

The Age of PowerShell

The IT industry is changing at a rapid pace, and gone are the days when administrators managed just a handful of servers. In today's rapid deployments, system administrators are required to handle tens of hundreds of servers that are spun up or destroyed, as demand dictates. The industry is increasingly dealing with this trend with a handful of technologies that automate many tasks that were done by hand in the past, including the creation of virtual machines, containers, configuration of operating systems, networking parameters, audit setting, and installation of applications or features. Tools such as Chef, Ansible and Puppet manage the configuration of servers, and others such as Kubernetes and Docker can be used to perform lightweight virtualization and application isolation, while Amazon Web Services (AWS) and Microsoft Azure cloud services give rise to fast and massive infrastructure with little capital expenditure for applications.

Linux and UNIX have long been prime examples of systems that supported these orchestration and automatization tools, given that most of their configuration was already performed either through small shell-based scripts or configuration files. But, a few years ago, even Microsoft saw the writing on the wall and began pushing technologies such as PowerShell and Desired State Configuration (DSC) to fill the gap.

Some auditors still rely on old and outdated scripts to harvest information and tables that have to be analyzed one by one in search of issues when they should perhaps be embracing those same orchestration and automation tools to help save time analyzing systems.

Enter PowerShell

PowerShell is a shell programming language and an automation tool. It is now on version 6.0 and has been open-sourced.¹ It also has versions for most major operating systems such as Linux, Windows and macOS and has gained many modules to administer other Microsoft technologies such as Active Directory, Structured Query Language (SQL) Servers, SharePoint and Skype for Business, to name a few.

Even as a shell, PowerShell is a bit odd for those familiar with KornShell (ksh) or Bash instead of this object-based language. If one is not a programmer outside of being an auditor, these terms may be unfamiliar, but they are not very difficult to understand.

Explaining PowerShell is easier through examples.

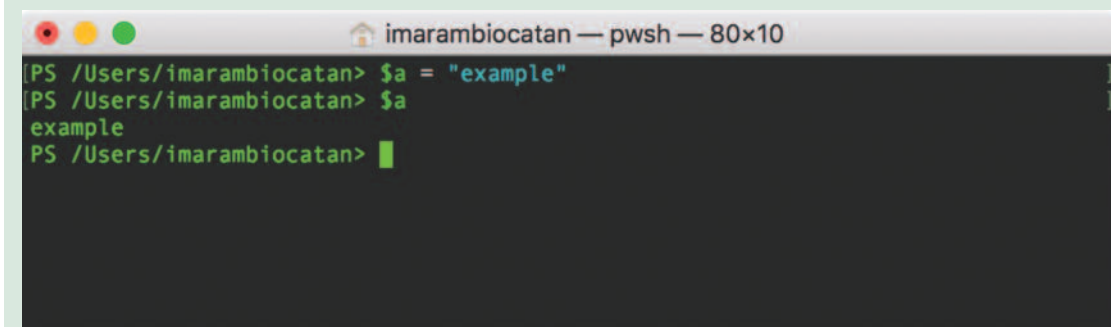
Figure 1 is an example that shows how to create a variable that contains some text.

Variables or “objects” in which one can store things for later use start with a \$ sign and, in this case, the variable is called *a* and has some text in it. Objects are just things that have abilities (methods) and properties. Objects are of a certain type that define their properties and methods. A real-life example of



Ignacio Marambio Catán, CISA, CRISC, CEH, CISSP, Security+ Has been an IT consultant for half of his professional career, after which he performed security operations, where the main objective was compliance. He also held audit-related positions with a focus on risk measurements. It was during his time performing security operations that he became a believer in automation, which later was used while auditing systems. Lately, Catán has been reading about machine learning and completed a Microsoft Professional Program in Data Science. In 2016, Catán was awarded the Certified Information Systems Auditor® (CISA®) Geographic Excellence Award.

Figure 1—A First PowerShell Example



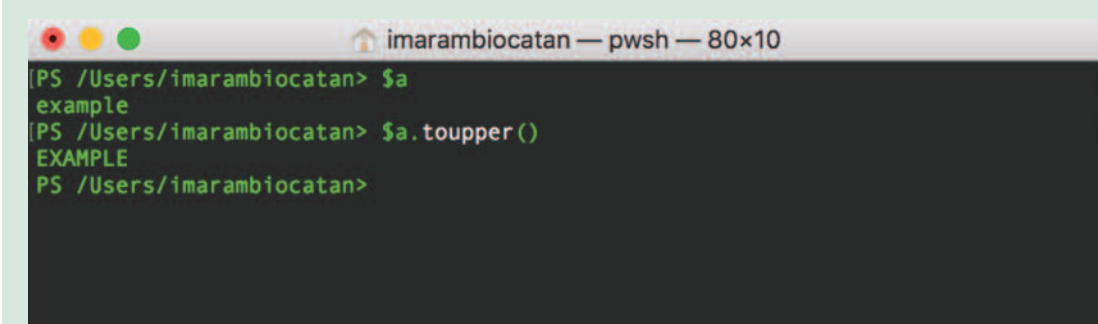
```
imarambiocatan — pwsh — 80x10
[PS /Users/imarambiocatan> $a = "example"
[PS /Users/imarambiocatan> $a
example
PS /Users/imarambiocatan> ]
```

an object would be a car; a property of the car would be its color (e.g., yellow) and it would have a method called "honk," which would honk the horn. For a PowerShell example, assume text is desired in uppercase.

The ToUpper method shown in **figure 2** exists because the variable *a* is of a given type or class; in this case, it is a string.

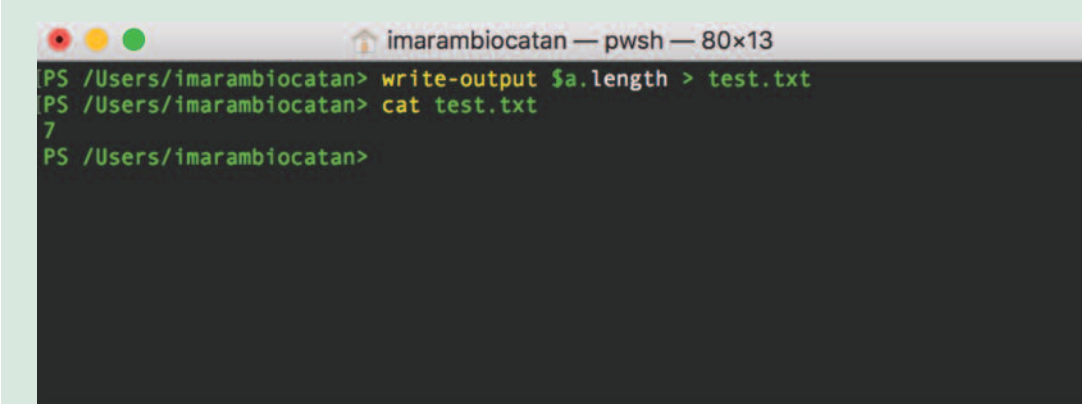
Creating variables is not the only thing that can be done with PowerShell, of course. There is the usual redirection with the > character and, as in UNIX, there is the *cat* command. *Length* is also a property of a string variable. For those who are not familiar with UNIX, *cat* is used to output the contents of a file to the screen. Also, the redirection operand is used to redirect the output of a given command to a file. An example can be seen in **figure 3**.

Figure 2—PowerShell Methods



```
imarambiocatan — pwsh — 80x10
[PS /Users/imarambiocatan> $a
example
[PS /Users/imarambiocatan> $a.toupper()
EXAMPLE
PS /Users/imarambiocatan> ]
```

Figure 3—Unix-Like Operands



```
imarambiocatan — pwsh — 80x13
[PS /Users/imarambiocatan> write-output $a.length > test.txt
[PS /Users/imarambiocatan> cat test.txt
7
PS /Users/imarambiocatan> ]
```

PowerShell also has another great feature, the pipe (see **figure 4**), which also works much as it does in UNIX.

ls, as in UNIX, lists files where the pipeline, the `|` character, passes objects one by one to the next command, which, in this case, is called *where*, and it is used to filter what is in the pipeline. What *-match* does should be obvious here. There are also *-eq*, *-ne*, *-lt*, *-gt* and many others² in addition to mean, equal, not equal, less than and greater than.

Objects can also be grouped in variables and there is also a function to go through them one by one.

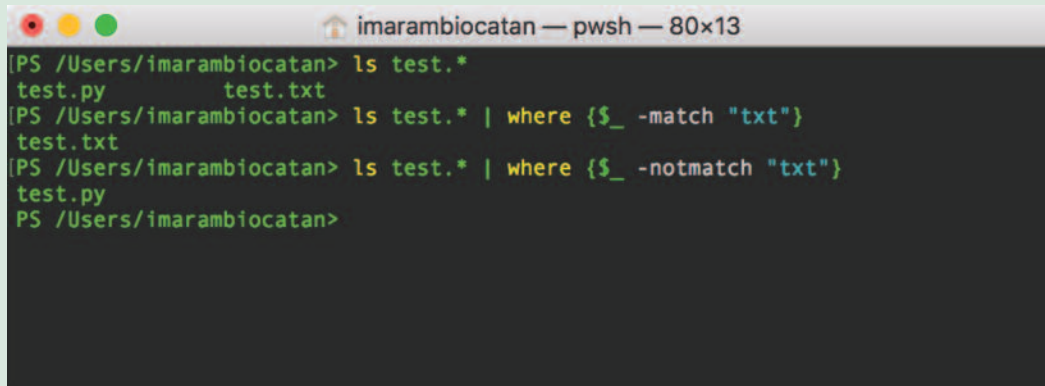
Figure 5 introduces the *if* control structure and blocks of code using the curly braces. There is much more to be discovered about structures, and readers are encouraged to investigate further on

their own. Modules and extensions to the core PowerShell commands to perform other advanced operations such as managing databases, mail servers and domain controllers can be installed using *install-module*, as can be seen in **figure 6**.

In the case of the *sqlserver* module, it includes about 93 commands related to the database, which can be used to create, modify or delete instances, databases, database users, store procedures, and change and show configurations.

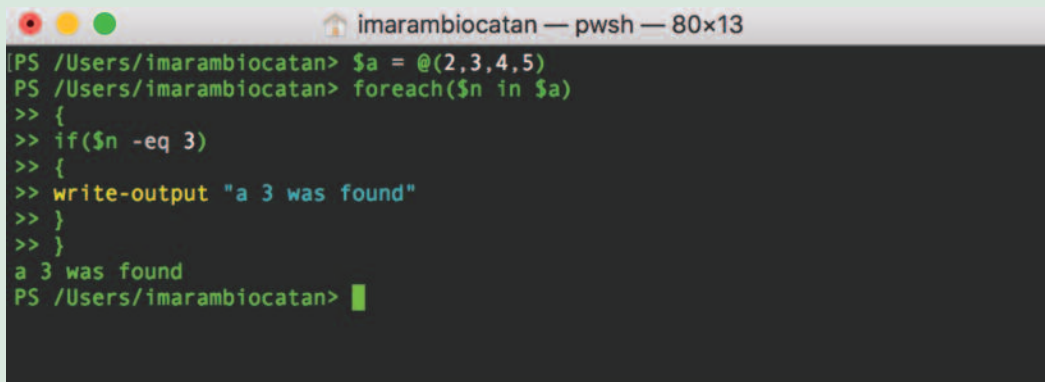
In **figure 7**, here using Windows 2012 R2, the example uses *get-sqlinstance* to get all the instances running in the server, and later a pipeline is used to get all the databases in all the instances. This is a newly installed test server, so there is very little here.

Figure 4—The Pipe



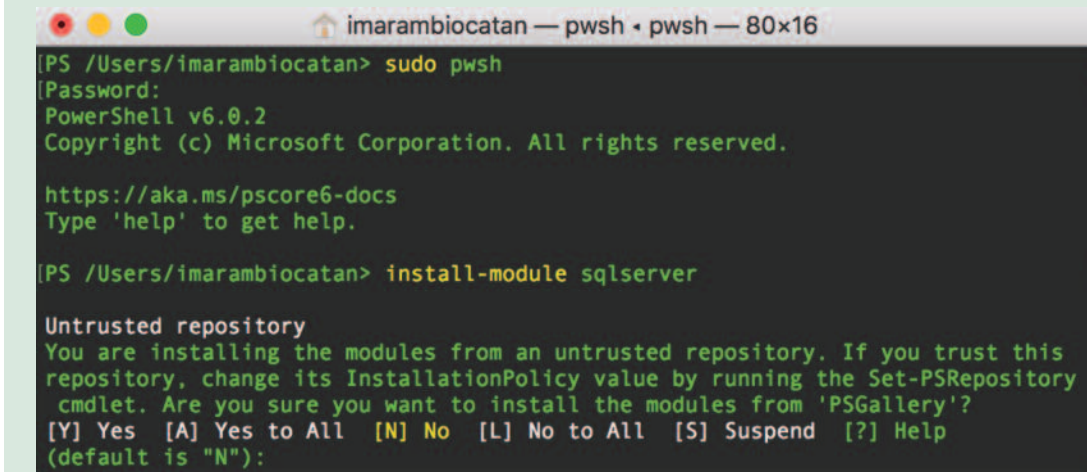
```
imarambiocatan — pwsh — 80x13
[PS /Users/imarambiocatan> ls test.*
test.py      test.txt
[PS /Users/imarambiocatan> ls test.* | where {$? -match "txt"}
test.txt
[PS /Users/imarambiocatan> ls test.* | where {$? -notmatch "txt"}
test.py
[PS /Users/imarambiocatan>
```

Figure 5—Control Flow



```
imarambiocatan — pwsh — 80x13
[PS /Users/imarambiocatan> $a = @(2,3,4,5)
[PS /Users/imarambiocatan> foreach($n in $a)
>> {
>> if($n -eq 3)
>> {
>> write-output "a 3 was found"
>> }
>> }
a 3 was found
[PS /Users/imarambiocatan>
```

Figure 6—Installing Modules



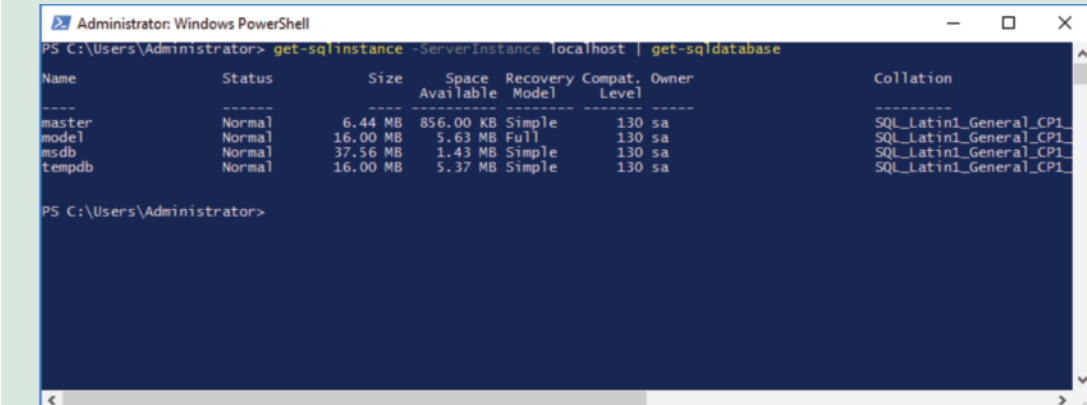
```
[PS /Users/imarambiocatan] sudo pwsh
[Password:
PowerShell v6.0.2
Copyright (c) Microsoft Corporation. All rights reserved.

https://aka.ms/pscore6-docs
Type 'help' to get help.

[PS /Users/imarambiocatan] install-module sqlserver

Untrusted repository
You are installing the modules from an untrusted repository. If you trust this
repository, change its InstallationPolicy value by running the Set-PSRepository
cmdlet. Are you sure you want to install the modules from 'PSGallery'?
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help
(default is "N"):
```

Figure 7—The SQL Server PowerShell Module



```
PS C:\Users\Administrator> get-sqlinstance -ServerInstance localhost | get-sqldatabase
```

Name	Status	Size	Space Available	Recovery Model	Compat. Level	Owner	Collation
master	Normal	6.44 MB	856.00 KB	Simple	130	sa	SQL_Latin1_General_CP1
model	Normal	16.00 MB	5.63 MB	Full	130	sa	SQL_Latin1_General_CP1
msdb	Normal	37.56 MB	1.43 MB	Simple	130	sa	SQL_Latin1_General_CP1
tempdb	Normal	16.00 MB	5.37 MB	Simple	130	sa	SQL_Latin1_General_CP1

```
PS C:\Users\Administrator>
```

On to DSC

DSC is the configuration management feature of PowerShell. Its main purpose is pushing settings to a set of servers. It is like a more extreme version of Group Policies, a Microsoft Windows technology used to manage configuration parameters on groups of servers and client computers that are part of a network.

To use DSC, a DSC configuration has to be written, and it relies on DSC resources. DSC configurations are PowerShell scripts written in a declarative language specifying how a given thing should be configured. The specific thing is relying on a DSC

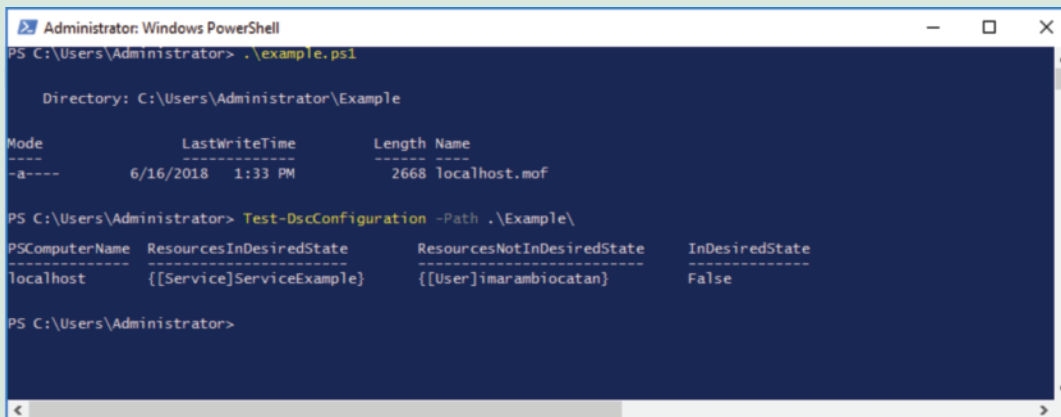
resource. Perhaps it is easier to again start with an example (figure 8) and build from there.

This little script uses two resources that are part of the *PSDesiredStateConfiguration* module. These are the *service* and *user* resources. This is, of course, a toy of an example that will ensure the Windows Update service is running and the user *imarambiocatan* exists. To use it, first compile it into a Managed Object Format (MOF)³ file, which is accomplished by running the example file. A test of the configuration using the *test-dscconfiguration* function is shown in figure 9.

Figure 8—A Simple DSC Example

```
configuration Example
{
    Import-DscResource -ModuleName PSDesiredStateConfiguration
    Node localhost
    {
        Service ServiceExample
        {
            Name           = "wuauserv"
            StartupType    = "Manual"
            State           = "Running"
        }
        User imarambiocatan
        {
            Ensure = "Present"
            UserName = "imarambiocatan"
        }
    }
}
Example
```

Figure 9—Testing a Server With DSC



The screenshot shows a Windows PowerShell window with the following commands and output:

```
PS C:\Users\Administrator> .\example.ps1

Directory: C:\Users\Administrator\Example

Mode                LastWriteTime         Length Name
----                -
-a-----         6/16/2018   1:33 PM           2668 localhost.mof

PS C:\Users\Administrator> Test-DscConfiguration -Path .\Example\

PSComputerName ResourcesInDesiredState ResourcesNotInDesiredState InDesiredState
-----
localhost      {[Service]ServiceExample} {[User]imarambiocatan}    False

PS C:\Users\Administrator>
```

From the output, it can be deduced that the *imarambiocatan* user does not exist in this server, but the Windows Update service is actually running. The exact same script can be used to fix the issues. But to illustrate how DSC works, in **figure 10**, the Windows update service is first stopped, and then DSC is used to fix the configuration of the system.

There are many available DSC resources by Microsoft and by others, and the things that can be done using DSC are endless, things that are as simple as starting a service or as complex as configuring an application with a database and a farm of distributed web servers. The DSC demo presented only explores so much, but an

administrator can also configure servers to fetch these MOF files from a centralized location, configure themselves accordingly and report their own status.⁴ The tools surrounding DSC are new and evolving in such a way that users can stumble into some issues with items such as the SQL Server DSC⁵ resources and submit the relevant bug reports along the way. Other tools such as DSC Environment Analyzer (DSCEA)⁶ can be used to generate Hypertext Markup Language (HTML) reports that span many hundreds of servers with very little help from the IT staff while also deploying the DSC resources needed by the servers to complete the tests.

Figure 10—Becoming Compliant

```

Administrator: Windows PowerShell
PS C:\Users\Administrator> stop-service "windows update"
PS C:\Users\Administrator> Test-DscConfiguration -Path .\Example\

PSComputerName ResourcesInDesiredState ResourcesNotInDesiredState InDesiredState
-----
localhost      -----
               {[Service]ServiceExample, [...] False

PS C:\Users\Administrator> Start-DscConfiguration -Path .\Example\

Id Name PSJobTypeName State HasMoreData Location Command
-- --
9 Job9 Configuratio... Running True localhost Start-DscConfiguratio...

PS C:\Users\Administrator> get-service "windows update"

Status Name DisplayName
-----
Running wuauerv Windows Update

PS C:\Users\Administrator> Get-LocalUser imarambiocatan

Name Enabled Description
-----
imarambiocatan True

PS C:\Users\Administrator>
  
```

Auditors should be able to leverage these tools to automate most of what today represents a sizeable time commitment on some of the IT-related engagements, and for once, sharing their scripts might make the life of administrators a lot easier in the future because they might be able to convert them into tools to automate the configuration of their own servers, resulting in a more secure environment. Furthermore, as more and more companies adopt development operations (DevOps), i.e., the practice that unifies operations and development to shorten the time between releases, making them more predictable and aligned to business objectives, automation will be key to their success. It is by understanding the tools that are being used for this automation that auditors will be able to effectively ask the right questions when it comes to auditing the processes in which they are involved.

Endnotes

- 1 Snover, J.; "PowerShell Is Open Sourced and Is Available on Linux," Microsoft Azure Blog, 18 August 2016, <https://azure.microsoft.com/en-us/blog/powershell-is-open-sourced-and-is-available-on-linux/>
- 2 Microsoft, "About Comparison Operators," 8 June 2017, https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.core/about/about_comparison_operators?view=powershell-6
- 3 Distributed Management Task Force, "Managed Object Format," 13 December 2012, https://www.dmtf.org/sites/default/files/standards/documents/DSP0221_3.0.0.pdf
- 4 Microsoft, "Azure Automation State Configuration Overview," 7 August 2018, <https://docs.microsoft.com/en-us/azure/automation/automation-dsc-overview>
- 5 Github, PowerShell/SqlServerDsc," <https://github.com/PowerShell/SqlServerDsc>
- 6 Microsoft, "DSCEA - DSC Environment Analyzer," <https://microsoft.github.io/DSCEA/>