PowerShell 2.0 Background Jobs

Jeffery Hicks

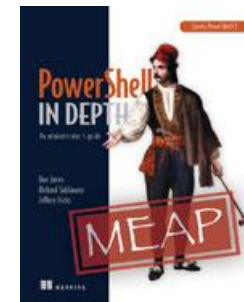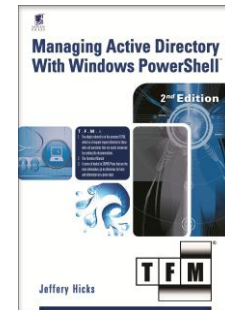**Windows PowerShell MVP**

jhicks@jdhitsolutions.com

AGENDA

- What is a PowerShell job?
- Job Requirements
- Creating local jobs
- Creating remote jobs
- Managing Jobs
- Working with Job Results
- Troubleshooting Jobs
- Q&A

WHO AM I?

- Windows PowerShell MVP
- PowerShell Author
  - Windows PowerShell 2.0: TFM (with Don Jones)
  - Managing Active Directory with Windows PowerShell: TFM 2nd Ed.
- IT trainer and consultant
- http://jdhitsolutions.com/blog
- http://twitter.com/jeffhicks

A NOTE...

- All demos will be made available
- Demos are written mostly as one-liners.
- Focus on results not language
- I'm covering PowerShell 2.0. PowerShell 3.0 adds even more fun and functionality!

http://jdhitsolutions.com/blog

WHAT IS A JOB?

- The standard PowerShell console is single threaded

- Running a command at the prompt "blocks" PowerShell until complete

- A background job "pushes" a PowerShell command to the "background" via a new runspace.

WHAT IS A JOB?

- Jobs consist of one parent or executive job and one or more child jobs

- When you are ready, you can retrieve the results, if any

- There is no job notification process

http://jdhitsolutions.com/blog

JOB REQUIREMENTS

- PowerShell 2.0
- PowerShell Remoting (WinRM) configured, even if running jobs locally.
- PowerShell Remoting enabled and configured on remote computers for remote jobs.

http://jdhitsolutions.com/blog

CREATING LOCAL JOBS

- Start-Job
  - Script block
  - PowerShell script
  - Script blocks and scripts can accept parameters
- Cmdlet –AsJob parameter
  - Get-WMIObject
  - Invoke-Command
- Optionally, you can define a job name and/or save job to a variable
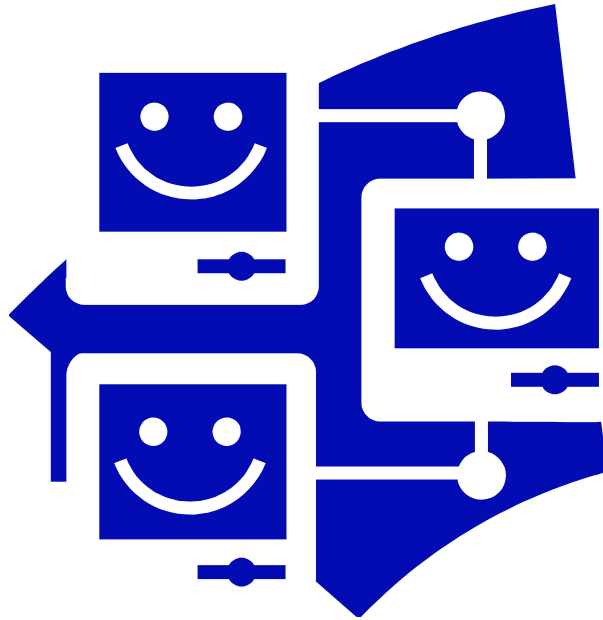
DEMO #1

CREATING REMOTE JOBS

- Use the Invoke-Command cmdlet
  - Computername
  - AsJob
  - Optional Credential
- You can create background jobs on multiple remote machines
  - Specify computer names
  - Specify a group of PSSessions
- Job queue is created locally but command runs remotely
- Results returned to local job queue

Demo #2

MANAGING JOBS

- Use Get-Job to retrieve one or more jobs
- Use Wait-Job to wait for a given job to complete. More useful in a PowerShell script
- Use Stop-Job to terminate a job
- Job queue cache ends when PowerShell session ends
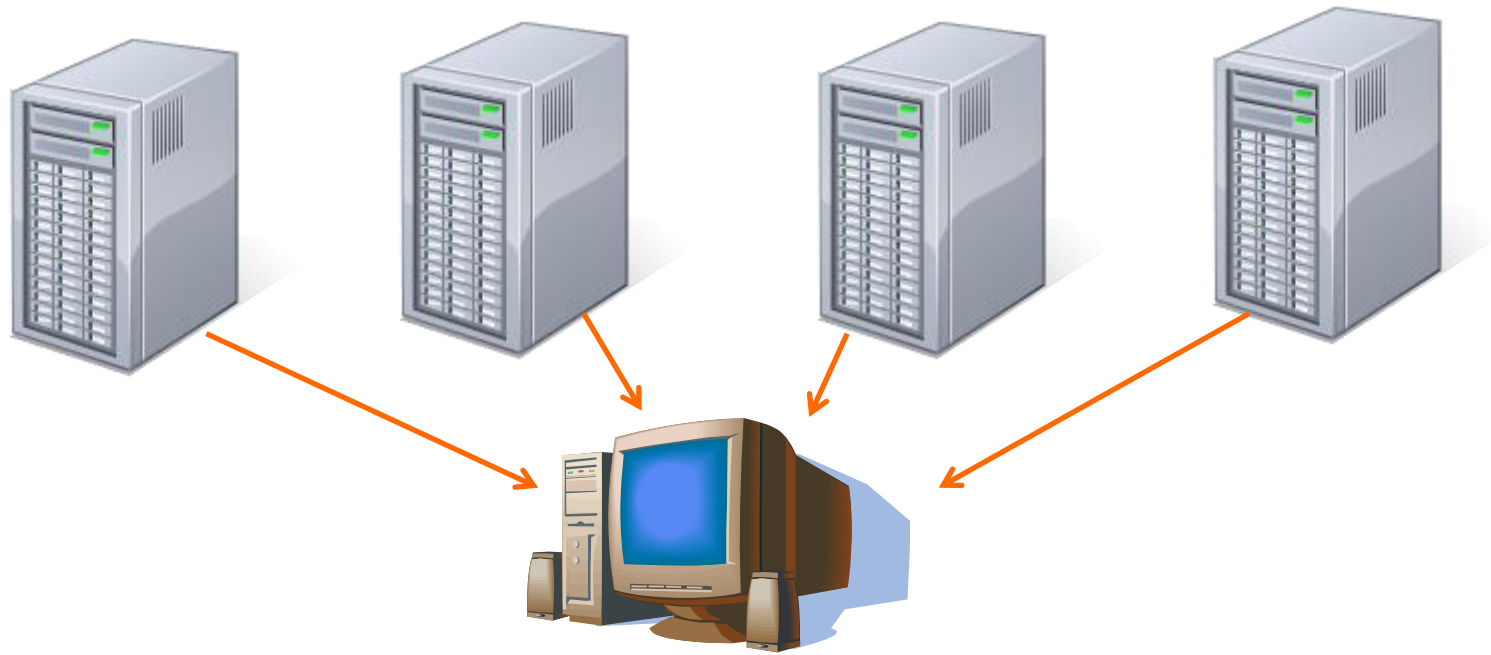- Use Remove-Job to manually clear one or more jobs

http://jdhitsolutions.com/blog

Working with Job Results

- Results stored in a local job queue cache
- Use Receive-Job to get results
- Results cleared unless you use –Keep

```
PS C:\> $data=Receive-Job 3 -keep
```
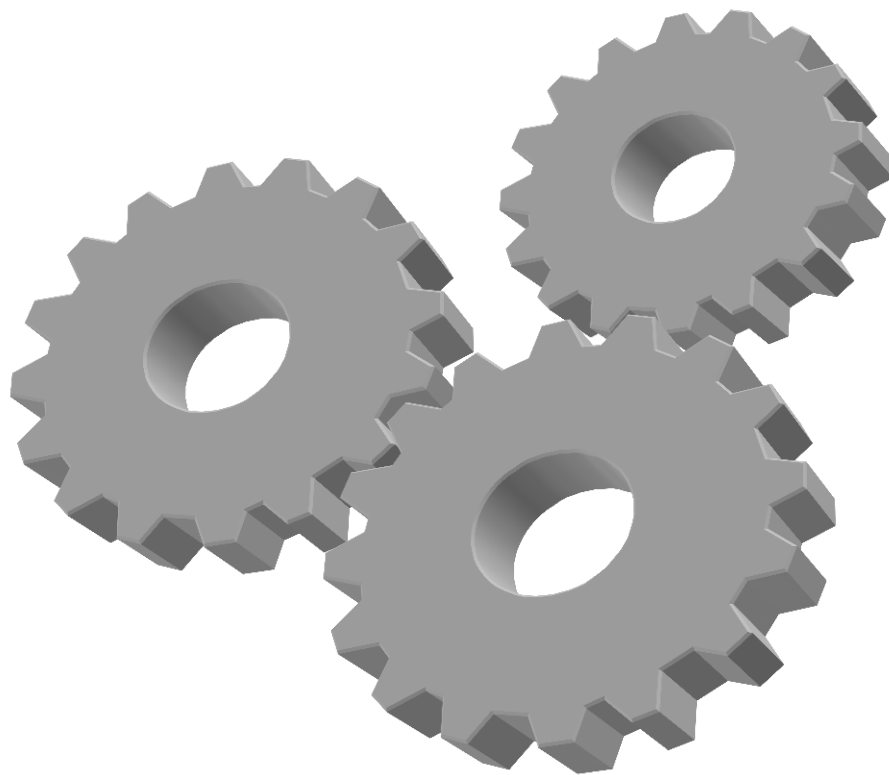
DEMO #3

TROUBLESHOOTING JOBS

- Verify your command runs locally and interactively before using in a job
- Verify WinRM and credentials with Test-WSMan
- Look at the JobStateInfo property on child jobs

```
PS C:\> (get-job 9).childjobs[0].jobstateinfo.reason
Access is denied. (Exception from HRESULT:
0x80070005 (E_ACCESSDENIED))
```

DEMO #4

QUESTIONS

RESOURCES

- Windows PowerShell 2.0: TFM by Don Jones and Jeffery Hicks
- Windows PowerShell in Action 2nd Ed. by Bruce Payette
- Windows PowerShell Cookbook 2nd Ed. by Lee Holmes
- Windows PowerShell Team blog (http://blogs.msdn.com/powershell)
- The Lonely Administrator (http://jdhitsolutions.com/blog)
- Prof. PowerShell (http://mcpmag.com/articles/list/prof-powershell.aspx

http://jdhitsolutions.com/blog

SUMMARY

- Jobs require PowerShell remoting
- Use Start-Job to create local background jobs with either script blocks or script files.
- Look for the –AsJob parameter in cmdlets
- Use Invoke-Command to create jobs for remote systems
- Keep job results
- Use Remove-Job to manually clear jobs
- Look at help and examples for all the job cmdlets
- PowerShell 3.0 is even better!

THANK YOU

- http://jdhitsolutions.com/blog
- jhicks@jdhitsolutions.com

http://jdhitsolutions.com/blog