

Creating and Managing Remote PowerShell Sessions



Liam Cleary

CEO / MICROSOFT MVP / MICROSOFT CERTIFIED TRAINER

@shareplicity www.shareplicity.com | @helloitsliam www.helloitsliam.com



Overview



Using "PSCredential" Objects for storing Credentials

Implementing Trusted Hosts

Creating and Removing PowerShell Sessions



Using "PSCredential" Objects for Storing Credentials



The PSPrincipal object
represents a set of security
credentials

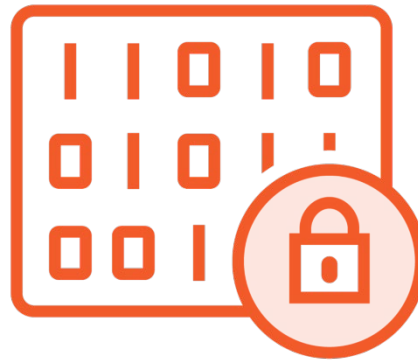
The object can be passed
as a parameter



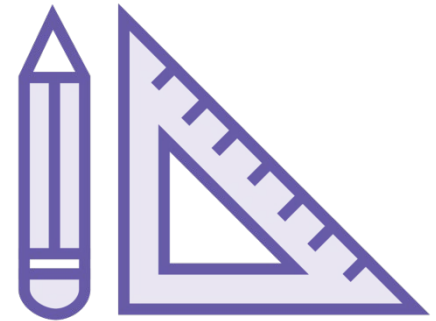
Creating PSCredential Objects



Create a Credential
Object using
Get-Credential Cmdlet



Create a Secure String
Object then Pass to
PSCredential Object



Create a PSCredential
Object

Creating PSCredential Objects

Creating PSCredential Object using Get-Credential

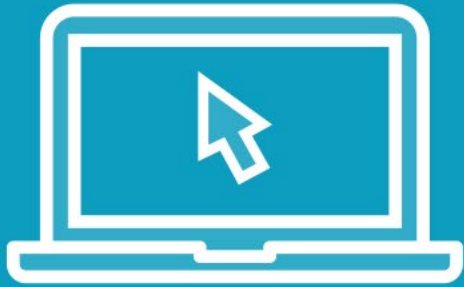
```
$creds = Get-Credential  
$creds = Get-Credential -Credential "GLOBOMANTICS\user"
```

Create PSCredential from Username and Secure Password

```
$user = "GLOBOMANTICS\user"  
$pwd = ConvertTo-SecureString "Password" -AsPlainText -Force  
  
$creds = New-Object System.Management.Automation.PSCredential ($user, $pwd)  
  
$creds = New-Object -TypeName System.Management.Automation.PSCredential `   
                -ArgumentList $user, $pwd
```



Demo



Creating PSCredential Objects



Implementing Trusted Hosts



What are Trusted Hosts?



Workgroup workstation allowed to Remotely Manage Domain Joined workstation or server



Different domain joined workstation allowed to Remotely Manage Domain Joined workstation or server



Workstation allowed to Remotely Management using different credentials



Why use Trusted Hosts?



Remote System is not
part of a Domain



Remote System is part
of an Untrusted
Domain



Connecting to Remote
System using
Local not Domain
Credentials

Trusted Host Entries



IP Version 4
Address



IP Version 6
Address



Fully Qualified
Domain Name
or Domain
Name



Wildcard

Creating Trusted Host Entries

Create using a Command Prompt

```
winrm set winrm/config/client @{TrustedHosts = "192.168.1.36"}
```

Create using PowerShell

```
Set-Item WSMan:\localhost\Client\TrustedHosts -Value "192.168.1.36" -Force
```

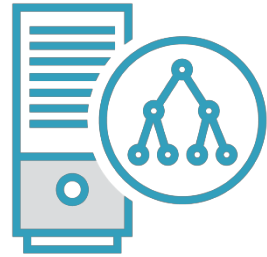
Adding Trusted Host Entries using PowerShell

```
$hosts = (Get-Item -Path WSMan:\localhost\Client\TrustedHosts).Value  
Set-Item -Path WSMan:\localhost\Client\TrustedHosts "$hosts, 192.168.1.36" -Force
```



Globomantics Trusted Hosts

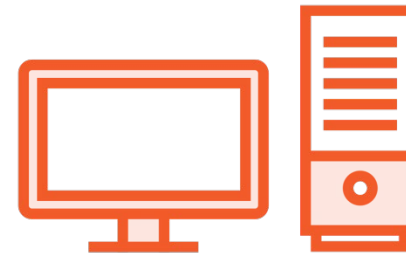
Domain
Joined



Active Directory
Server
(10.0.0.5/24)



File Server
(10.0.0.7/24)



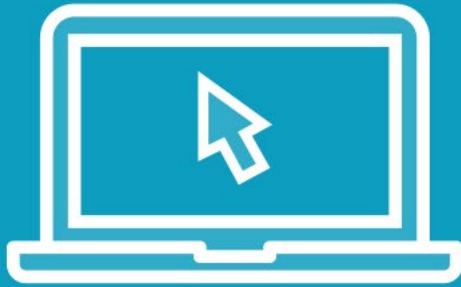
Administration
Workstation
(10.0.0.8/24)

Not Domain
Joined

```
Set-Item WSMAN:\localhost\Client\TrustedHosts `
-Value "10.0.0.8" -Force
```



Demo



Connect to Remote System without
Trusted Host Entry

Add Trusted Host Entry

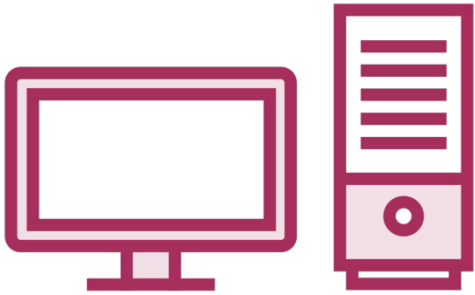
Connect to Remote System with Trusted
Host Entry



Creating and Removing PowerShell Sessions



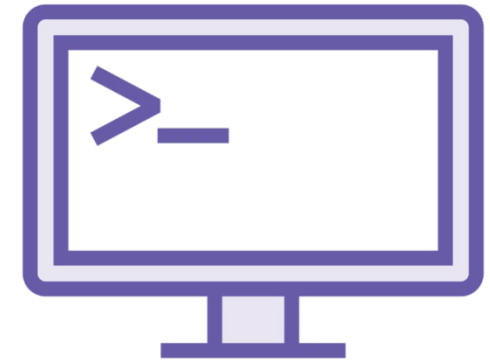
Create Remote PowerShell Sessions



Use `-ComputerName`
Parameter



Utilize
`Invoke-Command`



Use Interactive
PowerShell Session



Create a Session Using Invoke-Command

Execute a Command on a Remote Computer

```
Invoke-Command -ComputerName "10.0.0.5" -ScriptBlock { Get-ComputerInfo }
```

Execute a Script on a Remote Computer

```
Invoke-Command -ComputerName "10.0.0.5" -FilePath "C:\Scripts\Script.ps1"
```

Create Persistent Connection and Execute Command

```
$ps = New-PSSession -ComputerName "10.0.0.5"
```

```
Invoke-Command -Session $ps { Get-ComputerInfo }
```



Using Interactive Sessions

Connect to a PowerShell Session on the specified Computer

```
Enter-PSSession -ComputerName "10.0.0.5"
```

Create a New PowerShell Session on the specified Computer

```
New-PSSession -ComputerName "10.0.0.5"
```

Create a Persistent PowerShell Session on the specified Computer

```
$ps = New-PSSession -ComputerName "10.0.0.5"
```

Connect to PowerShell Session on the specified Computer, use Port and Credential

```
Enter-PSSession -ComputerName "10.0.0.5" -Port 99 -Credential "GLOBOMANTICS\user"
```

Remove a PowerShell Session

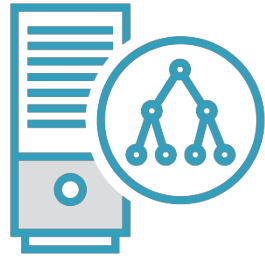
```
$ps = Get-PSSession
```

```
Remove-PSSession -Session $ps
```



Globomantics Sessions

Session 1



Active Directory
Server
(10.0.0.5/24)



File Server
(10.0.0.7/24)

Session 2



Administration
Workstation
(10.0.0.8/24)

\$s1 = 10.0.0.5
\$s2 = 10.0.0.7

```
Enter-PSSession -Session $s1 {Get-ComputerInfo}
```

```
Enter-PSSession -Session $s2 {Get-ComputerInfo}
```



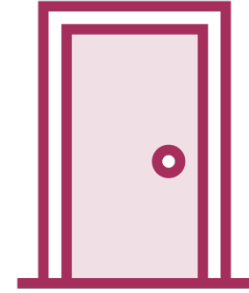
PowerShell Session Management



Get



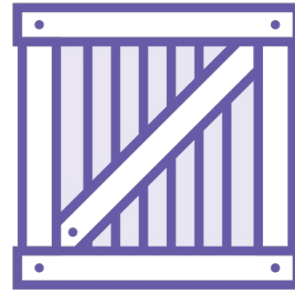
New / Remove



Exit



Connect



Import / Export



Enter



Modifying Session Properties

K	V

The New-PSSessionOption command modifies session properties

- Compression
- Access Properties
- Certificate Settings
- Encryption
- Language
- Application Arguments
- Timeouts



Create Session Options

Create Default Session Options

`New-PSSessionOption`

Define the Session Options

`$options = New-PSSessionOption`

`$options.OpenTimeout = (New-Timespan -Minutes 4)`

`$options.NoEncryption = $true`

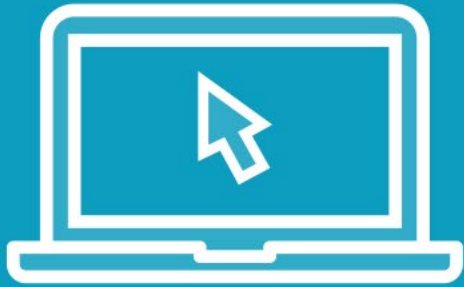
`$options.UICulture = (Get-UICulture)`

Create PowerShell Session using the created Session Options

`New-PSSession -ComputerName "10.0.0.5" -SessionOption $options`



Demo



Create PowerShell Remote Session

Reuse an Existing Remote PowerShell Session

Remove PowerShell Sessions



Summary



Goal: Manage Remote PowerShell Sessions

Used a "PSCredential" Object to Store Remote Credentials

Secured Remote PowerShell by using Trusted Hosts

Created and Managed Remote PowerShell Sessions

Up Next:

Executing Commands Using Multiple Remote Sessions

