote computers with Windows PowerShell
- Module 9: Managing Azure resources with PowerShell
- Module 10: Managing Microsoft 365 services with PowerShell
- Module 11: Using background jobs and scheduled jobs



Module 11: Using background jobs and scheduled jobs

Windows

Module overview

If you use the Windows PowerShell console to start a task whose execution takes a long time, you will not be able to run other commands from the same console until the task completes. Background jobs allow you to use the same console to work with other tasks while the long-running one is active. In this module, you'll learn about the Windows PowerShell jobs.

Lessons:

- Lesson 1: Use background jobs
- Lesson 2: Use scheduled jobs

Lesson 1: Use background jobs

PowerShell Profiles

Get-ExecutionPolicy

\$profile.CurrentUserAllHosts

Lesson 1 overview

When you run a command as a background job, Windows PowerShell performs the task asynchronously in its own thread. Even if the job takes a long time to complete, you regain access to the PowerShell prompt immediately.

This allows you to run other commands while the job runs in the background.

Topics:

- What are background jobs?
- Starting jobs
- Managing jobs
- Retrieving results for running jobs
- Demonstration: Using background jobs

What are background jobs?

- Run commands in the background
- Store command results in memory for retrieval
- Three basic job types:
 - Local
 - Remoting
 - CIM/WMI
- Each job type has different characteristics

Linux: ls &

Starting jobs

- Local jobs:

```
Start-Job -ScriptBlock { Dir }
```

- Remoting jobs:

```
Invoke-Command -ScriptBlock { Get-Service } -ComputerName LON-DC1 -AsJob
```

- CIM/WMI jobs:

```
Start-Job -ScriptBlock {Get-CimInstance -ClassName Win32_ComputerSystem}  
Get-WmiObject -Class Win32_BIOS-ComputerName LON-DC1 -AsJob
```


Managing jobs

Use the following commands:

- **Get-Job**

- Add *-ID* to retrieve a specific job by ID
- Add *-Name* to retrieve a specific job by name
- To get a list of child jobs, use the following syntax:

```
Get-Job -ID <parent_ID> -IncludeChildJobs
```

- **Stop-Job**

- **Remove-Job**

- **Wait-Job**

Retrieving results for running jobs

- Use **Receive-Job**:
 - Pipe jobs to it to specify jobs
 - Use *-ID* to specify by job ID
 - Use *-Name* to specify by job name
- Add *-Keep* to retain a copy of the results in memory. Otherwise, results aren't retained in memory
- Receiving results from a parent job will result in receiving results from all child jobs

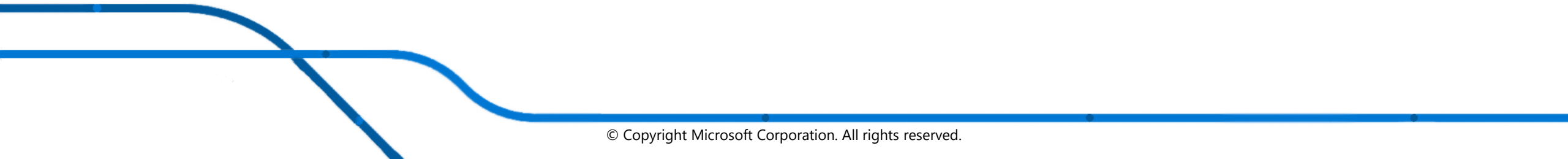
Demonstration: Using background jobs

In this demonstration, you'll learn how to create and manage local and remoting jobs.

1. Start local and remoting jobs.
2. Manage jobs.
3. Receive job results.



Lesson 2: Use scheduled jobs




Lesson 2 overview

This lesson covers scheduled jobs, which are similar to background jobs because they run asynchronously in the background. In Windows PowerShell, scheduled jobs are essentially scheduled tasks that follow all the same rules for actions, triggers, and other features, and run Windows PowerShell scripts by design.

Topics:

- Running Windows PowerShell scripts as scheduled tasks
- Demonstration: Using a Windows PowerShell script as a scheduled task
- What are scheduled jobs?
- Job options and job triggers
- Creating a scheduled job
- Retrieving scheduled job results
- Demonstration: Using scheduled jobs

Running Windows PowerShell scripts as scheduled tasks

- Windows PowerShell scripts can run in **Task Scheduler**:
 - Can use all options of **Task Scheduler** in a graphical user interface
- A scheduled task consists of:
 - **Action** —
 - **Principal** 
 - **Trigger** 2:30
 - **Additional settings** ps Admin
- To review the complete list of commands, run the following command:

```
Get-Command -Module ScheduledTasks
```

Demonstration: Using a Windows PowerShell script as a scheduled task

In this demonstration, you'll learn how to create and run a Windows PowerShell script as a scheduled task.



What are scheduled jobs?

- Scheduled jobs are a cross between background jobs and **Task Scheduler** tasks
- They have three components:
 - Job definition
 - Job options
 - Job triggers
- To review all scheduled job commands, use:

```
Get-Command -Module PSScheduledJob
```


Job options and job triggers

- Use **New-ScheduledJobOption** to create an option object.
- Parameters correspond to options in the **Task Scheduler** GUI and include:
 - *-RequireNetwork*
 - *-RunElevated*
 - *-WakeToRun*
 - *-HideInTaskScheduler*
- Use **New-JobTrigger** to create a trigger object.
- Five basic types of triggers:
 - *-Once*
 - *-Weekly*
 - *-Daily*
 - *-AtLogOn*
 - *-AtStartup*

Creating a scheduled job

- Use **Register-ScheduledJob** to create and register a new scheduled job.
- Windows PowerShell creates the job definition XML file on disk.
- Parameters include:
 - *-Name*
 - *-ScriptBlock* or *-FilePath*
 - *-Credential*
 - *-MaxResultCount*
 - *-ScheduledJobOption* (job option object)
 - *-Trigger* (job trigger object)

Retrieving scheduled job results

- Run **Get-Job** to review a list of **PSScheduledJob** jobs:
 - Each represents a running of the scheduled job.
 - Provides access to the job results.
- Run **Receive-Job** to retrieve results.
- Run **Remove-Job** to delete results and the job object.

Demonstration: Using scheduled jobs

In this demonstration, you'll learn how to create, run, and retrieve the results from a scheduled job.



Lab: Jobs management with PowerShell

Exercise 1: Starting and managing jobs



Exercise 2: Creating a scheduled job



Sign-in information for the exercise(s):

Virtual machines:

- **AZ-040T00A-LON-DC1**
- **AZ-040T00A-LON-SVR1**
- **AZ-040T00A-LON-CL1**

Username: **Adatum\Administrator**

Password: **Pa55w.rd**

Lab scenario

Background jobs provide a useful way to run multiple commands simultaneously and long-running commands in the background. In this lab, you'll learn to create and manage two of the three basic kinds of jobs.

You'll create and configure two scheduled jobs. You'll also create a scheduled task using a Windows PowerShell script that searches for and removes disabled accounts from a certain security group.

Lab-review questions



Get-CIMInstance doesn't have an *-AsJob* parameter. Why? How would you use **Get-CimInstance** in a job?



Is it possible to create a scheduled job without creating a job option object?

Lab-review answers



Get-CIMInstance doesn't have an *-AsJob* parameter. Why? How would you use **Get-CimInstance** in a job?

Microsoft is moving toward a standardized use pattern where **Invoke-Command** is used to run commands on remote computers and to manage that process in the background. You can use **Get-CIMInstance** inside the script block of **Invoke-Command** or inside the script block for **Start-Job**.



Is it possible to create a scheduled job without creating a job option object?

Yes. The *-ScheduledJobOption* parameter of **Register-ScheduledJob** is optional. You need to create a scheduled job option only if you require one of its features.