Introduction to PowerShell Background Jobs

Jeff Hicks | https://jdhitsolutions.github.io

Getting Started with PowerShell Background Jobs¶

Learning Objectives¶

- Understand the concept of background jobs in PowerShell.
- Learn how to start a background job using `Start-Job`.
- Understand how to interact with running background jobs and use `Get-Job` to monitor their status.
- Learn how to retrieve the results of background jobs with `Receive-Job`.
- Understand the limitations and considerations when using background jobs.

Introduction¶

- What are background jobs?
- Why use background jobs?
- What do I need?

Starting a Job¶

- Use `Start-Job` to run a command in the background.
- Specify a script block
- Specify a PowerShell script file
- You can give the job a name
- You can pass parameters to the job

```
$j = Start-Job -scriptblock {Get-Process | Sort-Object -Property WS -Descending}
```

This creates a job object.

```
Id     Name       PSJobTypeName   State         HasMoreData   Location     Command
--     ----       -------------   -----         -----------   --------     -------
10     Job10      BackgroundJob   Completed     True          localhost    Get-Process | Sort-Objec...
```

- The job ID is automatically generated.
- Do not assume it will start at `1`

Getting a Job¶

- Use `Get-Job`

```
PS C:\> Get-Job
```

```
Id     Name       PSJobTypeName   State       HasMoreData     Location    Command
--     ----       -------------   -----      -----------     --------    -------
8      Job8       BackgroundJob   Stopped    True            localhost    Get-Process | Sort-Objec...
```

```
10     Job10      BackgroundJob   Completed  True            localhost    Get-Process | Sort-Objec...
12     Job12      BackgroundJob   Running    True            localhost    dir D:\OneDrive\ -Recur...
```

- You can get jobs by ID, Name, or State.

```
PS C:\> Get-Job -State Completed

Id     Name       PSJobTypeName   State       HasMoreData    Location      Command
--     ----       -------------   -----       -----------    --------      ------
10     Job10      BackgroundJob   Completed   True           localhost     Get-Process | Sort-Objec...
12     Job12      BackgroundJob   Completed   True           localhost     dir D:\OneDrive\ -Recur...
```

Job Architecture¶

- Parent/Executive job
  - This is what we see with `Get-Job`
  - Manage this job with the job cmdlets
- One or more child jobs
  - Does the actual work
  - You don't have to do anything with these jobs
  - Generally can ignore unless troubleshooting
- The job status is based on the child job statuses

```
PS C:\> Get-Job -id 10 -IncludeChildJob

Id     Name      PSJobTypeName   State         HasMoreData    Location     Command
--     ----      -------------   -----         -----------    --------     -------
10     Job10     BackgroundJob   Completed     True           localhost    Get-Process | Sort-Objec...
11     Job11                     Completed     True           localhost    Get-Process | Sort-Objec...
```

There's always more than what you by default in PowerShell.

```
PS C:\> Get-Job 10 | Select *

State         : Completed
HasMoreData   : True
StatusMessage :
Location      : localhost
Command       : Get-Process | Sort-Object -Property WS -Descending
JobStateInfo  : Completed
Finished      : System.Threading.ManualResetEvent
InstanceId    : 813ca357-e7fd-4043-8c2b-9e512874f2e1
Id            : 10
Name          : Job10
ChildJobs     : {Job11}
PSBeginTime   : 4/5/2025 1:18:04 PM
PSEndTime     : 4/5/2025 1:20:03 PM
PSJobTypeName : BackgroundJob
Output        : {}
Error         : {}
Progress      : {}
Verbose       : {}
Debug         : {}
Warning       : {}
Information   : {}
```

- **Jobs do not persist across PowerShell sessions**.

Getting Job Results¶

- Any output is stored in the job object
- `HasMoreData`

```
PS C:\> Get-Job 10 | Select-Object Name,State,HasMoreData,Command

Name  State     HasMoreData Command
----  -----     ----------- -------
Job10 Completed        True Get-Process | Sort-Object -Property WS -Descending
```

- Get results with `Receive-Job`
- Results are removed unless you use `Keep`
- Results are the same as if you had run the command directly

```
PS C:\> $r = Receive-Job 10 -Keep
PS C:\> $r.count
360
PS C:\> $r | Select-Object -first 5

 NPM(K)    PM(M)      WS(M)     CPU(s)      Id  SI ProcessName
 ------    -----      -----     ------      --  -- -----------
      0     2.71     992.97      17.11    4584   0 Memory Compression
     69   609.79     608.44      74.00   29992   1 Code
     62   420.63     448.48     329.94   19308   1 Code
    189   269.05     416.88      37.61   23856   1 pwsh
    129   378.00     368.48     153.81   15752   1 thunderbird
```

- Use `Remove-Job` if you want to clean up your session.

```
Remove-Job -State Completed
```

Cmdlet Integration¶

- `Invoke-Command`
- `Foreach-Object`

```
#Discover
Get-Command -ParameterName AsJob
```

Example¶

```
PS C:\> $p = "c:\work","C:\scripts","C:\presentations" |
Foreach-Object -parallel {
    Get-ChildItem $_ -file -recurse | Measure-object length -sum
} -AsJob
PS C:\> $p | Get-Job -IncludeChildJob

Id     Name            PSJobTypeName   State         HasMoreData     Location             Command
--     ----            -------------   -----         -----------     --------             -------
14     Job14           PSTaskJob       Completed     True            PowerShell           ...
15     Job15           PSTaskChildJob  Completed     True            PowerShell           ...
16     Job16           PSTaskChildJob  Completed     True            PowerShell           ...
17     Job17           PSTaskChildJob  Completed     True            PowerShell           ...
```

Other Job Options¶

- `Stop-Job` to stop a running job

  - There may still be results to receive!
- `Wait-Job` to wait for a job to finish
  - You can use `-Timeout` to limit the wait time
  - This is a useful way to wait for a job to finish before continuing with a script
- Start any PowerShell command as a job by appending `&`
  - `dir d:\ -Recurse &`
  - Creates a background job
  - Cannot specify a job name or pass parameters

Thread Jobs¶

- A thread job is PowerShell job running in a separate *thread* instead of a child process.
- Requires less overhead than a background job.
- Requires the `Microsoft.PowerShell.ThreadJob` module.
- Scriptblock or PowerShell script File
- Can give the job a name
- Can pass parameters

```
PS C:\> $t = Start-ThreadJob { Get-Process | Sort WS -Descending | Select -First 10}
PS C:\> $t

Id     Name       PSJobTypeName   State       HasMoreData    Location       Command
--     ----       -------------   -----       -----------   --------        -------
3      Job3       ThreadJob       Completed   True           PowerShell     Get-Process | Sort WS -...

PS C:\> Receive-Job 3 -Keep

 NPM(K)    PM(M)      WS(M)     CPU(s)      Id  SI ProcessName
 ------    -----      -----     ------      --  -- -----------
      0     4.00   1,650.32      26.17    4584   0 Memory Compression
    101 1,489.49   1,562.47      39.47   40052   1 pwsh
    195 1,107.50   1,132.65      67.45   23856   1 pwsh
     64   736.76     585.64     244.80   29992   1 Code
     60   494.60     445.12     827.89   19308   1 Code
    192   249.71     334.98     117.20   24128   1 pwsh
    127   369.97     319.31     173.98   15752   1 thunderbird
  11965   134.10     270.97     348.66    3924   0 ekrn
    146   335.17     261.62     103.28   10572   1 explorer
    178   339.22     261.41     161.00   32420   1 Dropbox
```

Live Demo¶

- Jobs in Action

Read All About It¶

- help about_jobs
- about_Job_Details
- about_Remote_Jobs
- about_Thread_Jobs

Questions¶

- What else do you want to know?

Your Turn¶

- Create a background job to get the total size of all files in your TEMP directory, including the largest file and the smallest file.
  - Create the job with a name of `TempSize`
  - You can use `$ENV:Temp` PSDrive reference
  - Read help for `Measure-Object`
- View the job
- Receive the results

Solution

```
#start with a clean slate
Get-Job | Remove-Job -force
Start-Job -name TempSize -ScriptBlock {
    dir $env:TEMP -Recurse | Measure-Object -Property Length -sum -Maximum -minimum
}
Get-Job -name TempSize
Wait-Job TempSize
Receive-Job TempSize -Keep | Select-Object Count, Sum, Maximum, Minimum
```