Learning PowerShell in a Month of Lunches

Chapter 7 – Adding Commands

One Shell to Rule Them All

- How one shell can do everything? - Install the management tools for a given product (the option to install management tools is usually included in a product's Setup menu)

- About product-specific "management shells" - There isn't a separate PowerShell for Exchange or AD; it's all a single shell

- SQL Server 2008 and SQL Server 2008 R2 are exceptions. Their "product-specific" is a specially compiled version of PowerShell that runs only the SQL Server extensions.

Extensions: finding and adding snap-ins

- Proper name for a PowerShell snap-in is PSSnapin
  - Microsoft is moving away from the concept of snap-ins
  - PSSnapin consists of 1 or more DLL files, and additional XML files that contain configuration settings and help text.
- Get a list of available snap-ins by running:
  - Get-PSSnapin -registered
  - Nothing returned on M$ developer VM

Extensions: finding and adding modules

- More self-contained and somewhat easier to distribute.
- Simliar to PSSnapins, require a little more knowledge about them
- Must be loaded in the system path to auto-load
  - get-content env:psmodulepath
  - This is why <TAB> key works
- Run Import-Module and specify the complete path to the module to load a module that is not in the path
- Clear loaded modules
    - Get-Module | Remove-Module
    - Get-Module will also work on remote computer
- List the network modules
    - help *network*
- Help for a specific module
  - help Get-SmbServerNetworkInterface

Command conflicts / removing extensions

- Most PowerShell extensions add a short prefix to the noun portion of their command names tp prevent confusion
    - Get-ADUser or Invoke-Sql Cmd
- If 2 modules have the same cmdlet name then the last one loaded will run
- To prevent this the complete snap-in name must be used
    - MyCoolPowerShellSnapin\Get-User
- Other option is to remove the other module

Non-Windows OS

- PSModulePath environment variable will point elsewhere.
- A lot of the modules already out won't run due to dependencies
- There are modules available that will work on *nix and macOS that will not work on Windows

Playing with a new module

- Goal is to clear the DNS name resolution cache on our computer
- Find the DNS module functions
- help *dns*
- To investigate further load the modules
  - import-module -Name DnsClient
  - Can run Get-Module to validate it is loaded
  - Help or running the command would have loaded the module as well
  - get-command -Module DnsClient
- Find command details
  - help Clear-DnsClientCache
  - help Clear-DnsClientCache -Eamples
- No command line arguments but run with -verbose for fun
  - Clear-DnsClientCache -verbose
  - -verbose works on all commands but may not do anything different

Profile scripts: preloading extensions when the shell starts

- Issue is any snap-ins or modules that are loaded have to be reloaded for a new PowerShell session
- 3 ways to resolve (but only two mentioned)
- Creating a console file – only works with PSSnapins, not modules
    - Creates an XML file that can be used to open PowerShell
- Create a profile script (covered in Chapter 25)
    - Basically script the Add-PSSnapin and Import-Module commands

Getting modules from the internet

- PowerShellGet - search for, download, install, and update modules from online repositories. Similar to *nix repos
- Repo - http://powershellgallery.com
    - Repos have the install command similar to github clone
    - Manual downloads (offline install) available
- Microsoft has nothing to do with validation or endorsing i.e. use at your own risk
    - Does have a warning prompt
- Can setup a private gallery
- Run Register-PSRepository to add the URL of a repository
- Get-PSRepository will display what is set
- Find-Module to find modules in repositories (supports wildcards)
- Install-Module to download and install a module
    - Have to be admin (least on M$ VM)
    - Will prompt to setup NuGet on 1st run (M$ VM)
- Use Update-Module to update it

Common points of confusion

READ THE HELP!!!!
USE -example
USE -full

Lab

- Run the Networking troubleshooting pack
- Commands

```
get-module *trouble* -list
import-module TroubleShootingPack
get-command -Module TroubleShootingPack
help get-troubleshootingpack –full
help Invoke-TroubleshootingPack -full
dir C:\windows\diagnostics\system
pack=get-troubleshootingpack C:\windows\diagnostics\system\
Networking
Invoke-TroubleshootingPack $pack
```

Press <Enter>, 1, 2, then paste

https://www.pluralsight.com/browse/it-ops