Microsoft

PowerShell Basics

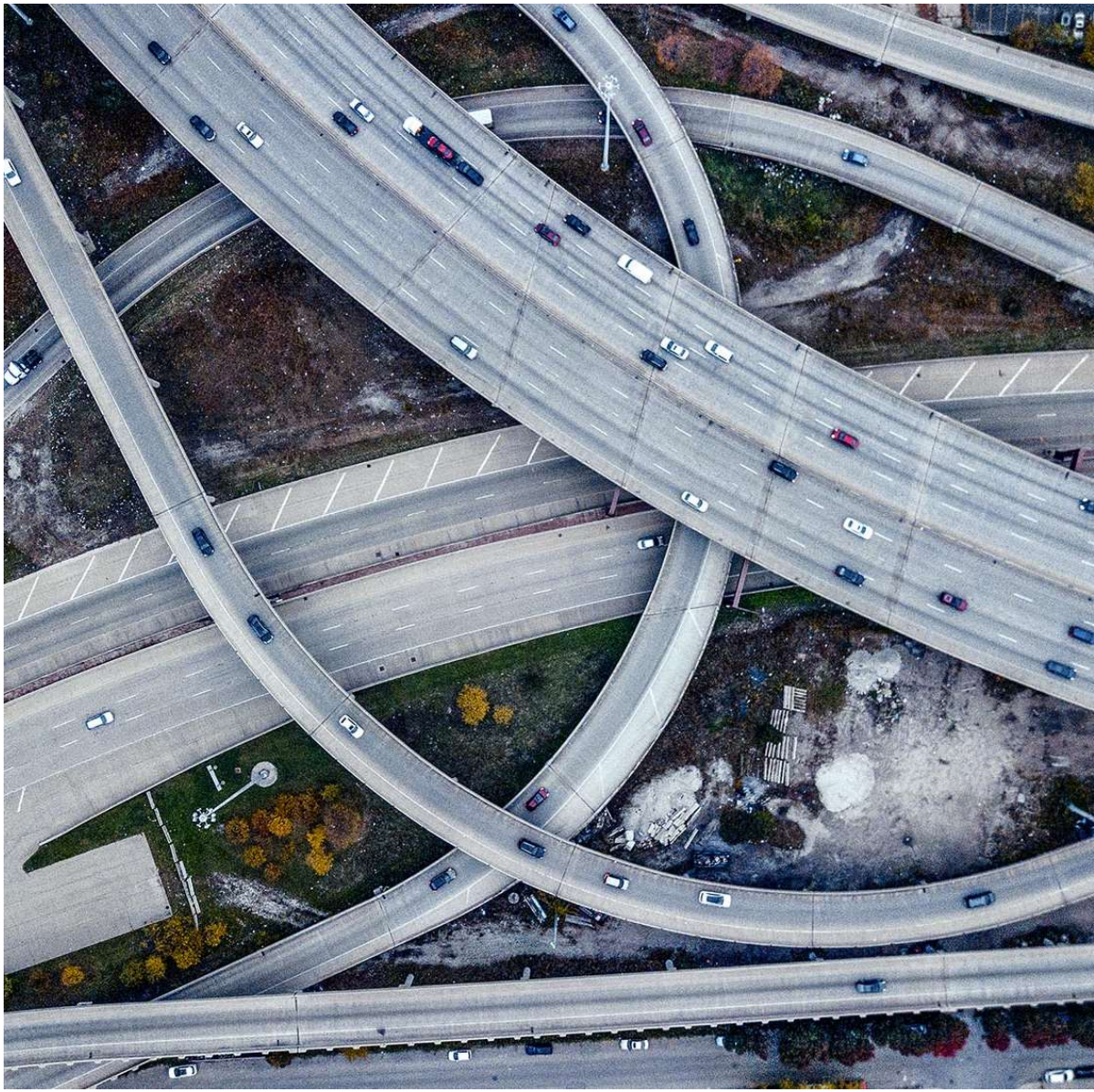
Marco Padilha
Sr Customer Engineer
mapadi@microsoft.com

Recording Notice

This session will be recorded. Your participation in this session serves as your consent to the recording.

Agenda

- Introduction
- ISE
- VSC
- Concepts
- Objects
- Basic Functions
- Script Overview
- PS Modules
- Pipeline
- Code Samples



“I hear and I forget. I see and I remember. I do and I understand”

Confucius

Problem: Different Operating Systems

- No common platform for management
- IT staff must be specialized in each OS
- Too many different scripting/coding options, need single management tool
- No easy way to use the same coding

New mission – digital transformation

- From **ANY** client
- Manage **ANY** server
- Running on **ANY** cloud
- Using **ANY** hypervisor
- With **ANY** storage
- Across **ANY** networking

Value proposition

Why should you use/learn PowerShell?

Because PowerShell is a single point, ubiquitous platform that:

- works with many products/vendors: SAN's, Citrix, cisco, VMware, Windows, Exchange, Oracle, SQL, System Center, and O365 (Delve, Planner, StaffHub, Dynamics, Security and Compliance, Teams, etc.)
- is a single administrative/development environment for heterogenous environments

PowerShell overview

- Task-based command-line shell and scripting language, built on the .NET framework
- Designed especially for system administration
- Helps IT professionals and power users control and automate the administration of several operating systems and applications that run on those operating systems
- Visual Studio free PowerShell plug-in available

Scripting language

- Automation
- Disaster Recovery
- High Availability
- Deployment
- Auditing
- Health Check
- Monitoring
- Reporting
- GUI over PowerShell
- And more…

Semantics

For clarification in this course:

- Windows PowerShell 5.1- will be referred to as **Windows PowerShell**
- PowerShell Core 6.x will be referred to as **PowerShell Core**
- PowerShell 7+ will be referred to as **PowerShell**

Ubiquity – Windows, mac, and Linux

- Common interface and cmdlets
- Runs natively on: Windows, Linux, and macOS X
- Open source on GitHub

Existing PS Users	Mgmt Products	App Developers	Linux Users
Manage Linux and Windows systems from any computer	Single stack to manage anything from anywhere	Framework transforms a small amount of code into rich automation and configuration	Another tool in your toolbox. Optimized for structured data, REST APIs, and object models

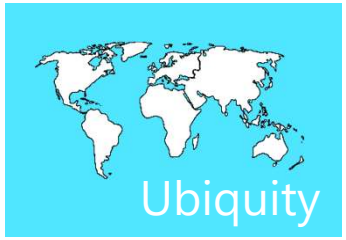
Ubiquity - PowerShell everywhere

- PowerShell and DSC play well with configuration management solutions, ex: Chef, Puppet, Ansible, Chocolatey, etc.
- Windows PowerShell down level to Server 2008 R2 and Windows 7
  - Open-source, cross-platform PowerShell Core 6.0+
  - Available on Windows, Nano, Mac, and Linux
- Based on the .NET CoreCLR
- PowerShell plugin for Visual Studio
  - Integrated terminal
  - Debugging

Cloud Era - Management Landscape

(Scale * complexity) exceeds skill
Cloud scale demands power

Ever-faster solution delivery needed
Pace of change is increasing



Heterogeneous environments are norm
IT spans on-prem, hybrid, and cloud

Azure Cloud Shell

- Interactive, browser-accessible shell
- **shell.azure.com**
- Bash experience available or PowerShell option
- Additional information in **Azure module** of this course

PowerShell differences

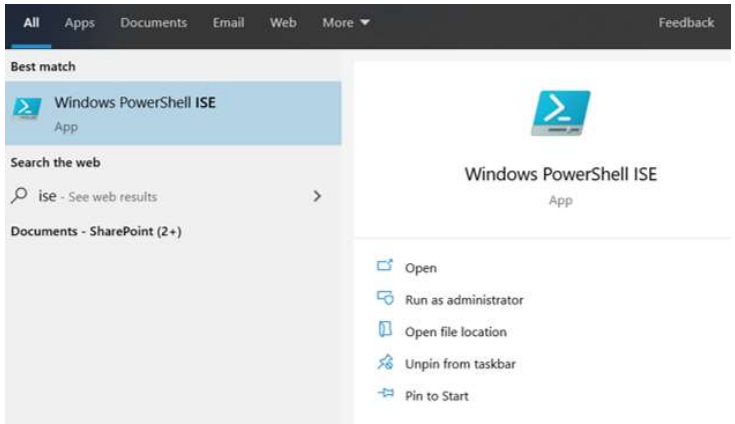
	Windows PowerShell	**PowerShell Core/PowerShell**
Versions	1.0 to 5.1	6.0+
Platforms	Windows only (client and server)	Windows, Mac OS, Linux
Dependency	.Net Framework	.Net Core
Usage	Relies on .Net Framework runtime	Relies on .Net Core runtime
Launched as	powershell.exe	pwsh.exe (Windows), pwsh (Mac and Linux)
$PSVersionTable.PSEdition	Set to Desktop	Set to Core
Update policy	critical bug fixes only	all updates (features, bugs)
Cmdlet count	2322+	1480+

Microsoft

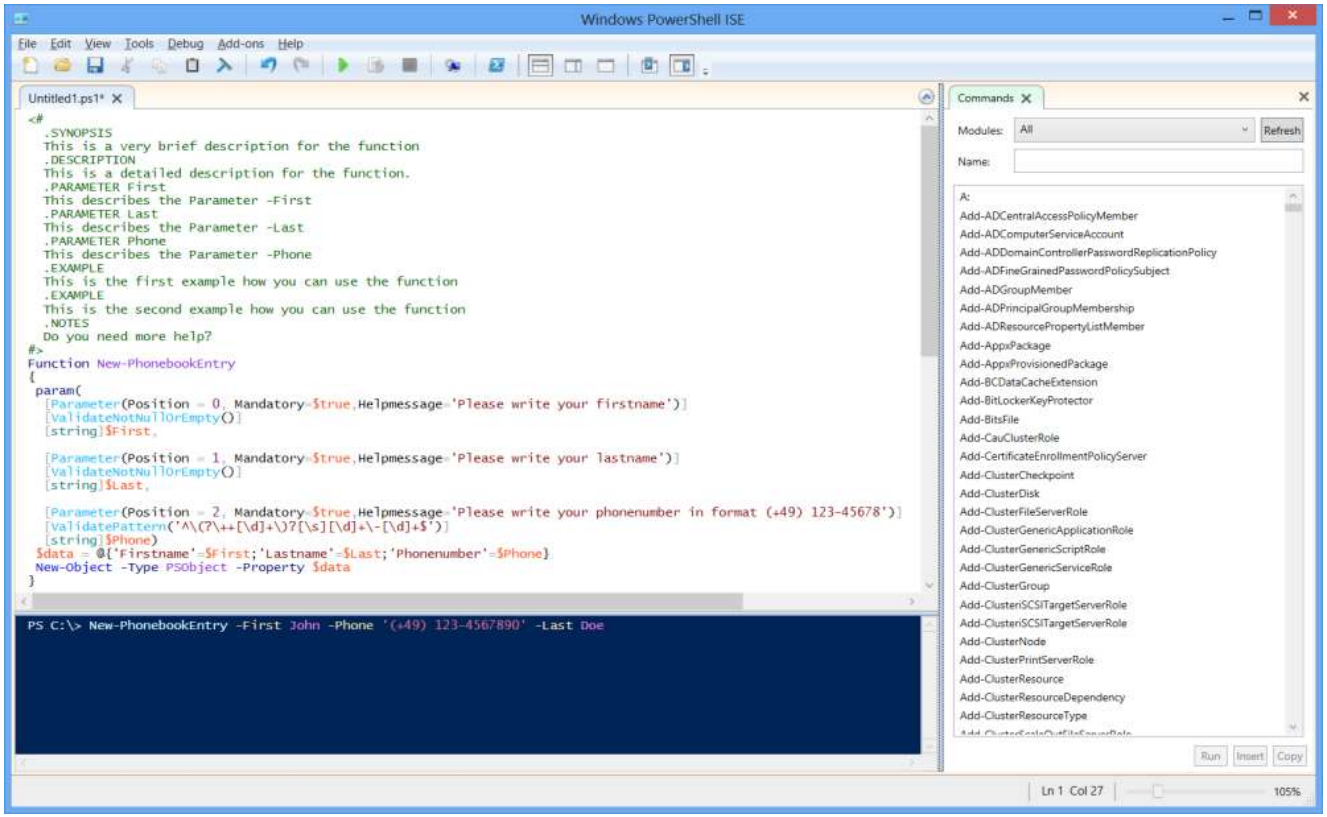
ISE editor

ISE layout

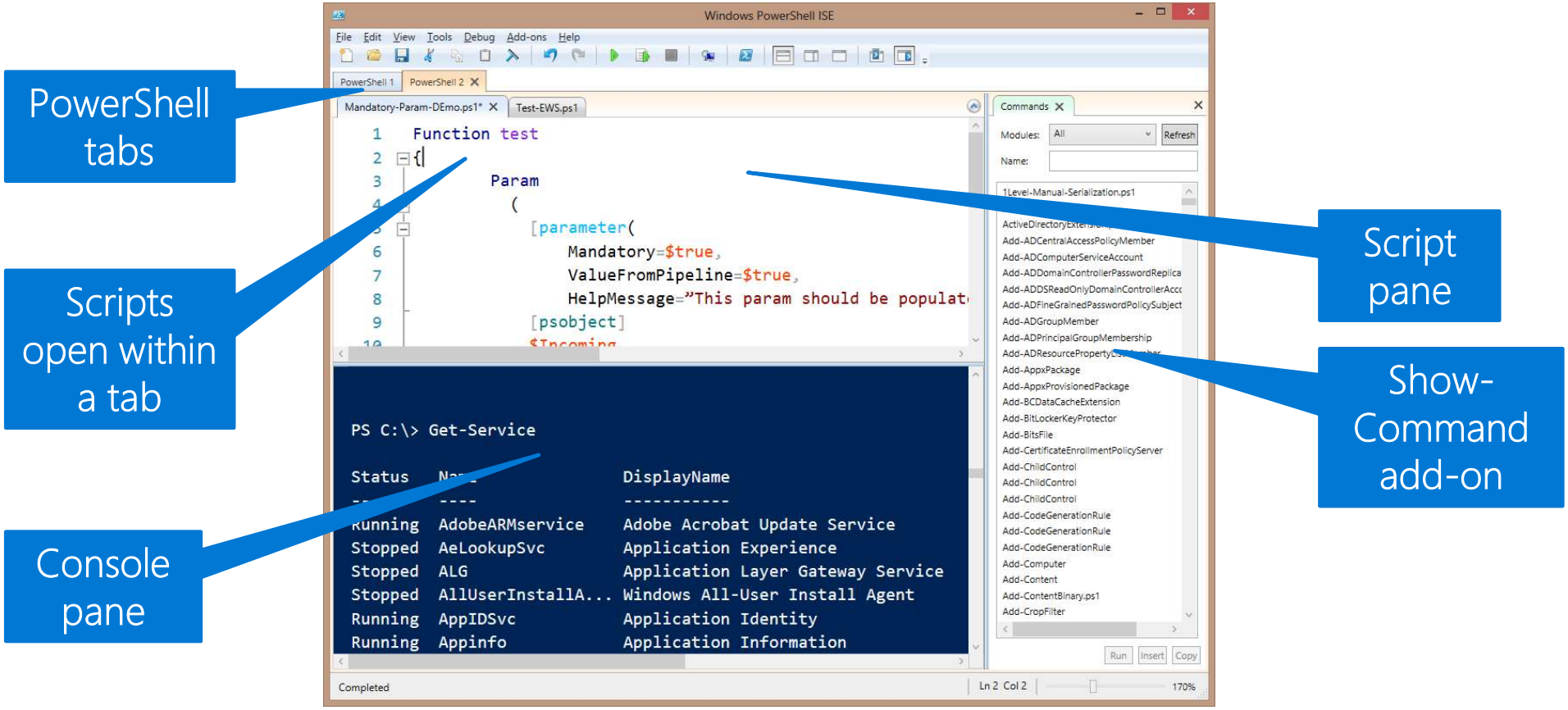
- Development Tool
- Graphical Editor
- Execution and Debugging



Start menu
Windows 10/Server 2012R2+



ISE anatomy


PowerShell tabs
Scripts open within a tab
Console pane
Script pane
Show-Command add-on
Windows PowerShell ISE
File Edit View Tools Debug Add-ons Help
PowerShell 1
PowerShell 2
Mandatory-Param-DEmo.ps1*
Test-EWS.ps1
Function test
{
Param
(
[parameter(
Mandatory=$true,
ValueFromPipeline=$true,
HelpMessage="This param should be populat
[psobject]
PS C:\> Get-Service
Status DisplayName
Running AdobeARMservice Adobe Acrobat Update Service
Stopped AeLookupSvc Application Experience
Stopped ALG Application Layer Gateway Service
Stopped AllUserInstallA... Windows All-User Install Agent
Running AppIDSvc Application Identity
Running Appinfo Application Information
Commands
Modules: All
Refresh
Name:
1Level-Manual-Serialization.ps1
Add-ADCentralAccessPolicyMember
Add-ADComputerServiceAccount
Add-ADGroupMember
Add-ADPrincipalGroupMembership
Add-AppxPackage
Add-AppxProvisionedPackage
Add-BCDataCacheExtension
Add-BitLockerKeyProtector
Add-BitsFile
Add-CertificateEnrollmentPolicyServer
Add-ChildControl
Add-CodeGenerationRule
Add-Computer
Add-Content
Add-ContentBinary.ps1
Add-CropFilter
Run
Insert
Copy
Completed
Ln 2 Col 2
170%

ISE future

- PowerShell ISE has been official editor for Windows PowerShell
- Advent of cross-platform PowerShell, needed something available cross platform
- Visual Studio Code (VSC) is now that editor from Microsoft
- Future efforts will be focused into VSC

Microsoft

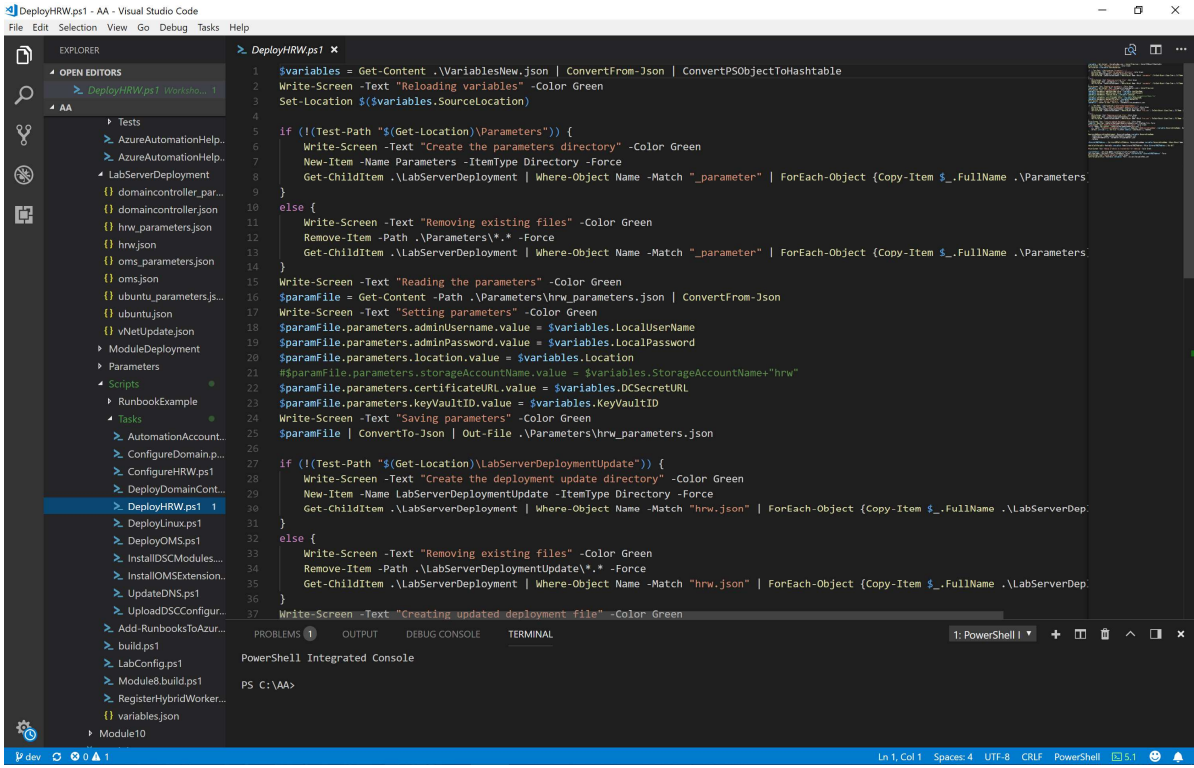
Overview of Visual Studio Code settings

Scenario Overview

- What coding application to use?
- Is there a free version that is compatible with *all* OS's?
- Visual Studio Code is the answer

Visual Studio Code (VS Code)

- Development Tool (many languages)
- Graphical Editor
- Execution and Debugging
- Source Control Integration



VS Code features from ISE

- Intellisense
- Auto save and crash recovery
- Syntax highlighting (themes)
- Collapsible code
- Brace matching
- Code snippets (Start typing then press **Tab Tab**)

Additional features

- Browse work as projects in a workspace folder
- Powerful search and replace
- View function definitions and references inline
- Code lens
- Source control integration (Git)
- Improved debugging experience
- Community extensions
- Testing integration

VS Code vs. Windows (ISE)

Why the two options?

- VS Code not full feature parity yet
- Customers aren’t always allowed to ‘install’ code editors on servers, ISE binaries are already there
- ISE in Windows Server 2019 is getting updates/improvements, but still without a (major/minor) version change

However …

- ISE is NOT available for non-Windows OS’s
- All future updates are focused on VSC

Conclusion: choice is key per situation

VS Code vs. VS Integrated Development Environment (IDE)

Why the two options?

- VS Code not full feature parity yet
- VS has greater functionality with TFS (Team Foundation Services) and Azure DevOps/VSTS (Visual Studio Team Services)

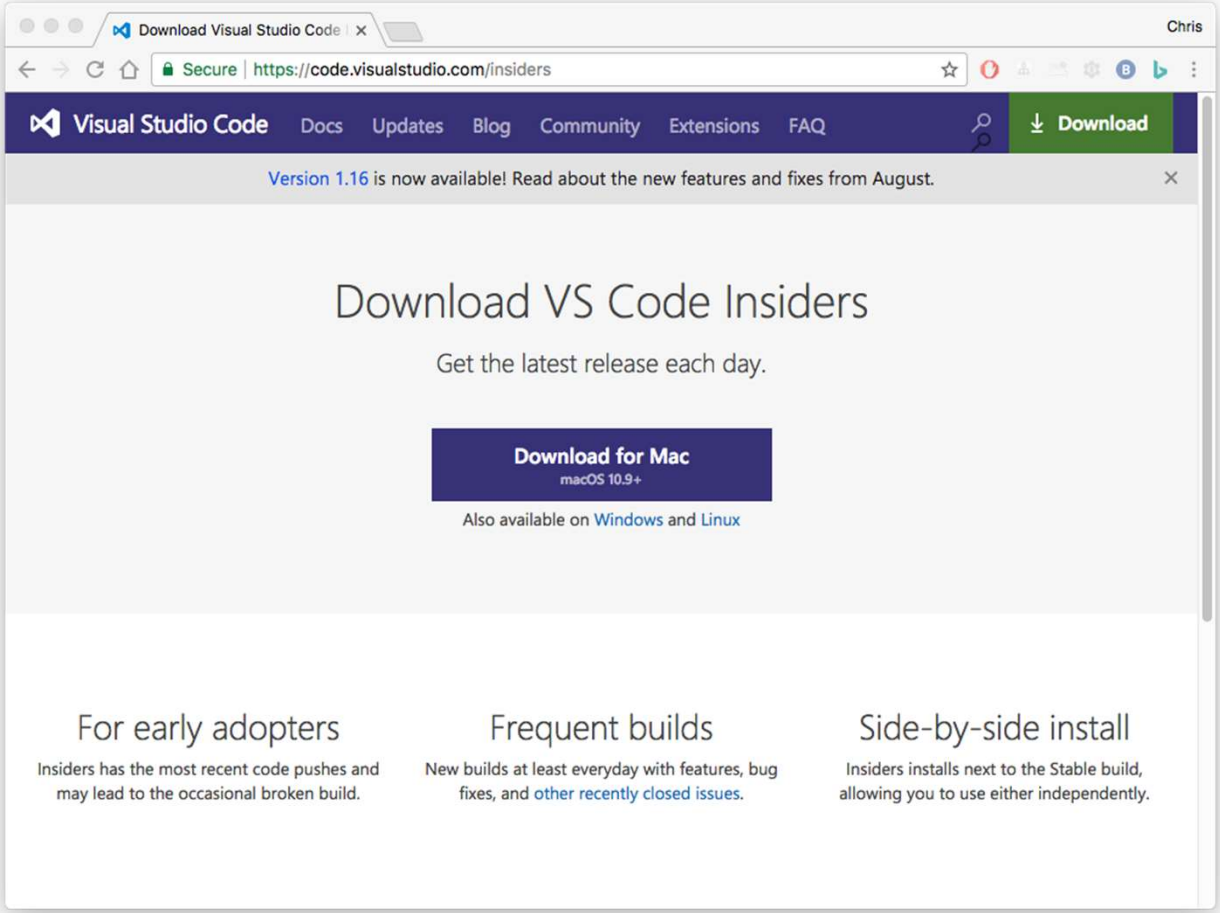
Conclusion: choice is key per situation

Built in options

1. Setup
2. Customization
3. Keyboard shortcuts
4. Editing and Code Navigation
5. Code Style and Correctness
6. Debugging
7. Tasks
8. Integrated Terminal
9. Source Code Control
10. Extensions, Marketplace

Setup

- Get the **Insiders** build
  - **Same builds**
  - **Early access to new features**
- Launch in 9+ Languages
  - **code --locale en-US**
  - **F1 >Configure Language**
- Add **shortcut** to **PATH**
  - F1 >Shell Command...

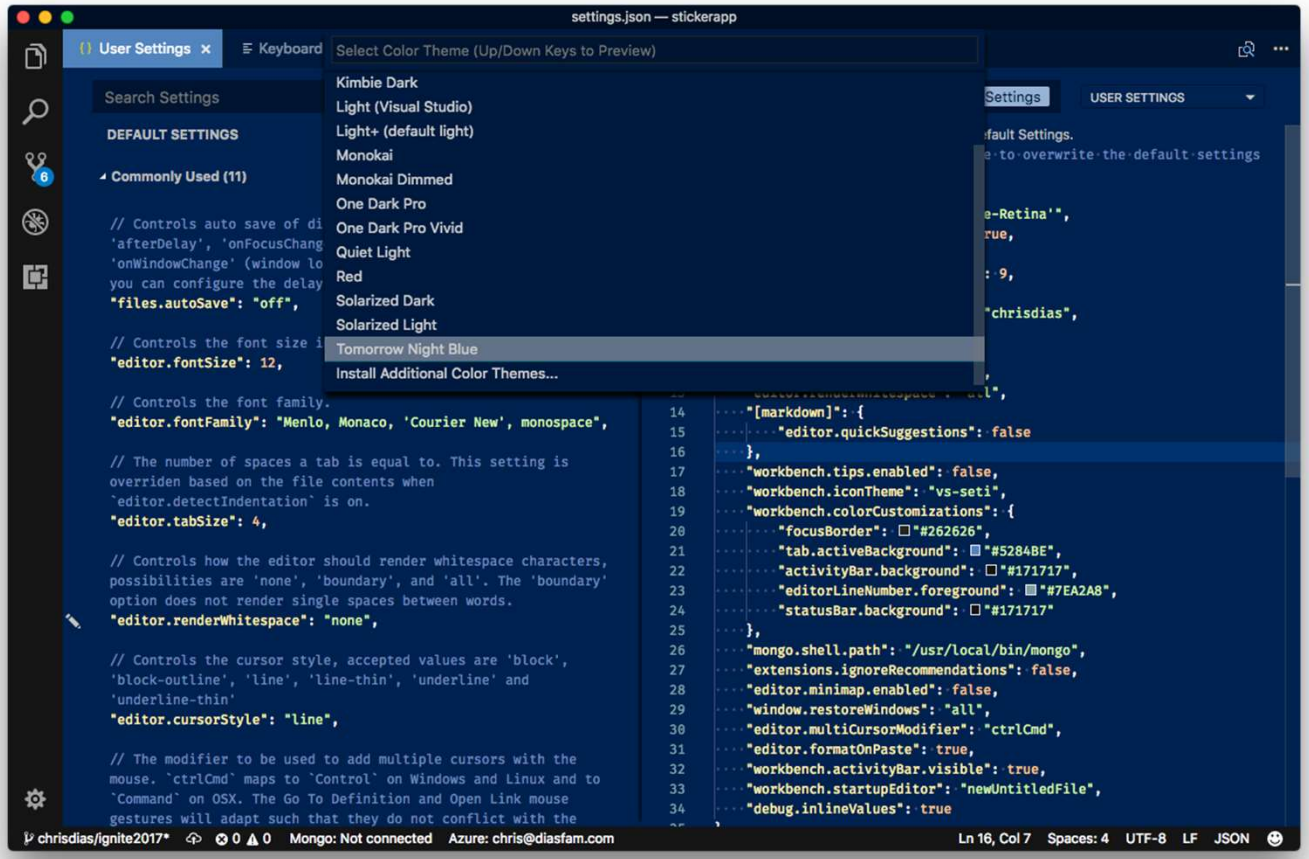


Customization

- 890+ Themes, Icons

```
"workbench.colorCustomizations": {
    "tab.activeBackground": "#5284BE"
}
```

- Settings: **CMD+,**
  - IntelliSense, validation
  - User, Workspace settings
- Fun favorites
  - AutoSave
  - Font ligatures
  - Workbench tips



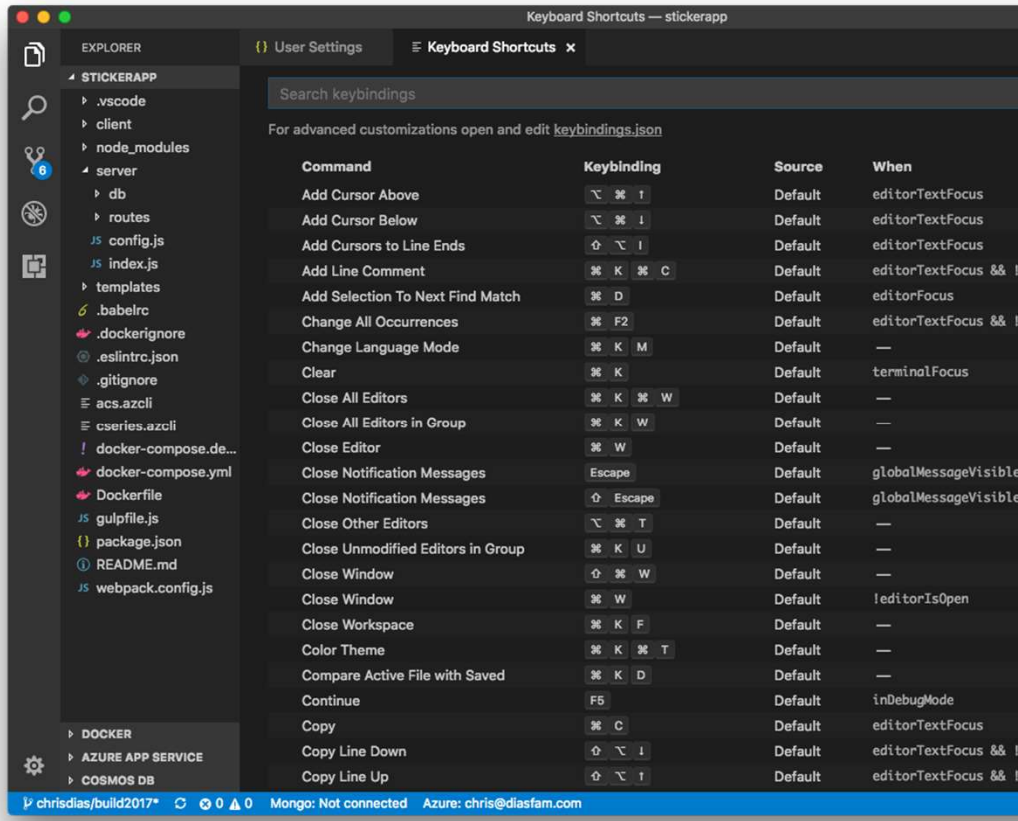
Keyboard shortcuts

- **Help > Keyboard Shortcut** Reference

- Customize shortcuts
- Editor, show conflicts
- Quick Outline in **keybindings.json**

- Keymap Extensions
- Sublime, VIM, Atom, …

- [Advanced Customization]



Problem matchers

- VS Code can process output from a task with a problem matcher
- Scan task output text for known warning or error strings and report inline to editor/Problems panel
- Can create custom problem matcher(s)

PS Script Analyzer

- Script Analyzer is built into VS Code
- Continual updates for VS Code
- Can be downloaded for ISE usage

```
function %

Get-Process | ? {}

'%' is an alias of 'ForEach-Object'. Alias can introduce possible problems and make scripts hard to maintain. Please consider
changing alias to its full content. PSScriptAnalyzer(PSAvoidUsingCmdletAliases)

Quick Fix...   Peek Problem

Measure-Command { 1..100000 | % {Get-Random -Minimum 1 -Maximum 10000} | Group-Object }

Measure-Command { 1..100000 | % {Get-Random -Minimum 1 -Maximum 10000} | Sort-Object }
Invoke-Command -ComputerName localhost -ScriptBlock {Get-Process name -eq pwsh}

Get-ProcessMeasure-Command {$a = Import-Csv foo.csv}
```

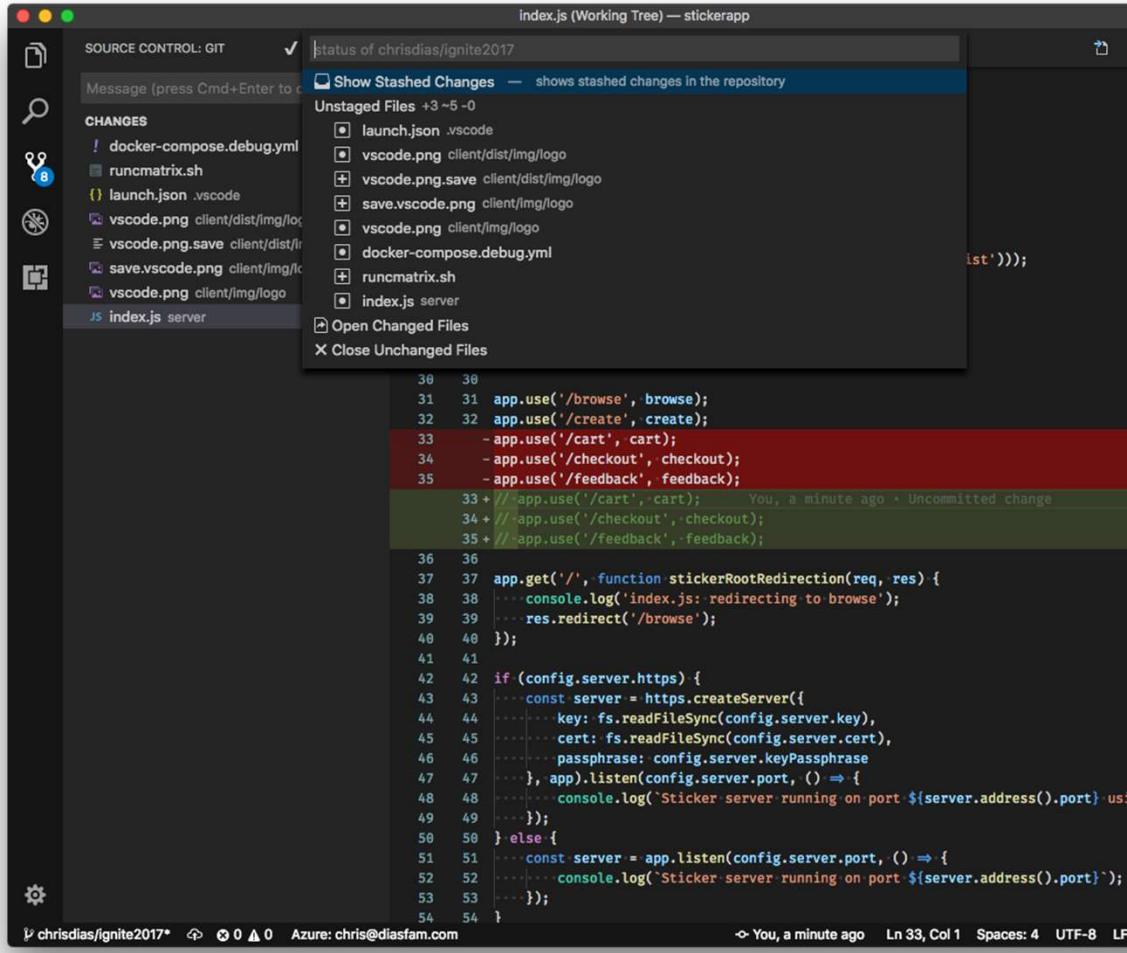
```
4
5
6   Measure-Command { 1..100000 | % {Get-Random -Minimum
7
8   Me                                                  um
9   In                                                  k {
10
11  Get-ProcessMeasure-Command {$a = Import-Csv foo.csv}
```

Replace % with ForEach-Object

Show documentation for "PSAvoidUsingCmdletAliases"

Source code control

- Multiple providers
  - Git, Hg, VSTS, Perforce
- Diff
  - Side by side, inline, accessible
  - git config --global core.editor code
- Easy branching, staging, stashing,
  and partial commits



Microsoft

PowerShell Concepts

PowerShell top cmdlets

- Get-Help
- Get/Set-ExecutionPolicy
- Get-Service
- ConvertTo-HTML/JSON
- Export-CSV
- Select-Object
- Get-EventLog
- Get/Stop-Process
- Get-Member

Get-Help

- Cmdlet help
- Concept Help
- Command Examples
- Detailed Syntax

Help for cmdlets

```
PS C:\> Get-Help Get-ChildItem
#or
PS C:\> Get-ChildItem -?

NAME
    Get-ChildItem
SYNOPSIS
    Gets the files and folders in a file system drive.
SYNTAX
    Get-ChildItem [[-Path] <String[]>] [[-Filter] <String>]...
...
DESCRIPTION
    The Get-ChildItem cmdlet gets the items in one or more...
...
RELATED LINKS
    Online version: http://technet.microsoft.com/library/h...
...
REMARKS
    To see the examples, type: "get-help Get-ChildItem ...
...
```

Get-Help parameters

```
PS C:\> Get-Help Get-ChildItem
PS C:\> Get-Help Get-ChildItem -Full
PS C:\> Get-Help Get-ChildItem -Examples
PS C:\> Get-Help Get-ChildItem -Detailed
```

Default Help Sections (no params)	All Help Sections (-Full)
NAME	NAME
SYNOPSIS	SYNOPSIS
SYNTAX	SYNTAX
DESCRIPTION	DESCRIPTION
RELATED LINKS	PARAMETERS
REMARKS	INPUTS
	OUTPUTS
	NOTES
	EXAMPLES
	RELATED LINKS

PowerShell help comments

Comment-based help keywords can be placed in functions/scripts for Get-Help display of information

```
<#
.SYNOPSIS
  Short description
.DESCRIPTION
  Long description
.EXAMPLE
  Example of how to use this cmdlet
.EXAMPLE
  Another example of how to use this cmdlet
.INPUTS
  Inputs to this cmdlet (if any)
.OUTPUTS
  Output from this cmdlet (if any)
.NOTES
  General notes
.COMPONENT
  The component this cmdlet belongs to
.ROLE
  The role this cmdlet belongs to
.FUNCTIONALITY
  The functionality that best describes this cmdlet
#>
```

Get-Service

- Cmdlet gets objects that represent the services on a computer
- Lists running and stopped services
- Not available on non-Windows OS's

```
PS C:\> Get-Service

Status     Name          DisplayName
------     ----          -----------
Stopped    AJRouter      AllJoyn Router Service
Stopped    ALG           Application Layer Gateway Service
Running    AppIDSvc      Application Identity
Running    Appinfo       Application Information
Stopped    AppMgmt       Application Management
...
```

ConvertTo-HTML

- cmdlet converts .NET Framework objects into HTML that can be displayed in a Web browser
- Use cmdlet to display output of a command in a Web page

ConvertTo-HTML Image

```
PS C:\Users\Student> ConvertTo-Html -InputObject (Get-Date)
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>HTML TABLE</title>
</head><body>
<table>
<colgroup><col/><col/><col/><col/><col/><col/><col/><col/><col/><col/><col/><col/><col/><col/><col/></colgroup>
<tr><th>DisplayHint</th><th>DateTime</th><th>Date</th><th>Day</th><th>DayOfWeek</th><th>DayOfYear</th><th>Hour</th><th>Kind</th><th>Millisecond</th><th>Minute</th><th>Month</th><th>Second</th><th>Ticks</th><th>TimeOfDay</th><th>Year</th></tr>
<tr><td>DateTime</td><td>Saturday, November 10, 2018 9:08:13 AM</td><td>11/10/2018 12:00:00 AM</td><td>10</td><td>Saturday</td><td>314</td><td>9</td><td>Local</td><td>764</td><td>8</td><td>11</td><td>13</td><td>636774376937643864</td><td>09:08:13.7643864</td><td>2018</td></tr>
</table>
</body></html>
```

HTML TABLE

file:///C:/tmep/out.html

DisplayHint	DateTime	Date	Day	DayOfWeek	DayOfYear	Hour	Kind	Millisecond	Minute	Month	Second	Ticks	TimeOfDay	Year
DateTime	Saturday, November 10, 2018 9:08:39 AM	11/10/2018 12:00:00 AM	10	Saturday	314	9	Local	479	8	11	39	636774377194796243	09:08:39.4796243	2018

ConvertTo-JSON

- Easily convert to/from JSON format

ConvertTo-Json Code

```
PS C:\> Get-Service | ConvertTo-Json | Out-File c:\temp\services.json
PS C:\> notepad.exe C:\temp\services.json
PS C:\> code . C:\temp\services.json
```


```
[
  {
    "CanPauseAndContinue": false,
    "CanShutdown": false,
    "CanStop": false,
    "DisplayName": "Agent Activation Runtime_28896f",
    "DependentServices": [
                    ],
    "MachineName": ".",
    "ServiceName": "AarSvc_28896f",
    "ServicesDependedOn": [
                    ],
    "ServiceHandle": null,
    "Status": 1,
    "ServiceType": 224,
    "StartType": 3,
    "Site": null,
    "Container": null,
    "Name": "AarSvc_28896f",
    "RequiredServices": [
                    ]
  },
```



```
[
    {
        "CanPauseAndContinue": false,
        "CanShutdown": false,
        "CanStop": false,
        "DisplayName": "Agent Activation Runtime_28896f",
        "DependentServices": [
                                ],
        "MachineName": ".",
        "ServiceName": "AarSvc_28896f",
        "ServicesDependedOn": [
                                ],
        "ServiceHandle": null,
        "Status": 1,
        "ServiceType": 224,
        "StartType": 3,
        "Site": null,
        "Container": null,
        "Name": "AarSvc_28896f",
        "RequiredServices": [
                                ]
    },
```

Export-CSV

- Creates a CSV file of the objects that are submitted
- Each object is represented as a line or row of the CSV file
- Row consists of a comma-separated list of the values of object properties
- Create spreadsheets and share data with programs/applications/scripts that take CSV files as input
- Do not format objects before sending them to the **Export-CSV** cmdlet
- To export only selected properties of an object, use the **Select-Object** cmdlet

*-Process

- cmdlet gets the processes on a local computer
- Can specify by process name or process ID (PID)
- cmdlet returns a process object that has detailed information about the process
- Supports methods that let you start and stop the process
- Starts/stops a program associated with the executable file

Note: * is for: Get, Start, Stop

PowerShell comments

Comments are extra annotation added for readability of script and additional information:

```
# Single line comment
```

```
<#
multi-line
comment
#>
```

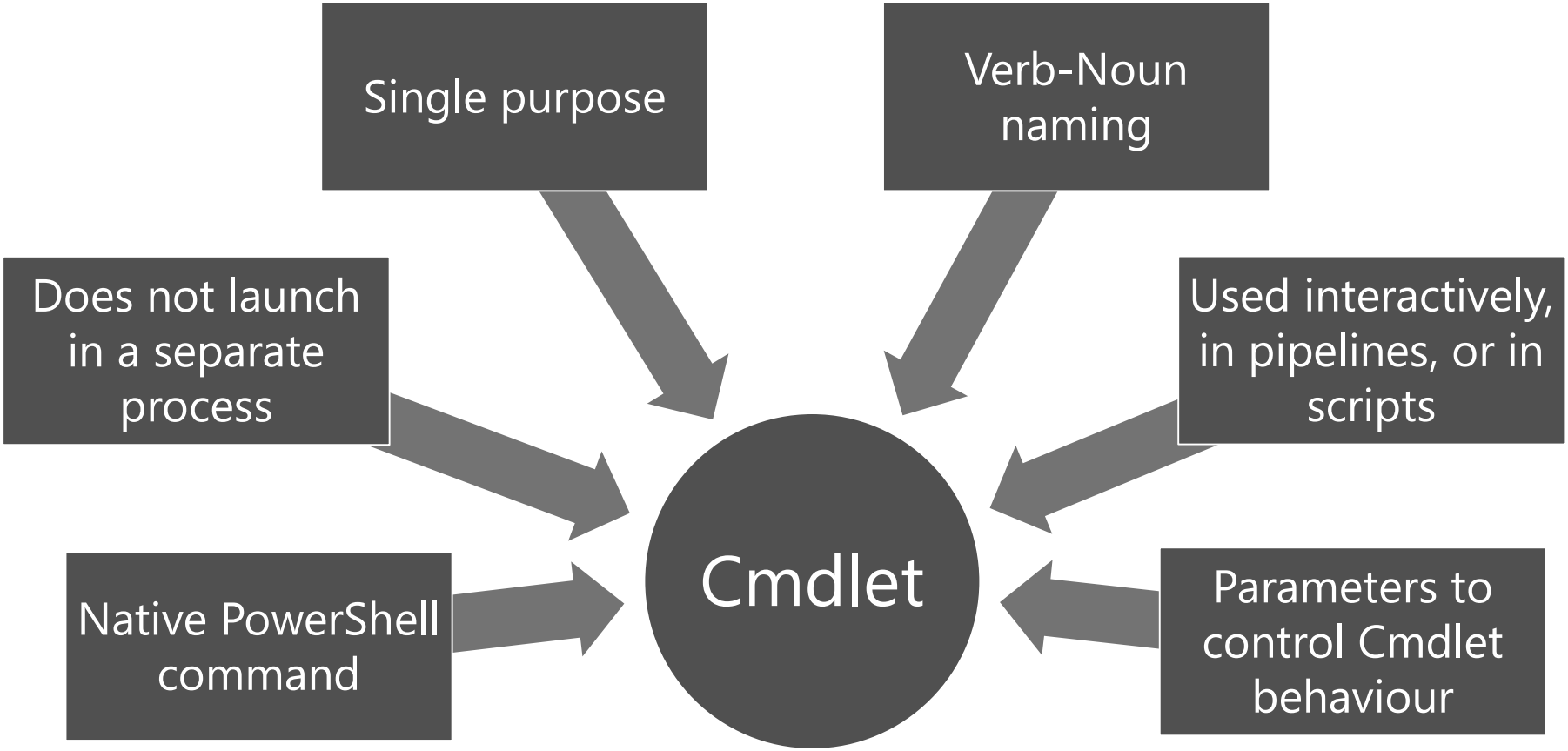
- Regions make code sections collapsible and easier to organize

```
#region

#endregion
```

Microsoft

Powershell cmdlets

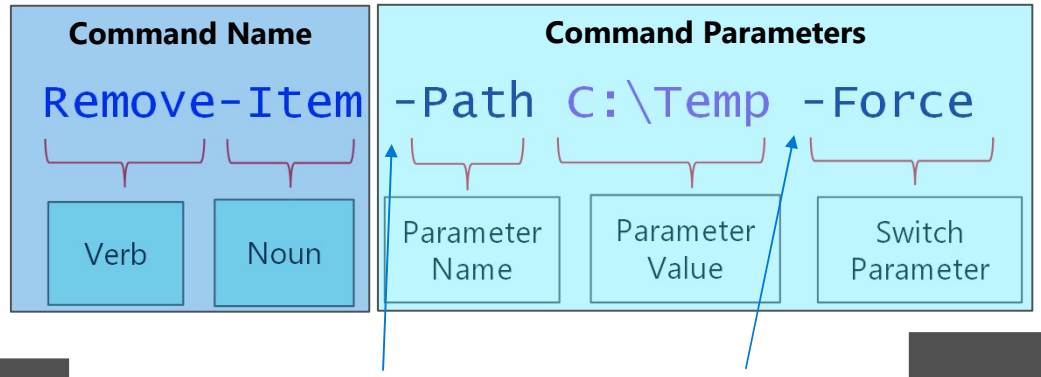
Cmdlet overview



Cmdlet anatomy

Parameters to control Cmdlet behaviour

Native PowerShell command



Verb-Noun Naming

Dashes precede parameter names

Does not Launch in Separate Process

Common parameters

- Parameters automatically available with any Cmdlet
- Implemented by PowerShell not Cmdlet developer
- Override system defaults or preferences

List of common parameters

Parameter	Action
-Debug (db)	Displays programmer-level detail
-ErrorAction (ea)	Determines how cmdlet responds to errors
-ErrorVariable (ev)	Stores error messages in a specified variable
-OutVariable (ov)	Stores output in a specified variable
-OutBuffer (ob)	Determines number of output objects to accumulate in a buffer
-PipelineVariable (pv)	Stores value of current pipeline* element as a variable
-Verbose (vb)	Displays detailed information
-WarningAction (wa)	Determines how cmdlet responds to warnings
-WarningVariable (wv)	Stores warnings in a specified variable

Risk mitigation parameters

- Many cmdlets also offer risk mitigation parameters
- Typically when the cmdlet changes the system or application

Parameter	Action
-WhatIf (wi)	Displays message describing the effect of the command, instead of executing the command
-Confirm (cf)	Prompts for confirmation before executing command

Get-Command

- Discover Commands (cmdlets, functions, scripts, aliases)
- Can show command syntax
- Can also discover external commands (.exe, .cpl, .msc)

Show-Command

- Not currently available in PowerShell
- **Get-Command** displays resources
- **Show/Hide** valid verbs

Microsoft

Objects and where do they come from

Objects and types

In PowerShell:

- Everything is an OBJECT
- Each OBJECT has a TYPE

Object types

Alias	Full Name	Description
Object	System.Object	Every type in PowerShell is derived from object
Boolean	System.Boolean	$true and $false
Char	System.Char	Stores UTF-16-encoded 16-bit Unicode code point
Int	System.Int32	-2147483648 to 2147483647
Long	System.Int64	-9223372036854775808 to 9223372036854775807
Double	System.Double	Double-precision floating-point number
Enum	System.Enum	Defines a set of named constants
Array	System.Array	One or more dimensions with 0 or more elements
DateTime	System.DateTime	Stores date and time values

How to find an object type

- PowerShell typically picks object type

Examples of PowerShell choosing appropriate

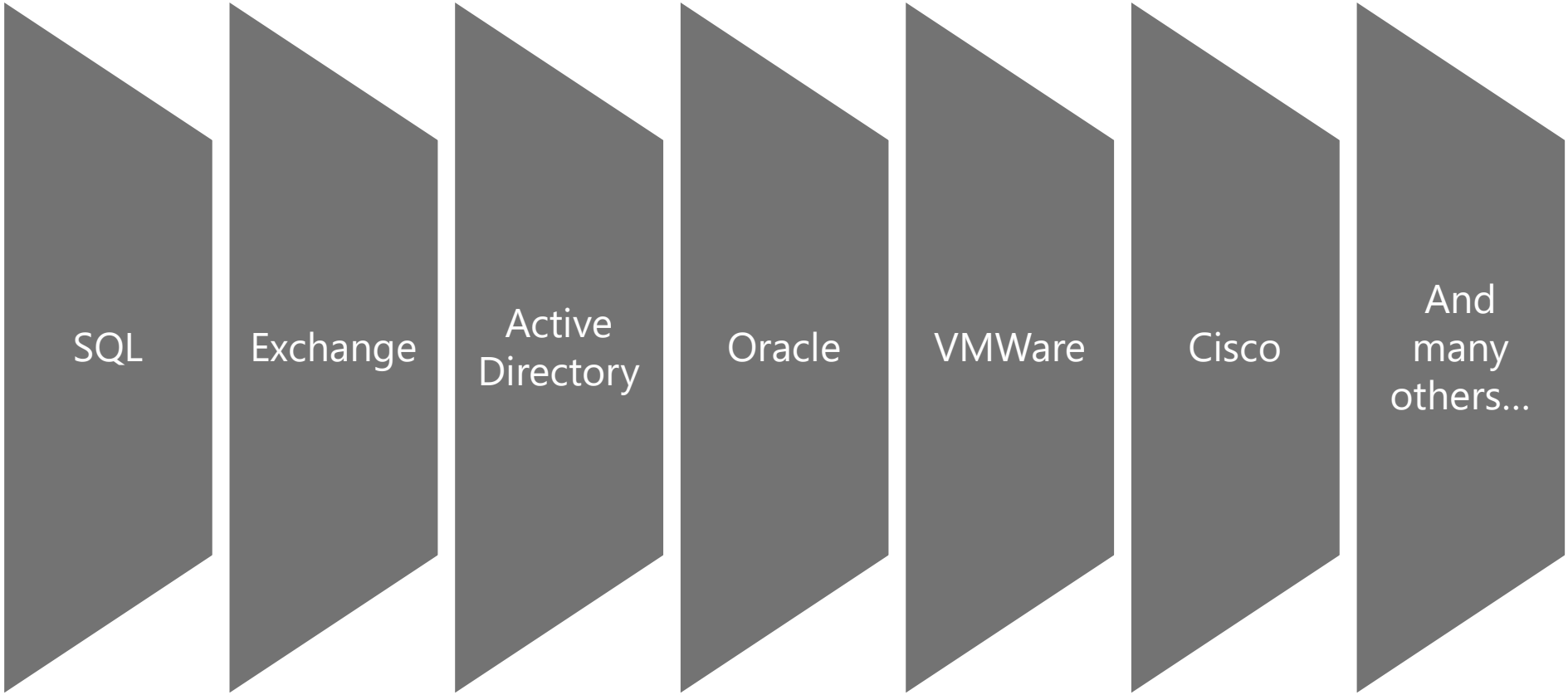
```
PS C:\> (1024).GetType().FullName
System.Int32

PS C:\> (1.6).GetType().FullName
System.Double

PS C:\> (1tb).GetType().FullName
System.Int64
```

Find type

- Pipe cmdlet to **Get-Member** to find object information

```
50    Get-Process | Get-Member

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

PS C:\> Get-Process | Get-Member


   TypeName: System.Diagnostics.Process

Name                       MemberType     Definition
----                       ----------     ----------
Handles                    AliasProperty  Handles = Handlecount
Name                       AliasProperty  Name = ProcessName
NPM                        AliasProperty  NPM = NonpagedSystemMemorySize64
PM                         AliasProperty  PM = PagedMemorySize64
SI                         AliasProperty  SI = SessionId
VM                         AliasProperty  VM = VirtualMemorySize64
WS                         AliasProperty  WS = WorkingSet64
Parent                     CodeProperty   System.Object Parent{get=GetParentProcess;}
Disposed                   Event          System.EventHandler Disposed(System.Object,
ErrorDataReceived          Event          System.Diagnostics.DataReceivedEventHandler
```

Additional types from providers



Microsoft

Basic functions

Function basics

- **Scriptblocks** are statements in braces

```
{Get-Process}
```

- **Functions:**
  - Are reusable, named scriptblocks
  - Can accept parameter values and return output
  - Advanced Functions behave like Cmdlets
  - Help topics that can be used with Get-Help (like cmdlets) available

```
function <Name>
 {
    param ($Parameter1, $Parameter2, $parameterNth)
        <statement list>
 }
```

Simple function

‘Function it up’, means turn simple repeatable code that is used often, into a simple Verb-Noun function

```
PS C:\> Get-Service –Name spooler –RequiredServices –ComputerName WINDC


Function Get-ServiceInfo {
         Get-service -Name Spooler -RequiredServices -ComputerName WINDC
}

#Run the function
PS C:\> Get-ServiceInfo
```

Simple function with parameters

Add parameters to control variable choices

```
PS C:\> Get-Service –Name spooler –RequiredServices –ComputerName WINDC


Function Get-ServiceInfo {
   param ($SVC, $Computer)
       Get-service -Name $SVC -RequiredServices -ComputerName $Computer
}

#Run the function
PS C:\> Get-ServiceInfo -SVC BITS -Computer localhost
```

Microsoft

PS Script Overview

What is a script

- A collection of statements or expressions within a text file with a .ps1 extension
- Scripts can be run in either the Windows PowerShell console, ISE or launched externally from cmd.exe
- The Param() statement enables the script to accept input parameters
- Production scripts **should** be signed with a trusted certificate

Execution policy scope

Apply Execution Policy Levels at one or more of these 5 scopes

```
PS C:\> Get-ExecutionPolicy -List

Scope           ExecutionPolicy
-----           ---------------
MachinePolicy   Undefined
UserPolicy      Undefined
Process         Undefined
CurrentUser     Undefined
LocalMachine    RemoteSigned
```

Effective Policy

AD Group Policy – Computer

- Affects all users on targeted computer
- Edited through GPO Tools

AD Group Policy – User

- Affects users targeted only
- Edited through GPO Tools

Process

- Console or ISE Command-line Parameter:
  i.e.: *c:\> powershell.exe –executionpolicy remotesigned*
- Affects current PowerShell Host session only
- Lost upon exit of session (i.e. host process)

Registry – User

- Affects current user only
- Stored in HKCU registry subkey

Registry – Computer

- Affects all users on computer
- Stored in HKLM registry subkey (Admin access needed to change)

Highest Priority Wins

Script execution

- Execution Policy determines which scripts can and cannot be run
- Default setting is restricted, which means that no scripts can be run, including Profiles
- PowerShell can be used as Interactive only
- Control on Windows machines via GPO
- Available to control on OS’s using Desired State Configuration (DSC)

Running a script (Windows)

Full path and file name

```
PS C:\> c:\scripts\script.ps1
```

Script in current directory

```
PS C:\Scripts> .\script.ps1
```

Spaces in path (tab completion helps)

```
PS C:\> & “c:\scripts\my script.ps1”
```

Script is in environment path

```
PS C:\> Script.ps1
```

Running a script (Linux)

Full path and file name

```
PS /> /home/user/Documents/script.ps1
```

Script in current directory

```
PS /> /home/user/Documents> ./script.ps1
```

Spaces in path

```
PS /> & “/home/user/My script.ps1”
```

Microsoft

Anatomy of PS modules

What is a module

- A package of commands
- All cmdlets and providers in a session are added by a module or a snap-in
- Modules can contain cmdlets, scripts, functions, variables, aliases, and other useful items
- Modules are useful for sharing code

Get-Module – Imported Modules

- Gets modules that have been imported into current session

List imported modules

```
PS C:\> Get-Module

ModuleType Version Name                        ExportedCommands
---------- ------- ----                        ----------------
Script     1.0.0.0 ISE                         {Get-IseSnipp...
Manifest   3.1.0.0 Microsoft.PowerShell.Ma... {Add-Computer...
Manifest   3.1.0.0 Microsoft.PowerShell.Ut... {Add-Member, ...
```

Script module

- Script module can be as simple as script with .psm1 extension
- Typically contain functions for company or department to re-use
- Script runs upon module import
- By default, functions and variables remain available for use in PowerShell session

MyModule.psm1

```
Function Get-Data
{
    “Simple function”
}
```

1. Create Module Folder in PSModule path with same name as .psm1, “MyModule”
2. Place MyModule.psm1 file in named folder
3. Import-Module MyModule

Microsoft

Pipeline Introduction

What is a Pipeline?

- Series of commands connected by pipeline character
- Vertical bar character |
- Sends output of command as input to another (left to right)
- Passes Objects, not text
- Filtering, Formatting, and Outputting available
- Cmdlets designed to chain together into ‘pipelines’

Pipeline use

Initial pipeline input provided by:

- Get-* & Import-* cmdlets, text files & external commands

Pipeline objects are manipulated using:

- Sort-*, Select-*, Group-*, Measure-Object, cmdlets & more

Pipeline output via:

- Format-* cmdlets (should always be last)
- Export-* cmdlets
- Out-* cmdlets
- Variables

‘Get’ cmdlets

- Typically placed first in the pipeline
- Provides input to be processed

Returns schedule and bits services

```
PS C:\> Get-Service -Name Schedule , BITS | Start-Service
```

Takes an action on the services

Text file input

- Text files provide input to be processed by the piepline

Reads a text file

```
PS C:\> Get-Content .\services.txt | Get-Service
```

Takes an action on each line in the file

Object cmdlets

Name	Description
Sort-Object	Sorts objects by property values
Select-Object	Selects object properties
Group-Object	Groups objects that contain the same value for specified properties
Measure-Object	Calculates numeric properties of objects, and the characters, words, and lines in string objects, such as text files
Compare-Object	Compares two sets of objects

Format List (FL)

```
PS C:\> Get-Process -Name powershell | Format-List

Id      : 6400
Handles : 472
CPU     : 0.78125
Name    : powershell
```

- Output is in list format
- Properties chosen are based on default formatting in PowerShell by object type

```
PS C:\> Get-Process -Name powershell |
Format-List -Property Name, BasePriority, PriorityClass

Name          : powershell
BasePriority  : 8
PriorityClass : Normal
```

- Output in list format
- Consists of specified properties

Format Table (FT)

```
PS C:\> Get-Process | Format-Table

 Handles  NPM(K)    PM(K)      WS(K) VM(M)   CPU(s)     Id ProcessName
-------  ------    -----      ----- -----   ------     -- -----------
     82       7     1308       1420    45     0.14   2308 armsvc
    195      13     2568       3440    94     3.78   1192 atieclxx
    110       6      852       1172    23     0.09    868 atiesrxx
    565      20     6384       7092   113    42.14   4308 BasisSync
    180      12     2276       2660    89     0.41   7744 BDAppHost
    …
```

```
PS C:\> Get-Process |
Format-Table -Property name,workingset,handles

Name                          WorkingSet            Handles
----                          ----------            -------
csrss                             847872                216
csrss                             356352                 91
csrss                           15646720                183
dwm                              7045120                176
…
```

Pipeline variable

- When multiple objects are piped, PowerShell sends objects one at a time
- Built-in variables $_ and $PSItem represent current object on pipeline
- Used to perform an action on every object
- Use -PipelineVariable parameter to name variable on the pipeline
- Scoped only to current pipeline

```
Get-ChildItem | Where-Object { $psitem.Length -gt 1MB }
```

- Alternatively, create a custom pipeline variable using –PipelineVariable

```
Get-Process -PipelineVariable CurrentProcess |
    Where-Object {$CurrentProcess.ws -gt 100MB}
```

- Where-Object has a simple syntax (new to v3)

```
Get-ChildItem | Where-Object { $_.Length -gt 1MB }
    Get-ChildItem | Where-Object Length -gt 1MB
```

Other object cmdlets

Cmdlet	Action – Aliases
ForEach-Object	Performs an operation against each item. Aliases: • % • ForEach
Where-Object	Filters objects in the pipeline Aliases: • ? • Where

Microsoft

Code samples

Azure Examples

https://github.com/mapadi/ps1examples

- Get Subscription
- Change Subscription
- Create VM
- Create Network

Overview of Azure PowerShell | Microsoft Docs

Filtering on attributes

```
Get-User -filter {Department -like "marketing"} | Format-List

Get-User -filter {Department -like "marketing"} | Format-Table name, department, office

Get-User -filter {Department -like "marketing"} | Get-Mailbox

Get-User -filter {Department -like "marketing"} | Set-User -Office "chicago"

Get-User -filter {(Department -like "*marketing*") -AND (RecipientType -eq "UserMailbox")}
|ft name, Department, RecipientType

Get-User -filter {(Department -like "*marketing*") -AND (RecipientType -eq "UserMailbox")}
| Set-Mailbox -IssueWarningQuota 600mb
```

Logging onto Exchange online

```
$Cred = Get-Credential
$Session = New-PSSession –ConfigurationName Microsoft.Exchange `
      -ConnectionUri https://ps.outlook.com/powershell `
      -Credential $Cred `
      -Authentication Basic `
      -AllowRedirection
Import-PSSession $Session -Prefix EXO
Import-Module msonline
Connect-Msolservice
```

Send e-mail

```
Send-MailMessage –From user@domain.com –To user@Hotmail.com –Subject
“Test Email” –Body “Test SMTP Relay Service” -SmtpServer
smtp.office365.com -Credential $msolcred -UseSsl -Port 587
```

Send mass e-mail

```
1..3600 | ForEach-Object {Send-MailMessage -To
User@ReceivingTenant.com -From external@SendingTenant.com -
SmtpServer smtp.office365.com -Subject "Test Message $_" -Body
"This is the body of Message $_" ; write-host “Sending Message
$_”}
```

Create mailboxes and users

```
1..1000000000 | % {New-Mailbox -Name User$_ -Alias User$_
–DisplayName “User$_” -Password (ConvertTo-SecureString "Password1"
-AsPlainText -Force) -UserPrincipalName "user$_@contoso.com"
-OrganizationalUnit “contoso.com/Accounts"}
```

O365 networking

```
# Peering points connections
 Start-Process http://www.peeringdb.com/view.php?asn=8075

 # Test for Peering point from current workstation
 tracert outlook.office365.com

 #Test Connectivity site
 Start-Process http://testconnectivity.microsoft.com/

 #Hybrid Envrionment Free/busy site
 Start-Process http://support.microsoft.com/kb/2555008

 #DNS check for autodiscover (Replace ‘tenant’ with valid tenant name.)
 Resolve-DnsName Autodiscover.tenant.onmicrosoft.com
```

Restart Autodiscover app pool

- Quickly restart the Autodiscover app pool within IIS
- Can be used for other app pools on non-Exchange servers
- Get-ExchangeServer could be replaced with Get-Content and list of servers needed
- “autod” could be replaced by name of specific app pool to recycle

```
Function Restart-AutoDAppPool {
      Get-ExchangeServer | foreach {
            Write-Output "Recycling $_" ; Invoke-Command `
            -ComputerName $_.name `
            -ScriptBlock { Get-ChildItem IIS:\AppPools | `
            Where-Object { $_.name -like "*autod*" } |
      Restart-WebAppPool
      }
   }
}
```

SQL

```
get-psdrive

cd SQLSERVER:\sql\uspauljfwi10\default\databases #location within runspace, as a
provider
dir
#SQLPS module for 2012+ Mapps a 'drive' to expose all of the SQL backend.

get-help Invoke-SQLcmd -Examples #Displays get-help of the SQL module
#Can expose all of the SQL application.

Invoke-Sqlcmd "select *,db_name(dbid) as DBname from sys.sysprocesses where spid >
50" | select spid, blocked, cmd, DBname, hostname, loginame | ft #Select statement
of all system processes running on the SQL server and then displays the most
pertinent information.
```

Active Directory

Active Directory Powershell Blog | Microsoft Docs

Additional Resources

- Learn:
  PowerShell Documentation - PowerShell | Microsoft Docs
  PowerShell Gallery | Home
  https://code.visualstudio.com/docs

- Insiders:
  https://code.visualstudio.com/insiders

- Contribute:
  https://github.com/microsoft/vscode

- Tips:
  https://github.com/microsoft/vscode-tips-and-tricks

Reminder

Before we start Q&A, we would like to remind everyone that this session is being recorded.

Microsoft

Questions?

Microsoft

Thank you