APIs - PowerShell Cmdlets

PowerShell Cmdlets

To get started with Nutanix PowerShell Cmdlets, see PowerShell Cmdlets Reference on the Nutanix Support Portal.

The below will cover the Nutanix PowerShell Cmdlets, how to use them and some general background on Windows PowerShell.

PowerShell Versions

At the time of writing, the latest available version is PowerShell 7.3.4

Commands outlined in this Nutanix Bible chapter may not apply to earlier PowerShell versions.

Nutanix Cmdlets Versions

At the time of writing, the latest available version is Nutanix Cmdlets v2.0

Commands outlined in this Nutanix Bible chapter may not apply to other versions of the Nutanix Cmdlets.

PowerShell Basics

Windows PowerShell is a powerful shell and scripting language built on the .NET framework. It is a very simple to use language and is built to be intuitive and interactive. Within PowerShell there are a few key constructs/Items:

Cmdlets

Cmdlets are commands or .NET classes which perform a particular operation. They are usually conformed to the Getter/Setter methodology and typically use a Verb-Noun based structure. For example: Get-Process, Set-Partition, etc.

Piping Or Pipelining

Piping is an important construct in PowerShell (similar to its use in Linux) and can greatly simplify things when used correctly. With piping you're essentially taking the output of one section of the pipeline and using that as input to the next section of the pipeline. The pipeline can be as long as required (assuming there remains output which is being fed to the next section of the pipe). A very simple example could be getting the current processes, finding those that match a particular trait or filter and then sorting them:

```
Get-Service | where {$_.Status -eq "Running"} | Sort-Object Name
```

Piping can also be used in place of for-each, for example:

```
# For each item in my array
$myArray | %{
  # Do something
}
```

Key Object Types

Below are a few of the key object types in PowerShell. You can easily get the object type by using the .getType() method, for example: $someVariable.getType() will return the objects type.

Variables

```
$myVariable = "foo"
```

Note: You can also set a variable to the output of a series or pipeline of commands:

```
$myVar2 = (Get-Service | where {$_.Status -eq "Running"})
```

```
$myVar3 = (Get-Process | where {$_.ProcessName -eq "Chrome"})
```

In this example the commands inside the parentheses will be evaluated first then variable will be the outcome of that.

Array

```
$myArray = @("Value","Value")
```

Note: You can also have an array of arrays, hash tables or custom objects

Hash Table

```
$myHash = @{"Key" = "Value";"Key" = "Value"}
```

Useful Commands

Get the help content for a particular Cmdlet (similar to a man page in Linux)

```
Get-Help Cmdlet Name
```

Example:

```
Get-Help Get-Process
```

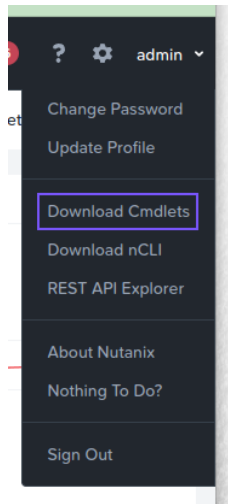List properties and methods of a command or object

```
<expression or object> | Get-Member
```

Example:

```
$someObject | Get-Member
```

Core Nutanix Cmdlets And Usage

The Nutanix Cmdlets can be downloaded directly from the Prism Central UI and can be found on the drop down in the upper right hand corner:

Nutanix Cmdlets v2.0 - Usage Examples

The current Nutanix Cmdlet v2.0 reference can be found on the Nutanix Portal.

List Nutanix Cmdlets

```
Get-Command -Module Nutanix.Prism.Common
```

```
Get-Command -Module Nutanix.Prism.Ps.Cmds
```

Connect To Prism Central

Prompt for all connection details and accept valid SSL certificates only:

```
Connect-PrismCentral
```

Supply connection details (excluding password), accept valid and invalid SSL certificates:

```
Connect-PrismCentral -AcceptInvalidSSLCerts -Server <prism_central_ip_addres> -UserName <username>
```

List All Virtual Machines

```
Get-VM
```

Get Nutanix VMs Matching A Certain String

Assign to variable:

```
$searchString = "my-vm-name"
$vms = Get-VM | where {$_.vmName -match $searchString}
```

Interactive:

```
$myVm = Get-VM | where {$_.vmName -match "myVmName"}
```

Interactive and formatted:

```
$myVm = Get-VM | where {$_.vmName -match "myVmName"} | ft
```

List All Storage Containers

```
Get-StorageContainer
```

Get Storage Container With Similar Names

```
Get-StorageContainer | where {$_.name -like "Nutanix*"}
```

List All Networks

```
Get-Network
```

Get Specific Network By Name

```
Get-Network | where ${_.name -eq "Default"}
```

Disconnect From Prism Central

```
Disconnect-PrismCentral
```

Legacy Nutanix Cmdlets Information

The information in the following section applies to legacy Nutanix Cmdlets and PowerShell versions only. It has been left here for legacy support. Where possible, new users are recommended to install the latest version of both PowerShell and the Nutanix Cmdlets.

Nutanix Cmdlets v1.0 - Usage Examples

The Nutanix Cmdlet v1.0 reference can be found on the Nutanix Portal.

Load Nutanix Snapin

Check if snapin is loaded and if not, load

```
if ( (Get-PSSnapin -Name NutanixCmdletsPSSnapin -ErrorAction SilentlyContinue) -eq $null )
{
    Add-PsSnapin NutanixCmdletsPSSnapin
}
```

List Nutanix Cmdlets

```
Get-Command | Where-Object{$_.PSSnapin.Name -eq "NutanixCmdletsPSSnapin"}
```

Connect To A Nutanix cluster

```
Connect-NutanixCluster -Server $server -UserName "myuser" -Password (Read-Host "Password: " -AsSecureString) -AcceptInvalidSSLCerts
```

Get Nutanix VMs Matching A Certain Search String

Set to variable

```
$searchString = "myVM"
$vms = Get-NTNXVM | where {$_.vmName -match $searchString}
```

Interactive

```
Get-NTNXVM | where {$_.vmName -match "myString"}
```

Interactive and formatted

```
Get-NTNXVM | where {$_.vmName -match "myString"} | ft
```

Get Nutanix vDisks

Set to variable

```
$vdisks = Get-NTNXVDisk
```

Interactive

```
Get-NTNXVDisk
```

Interactive and formatted

```
Get-NTNXVDisk | ft
```

Get Nutanix Storage Containers

Set to variable

```
$containers = Get-NTNXContainer
```

Interactive

```
Get-NTNXContainer
```

Interactive and formatted

```
Get-NTNXContainer | ft
```

Get Nutanix Protection Domains

Assign to variable

```
$pds = Get-NTNXProtectionDomain
```

Interactive

```
Get-NTNXProtectionDomain
```

Interactive and formatted

```
Get-NTNXProtectionDomain | ft
```

Get Nutanix Consistency Groups

Set to variable

```
$cgs = Get-NTNXProtectionDomainConsistencyGroup
```

Interactive

```
Get-NTNXProtectionDomainConsistencyGroup
```

Interactive and formatted

```
Get-NTNXProtectionDomainConsistencyGroup | ft
```