

Descubra o Poder do Windows Powershell



Consultoria & TI

Agenda

- ▶ A empresa H2G2
- ▶ O Poder do Powershell
- ▶ Demonstração

H2G2 Consultoria e TI

A H2G2 Consultoria e TI é uma empresa jovem, formada por especialistas com o objetivo de excelência na prestação de serviços TI.

Através dos serviços de consultoria e suporte nós procuramos entender as necessidades dos clientes e entregar de forma mais eficiente os serviços de TI para que o negócio alcance seus resultados.

Prezamos pela entrega de soluções que tragam diferenciais competitivos, alinhando com as estratégias de negócio e baseado em princípios de alta-disponibilidade e segurança.

O parceiro tecnológico para sua empresa manter-se focada em seu negócio!

Consultoria

Infraestrutura e Redes



▶ Consultoria em infraestrutura de redes e segurança Linux e Windows

▶ Microsoft

- ▶ Dynamics CRM
- ▶ Exchange
- ▶ System Center
- ▶ Sharepoint
- ▶ Lync Server
- ▶ SQL Server

▶ Linux

- ▶ Zimbra
- ▶ Zabbix
- ▶ OpenLDAP

▶ Segurança

- ▶ PFSense
- ▶ Fortigate

▶ Gestão de TI

- ▶ Mapeamento de Processos
- ▶ Definição de Indicadores
- ▶ OTRS
- ▶ ProcessMaker

▶ Cloud Computing

- ▶ Windows Azure
- ▶ Office 365
- ▶ Dynamics Online
- ▶ Sharepoint Online
- ▶ OMS (Operations Management Suite)

Consultoria

Eventos e Treinamentos



Windows Powershell



Segurança Microsoft



Microsoft Deployment
Com MDT



Gerenciando Projetos
Colaborativos



Treinamentos Oficiais
Microsoft



Montando um Service
Desk com OTRS

Licenciamento

Microsoft



Microsoft Partner

- ▶ Open License

 - Licenciamento por volume para pequenas e médias empresas

- ▶ SPLA Licensing

 - Assinatura de licença com baixo custo mensal

- ▶ CSP Licensing

 - Assinatura de serviços de nuvem gerenciados com baixo custo mensal

- ▶ Academic

 - Licenciamento para instituições vinculadas ao MEC

Redes Sociais



<http://www.youtube.com/c/H2G2BRITSM>



<https://www.facebook.com/H2G2ITSM>



<https://twitter.com/H2G2TI>



Hangout

Descubra o Poder do Powershell - Google Chrome

<https://plus.google.com/u/1/hangouts/onair/watch?hid=hoaevent%2Fcojqcg5mp00c10hdktvg9e2oq6o&hl=pt-BR>



Descubra o Poder do Powershell

Rodrigo Camarão  ? 

Q&A
P&R

Showcase

Showcase

Ainda não há itens em destaque. Confira novamente mais tarde.

O Poder do Powershell

Visão Geral do Powershell

- ▶ Introduzido em 2006
- ▶ Implementado como uma “engine” que pode ser integrada a GUI ou utilizada diretamente por linha de comando

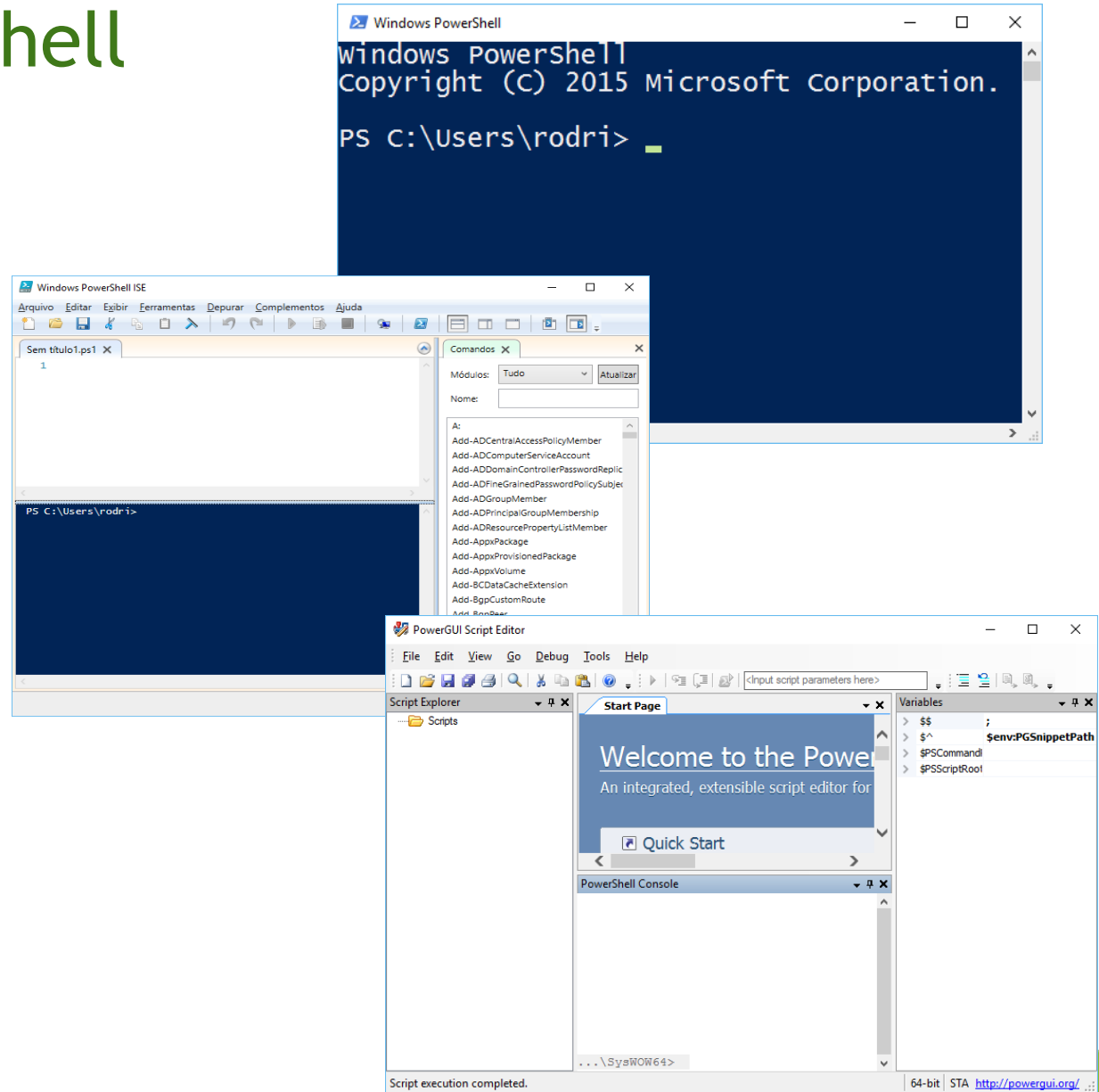
- ▶ Funcionalidades
 - ▶ Cmdlets (pronuncia-se “command-lets”)
 - ▶ Módulos
 - ▶ Funções
 - ▶ Workflow
 - ▶ Mais..

Visão Geral do Powershell

► Console

► ISE (Script Environment)

► Third-party



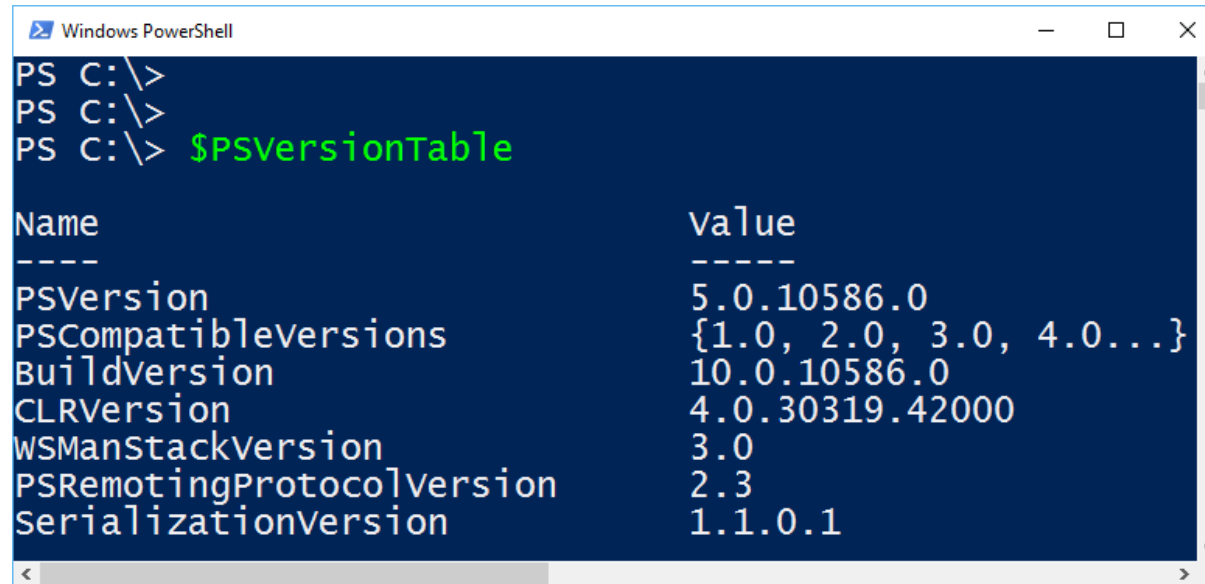
Visão Geral do Powershell

	2.0	3.0	4.0
Windows XP	Available	No	No
Windows Server 2003	Available	No	No
Windows Vista	Available	No	No
Windows Server 2008	Available	Available with SP2	No
Windows 7	Installed	Available with SP1	Available
Windows Server 2008 R2	Installed	Available with SP2	Available
Windows 8	No	Installed	Available
Windows Server 2012	No	Installed	Available
Windows 8.1 e 2012 R2	No	No	Installed

Qual versão do Powershell?

▶ Para descobrir a versão do Powershell que você está usando:

▶ `$PSVersionTable`



```
Windows PowerShell
PS C:\>
PS C:\>
PS C:\> $PSVersionTable

Name                Value
----                -
PSVersion            5.0.10586.0
PSCompatibleVersions {1.0, 2.0, 3.0, 4.0...}
BuildVersion         10.0.10586.0
CLRVersion            4.0.30319.42000
WSManStackVersion    3.0
PSRemotingProtocolVersion 2.3
SerializationVersion 1.1.0.1
```

▶ Para executar uma versão anterior execute:

▶ `Powershell.exe -version 2.0`

Visão Geral do Powershell

- ▶ Escolha uma FONTE que facilite a diferenciação entre

“ ‘ ` ’ ({ [| <

- ▶ Write-Host ' This is a message '

- ▶ \$var = ' World '

- ▶ \$out = " Hello \$var "

- ▶ \$out agora contém ' Hello World '

- ▶ \$query = " SELECT * FROM Customers WHERE Name LIKE ' %RODRIGO% ' "



Dica!

Cmd-Lets em Powershell

▶ Comandos familiares

- ▶ Dir
- ▶ Cd
- ▶ Type

Aliases

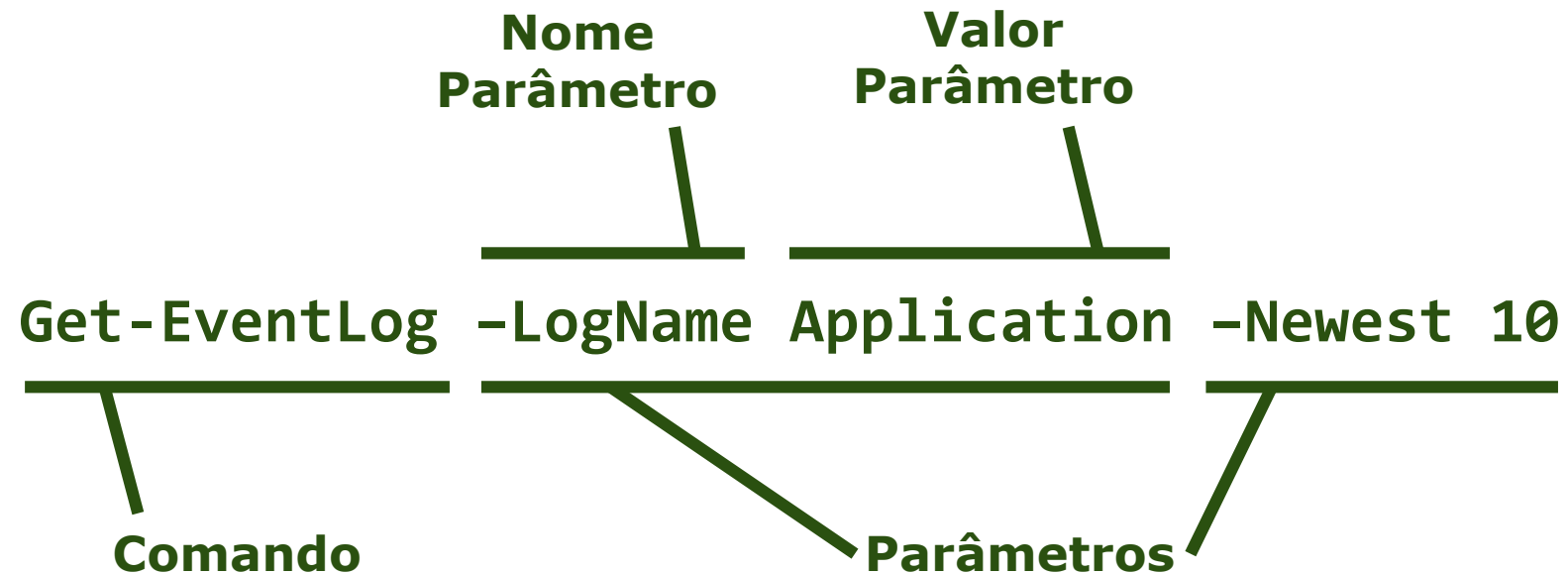
▶ Sintaxe do PowerShell

- ▶ Get-ChildItem

↓
VERBO

↓
SUBSTANTIVO

Sintaxe completa em Powershell



Sintaxe completa em Powershell

▶ Cuidado com os espaços

- ▶ Powershell utiliza o espaço como delimitador para:
 - ▶ Comandos e Parametros
 - ▶ Parametros e Valores

```
Get-EventLog -LogName Application -Newest 10
```

▶ Cuidado com Maiúsculo e Minúsculo

- ▶ Powershell é case INSENSITIVE
- ▶ Mas alguns parâmetros e valores podem ser case SENSITIVE



Dica!

Entendendo o Pipeline

- ▶ O Powershell executa os comandos em um *pipeline*
- ▶ No console, cada comando completo é um pipeline
- ▶ É possível executar múltiplos comandos

```
Get-Service | Out-File ServiceList.txt
```

Descobrendo um Objeto

- ▶ Objetos incluem
 - ▶ *Propriedades*
 - ▶ *Métodos*
 - ▶ *Eventos*

- ▶ Utilize o Get-Member para ver os detalhes do objeto

Get-Service | **Get-Member**

Selecionando Propriedades de um Objeto

- ▶ Utilize `Select-Object` para pegar uma informação

```
Get-Process | Select-Object -First 10
```

```
Get-Process | Select-Object -Property Name,ID,VM,PM
```

VM - Virtual Memory é exibido em MB

PM - Paged Memory é exibido em KB

Criando Propriedades Calculadas

- ▶ Propriedades calculadas (custom) facilitam a interpretação
- ▶ Hash Table (Associative Array)
 - ▶ *label, l, name ou n*
 - ▶ *expression ou e*

```
@{  
    n='VirtualMemory';  
    e={$PSItem.PM}  
}
```

Get-Process | Select-Object

Name,

ID,

@{n='VirtualMemory';e={\$PSItem.PM}},

@{n='PagedMemory';e={\$PSItem.PM}}

Criando Propriedades Calculadas

- ▶ Formatação de Propriedades
- ▶ Hash Table (Associative Array)
 - ▶ *label, l, name ou n*
 - ▶ *expression ou e*



Dica!

```
@{  
    n='VirtualMemory';  
    e={$PSItem.PM}  
}
```

Get-Process | Select-Object

Name,

ID,

```
@{n='VirtualMemory(MB)';e='{0:N2}' -f ($PSItem.VM / 1MB)}},
```

```
@{n='PagedMemory(MB)';e='{0:N2}' -f ($PSItem.PM / 1MB) }
```

Convertendo, Importando e Exportando

- ▶ Conversão modifica os dados e facilita a manipulação
- ▶ 2 Verbos
 - ▶ *ConvertTo*
 - ▶ *Export*

Get-Command -Verb ConvertTo,Export

Convertendo, Importando e Exportando

- ▶ Conversão modifica os dados e facilita a manipulação

```
Get-Service | ConvertTo-CSV | Out-File Services.csv
```

```
Get-Service | Export-Csv Services.csv
```


Filtrando

- ▶ O comando Where-Object suporta múltiplas condições
- ▶ Utilize o \$PSItem ou \$_ para representar o objeto

```
Get-Service | Where-Object -FilterScript { $_.Status -eq 'Running' }
```

```
Get-Service | Where Status -eq Running
```

```
Get-Service | ? Status -eq Running
```

Filtrando

► Operadores

Comparação	Case-Insensitive	Case-Sensitive
Igualdade	-eq	-ceq
Diferença	-ne	-cne
Maior	-gt	-cgt
Menor	-lt	-clt
Maior Igual	-ge	-cge
Menor Igual	-le	-cle
Wildcard	-like	-clike

Trabalhando com Providers

- ▶ Adapta Data Stores para parecerem drives no Powershell
- ▶ Active Directory
- ▶ Registro
- ▶ Certificados Digitais
- ▶ WS-Management
- ▶ FileSystem

Get-PSDrive

Trabalhando com FileSystem

Comando DOS	Cmdlet Powershell
Dir	Get-ChildItem
Move	Move-Item
Ren	Rename-Item
Del	Remove-Item
Copy	Copy-Item
Mkdir	New-Item
Cd	Set-Location
	Get-Location
	Get-ItemProperty
	Set-Item
	Set-ItemProperty

Get-Command -Noun Item,ItemProperty

Trabalhando com Active Directory

Computadores

Get-ADComputer

Set-ADComputer

Remove-ADComputer

New-ADComputer

Usuários

Get-ADUser

Set-ADUser

Remove-ADUser

New-ADUser

Accounts

Search-ADAccount

Enable-ADAccount

Disable-ADAccount

Unlock-ADAccount

Grupos

Get-ADGroup

Set-ADGroup

Remove-ADGroup

New-ADGroup

Get-Command -Module ActiveDirectory

Trabalhando com Active Directory

▶ Localizando contas de Usuários

▶ Desabilitados

Search-ADAccount -AccountDisabled

▶ Último Logon

Get-ADUser -Identity <username> -properties LastLogOnDate



Dica!

Trabalhando com Active Directory

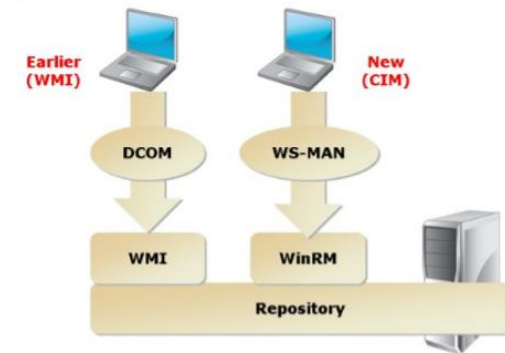


Dica!

```
$domain = "ADATUM.COM"  
$DaysInactive = 90  
$time = (Get-Date).Adddays(-($DaysInactive))  
  
$Users = Get-ADUser -Filter {LastLogonTimeStamp -lt $time}  
           -Properties LastLogonTimeStamp  
  
$Users | select-object  
Name,  
@{n="Data";  
   e={[DateTime]::FromFileTime($_.lastLogonTimestamp).ToString('yyyy-MM-dd')}}}
```

Trabalhando com WMI e CIM

- ▶ WMI - Windows Management Instrumentation
- ▶ CIM - Common Information Model



```
Get-WmiObject -Namespace root\cimv2 -List
```

```
Get-CimClass -Namespace root\CIMv2
```


Trabalhando com WMI e CIM



Dica!

► Descobrindo o ServiceTag

```
Get-CimInstance Win32_SystemEnclosure | Select SerialNumber
```

► Descobrindo a Placa de Video

```
Get-CimInstance Win32_VideoController | select Caption
```

► Descobrindo o Disco Rígido

```
Get-CimInstance Win32_LogicalDisk -Filter "DriveType=3" |  
Select DeviceID,@{n='Size';e='{0:N2}' -f ($_.Size/1GB)}
```

Trabalhando com Scripts Seguros

▶ Variáveis

- ▶ `$User = Get-ADUser -Identity rodrigo.camarao`
- ▶ `[int]$Valor = 100`
- ▶ `$Data = Get-Date`

▶ Arrays

- ▶ `$Users = Get-ADUser -Filter *`

Trabalhando com Scripts Seguros

- ▶ Recursos de segurança do Powershell servem para
 - ▶ Prevenir contra erros de usuário
 - ▶ Evitar scripts não autorizados
- ▶ Políticas de Execução de script
 - ▶ Restricted (Default) - Previne execução de scripts
 - ▶ AllSigned - Executa apenas scripts assinados
 - ▶ RemoteSigned - Executa todos scripts locais, mas remotos somente assinados
 - ▶ Unrestricted - Executa todos os scripts
 - ▶ Bypass - Segurança gerenciada pelo script

Set-ExecutionPolicy RemoteSigned -Force

Trabalhando com Scripts Seguros

▶ Extensão de arquivos

- ▶ .ps1 - Script Powershell
- ▶ .psm1 - Script para módulos de Powershell
- ▶ .psc - Configuração de console

▶ A execução de scripts deve ser com caminho relativo ou absoluto

C:\script.ps1

.\script.ps1

Trabalhando com Scripts Seguros

► Passagem de parâmetros

```
[CmdletBinding()]  
Param(  
    [Parameter(Mandatory=$True)]  
    [string]$ComputerName,  
  
    [int]$EventID = 4624  
)
```

```
Get-EventLog -LogName Security -ComputerName $ComputerName |  
Where EventID -eq $EventID | Select -First 50
```

Relatório de Logon de usuários



```
[CmdletBinding()]
param(
    $ComputerName="localhost"
)

$UserProperty = @{n="User";e={(New-Object
System.Security.Principal.SecurityIdentifier
$_.ReplacementStrings[1]).Translate([System.Security.Principal.NTAccount])}
}
$TypeProperty = @{n="Action";e={if($_.EventID -eq 7001) {"Logon"} else
{"Logoff"}}}
$TimeProperty = @{n="Time";e={$_.TimeGenerated}}
$MachineNameProperty = @{n="MachinenName";e={$_.MachineName}}

foreach ($computer in $ComputerName) {
    Get-EventLog System -Source Microsoft-Windows-Winlogon | select
$UserProperty,$TypeProperty,$TimeProperty,$MachineNameProperty |
Export-CSV logins.csv
}
```



“Nosso principal objetivo é prover serviços de TI com excelência alinhado às necessidades do negócio, buscando otimização de recursos e redução de custos.”

Obrigado!

Contatos:

Tel: (+55) 51 4063-8203 | 31 4063-9011

E-mail: comercial@h2g2.com.br

<http://www.h2g2.com.br>

Treinamentos

Windows Powershell

28/Jan/2016



20%
Desconto