PowerShell

Connect commands into a pipeline

This hands-on lab demonstrates how to effectively use PowerShell pipelines to process objects. You'll explore how to retrieve and inspect system processes, chain cmdlets together using the pipeline (|), and apply principles like filtering leftand formatting right. You'll also use tools like Get-Member, Select-Object, and Format-* to understand the structure and behavior of objects passed through the pipeline. This lab helps you build a mental model of PowerShell's object-oriented pipeline.

1 Inspect Process Objects with Get-Member

Displays the object type and all available properties and methods returned by `Get-Process`.

```
PowerShell> Get-Process | Get-Member
```

```
  TypeName: System.Diagnostics.Process

Name                    MemberType    Definition
----                ----------     ----------
Handles                  AliasProperty  Handles = Handlecount
Name                    AliasProperty  Name = ProcessName
```

When you pass the results of a command to Get-Member, Get-Member returns information about an object, like:

- The type of object being passed to Get-Member.
- The Properties of the object that may be evaluated.
- The Methods of the object that may be executed.

2 Get a Specific Process (zsh)

Returns information about the currently running `zsh` process.

```
PowerShell> Get-Process zsh
```

```
 NPM(K)    PM(M)      WS(M)     CPU(s)      Id  SI ProcessName
 ------    -----      -----     ------      --  -- -----------
      0     0.00       0.41       0.05   17244 …44 zsh
```

3 Display All Properties in List Format

Formats all properties of the process vertically for easier reading.

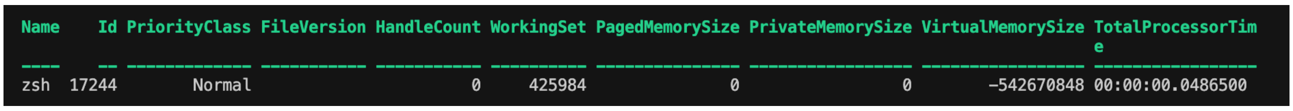
```
PowerShell> Get-Process zsh | Format-List -Property *
```

```
Name                      : zsh
Id                    : 17244
PriorityClass             : Normal
FileVersion               :
HandleCount               : 0
…
```

4 Display All Properties in Table Format

Formats all properties into a wide table, ideal for viewing a few properties side by side.

```
PowerShell> Get-Process zsh | Format-Table -Property *
```



```
Name    Id PriorityClass FileVersion HandleCount WorkingSet PagedMemorySize PrivateMemorySize VirtualMemorySize TotalProcessorTim
                                                                                                                e
----    -- ------------- ----------- ----------- ---------- --------------- ----------------- ----------------- -----------------
zsh  17244        Normal                       0     425984               0                 0        -542670848 00:00:00.0486500
```

5 Filter Members Starting with ‘C’

Finds members of the object whose names begin with the letter "C", such as `CPU`.

```
PowerShell> Get-Process zsh | Get-Member -Name C*
```

```
   TypeName: System.Diagnostics.Process

Name          MemberType     Definition
----          ----------     ----------
…

CPU           ScriptProperty System.Object CPU {get=$this.TotalProcessorTime.TotalSeconds;}
```

6 Select Specific Properties of a Process

Extracts the `Id`, `Name`, and calculated `CPU` usage of the process.

```
PowerShell> Get-Process zsh | Select-Object -Property Id, Name, CPU
```

```
   Id Name   CPU
   -- ----   ---
17244 zsh  0.049
```

7 Sort All Processes by Name Descending

Sorts the list of processes alphabetically in reverse order by name.

```
PowerShell> Get-Process | Sort-Object -Descending -Property Name
```

```
 NPM(K)    PM(M)      WS(M)     CPU(s)      Id  SI ProcessName
 ------    -----      -----     ------      --  -- -----------
      0     0.00       0.41       0.05   17244 …44 zsh
      0     0.00       0.00       0.00     541 541 WirelessRadioManagerd
      0     0.00       0.00       0.00     414 414 WindowServer
```

8 Sort Processes by CPU Usage and Name

Sorts processes first by highest CPU time, then by name for consistent secondary sorting.

```
PowerShell> Get-Process | Sort-Object -Descending -Property CPU, Name
```

```
 NPM(K)    PM(M)      WS(M)     CPU(s)      Id  SI ProcessName
 ------    -----      -----     ------      --  -- -----------
      0     0.00      16.77   2,083.77    3688   1 Tempus Stopwatch
      0     0.00      83.72     617.78    4625   1 Safari
```

9 Get Top 3 High-CPU Processes (>2 sec)

Filters for CPU usage over 2 seconds, sorts the results, and selects the top 3 processes.

```
PowerShell> Get-Process | Where-Object CPU -gt 2 | Sort-Object CPU -Descending | Select-
Object -First 3
```

```
 NPM(K)    PM(M)      WS(M)     CPU(s)      Id  SI ProcessName
 ------    -----      -----     ------      --  -- -----------
```

```
      0     0.00      26.41   2,129.69    3688   1 Tempus Stopwatch
      0     0.00      82.59     632.72    4625   1 Safari
      0     0.00     359.78     515.33    4653 …53 com.apple.WebKit.WebContent
```
———————————————————————————————————————

- **Get-Process**
  - Retrieves a list of all processes running on the local system.
  - Each process object includes properties like CPU, Id, ProcessName, StartTime, etc.
- | (Pipeline)
  - Sends the output of Get-Process to the next command for further processing.
- **Where-Object CPU -gt 2**
  - Filters the process list to include only those where the CPU property (total processor time in seconds) is greater than 2 seconds.
  - Equivalent full syntax:

    ```
    Where-Object { $_.CPU -gt 2 }
    ```

- **Sort-Object CPU -Descending**
  - Sorts the filtered processes by their CPU usage in descending order (highest CPU usage first).
- **Select-Object -First 3**
  - Selects the top 3 processes from the sorted list (i.e., the 3 processes with the highest CPU usage over 2 seconds).

10 Get Bottom 3 High-CPU Processes (>2 sec)
Returns the last 3 entries among high-CPU processes after sorting in descending order.

```
PowerShell> Get-Process | Where-Object CPU -gt 2 | Sort-Object CPU -Descending | Select-Object -Last 3
```
———————————————————————————————————————
```
 NPM(K)    PM(M)      WS(M)     CPU(s)      Id  SI ProcessName
 ------    -----      -----     ------      --  -- -----------
      0     0.00      12.02       2.35    8147   1 PowerChime
      0     0.00      11.94       2.31     629   1 usernotificationsd
      0     0.00       9.95       2.13    4193 …93 com.apple.WebKit.GPU
```
———————————————————————————————————————

11 Filtering Late: A Suboptimal Pattern
Selects all process names and then filters by name—inefficient and contrary to best practice.

In a pipeline statement, filtering left means filtering for the results you want as early as possible.

```
PowerShell> Get-Process | Select-Object Name | Where-Object Name -eq "Google Chrome"
```
———————————————————————————————————————
```
Name
----
Google Chrome
```
———————————————————————————————————————

This statement doesn't follow the filtering left principle, because it operates on all the processes, attempts to format the response, and then filters at the end.

12 Filtering Left: Preferred Pattern

Filters the process list first, then selects only the desired columns for output.

```
PowerShell> Get-Process | Where-Object Name -eq "Safari" | Select-Object Name
————————————————————————————————————————
Name
----
Safari
————————————————————————————————————————
```

13 Filtering with Parameter Instead of Pipeline

Uses the `-Name` parameter directly to filter, improving performance and readability.

```
PowerShell> Get-Process -Name "AnkiApp" | Select-Object Name
————————————————————————————————————————
Name
----
AnkiApp
————————————————————————————————————————
```

In this version, the parameter -Name does the filtering

14 Formatting Right: Preserving Object Structure

Formats selected properties and inspects the resulting object, which is still process-based.

Whereas filtering left means to filter something as early as possible in a statement, formatting right means to format something as late as possible in the statement.

```
PowerShell> Get-Process zsh | Select-Object Name, CPU | Get-Member
————————————————————————————————————————
  TypeName: Selected.System.Diagnostics.Process

Name        MemberType   Definition
----        ----------   ----------
Equals      Method       bool Equals(System.Object obj)
GetHashCode Method       int GetHashCode()
GetType     Method       type GetType()
ToString    Method       string ToString()
CPU         NoteProperty System.Double CPU=0.04865
Name        NoteProperty string Name=zsh
————————————————————————————————————————
```

15 Formatting Too Early Breaks the Pipeline

Uses `Format-Table` before piping to another cmdlet—returns a formatting object, not a process.

```
PowerShell> Get-Process zsh | Format-Table Name, CPU | Get-Member
————————————————————————————————————————
  TypeName: Microsoft.PowerShell.Commands.Internal.Format.FormatStartData

Name                                    MemberType Definition
----                                    ---------- ----------
Equals                                  Method     bool Equals(System.Object obj)
GetHashCode                             Method     int GetHashCode()
————————————————————————————————————————
```

Focusing just on the types we get back, we're getting back something different.

16 Format-Table Disrupts Data Access

After formatting with `Format-Table`, selecting properties fails due to lost object structure.

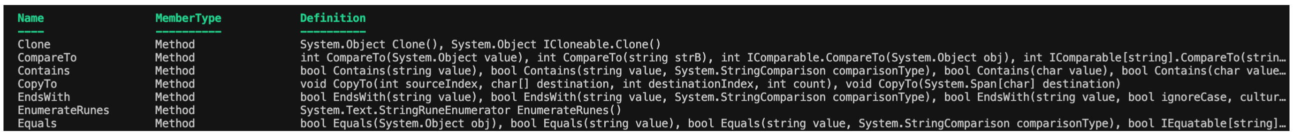
```
PowerShell> Get-Process zsh | Format-Table Name, CPU | Select-Object Name, CPU
———————————————————————————————————————————
Name CPU
---- —
———————————————————————————————————————————
```

It's empty, because Format-Table transformed the object containing your results by placing data into other properties.

17 Inspecting a String Object

Displays the type and available methods of a simple string using `Get-Member`.

```
PowerShell> "a string" | Get-Member
```



```
Name           MemberType    Definition
----           ----------    ----------
Clone          Method        System.Object Clone(), System.Object ICloneable.Clone()
CompareTo      Method        int CompareTo(System.Object value), int CompareTo(string strB), int IComparable.CompareTo(System.Object obj), int IComparable[string].CompareTo(strin…
Contains       Method        bool Contains(string value), bool Contains(string value, System.StringComparison comparisonType), bool Contains(char value), bool Contains(char value…
CopyTo         Method        void CopyTo(int sourceIndex, char[] destination, int destinationIndex, int count), void CopyTo(System.Span[char] destination)
EndsWith       Method        bool EndsWith(string value), bool EndsWith(string value, System.StringComparison comparisonType), bool EndsWith(string value, bool ignoreCase, cultur…
EnumerateRunes Method        System.Text.StringRuneEnumerator EnumerateRunes()
Equals         Method        bool Equals(System.Object obj), bool Equals(string value), bool Equals(string value, System.StringComparison comparisonType), bool IEquatable[string]…
```

18 Format Output as a Table

Formats the members of the string object into a readable table layout.

```
PowerShell> "a string" | Get-Member | Format-Table
———————————————————————————————————————————
   TypeName: System.String

Name             MemberType       Definition
----             ----------       ----------
Clone            Method           System.Object Clone(), System.Object ICloneable.Clone()
CompareTo        Method           int CompareTo(System.Object value), int CompareTo(string
strB), int IComparable.CompareTo(System.Object obj), int IComparable[string].CompareTo(strin…
Contains         Method           bool Contains(string value), bool Contains(string value,
System.StringComparison comparisonType), bool Contains(char value), bool Contains(char
value…
CopyTo           Method           void CopyTo(int sourceIndex, char[] destination, int
destinationIndex, int count), void CopyTo(System.Span[char] destination)
EndsWith         Method           bool EndsWith(string value), bool EndsWith(string value,
System.StringComparison comparisonType), bool EndsWith(string value, bool ignoreCase,
cultur…
EnumerateRunes   Method           System.Text.StringRuneEnumerator EnumerateRunes()
———————————————————————————————————————————
```

19 Format Output as a List

Displays member data in a vertical format—great for verbose detail.

```
PowerShell> "a string" | Get-Member | Format-List
————————————————————————————————————————
TypeName   : System.String
Name       : Clone
MemberType : Method
Definition : System.Object Clone(), System.Object ICloneable.Clone()

TypeName   : System.String
Name       : CompareTo
MemberType : Method
Definition : int CompareTo(System.Object value), int CompareTo(string strB), int
IComparable.CompareTo(System.Object obj), int IComparable[string].CompareTo(string other)

…
————————————————————————————————————————
```