Admin Server 2 - Manage a core Install | Anthony Adams

This week we will look at managing a server without a GUI.

We will look at some commands you will use. Also we will look at how PowerShell can be used to do a level of programming. It isn't the most robust and you wont be building games or websites with it but it can handle tasks you need to manage your server.

Built In Variables

One of the things you will need to learn to do this is variables. PowerShell has some built in system variables that can be used in scripts or in the command line. Let's look over the some of the basics.

$true – a system Boolean variable that contains or can be compared to the Boolean value True

$false – a system Boolean variable that contains or can be compared to the Boolean
value False

$null – a system variable that contains or can be compared to a null value, this means no value has been set. It is not = to 0 or blank. For example if there is an input and the user does not enter a value. It will return $null.

$host – a system object variable that contains information about the PowerShell host system. So it contains all the information for the settings of the host system.

$_ - current instance of an object (acts as a place holder), current value in an array.

Comparison Operators

PowerShell uses a number of Comparison Operators that can be used in scripts and on the command
line. The following is not a complete list.

-eq value is equal to

-ne value is not equal to

-like string wild card comparison

-lt value is less than

-gt value is greater than

-ge value is greater than or equal to

-le value is less than or equal to

$true –ne $false – returns true || True not equal to false (true)
“This” –like “T*” – returns true || “This” compared to “T*” is true because the * “T*” is a wild card and can be anything.
“This” –eq “This” –returns true || “This” and “This” are identical

Cmdlets

Cmdlests for managing processes:

Get-process - Gets the processes that are running on the local computer

Without parameters, this cmdlet gets all of the processes on the local computer. You can also specify a particular process by process name or process ID (PID) or pass a process object through the pipeline to this cmdlet.

Start-process - Starts one or more processes on the local computer.

The Start-Process cmdlet starts one or more processes on the local computer. By default, Start-Process creates a new process that inherits all the environment variables that are defined in the current process.

Stop-process - Stops one or more running processes.

The Stop-Process cmdlet stops one or more running processes. You can specify a process by process name or process ID (PID), or pass a process object to Stop-Process. Stop-Process works only on processes running on the local computer.

Cmdlets for managing services:

Get-service - Gets the services on the computer.
The Get-Service cmdlet gets objects that represent the services on a computer, including running and stopped services. By default, when Get-Service is run without parameters, all the local computer's services are returned.

Start-service - Starts one or more stopped services (Windows only)
The Start-Service cmdlet sends a start message to the Windows Service Controller for each of the specified services. If a service is already running, the message is ignored without error. You can specify the services by their service names or display names, or you can use the **InputObject** parameter to supply a service object that represents the services that you want to start.

Restart-service - Stops and then restarts one or more services (Windows only)
The Restart-Service cmdlet sends a stop message and then a start message to the Windows Service Controller for a specified

service. If a service was already stopped, it is started without notifying you of an error. You can specify the services by their service names or display names, or you can use the InputObject parameter to pass an object that represents each service that you want to restart.

Stop-service - Stops one or more running services (Windows only)
The Stop-Service cmdlet sends a stop message to the Windows Service Controller for each of the specified services. You can specify the services by their service names or display names, or you can use the InputObject parameter to pass a service object that represents the service that you want to stop.

Set-service - Starts, stops, and suspends a service, and changes its properties. (Windows only)
The Set-Service cmdlet changes the properties of a service such as the Status, Description, DisplayName, and StartupType. Set-Service can start, stop, suspend, or pause a service. To identify a service, enter its service name or submit a service object. Or, send a service name

or service object down the pipeline to Set-Service.

Process vs Service

A process is an instance of a running program, a service is a process running running in the background like a protocol, it does not interact with the desktop. A process can spawn multiple processes. If you check the task manager you can see this in action. An easy example of this is when running a browser, you will see multiple processes running as a result of that browser being open.

Selecting specific object properties or instances

Select Object - Selects objects or object properties.

The Select-Object cmdlet selects specified properties of an object or set of objects. It can also select unique objects, a specified number of objects, or objects in a specified position in an array.

Cmdlet for sorting objects

Sort-object - Sorts objects by properties or values.

The Sort-Object cmdlet sorts objects in ascending or descending order based on object property values. If sort properties aren't included in a command, PowerShell uses default sort properties of the first input object. If the input object's type has no default sort properties, PowerShell attempts to compare the objects themselves.

Filtering objects based on conditions

Where-object - Selects objects from a collection based on their property values.

The Where-Object cmdlet selects objects that have particular property values from the collection of objects that are passed to it. For example, you can use the Where-Object cmdlet to select files that were created after a certain date, events with a particular ID, or computers that use a particular version of Windows.

Managing Windows server roles and features

Get-WindowsFeature

Gets information about Windows Server roles,

role services, and features that are available for installation and installed on a specified server.

Install-WindowsFeature

Installs one or more roles, role services, or features on either the local or a specified remote server that is running Windows Server.

Uninstall-WindowsFeature

Uninstalls specified Windows Server roles, role services, and features from a computer that is running Windows Server. By adding the Remove parameter, also deletes feature files or payload, from a computer.

Cmdlet for obtaining a windows credential object (Username and password)

Get-Credential

Gets a credential object based on a user name and password.

Forcing order of operations

Brackets can be used to force a specific order of

execution. All commands within
the brackets are executed first, then the rest of the command will be executed

The following command will execute the Get-Credential cmdlet to obtain
credentials to pass to the Start-Process cmdlet's –Credential parameter

Start-Process cmd –Credential (Get-Credential Acme\Anthony.Green)

Object properties and methods

• In the following command the Get-Process PowerShell
cmdlet will execute, returning all processes whose name
is PowerShell

• Then the .ID portion of the command will list the ID
property for all returned processes
(Get-Process PowerShell).ID

• The following command will use the Stop method of the
returned process objects to stop each instance

```
(Get-Process PowerShell).Stop()
```