

# PowerShell Cardiff User Group

## Ways to run code in parallel with little effort

with Daniel Krebs

12 March 2019

@GetPSugUK



# Inspiration

A friend, Martynas Valkunas, asked me about how to execute code in parallel in PowerShell. He was developing an clean-up script for Azure Resource Groups.



# What is available to us?

- Start-Process cmdlet without –Wait switch
- Invoke-Command
- PowerShell Workflow
- PowerShell Background Jobs
- 3<sup>rd</sup>-party modules (PowerShell Gallery):
  - **PSParallel (only contains Invoke-Parallel cmdlet)**
  - **PoshRSJobs**
  - ThreadJob
  - SplitPipeline
  - PForEach
  - Jojoba (dependency on PoshRSJobs)



# Basic anatomy of PowerShell host process

Host Process (PowerShell.exe & others)

Runspace Pool (Thread Pool)

Runspace (Thread)

Session

- Automatic session variables
- Environment variables
- Loaded functions, modules, providers, etc.



# Start-Process

```
$data | ForEach-Object {  
    Start-Process `  
        -FilePath 'powershell.exe' `  
        -ArgumentList @(  
            '-File', 'C:\Scripts\Azure\ResourceGroupJanitor.ps1',  
            '-Subscription', $PSItem['SubscriptionId'],  
            '-ResourceGroup', $PSItem['ResourceGroupName']  
        ) `  
        -WindowStyle Hidden `  
        # -Wait  
    }  
}
```



# Invoke-Command

```
$minDays = 35
```

```
Invoke-Command `
    -ComputerName $serverList `
    -ThrottleLimit 365 `
    -ScriptBlock {
        param (
            [Int]
            $MinimumDays
        )
        # ...
    } `
    -ArgumentList $minDays
```

```
$minDays = 35
```

```
Invoke-Command `
    -ComputerName $serverList `
    -ThrottleLimit 365 `
    -FilePath 'C:\Scripts\ProfileCleanup.ps1'
    -ArgumentList $minDays
```



# Windows PowerShell Workflow – parallel {}

```
workflow Get-InventoryData {  
    parallel {  
        Get-CimInstance -ClassName Win32_ComputerSystem  
        Get-CimInstance -ClassName Win32_BIOS  
    }  
}  
  
Get-InventoryData -PSComputerName 'srv01', 'srv02', 'srv03'
```



# Windows PowerShell Workflow – foreach –parallel

```
workflow Get-InventoryData {  
    param (  
        [String[]]  
        $ComputerName  
    )  
  
    foreach -parallel ($Name in $ComputerName) {  
        Get-WmiObject -Class Win32_BIOS -PSComputerName $Name  
        Get-WmiObject -Class Win32_ComputerSystem -PSComputerName $Name  
    }  
}
```





# Windows PowerShell Workflow – sequence {

```
workflow Get-InventoryData {  
    parallel {  
        Get-CimInstance -ClassName Win32_ComputerSystem  
        Get-CimInstance -ClassName Win32_BIOS  
  
        sequence {  
            Get-Process  
            Get-Service  
        }  
    }  
}  
  
Get-InventoryData -PSComputerName 'srv01', 'srv02', 'srv03'
```



# Windows PowerShell Workflow – InlineScript {}

```
workflow InvokeMe {  
  
    # Execute any standard PowerShell code  
    InlineScript {  
        | Get-AWSRegion  
    }  
  
    Get-ChildItem -Path 'C:\Users\' -Directory:$true |  
        | ForEach-Object {  
            | InlineScript {  
                | # Does not work inside ForEach-Object  
            }  
        }  
    }  
}
```



# PowerShell Gallery



PowerShell Gallery

Packages

Publish

Statistics

Documentation

Sign in

runspace



Filter By

Displaying results 1 - 11.

Sort By Relevance ▼

## Package Types

- ☒ Module
- ☒ Script

## Categories

- ☐ Cmdlet
- ☐ DSC Resource
- ☐ Function



## PoshRSJob

Module

By: [boe.prox](#) | 30,141 downloads | Last Updated: 24/02/2018 | Latest Version: 1.7.4.4

Module designed to use PowerShell runspaces to create jobs that allow throttling and quicker execution of commands

Tags

PoshRSJob

Runspace

RunspacePool

Linux

PowerShellCore

RSJob



# PowerShell Background Jobs – Start-Job ...

```
Get-ChildItem -Path 'C:\Users\' -Directory |  
    ForEach-Object {  
        Start-Job  
            -Name $PSItem.Name `  
            -ScriptBlock {  
                param (  
                    [String]  
                    $UserPath  
                )  
                # Do work  
            } -ArgumentList $PSItem.FullName  
    }
```

```
$functionName = 'Function:Start-Cleanup'  
$scriptBlock = Get-Item $functionName |  
    Select-Object -ExpandProperty ScriptBlock  
  
Get-ChildItem -Path 'C:\Users\' -Directory |  
    ForEach-Object {  
        Start-Job -Name $PSItem.Name `  
            -ScriptBlock $scriptBlock `  
            -ArgumentList $PSItem.FullName  
    }
```



# PowerShell Background Jobs - Cmdlets

```
... = ... | Start-Job ... | Wait-Job | Receive-Job  
... = ... | Start-Job ... | Wait-Job | Receive-Job -Keep  
|  
# Background Job cmdlets  
Debug-Job  
Get-Job  
Receive-Job  
Remove-Job  
Resume-Job  
Start-Job  
Stop-Job  
Suspend-Job  
Wait-Job
```



# PoshRSJobs – Start-RSJob ...

```
Get-ChildItem -Path 'C:\Users\' -Directory |  
    ForEach-Object {  
        Start-RSJob  
            -Name $PSItem.Name `  
            -ScriptBlock {  
                param (  
                    [String]  
                    $UserPath  
                )  
                # Do work  
            } `br/>            -ArgumentList $PSItem.FullName `br/>            -Throttle 32 # PoshRSJobs feature  
    }
```

```
$functionName = 'Function:Start-Cleanup'  
$scriptBlock = Get-Item $functionName |  
    Select-Object -ExpandProperty ScriptBlock  
  
Get-ChildItem -Path 'C:\Users\' -Directory |  
    ForEach-Object {  
        Start-RSJob -Name $PSItem.Name `br/>            -ScriptBlock $scriptBlock `br/>            -ArgumentList $PSItem.FullName `br/>            -Throttle 13  
    }
```



# PoshRSJobs - Cmdlets

```
... = ... | Start-RSJob ... | Wait-RSJob | Receive-RSJob
```

```
... = ... | Start-RSJob ... | Wait-RSJob | Receive-RSJob -Keep
```

```
# PoshRSJobs cmdlets
```

```
Get-Job
```

```
Receive-Job
```

```
Remove-Job
```

```
Start-Job
```

```
Stop-Job
```

```
Wait-Job
```



# Invoke-Parallel

```
Get-ChildItem -Path 'C:\Users\' -Directory |  
    Invoke-Parallel `  
        -ScriptBlock {  
            param (  
                [DirectoryInfo]  
                $Directory  
            )  
            # Do work  
        } -ThrottleLimit 128
```

```
$functionName = 'Function:Start-Cleanup'  
$scriptBlock = Get-Item $functionName |  
    Select-Object -ExpandProperty ScriptBlock  
  
Get-ChildItem -Path 'C:\Users\' -Directory |  
    Invoke-Parallel `  
        -ScriptBlock $scriptBlock `  
        -ThrottleLimit 128
```





# Thank you / Questions ?!?



JOIN US ON SLACK 



<https://slofile.com/slack/get-psuguk>