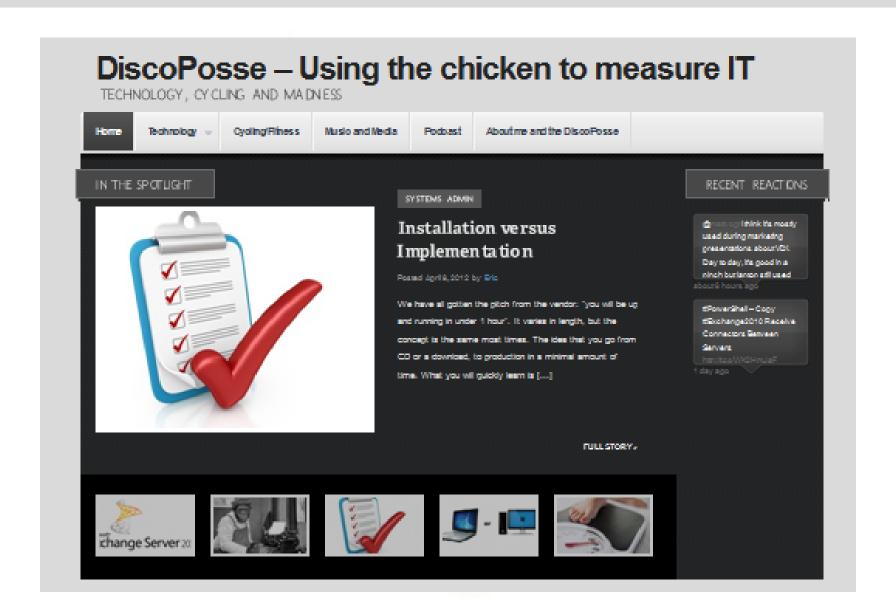
# Admin Scripting, Tips and Tricks

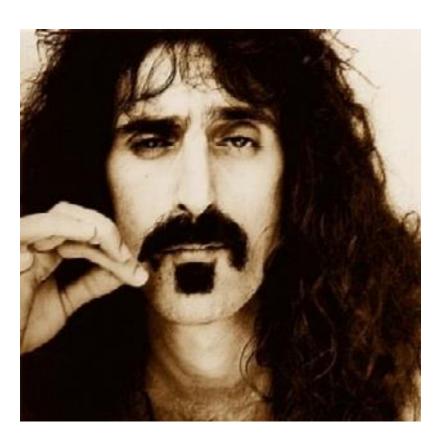
Managing your environment with simple effective scripting techniques

**Eric Wright** 



# Using the chicken to measure IT?





Measurability, tangibility and Frank Zappa

# These are your friends









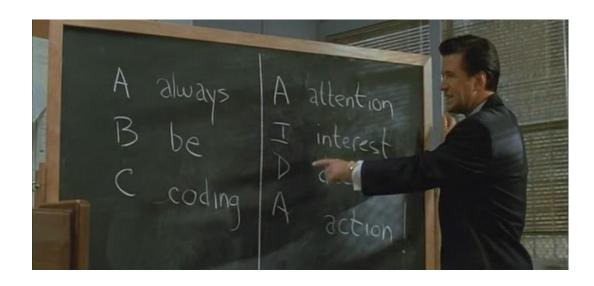
#### **SCRIPTING BASICS AND CONCEPTS**

# Why? DRY, that's why.

- DRY Don't Repeat Yourself
  - Repetition opens the door for human error
  - Productivity lost when manual processes are used
  - Scheduling is not firm, nor is it guaranteed
  - Scripts don't get the flu

#### **ABC**

- A always
- B be
- C coding



# You only get better with practice

- More exposure to scripting increases your base of knowledge
- PowerShell is becoming an industry standard for many infrastructure systems
- Windows Server 8 extends even further with PowerShell v3

# Don't hesitate, automate!

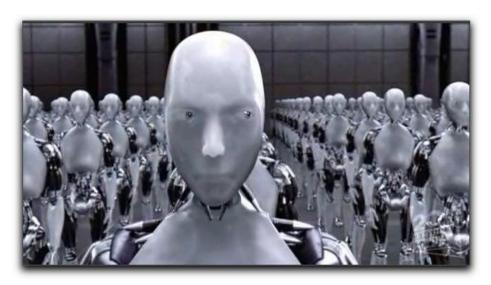


# Win with scripting!

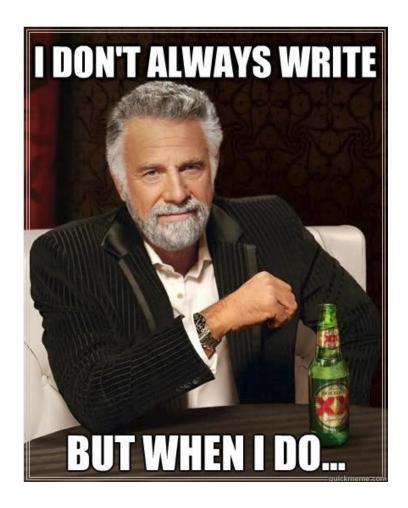
- Automate current tedious processes
- Build maintenance tasks easily with native scripting tools
- Design and standardize installations and application configurations using scripts
- Be prepared for when scripting is a requirement during issues and emergencies

#### But I can't control it when it's automated?

- Many administrators do not trust automated processes
- The machine will never become smarter than the creator
- The robots have yet to replace the people who built them



#### **DOCUMENTATION**



I write documentation!

#### DiscoPosse – Using the chicken to measure IT

TECHNOLOGY, CYCLING AND MADNESS

# Challenge yourself...

- Create one new and one legacy document a week
- You will learn as you write.
- Teaching others forces us to teach ourselves first
- Testing and Documentation are usually the first victims when reeling in project timelines

NEVER HAVE I FELT SO CLOSE TO ANOTHER SOUL AND YET 50 HELPLESSLY ALONE AS WHEN I GOOGLE AN ERROR AND THERE'S ONE RESULT A THREAD BY SOMEONE WITH THE SAME PROBLEM ANO NO ANSWER LAST POSTED TO IN 2003



HTTP://XKCD.COM/979/

# What does sharing do for you?

- Opens up for new ideas and opinions
- Pair programming concept realized through social means (internally or externally)
- Create a repository for others who are in similar situations
- Improves your ability to write documentation and code

# Let's Get to the Coding

#### But I'm not a coder!

Not necessary. Simple scripting is just that. Simple. Grow with your comfort level as you learn.

# 3 types of scripting

- Report
  - Tell me about point in time information
- Monitor
  - Information gathering over a period of time
- Act
  - Take action based on a set of criteria

# The one question quiz about your script

# You are totally...

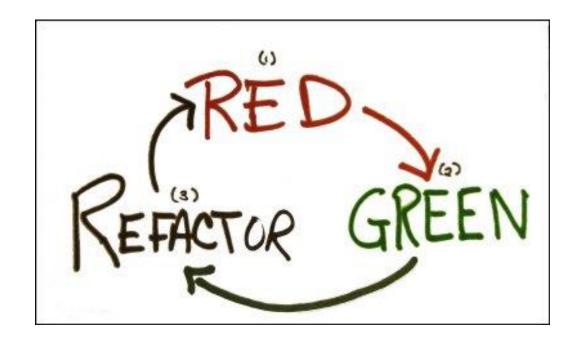
- Right
- **□**Wrong
- ✓ All of the above

# Code is never complete

#### You will always be revisiting code because:

- Products and source data change
- You learn different techniques
- Digital rot happens
- It just makes sense

# The art of refactoring



# TDD – Test Driven Development

- Not just for application development
- It should not work the first time
- No seriously...it is supposed to fail
- Handling exceptions requires repetition

# Testing the process

- If you don't already have one, create a test environment
- It is seldom said that a script or application was tested too much

#### What If?



- In many PowerShell CmdLets there is a parameter WhatIf
- The WhatIf parameter test drives the action without committing the change

# Comment, Comment, Comment

- Comments in PowerShell begin with #
- Use comments to walk the reader through the natural flow of the script
- You totally understand the script now. Just wait 3 months and you will find out how short term memory fades

# Logging



# Logging to the console

- Output to screen as you build
- Use Write-Host with your parameters to confirm results

# Logging to file

- Use the *Out-File* CmdLet to direct output to the filesystem
- Use the *append* option

```
$win7Computers=Get-ADComputer -Filter {OperatingSystem -like "Windows 7*"} -Properties *
ForEach ($win7Computer in $win7Computers) {
    $win7ComputerName = $win7Computer.name
    $win7OU = $win7Computer.DistinguishedName -replace "$win7Computername.",""
    $logOutput = "$win7ComputerName is located in $win7OU"
    $logOutput | Out-File "X:\path\win7Computers.txt" -append
}
```

# Readability

- Comments are key, but readability is equally important
- Space, tab, and comment to make the script more natural to the eyes when revisiting

```
# Environment setup
# Import the ActiveDirectory module to enable the Get-ADComputer CmdLet
Import-Module ActiveDirectory
$SCCMServer = "YOUR SCCM SERVER"
$sitename = "YOUR SCCM SITE"
$old = (Get-Date).AddDays(-
45) # The threshold for what we consider to be old (current set as 45 days)
# Find the computers in Active Directory which are "old"
$oldComputers = Get-ADComputer -Filter {PasswordLastSet -le $old} -Properties *
ForEach ($oldComputer in $oldComputers) {
    # Select the computer(s)
    $computername = $oldComputer.name
    # Get the resourceID from SCCM
    $resID = Get-WmiObject -computername $SCCMServer -
query "select resourceID from sms r system where name like `'$computername`'" -
Namespace "root\sms\site $sitename"
    $computerID = $resID.ResourceID
```

# Readability

This is a tick: Not Good This is a backtick: Good





# Using the backtick

 By adding a backtick, you can extend a command onto multiple lines

```
# Now we set the additional properties that aren't accessible by the New-MailContact
cmdlet
Set-Contact -Identity $sourceEmail `
-City $sourceAddressCity `
-Company $sourceBranchLegalName `
-CountryOrRegion $sourceAddressCountry `
-Department $sourceBranchCode
-DisplayName $targetDisplayName
-SimpleDisplayName $targetDisplayName
-FirstName $targetFirstName `
-LastName $sourceLastName `
-Phone $targetPhoneNumber `
-PostalCode $sourceAddressZip
-StateOrProvince $sourceAddressState
-StreetAddress $targetStreetAddress
-Title $sourceJobTitle
-WebPage "RJFAccountFlag" `
-WindowsEmailAddress $sourceEmail-WhatIf
```

#### The art of the one-liner



"Doctor, my leg hurts. What can I do?" The doctor says, "Limp!"

# Pipeline

- Passing output from one CmdLet into another CmdLet
- Uses the pipe symbol: |

Example:

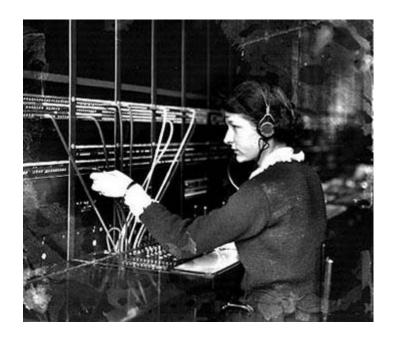
**Get-VM | sort PowerState** 

### Pipelines can be extended

Export a list of all non-server OS computers into a CSV format where the computer account hasn't contacted AD for over 30 days

```
Get-ADcomputer -Filter { OperatingSystem -notlike "Windows Server*" } -Properties PasswordLastSet | Where-Object { (((Get-Date) - $_.PasswordLastSet).Days) -gt 30} | select-object Name, @{Name="Age";Expression={ (((Get-Date) - $_.PasswordLastSet).Days) }} | Export-CSV ComputerListByAge.csv
```

### Operators



### Operators for PowerShell

```
Mathematical
   +,-,*,/,=
Replace
   -replace (e.g. "It is sunny" -replace "sunny","cloudy")
   -ireplace (case insensitive replace)
   -creplace (case sensitive replace)
Value
   -is
   -isnot
```

### Comparison Operators for PowerShell

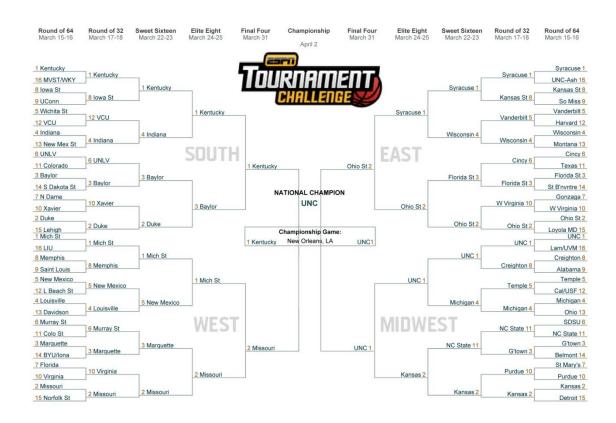
Used for numeric and string comparisons

- -**It** (less than)
- -le (less than or equal to)
- -gt (greater than)
- -ge (greater than or equal to)
- -like (wildcard pattern matching)
- -notlike (wildcard pattern matching)
- -match (regular expression pattern matching)
- -notmatch (reg. expression pattern matching)
- -contains (returns \$True or \$False result)
- -notcontains (returns \$True or \$False)

### Bitwise and Logical Operators

- Used for inclusive or exclusive operations
  - -and
  - -or
  - -xor
  - -not
  - -band
  - -bor
  - -bxor
  - -bnot

### **Brackets**



No, not those brackets...

### **Brackets**

- Understand where you need brackets (aka parentheses)
- Order is important
- Mathematical operations use BEDMAS logic
- Command operations use parenthetical order
- Not always necessary

#### **Brackets**

- Valid
  - \$yesterday = (Get-Date).AddDays(-1)
- Also valid, but unnecessary
  - \$yesterday = ((Get-Date).AddDays(-1))

### **Arrays**

- Gather data as an array
   \$AllVMs = Get-VM
- Access array elements (remember first element is 0)
   \$AIIVMs[0]
- Access properties
   \$AllVMs[0].PowerState
- Count elements in an array \$AllVMs.count

### Clean up after your pet

Clear-Variable \$variablename

[gc]::collect()

[gc]::WaitForPendingFinalizers()

### Modules and Snapins



# Extending the reach of your shell environment

### Import-Module

**Add-PSSnapin** 

- PowerCLI
- Exchange 2007/2010
- SQL 2008 or higher
- Active Directory 2008 or higher
- Quest ActiveRoles

### One shell to rule them all

Create a script named MyConsole.ps1

```
# Import the Active Directory extensions
Import-Module ActiveDirectory
# Log in to vCenter to allow remote management
Connect-VIServer <Your vCenter Server Name Here>
# Create and import a PowerShell remote session to let you run
# Exchange 2010 commands from your single shell
$Session = New-PSSession -ConfigurationName Microsoft.Exchange -
ConnectionUri http://<FQDN of Exchange 2010 server>/PowerShell/ -
Authentication Kerberos
Import-PSSession $Session
# Add the Quest PowerShell snapin
Add-PsSnapIn Quest.ActiveRoles.ADManagement -ErrorAction SilentlyContinue
```

### Let's **Get** going

First project: Get everything...literally

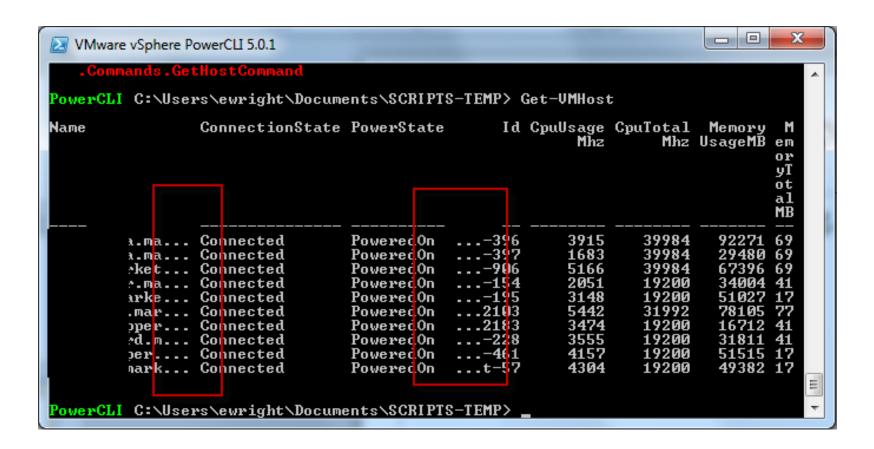
**Get-VM** 

**Get-VMHost** 

**Get-Help Get\*** 

**Get-Command Get\*** 

### Where is my output?



#### Format-List

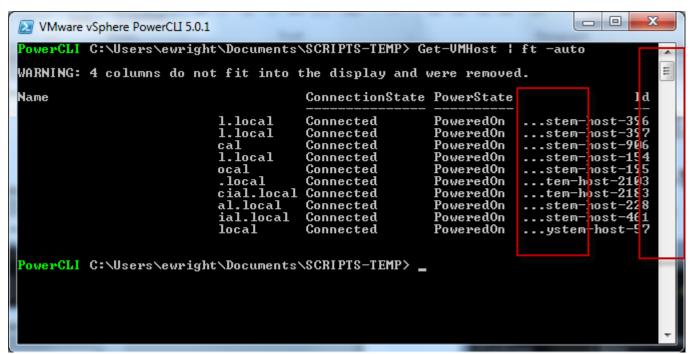
#### Get-VMHost | Format-List Get-VMHost | fl

```
VMware vSphere PowerCLI 5.0.1
WARNING: 'State' property is obsolete. Use 'ConnectionState' instead.
                      : Connected
ConnectionState
                      : Connected
PowerState
                      : PoweredOn
{\sf UMSwapfileDatastoreId}:
UMSwapfilePolicy
                     : Inherit
                     : ClusterComputeResource-domain-c392
ParentId
IsStandalone
                      : False
Manufacturer
                      : HP
Mode 1
                      : ProLiant DL380 G7
NumCpu
                      : 12
                     : 39984
CpuTota1Mhz
CpuUsageMhz
MemoryTota1MB
                      : 5835
                     : 122869
MemoryUsageMB
                     : 92263
ProcessorType : Inte
HyperthreadingActive : True
                     : Intel(R) Xeon(R) CPU
                                                      X5680 @ 3.33GHz
                     : UTC
: 4.1.0
TimeZone
Version
Build
                      : 433742
Parent
UMSwapfileDatastore
```

### Format-Table -auto

## Get-VMHost | Format-Table –auto Get-VMHost | ft –auto

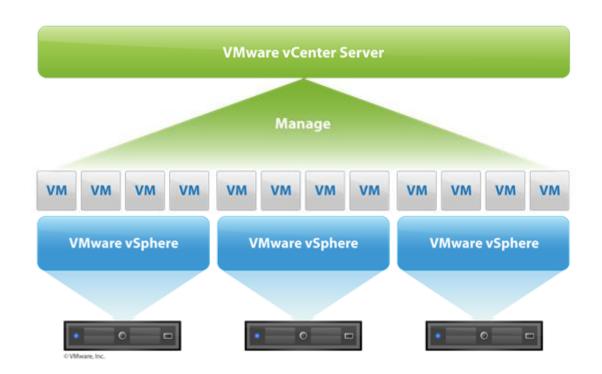
Maps to your console width



### Start small and build

**Connect-VIServer** 

**Get-Cluster | Get-VMHost | Get-VM** 



### DiscoPosse – Using the chicken to measure IT

TECHNOLOGY, CYCLING AND MADNESS

#### **PUTTING IT ALL TOGETHER**

### What did we learn?

- PowerShell is a directional strategy for many vendors
- Leverage -WhatIf wherever possible
- Fail early, fail often (It's an IBM thing)
- Commenting takes time, but makes you a hero
- Log your process
- Readability is key
- Practice, practice, practice

### Much, much more

- Sorting
- Out-GridView
- Regular Expressions
- String conversion
- Memory management

### How do I get help?

- Get-Help \*
- Get-Help CmdLet
- PowerCLI Community (Search PowerCLI on http://vmware.com)
- TechNet Gallery http://gallery.technet.microsoft.com
- Twitter #PowerShell #PowerCLI
- DiscoPosse.com ©

#### Let PowerShell and PowerCLI do this for you



### DiscoPosse – Using the chicken to measure IT

TECHNOLOGY, CYCLING AND MADNESS

www.discoposse.com