

TÀI LIỆU HƯỚNG DẪN HỌC MICROSOFT POWERSHELL CƠ BẢN

Created by: Lưu Hồ Phương

Nội dung

Chương 1: Giới thiệu PowerShell.....	3
1.1 PowerShell là gì?	3
1.2 Lợi ích của việc sử dụng PowerShell	3
1.3 Cài đặt PowerShell và cấu hình môi trường	3
Chương 2: Cú pháp cơ bản của PowerShell và Cmdlets	4
2.1 Cú pháp cơ bản của PowerShell	4
2.2 Sử dụng Cmdlets cơ bản	4
2.3 Ví dụ cụ thể	4
Chương 3: Biến, Điều kiện và Vòng lặp trong PowerShell	5
3.1 Biến trong PowerShell	5
3.2 Quản lý tiến trình process	7
3.3 Lệnh điều kiện trong PowerShell	9
3.4 Vòng lặp trong PowerShell	9
3.5 Ví dụ cụ thể	9
Chương 4: Hàm và Module trong PowerShell	13
4.1 Định nghĩa hàm	13
4.2 Tham số trong PowerShell	13
4.3 Module trong PowerShell	13
4.4 Ví dụ cụ thể	14
Chương 5: Thực Hành và Luyện Tập với PowerShell trong Windows Server	17
5.1 Thực Hiện Các Tác Vụ Thực Tế	17
5.2 Tự Động Hóa	17
Chương 6: Sử dụng PowerShell với Đám Mây Azure	22
6.1. Giới thiệu	22
6.2. Cài đặt và Cấu hình	22
6.3 Quản lý Tài Nguyên Đám Mây Azure	23
6.4 Quản lý danh tính Azure (Azure Identities)	32
6.5 Scripting Azure PowerShell	34

- **Tham khảo các khóa học của giảng viên Lưu Hồ Phương tại trang Udemy.com:**

1. Khóa học *AZ-104 Microsoft Azure Administrator Exam Prep (Tiếng Việt)*

Link: <https://www.udemy.com/course/tim-hieu-am-may-azure-voi-chung-chi-az-104-microsoft-azure/?couponCode=55E82A014F8A541C7A9A>

2. Khóa học *AWS Certified Solutions Architect - Associate (Tiếng Việt)*

Link: <https://www.udemy.com/course/aws-certified-solutions-architect-associate-tieng-viet/?couponCode=1E86455C1D8E9DA99B93>

3. Khóa học *LPIC-1: Linux System Administrator Exam 101&102 (Tiếng Việt)*

Link: <https://www.udemy.com/course/lpic-1-system-administrator-exam-101-102tieng-viet/?couponCode=504E5B8374080543F841>

4. Khóa học *AWS Certified Cloud Practitioner (Tiếng Việt)*

Link: <https://www.udemy.com/course/aws-certified-cloud-practitioner-tieng-viet/?couponCode=CF85DD09F54E32035CFB>

Chương 1: Giới thiệu PowerShell

1.1 PowerShell là gì?

PowerShell là một môi trường dòng lệnh và kịch bản (scripting environment) được phát triển bởi Microsoft, được thiết kế đặc biệt để quản lý và tự động hóa các tác vụ trong hệ điều hành Windows.

1.2 Lợi ích của việc sử dụng PowerShell

- **Tự động hóa:** PowerShell cho phép tự động hóa các tác vụ quản trị hệ thống, giúp tiết kiệm thời gian và giảm thiểu lỗi do con người.
- **Quản lý hiệu suất:** Với PowerShell, bạn có thể quản lý và giám sát hiệu suất hệ thống một cách hiệu quả.
- **Linh hoạt:** PowerShell hỗ trợ nhiều kiểu dữ liệu và cung cấp một loạt các cmdlets (lệnh) cho phép bạn thực hiện nhiều tác vụ khác nhau.
- **Mạnh mẽ và Bảo mật:** PowerShell có thể thực hiện các tác vụ phức tạp và hỗ trợ các cơ chế bảo mật để đảm bảo an toàn cho hệ thống.

1.3 Cài đặt PowerShell và cấu hình môi trường

- **Cài đặt:** PowerShell được tích hợp sẵn trong hệ điều hành Windows. Bạn có thể cài đặt các phiên bản cập nhật mới nhất từ Microsoft.
- **Cấu hình Môi trường:** Sau khi cài đặt, bạn có thể cấu hình các biến môi trường và các tùy chọn khác cho PowerShell để phù hợp với nhu cầu của mình.

Chương 2: Cú pháp cơ bản của PowerShell và Cmdlets

2.1 Cú pháp cơ bản của PowerShell

- **Cmdlets:** Cmdlets là các lệnh cơ bản trong PowerShell, mỗi cmdlet có cấu trúc **Verb-Noun (Động từ-Danh từ)**, **ví dụ:** `Get-Process`, `New-Item`.
- **Tham số:** Cmdlets thường được sử dụng với các tham số để chỉ định hành vi cụ thể. Tham số được đặt sau tên cmdlet và có thể có giá trị cụ thể.
- **Biến:** PowerShell sử dụng biến để lưu trữ và truy cập các giá trị. Biến được bắt đầu bằng dấu `$`, ví dụ: `$variable`.
- **Pipe (|):** Pipe được sử dụng để chuyển kết quả của một cmdlet (lệnh) cho đầu vào của một cmdlet (lệnh) khác. Ví dụ: `Get-Process | Stop-Process`.
- **Comment:** Bạn có thể thêm comment trong mã PowerShell bằng cách sử dụng ký hiệu `#`.

2.2 Sử dụng Cmdlets cơ bản

- **Get-Command:** Lệnh này giúp bạn liệt kê tất cả các cmdlet (Lệnh) có sẵn trong PowerShell.
- **Get-Help:** `Get-Help` là một cmdlet để xem hướng dẫn sử dụng của một cmdlet cụ thể.
Ví dụ: `Get-Help Get-Process`.
- **Get-Process:** Cmdlet này được sử dụng để liệt kê tất cả các tiến trình đang chạy trên hệ thống.

2.3 Ví dụ cụ thể

- **Tạo một thư mục mới:** `New-Item -ItemType Directory -Path C:\NewFolder`
- **Thay đổi tên tập tin:** `Rename-Item -Path C:\OldFile.txt -NewName C:\NewFile.txt`
- **Xóa một thư mục:** `Remove-Item -Path C:\OldFolder -Recurse`
- **Xóa một file:** `Remove-Item -Path C:\test.txt`

Trong chương tiếp theo, chúng ta sẽ tìm hiểu về các chủ đề nâng cao hơn trong PowerShell, bao gồm biến, điều kiện và vòng lặp.

Chương 3: Biến, Điều kiện và Vòng lặp trong PowerShell

3.1 Biến trong PowerShell

- **Khai báo biến:** Bạn có thể khai báo một biến bằng cách gán giá trị cho nó.
Ví dụ: `$name = "John"`.
- **Truy cập biến:** Để truy cập giá trị của biến, bạn sử dụng dấu `$` trước tên biến.
Ví dụ: `Write-Host $name`.
- **Kiểu dữ liệu:** PowerShell tự động xác định kiểu dữ liệu của biến dựa trên giá trị được gán.
 - **Trong PowerShell, có nhiều loại biến khác nhau được sử dụng để lưu trữ và thao tác dữ liệu. Dưới đây là một số biến cơ bản trong PowerShell:**
 1. **Biến môi trường (Environment Variables):** Là các biến được đặt ra bên ngoài của PowerShell và lưu trữ thông tin môi trường hệ thống như đường dẫn file, tên máy tính, ngôn ngữ mặc định, v.v.
Ví dụ: `$env:USERNAME` lưu trữ tên người dùng hiện tại.
 2. **Biến mảng (Array Variables):** Là biến có thể chứa nhiều giá trị, mỗi giá trị được định danh bằng một chỉ số. Ví dụ: `$myArray = @(1, 2, 3, 4, 5)`.
 3. **Biến chuỗi (String Variables):** Là biến lưu trữ chuỗi ký tự. Ví dụ: `$myString = "Hello, World!"`.
 4. **Biến số (Numeric Variables):** Là biến lưu trữ các giá trị số. Có thể là số nguyên (integer) hoặc số thực (floating-point). Ví dụ: `$myNumber = 10`.
 5. **Biến đối tượng (Object Variables):** Là biến lưu trữ một đối tượng, có thể là kết quả trả về từ các lệnh hoặc tác vụ khác. Ví dụ: `$myObject = Get-Process`.
 6. **Biến logic (Boolean Variables):** Là biến chỉ có hai giá trị là `True` hoặc `False`, thường được sử dụng trong các điều kiện. Ví dụ: `$isSuccess = $true`.
 7. **Biến hàm (Function Variables):** Là biến lưu trữ một hàm hoặc phương thức, có thể được gọi để thực hiện một loạt các tác vụ. Ví dụ: `$myFunction = { Write-Host "Hello, World!" }`.
- *Các biến này giúp PowerShell linh hoạt trong việc lưu trữ và thao tác dữ liệu trong quá trình thực thi các tác vụ và lệnh.*
- ❖ **Lập trình hướng đối tượng trong PowerShell** cho phép bạn tạo ra các đối tượng và thực hiện các hoạt động trên chúng theo cách mà các đối tượng tương tác với nhau trong thế giới thực. Sau đây là một số khái niệm cơ bản về lập trình hướng đối tượng trong PowerShell:
 1. **Lớp (Class):** Lớp là một mô hình hoặc khuôn mẫu cho các đối tượng. Nó định nghĩa các thuộc tính và phương thức mà một đối tượng cụ thể sẽ có.
 2. **Đối tượng (Object):** Đối tượng là một thực thể cụ thể được tạo ra từ một lớp. Nó có các thuộc tính và phương thức được định nghĩa trong lớp.
 3. **Thuộc tính (Property):** Thuộc tính là các đặc điểm hoặc thông tin của một đối tượng. Ví dụ, một đối tượng "Car" có thể có thuộc tính "Color" để chỉ màu sắc của xe.
 4. **Phương thức (Method):** Phương thức là các hành động hoặc chức năng mà một đối tượng có thể thực hiện. Ví dụ, một đối tượng "Car" có thể có phương thức "Start" để khởi động động cơ.
 5. **Kế thừa (Inheritance):** Kế thừa là khả năng một lớp con kế thừa các thuộc tính và phương thức từ một lớp cha. Điều này giúp tái sử dụng mã và tạo ra mối quan hệ chặt chẽ giữa các lớp.
 6. **Namespace:** Namespace là một cách để tổ chức và quản lý các lớp và đối tượng. Nó giúp tránh xung đột tên và làm cho mã trở nên dễ đọc và dễ hiểu hơn.

Sau đây là một số ví dụ cụ thể về lập trình hướng đối tượng trong PowerShell:

Ví dụ 1: Tạo lớp và đối tượng (Hãy copy code dưới và save thành file có đuôi .ps1 để test thử)

```
# Định nghĩa lớp "Car"

class Car {

    [string]$Color

    [int]$Speed

    Car([string]$color, [int]$speed) {

        $this.Color = $color

        $this.Speed = $speed

    }

    [void] Accelerate([int]$increment) {

        $this.Speed += $increment

    }

    [void] Brake() {

        $this.Speed = 0

    }

}

# Tạo đối tượng từ lớp "Car"

$myCar = [Car]::new("Red", 0)

# Thực hiện các hành động trên đối tượng

$myCar.Accelerate(20)

Write-Host "Current speed: $($myCar.Speed)"

$myCar.Brake()

Write-Host "Current speed after braking: $($myCar.Speed)"
```

```
PS E:\> .\test.ps1
Current speed: 20
Current speed after braking: 0
PS E:\> █
```

Ví dụ 2: Kế thừa

```
# Lớp cha "Animal"

class Animal {

    [string]$Species

    Animal([string]$species) {

        $this.Species = $species

    }

    [void] MakeSound() {

        Write-Host "$($this.Species) makes a sound"

    }

}

# Lớp con "Dog" kế thừa từ lớp "Animal"

class Dog : Animal {

    Dog() : base("Dog") {}

}

# Tạo đối tượng từ lớp "Dog" và gọi phương thức

$myDog = [Dog]::new()

$myDog.MakeSound()
```

- *Các ví dụ trên giúp mình họa cách sử dụng lập trình hướng đối tượng trong PowerShell và cách thực hiện các khái niệm như lớp, đối tượng, kế thừa.*

3.2 Quản lý tiến trình process

Quản lý tiến trình là một phần quan trọng của việc quản trị hệ thống máy chủ. Trong PowerShell, chúng ta có thể thực hiện các thao tác như khởi chạy và dừng tiến trình một cách dễ dàng.

3.2.1 Khởi chạy tiến trình: Để khởi chạy một tiến trình, chúng ta sử dụng lệnh **Start-Process**.

Ví dụ, nếu bạn muốn khởi chạy một ứng dụng trên máy tính của bạn, bạn có thể sử dụng lệnh sau:

Start-Process -FilePath "C:\Path\To\YourProgram.exe"

Trong đó **"C:\Path\To\YourProgram.exe"** là đường dẫn tới tập tin thực thi của chương trình bạn muốn chạy.

3.2.2 Dừng Tiến trình: Để dừng một tiến trình đang chạy, chúng ta sử dụng lệnh

Stop-Process. Ví dụ, nếu bạn muốn dừng một tiến trình theo tên của nó, bạn có thể sử dụng lệnh sau:

```
Stop-Process -Name "YourProcessName"
```

Trong đó "**YourProcessName**" là tên của tiến trình bạn muốn dừng.

3.2.3 Để hiển thị các tiến trình đang chạy trên hệ thốngTrong PowerShell, chúng ta sử dụng lệnh **Get-Process** để làm điều này. Dưới đây là cách thực hiện:

```
Get-Process
```

Lệnh này sẽ trả về danh sách các tiến trình đang chạy trên máy tính của bạn, bao gồm tên tiến trình, ID của tiến trình, CPU sử dụng, và nhiều thông tin khác.

Bạn cũng có thể chạy lệnh **Get-Process** với các tham số tùy chọn để lọc hoặc sắp xếp kết quả hiển thị theo ý muốn.

- Lọc theo tên tiến trình: Để chỉ hiển thị các tiến trình có tên nhất định, bạn có thể sử dụng tham số **-Name** với tên tiến trình mong muốn. Ví dụ, để xem các tiến trình PowerShell đang chạy, bạn có thể sử dụng:

```
Get-Process -Name powershell
```

- Lọc theo ID của tiến trình:

Để chỉ hiển thị một tiến trình cụ thể dựa trên ID của nó, bạn có thể sử dụng tham số **-Id** với ID tiến trình mong muốn.

Ví dụ, để xem thông tin chi tiết của một tiến trình với ID là 1234:

```
Get-Process -Id 1234
```

- Sắp xếp danh sách tiến trình theo CPU sử dụng: Để hiển thị danh sách các tiến trình được sắp xếp theo CPU sử dụng (từ cao đến thấp), bạn có thể sử dụng tham số **-Sort** với giá trị **CPU**. Ví dụ:

```
Get-Process | Sort-Object CPU -Descending
```

Lệnh này sẽ hiển thị danh sách các tiến trình được sắp xếp theo CPU sử dụng, với tiến trình sử dụng nhiều CPU nhất xuất hiện đầu tiên.

Quản lý tiến trình bằng PowerShell giúp bạn có thể kiểm soát và điều chỉnh các tiến trình hoạt động trên hệ thống của mình một cách dễ dàng và hiệu quả.

3.2.4 Ví dụ cụ thể

- Khởi chạy tiến trình soạn thảo notepad:

```
Start-Process -FilePath "C:\Windows\System32\notepad.exe"
```

- Dừng tiến trình soạn thảo notepad:

```
Stop-Process -Name "notepad"
```

3.3 Lệnh điều kiện trong PowerShell

- **If statement:** Câu lệnh **if** được sử dụng để thực hiện một hành động nếu điều kiện được chỉ định là đúng. Ví dụ:

```
$name = "John"
# eq là =
if ($name -eq "John") {
    Write-Host "Hello John!"
}
```

- **Else statement:** Bạn có thể sử dụng câu lệnh **else** để thực hiện một hành động khi điều kiện là sai. Ví dụ:

```
if ($name -eq "John") {
    Write-Host "Hello John!"
} else {
    Write-Host "Hello Stranger!"
}
```

3.4 Vòng lặp trong PowerShell

- **For loop:** Vòng lặp **for** được sử dụng để lặp qua một tập hợp các giá trị. Ví dụ:
for (\$i = 1; \$i -le 5; \$i++) {
 Write-Host "Number: \$i"
}
- **Foreach loop:** Vòng lặp **foreach** được sử dụng để lặp qua các phần tử trong một mảng hoặc một tập hợp. Ví dụ :
\$colors = "Red", "Green", "Blue"
foreach (\$color in \$colors) {
 Write-Host "Color: \$color"
}

3.5 Ví dụ cụ thể

- **Tìm kiếm tập tin:** Sử dụng vòng lặp **foreach** để duyệt qua tất cả các tập tin trong một thư mục và hiển thị tên của chúng.
Đường dẫn thư mục cần tìm kiếm
\$path = "C:\Path\To\Directory"

Sử dụng cmdlet Get-ChildItem để lấy danh sách tất cả các tập tin trong thư mục
\$files = Get-ChildItem -Path \$path

Duyệt qua từng tập tin và hiển thị tên của chúng

```
foreach ($file in $files) {
    Write-Host $file.Name
}
```

```
Administrator: Windows PowerShell
PS C:\Users\Administrator> # Đưa ra tên thư mục cần tìm kiếm
PS C:\Users\Administrator> $path = "C:\thumuc1\thumuc2"
PS C:\Users\Administrator> # Sử dụng cmdlet Get-Childitem để lấy danh sách tất cả các tập tin trong thư mục
PS C:\Users\Administrator> $files = Get-Childitem -Path $path
PS C:\Users\Administrator> # Duyệt qua từng tập tin và hiển thị tên của chúng
PS C:\Users\Administrator> foreach ($file in $files) {
>>     Write-Host $file.Name
>> }
MicrosoftEdgeSetup.exe
StorageSyncAgent_ws2016.msi
PS C:\Users\Administrator> dir C:\thumuc1\thumuc2

Directory: C:\thumuc1\thumuc2

Mode                LastWriteTime         Length Name
----                -
-a----           9/24/2023   9:10 PM         1600880 MicrosoftEdgeSetup.exe
-a----           9/24/2023   9:04 PM         64864256 StorageSyncAgent_ws2016.msi

PS C:\Users\Administrator>
```

- **Xử lý batch:** Sử dụng vòng lặp **for** để thực hiện một loạt các lệnh trên một tập hợp các đối tượng.
- Trong ví dụ này, **\$userName -eq "User1"** kiểm tra xem người dùng hiện tại có phải là "User1" không. Nếu là "User1", nó sẽ cấp quyền đọc cho thư mục \$path. Bạn cần thay đổi đường dẫn của thư mục với biến \$path sao cho đúng với đường dẫn của thư mục trên hệ thống của bạn.

```
$userNames = @("User1", "User2", "User3")
```

```
$path = "E:\Data" # Thay đổi đường dẫn thư mục tùy vào vị trí thực tế của thư mục trong PC của bạn
```

```
# Sử dụng vòng lặp for để thực hiện một loạt các lệnh trên mỗi người dùng
```

```
for ($i = 0; $i -lt $userNames.Length; $i++) {
```

```
    $userName = $userNames[$i]
```

```
    # Thực hiện các lệnh tương ứng với mỗi người dùng, ví dụ: đặt quyền truy cập, tạo thư mục, v.v.
```

```
    Write-Host "Processing user: $userName"
```

```
    # Đoạn mã trong này là nơi bạn thực hiện các lệnh tương ứng với mỗi người dùng
```

Ví dụ: Set permissions, create directories, etc.

Cấp quyền đọc cho thư mục "test" cho người dùng "User1"

```
if ($UserName -eq "User1") {  
  
    $acl = Get-Acl -Path $path  
  
    $permission = "BUILTIN\Users","ReadAndExecute","Allow"  
  
    $rule = New-Object System.Security.AccessControl.FileSystemAccessRule $permission  
  
    $acl.SetAccessRule($rule)  
  
    Set-Acl -Path $path -AclObject $acl  
  
    Write-Host "Permissions set for User1 to read the test folder."  
  
}  
  
}
```

```
PS C:\Users\Administrator> $userNames = @("User1", "User2", "User3")  
PS C:\Users\Administrator> $path = "E:\Data"  
PS C:\Users\Administrator> # Sử dụng vòng lặp for để thực hiện một loạt các lệnh trên mỗi người dùng  
PS C:\Users\Administrator> for ($i = 0; $i -lt $userNames.Length; $i++) {  
>>     $userName = $userNames[$i]  
>>     # Thực hiện các lệnh tương ứng với mỗi người dùng, ví dụ: đặt quyền truy cập, tạo thư mục, v.v.  
>>     Write-Host "Processing user: $userName"  
>>  
>>     # Đoạn mã trong này là nơi bạn thực hiện các lệnh tương ứng với mỗi người dùng  
>>     # ví dụ: set permissions, create directories, etc.  
>>  
>>     # cấp quyền đọc cho thư mục "test" cho người dùng "User1"  
>>     if ($UserName -eq "User1") {  
>>         $acl = Get-Acl -Path $path # Thay đổi đường dẫn thư mục tùy vào vị trí thực tế của thư mục "test"  
>>         $permission = "BUILTIN\Users","ReadAndExecute","Allow"  
>>         $rule = New-Object System.Security.AccessControl.FileSystemAccessRule $permission  
>>         $acl.SetAccessRule($rule)  
>>         Set-Acl -Path $path -AclObject $acl # Thay đổi đường dẫn thư mục tùy vào vị trí thực tế của thư mục "test"  
>>         Write-Host "Permissions set for User1 to read the test folder."  
>>     }  
>> }  
Processing user: User1  
Permissions set for User1 to read the test folder.  
Processing user: User2  
Processing user: User3  
PS C:\Users\Administrator> _
```

* Cấp quyền cho đọc cho file:

\$path = "E:\data\test.txt"

\$ACL = Get-ACL -Path \$path

\$AccessRule = New-Object
System.Security.AccessControl.FileSystemAccessRule("User1","Read","Allow")

\$ACL.SetAccessRule(\$AccessRule)

\$ACL | Set-Acl -Path \$path

(Get-ACL -Path \$path).Access | Format-Table

IdentityReference,FileSystemRights,AccessControlType,IsInherited,InheritanceFlags -AutoSize

```
PS E:\Data> dir

Directory: E:\Data

Mode                LastWriteTime         Length Name
----                -
-a----          9/24/2023   8:55 PM        15359 Configuring Azure File Sync.txt
-a----          2/18/2024   8:06 PM           6 test.txt

PS E:\Data> $path = "E:\data\test.txt"
PS E:\Data> $ACL = Get-ACL -Path $path
PS E:\Data> $AccessRule = New-Object System.Security.AccessControl.FileSystemAccessRule("User1","Read","Allow")
PS E:\Data> $ACL.SetAccessRule($AccessRule)
PS E:\Data> $ACL | Set-Acl -Path $path

PS E:\Data> (Get-ACL -Path $path).Access | Format-Table IdentityReference,FileSystemRights,AccessControlType,IsInherited
,InheritanceFlags -AutoSize

IdentityReference      FileSystemRights AccessControlType IsInherited InheritanceFlags
-----
BUILTIN\Administrators FullControl      Allow          False       None
WIN-2QHT1A8LGL\user1  Read, Synchroni Allow          False       None
NT AUTHORITY\SYSTEM   FullControl      Allow          True        None
BUILTIN\Administrators FullControl      Allow          True        None
WIN-2QHT1A8LGL\Administrator FullControl      Allow          True        None

PS E:\Data> _
```

Chương 4: Hàm và Module trong PowerShell

4.1 Định nghĩa hàm

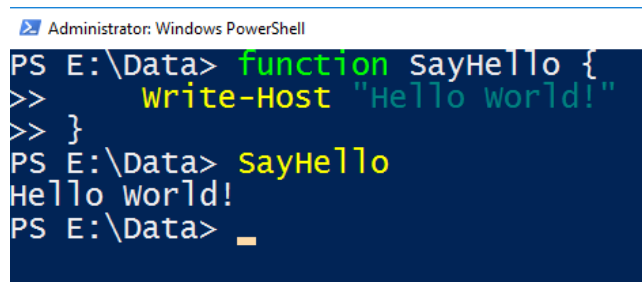
Bạn có thể định nghĩa một hàm bằng cách sử dụng từ khóa **function**. Ví dụ:

```
function SayHello {  
    Write-Host "Hello World!"  
}
```

Gọi hàm: Để gọi một hàm, chỉ cần gọi tên của hàm với các tham số cần thiết (nếu có).

Ví dụ:

SayHello



```
Administrator: Windows PowerShell  
PS E:\Data> function SayHello {  
>>     Write-Host "Hello world!"  
>> }  
PS E:\Data> SayHello  
Hello world!  
PS E:\Data> _
```

4.2 Tham số trong PowerShell

Định nghĩa tham số: Bạn có thể định nghĩa các tham số cho một hàm để nhận đầu vào từ bên ngoài. Ví dụ:

```
function Greet {  
    param (  
        [string] $name  
    )  
    Write-Host "Hello $name!"  
}
```

Gọi hàm với tham số: Khi gọi hàm, bạn cần cung cấp giá trị cho các tham số đã được định nghĩa. Ví dụ:

Greet -name "John"

4.3 Module trong PowerShell

- **Định nghĩa module:** Một module là một bộ sưu tập các hàm, biến và tài nguyên khác được đóng gói lại để tái sử dụng và quản lý dễ dàng. Ví dụ:

```
# Module file: MyModule.psm1  
  
function Get-DateModified {  
  
    param (  

```

```

        [string] $path
    )

    $file = Get-Item $path

    return $file.LastWriteTime
}

```

Nhập module: Bạn có thể nhập module vào phiên làm việc hiện tại bằng cách sử dụng lệnh Import-Module. Ví dụ:

```
Import-Module MyModule.psm1
```

4.4 Ví dụ cụ thể

- **Quản lý tập tin:** Tạo một module chứa các hàm để quản lý tập tin và thư mục, bao gồm việc sao chép, di chuyển, và xóa tập tin.

a. Viết các hàm PowerShell:

Đầu tiên, tạo một file PowerShell mới có tên **FileManagement.ps1**. trong file này, định nghĩa các hàm PowerShell để thực hiện các tác vụ quản lý tập tin và thư mục như sau:

Copy-File: Sao chép một tập tin từ một vị trí nguồn đến một vị trí đích.

Move-File: Di chuyển một tập tin từ một vị trí nguồn đến một vị trí đích.

Delete-File: Xóa một tập tin khỏi hệ thống.

Make-Directory: Tạo một thư mục mới trên hệ thống.

Ví dụ:

```

function Copy-File {
    param(
        [string]$SourcePath,
        [string]$DestinationPath
    )

    Copy-Item -Path $SourcePath -Destination $DestinationPath
}

function Move-File {
    param(
        [string]$SourcePath,
        [string]$DestinationPath
    )

    Move-Item -Path $SourcePath -Destination $DestinationPath
}

```

```

function Delete-File {
    param(
        [string]$FilePath
    )

    Remove-Item -Path $FilePath
}

function Make-Directory {
    param(
        [string]$DirectoryPath
    )

    New-Item -Path $DirectoryPath -ItemType Directory
}

```

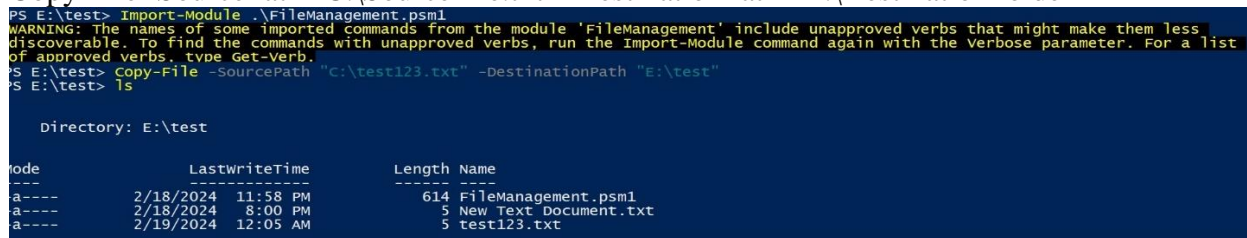
b. Tạo module PowerShell

Sau khi đã định nghĩa các hàm PowerShell, bạn có thể tạo một module PowerShell để tổ chức và quản lý các hàm này. Để làm việc này, chúng ta có các bước sau:

1. Tạo một thư mục mới có tên là **FileManagement**.
2. Di chuyển các file PowerShell (**FileManagement.ps1**) vào thư mục này.
3. Đổi tên file PowerShell thành **FileManagement.psm1**
4. Nhập module vào phiên làm việc hiện tại bằng lệnh **Import-Module**, và Gọi các hàm đã được định nghĩa trong module để thực hiện các tác vụ quản lý tập tin và thư mục. Ví dụ

```
Import-Module FileManagement.psm1
```

```
Copy-File -SourcePath "C:\SourceFile.txt" -DestinationPath "D:\DestinationFolder"
```



```

PS E:\test> Import-Module .\FileManagement.psm1
WARNING: The names of some imported commands from the module 'FileManagement' include unapproved verbs that might make them less discoverable. To find the commands with unapproved verbs, run the Import-Module command again with the verbose parameter. For a list of approved verbs, type Get-Verb.
PS E:\test> Copy-File -SourcePath "C:\test123.txt" -DestinationPath "E:\test"
PS E:\test> ls

Directory: E:\test

Mode                LastWriteTime         Length Name
----                -
a----             2/18/2024   11:58 PM           614 FileManagement.psm1
a----             2/18/2024    8:00 PM             5 New Text Document.txt
a----             2/19/2024   12:05 AM             5 test123.txt

```

❖ Quản lý máy chủ

a) Định nghĩa các hàm PowerShell

Tương tự như phần quản lý tập tin và thư mục, bạn sẽ định nghĩa các hàm PowerShell để quản lý các tác vụ trên máy chủ như Start, Stop, Restart, Update, etc. **Ví dụ:**

```

function Start-Server {

    param(

```

```

        [string]$ServerName
    )

    Start-Service -Name $ServerName
}

function Stop-Server {

    param(

        [string]$ServerName
    )

    Stop-Service -Name $ServerName
}

# Thêm các hàm khác tùy thuộc vào nhu cầu

```

b) Tạo module PowerShell

Làm tương tự như bước tạo module PowerShell như trong quản lý tập tin và thư mục.

3. Tích hợp với các quy trình tự động hóa

- Sử dụng các hàm và module PowerShell đã được tạo để xây dựng các kịch bản tự động hóa. Ví dụ, bạn có thể viết các script PowerShell để tự động sao lưu dữ liệu từ máy chủ, quản lý các tập tin logs, hoặc triển khai các cập nhật các bản vá lỗi, phần mềm.

Ghi chú:

- Đảm bảo rằng bạn đã thử nghiệm các hàm và module PowerShell của bạn trước khi triển khai chúng vào môi trường chạy thật production.
- Luôn tạo bản sao lưu (backup) của các dữ liệu quan trọng trước khi thực hiện bất kỳ thay đổi nào trong hệ thống máy chủ.

Chương 5: Thực Hành và Luyện Tập với PowerShell trong Windows Server

5.1 Thực Hiện Các Tác Vụ Thực Tế

Trong chương này, chúng ta sẽ thực hiện một số tác vụ thực tế bằng PowerShell trên Windows Server để tự động hóa quản lý người dùng và phân quyền.

5.1.1 Tạo Người Dùng mới

Bước 1: Mở PowerShell với quyền quản trị Administrator.

Bước 2: Sử dụng cmdlet **New-LocalUser** để tạo người dùng mới.

```
New-LocalUser -Name "UserName" -Password (ConvertTo-SecureString -AsPlainText "P@ssw0rd" -Force)
```

Bước 3: Xác định các thuộc tính khác cho người dùng mới, nếu cần.

5.1.2 Gán Vai Trò (role) và Quyền Truy Cập

Bước 1: Sử dụng cmdlet **New-LocalGroup** để tạo một nhóm mới.

```
New-LocalGroup -Name "Admin"
```

Bước 2: Sử dụng cmdlet **Add-LocalGroupMember** để thêm người dùng vào nhóm.

```
Add-LocalGroupMember -Group "Admin" -Member "UserName"
```

Bước 3: Sử dụng cmdlet **Get-LocalGroupMember** để kiểm tra thành viên của nhóm.

```
Get-LocalGroupMember -Group "Admin"
```

5.2 Tự Động Hóa

a. Viết các script PowerShell để tự động hóa việc tạo người dùng mới, gán vai trò và quyền truy cập. Sử dụng vòng lặp, các điều kiện và cmdlet PowerShell để tự động hóa process.

```
# Script để tạo người dùng mới và thêm vào nhóm quản trị Admin
$UserName = "NewUser"
$Password = ConvertTo-SecureString "P@ssw0rd" -AsPlainText -Force

New-LocalUser -Name $UserName -Password $Password

Add-LocalGroupMember -Group "Admin" -Member $UserName
```

b. Trong ví dụ này, chúng ta sẽ viết một script PowerShell đơn giản để tạo bản sao lưu của dữ liệu từ một thư mục nguồn đến một thư mục đích, cũng như lập lịch tự động backup trong Task Scheduler. Đây là một phần quan trọng của việc bảo vệ dữ liệu và tự động hóa quản lý thông tin.

- Tạo một file script PowerShell dành riêng cho hàm Backup-Data, ví dụ là

BackupScript.ps1 với nội dung như sau:

```
param (
    [string]$sourceFolder,
    [string]$destinationFolder
)
# Tạo đường dẫn đến thư mục backup
$backupFolder = Join-Path -Path $destinationFolder -ChildPath (Get-Date -Format
"yyyyMMdd_HH:mm:ss")

# Tạo thư mục backup
New-Item -Path $backupFolder -ItemType Directory
# Sao chép dữ liệu từ thư mục nguồn sang thư mục backup
Copy-Item -Path $sourceFolder -Destination $backupFolder -Recurse -Force
```

- Tạo file script PowerShell gốc cho việc tạo bản sao lưu dữ liệu với khả năng cấu hình lịch backup và lựa chọn backup ngay lập tức, VD là backup.ps1:

```
# Hỏi người dùng nhập đường dẫn thư mục nguồn
do {
    $sourceFolder = Read-Host "Enter the source folder path:"
} while (-not $sourceFolder -or -not (Test-Path -Path $sourceFolder -PathType Container))

# Hỏi người dùng nhập đường dẫn thư mục đích
do {
    $destinationFolder = Read-Host "Enter the destination folder path:"
} while (-not $destinationFolder)

# Kiểm tra xem thư mục đích đã tồn tại chưa, nếu chưa thì tạo mới
if (-not (Test-Path -Path $destinationFolder -PathType Container)) {
    New-Item -Path $destinationFolder -ItemType Directory
}

# Hỏi người dùng nhập thông tin thời gian backup
do {
    $backupTime = Read-Host "Enter the backup time (format: HH:MM):"
} while (-not $backupTime)

# Hàm sao lưu dữ liệu
function Backup-Data {
    param (
        [string]$sourceFolder,
        [string]$destinationFolder
    )

    # Tạo đường dẫn đến thư mục backup
```

```

$backupFolder = Join-Path -Path $destinationFolder -ChildPath (Get-Date -Format
"yyyyMMdd_HH:mm:ss")

# Tạo thư mục backup
New-Item -Path $backupFolder -ItemType Directory

# Sao chép dữ liệu từ thư mục nguồn sang thư mục backup
Copy-Item -Path $sourceFolder -Destination $backupFolder -Recurse -Force
}

# Hàm để cấu hình lịch backup
function Configure-BackupSchedule {
    param (
        [string]$backupTime,
        [string]$sourceFolder,
        [string]$destinationFolder
    )

    # Lập lịch backup
    $trigger = New-ScheduledTaskTrigger -Daily -At $backupTime

    # Đường dẫn tuyệt đối của tệp script BackupScript.ps1
    $scriptPath = "C:\path\to\BackupScript.ps1"

    # Tạo công việc backup
    $action = New-ScheduledTaskAction -Execute "PowerShell.exe" -Argument "-File $scriptPath
-sourceFolder '$sourceFolder' -destinationFolder '$destinationFolder'"

    # Tạo công việc backup với trigger và action tương ứng
    Register-ScheduledTask -TaskName "BackupTask" -Trigger $trigger -Action $action -
Description "Backup task" -User "SYSTEM" -RunLevel Highest
}

# Hỏi người dùng liệu có muốn backup ngay lập tức không
$backupNow = Read-Host "Do you want to backup now? (yes/no):"

if ($backupNow -eq "yes") {
    # Gọi hàm Backup-Data để thực hiện sao lưu ngay lập tức
    Backup-Data $sourceFolder $destinationFolder
} else {
    # Cấu hình lịch backup
    Configure-BackupSchedule $backupTime $sourceFolder $destinationFolder
}

# Hiển thị thông điệp xác nhận
Write-Host "Backup configuration completed successfully!"

```

```

Select Administrator: Windows PowerShell
PS E:\> .\backup.ps1
Enter the source folder path:: C:\thumuc1
Enter the destination folder path:: e:\backup
Enter the backup time (format: HH:MM):: 22:00
Do you want to backup now? (yes/no):: yes

Directory: E:\backup

Mode                LastWriteTime         Length Name
----                -
d-----          2/19/2024   6:05 AM             20240219_060510
Backup configuration completed successfully!

PS E:\> .\backup.ps1
Enter the source folder path:: C:\thumuc1
Enter the destination folder path:: e:\backup
Enter the backup time (format: HH:MM):: 22:00
Do you want to backup now? (yes/no):: no

TaskPath                TaskName                State
-----                -
\                        BackupTask              Ready
Backup configuration completed successfully!

```

c. Dưới đây là một đoạn code PowerShell để kiểm tra trạng thái của máy chủ, ghi log và tạo một dịch vụ Windows để tự động kiểm tra trạng thái Server và ghi log:

- Copy đoạn code dưới và tạo file Check-ServerStatusAndLog.ps1 :

```

# Hàm kiểm tra trạng thái máy chủ và ghi vào log
function Check-ServerStatusAndLog {
    $logFile = "C:\Path\To\ServerStatus.log"

    # Thời gian hiện tại
    $timestamp = Get-Date -Format "yyyy-MM-dd HH:mm:ss"

    # Thực hiện các kiểm tra để đảm bảo máy chủ hoạt động bình thường
    # Trong ví dụ này, chúng ta chỉ kiểm tra kết nối mạng và dịch vụ cụ thể
    $networkStatus = Test-NetConnection -ComputerName "localhost" -Port 80

    # Kiểm tra kết nối mạng và ghi vào log
    if ($networkStatus.TcpTestSucceeded) {
        $networkStatusMessage = "Server is reachable."
    } else {
        $networkStatusMessage = "Server is unreachable."
    }
    Add-Content -Path $logFile -Value "$timestamp - $networkStatusMessage"

    # Kiểm tra trạng thái của một dịch vụ cụ thể, ví dụ: dịch vụ IIS
    $serviceStatus = Get-Service -Name "W3SVC"
    if ($serviceStatus.Status -eq "Running") {
        $serviceStatusMessage = "IIS is running."
    } else {
        $serviceStatusMessage = "IIS is not running."
    }
    Add-Content -Path $logFile -Value "$timestamp - $serviceStatusMessage"
}

# Kiểm tra trạng thái máy chủ và ghi log
Check-ServerStatusAndLog

```

- Có thể tạo dịch vụ (service) trong Windows để tự động kiểm tra trạng thái máy chủ và ghi log như sau:

```
# Khai báo biến

$serviceName = "ServerStatusService"

$serviceDisplayName = "Server Status Service"

$serviceDescription = "Automatically checks the status of the server and logs it."

$serviceExecutable = "$($env:SystemRoot)\System32\WindowsPowerShell\v1.0\powershell.exe"

$serviceArguments = "-NoProfile -ExecutionPolicy Bypass -File
C:\Path\To\CheckServerStatus.ps1"

# Tạo dịch vụ Windows

New-Service -Name $serviceName -DisplayName $serviceDisplayName -Description
$serviceDescription -BinaryPathName $serviceExecutable -StartupType Automatic

# Khởi động dịch vụ

Start-Service -Name $serviceName
```

-Trong hai đoạn code trên:

- Hàm **Check-ServerStatusAndLog** được sử dụng để kiểm tra trạng thái của máy chủ và ghi vào log file.
- Log file được ghi vào đường dẫn được chỉ định trong biến **\$LogFile**.
- Sau khi kiểm tra trạng thái máy chủ và ghi log, chúng ta tạo một dịch vụ Windows bằng cách sử dụng cmdlet **New-Service**. Dịch vụ này sẽ chạy script kiểm tra trạng thái máy chủ và ghi log.
- Cuối cùng, dịch vụ được khởi động bằng lệnh **Start-Service**.

Chương 6: Sử dụng PowerShell với Đám Mây Azure

6.1. Giới thiệu

Trong chương này, chúng ta sẽ tìm hiểu cách sử dụng PowerShell để tương tác với Đám Mây Azure, bao gồm việc quản lý tài nguyên, triển khai ứng dụng, và thực hiện các tác vụ quản trị khác trên nền tảng Đám Mây Azure.

6.2. Cài đặt và Cấu hình

Đầu tiên, hãy cài đặt module PowerShell cho Azure trong máy win10 của bạn, và cấu hình nó để kết nối với tài khoản Azure của bạn.

- **Với windows 10 trở lên yêu cầu:**

- Update to Windows PowerShell 5.1
- Cài đặt .Net Framework 4.7.2 trở lên
- Update PowerShell 5.1 với quyền administrator:

`Install-Module -Name PowerShellGet -Force`

- **Thiết đặt PowerShell execution policy để truy cập từ xa hoặc ít hạn chế hơn**

- Kiểm tra PowerShell execution policy:
`Get-ExecutionPolicy -List`
- Thiết đặt PowerShell execution policy để truy cập từ xa
`Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope CurrentUser`

- **Cài đặt Az PowerShell module**

`Install-Module -Name Az -Repository PSGallery -Force`

- **Có thể update phiên bản mới nhất của Az PowerShell module sử dụng lệnh:**

`Update-Module -Name Az -Force`

- Để bắt đầu sử dụng powershell từ máy win10 của bạn để quản lý các tài nguyên (resources), sử dụng lệnh sau trong powershell để đăng nhập vào tài khoản Azure của bạn:

`Connect-AzAccount`

```
Administration: Windows PowerShell
PS C:\Windows\system32> $PSVersionTable.PSVersion

Major Minor Build Revision
-----
5      1      18362  145

PS C:\Windows\system32> Install-Module -Name PowerShellGet -Force

NuGet provider is required to continue
PowerShellGet requires NuGet provider version '2.8.5.281' or newer to interact with NuGet-based repositories. The NuGet provider must be available in 'C:\Program
Files\PackageManagement\ProviderAssemblies' or 'C:\Users\pc-test01\AppData\Local\PackageManagement\ProviderAssemblies'. You can also install the NuGet provider by running
'Install-PackageProvider -Name NuGet -MinimumVersion 2.8.5.281 -Force'. Do you want PowerShellGet to install and import the NuGet provider now?
[Y] Yes [N] No [S] Suspend [?] Help (default is "Y"): Y
PS C:\Windows\system32> $PSVersionTable.PSVersion

Major Minor Build Revision
-----
5      1      18362  145

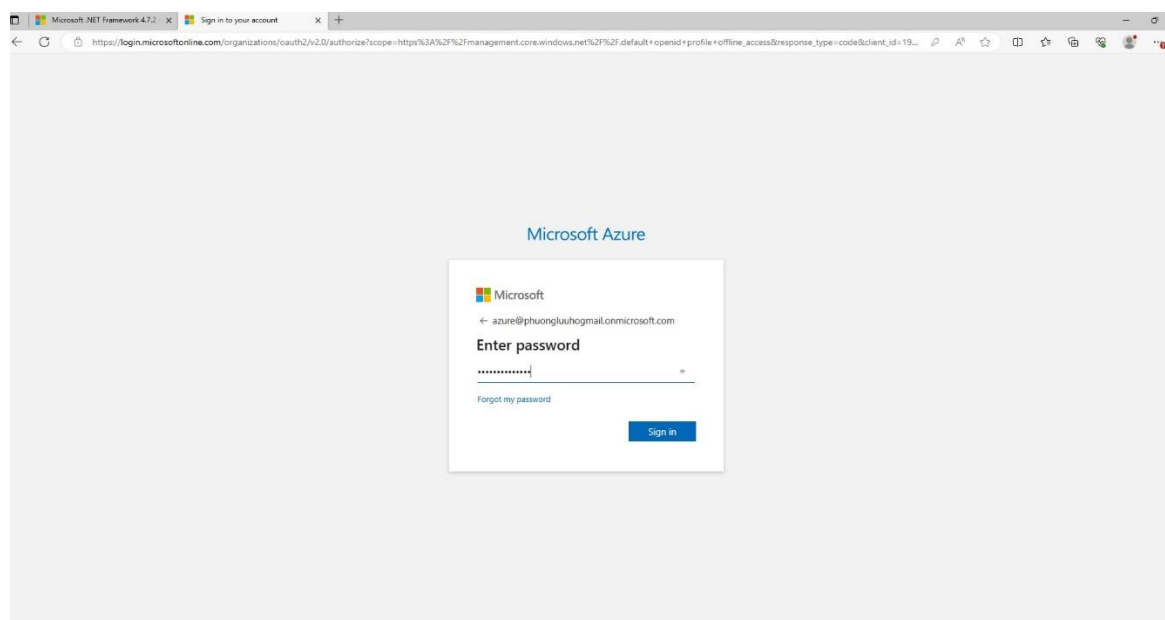
PS C:\Windows\system32> Get-ExecutionPolicy -List

Scope ExecutionPolicy
-----
MachinePolicy Undefined
UserPolicy Undefined
Process Undefined
CurrentUser Undefined
LocalMachine Undefined

PS C:\Windows\system32> Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope CurrentUser

Execution Policy Change
The execution policy helps protect you from scripts that you do not trust. Changing the execution policy might expose you to the security risks described in the
about_Execution_Policies help topic at https://go.microsoft.com/fwlink/?LinkID=135170. Do you want to change the execution policy?
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is "N"): Y
PS C:\Windows\system32> Install-Module -Name Az -Repository PSGallery -Force
PS C:\Windows\system32> Connect-AzAccount
WARNING: Unable to acquire token for tenant 'e54ef5cd-043d-4f06-b96a-82a1dd13e29c' with error 'SharedTokenCacheCredential authentication unavailable. Token acquisition
failed for user azure@phuongluuho@gmail.onmicrosoft.com. Ensure that you have authenticated with a developer tool that supports Azure single sign on.'
WARNING: Unable to acquire token for tenant 'e54ef5cd-043d-4f06-b96a-82a1dd13e29c' with error 'SharedTokenCacheCredential authentication unavailable. Token acquisition
```

- Sau đó nhập tài khoản và mật khẩu của bạn vào Azure Portal



6.3 Quản lý Tài Nguyên Đám Mây Azure

- Liệt kê các subscriptions (thuê bao):
Get-AzSubscription
- Liệt kê tất cả các Resource Group (nhóm tài nguyên) trong tài khoản Azure của bạn:
Get-AzResourceGroup
- Liệt kê tất cả các tài nguyên trong toàn bộ tài khoản Azure
Get-AzResource

- **Liệt kê tất cả các tài nguyên trong một Resource Group**

Get-AzResource -ResourceGroupName Your_ResourceGroup_Name

- **VD:** Get-AzResource -ResourceGroupName Dev-rg

```
PS C:\Users\pc-test01> Get-AzSubscription
WARNING: Unable to acquire token for tenant 'e54ef5cd-043d-4f06-b96a-82aidd13e29c' with error 'SharedTokenCacheCredential authentication unavailable. Token acquisition failed for user 'azure@phuongluuhogmail.onmicrosoft.com'. Ensure that you have authenticated with a developer tool that supports Azure single sign on.'
```

Name	Id	State
Azure subscription 1	c496a4db-c3af-485c-80a3-c0d4961071ff	Enabled

```
PS C:\Users\pc-test01> Get-AzResourceGroup

ResourceGroupName : Dev-rg
Location           : southcentralus
ProvisioningState   : Succeeded
Tags               :
ResourceId          : /subscriptions/c496a4db-c3af-485c-80a3-c0d4961071ff/resourceGroups/Dev-rg

ResourceGroupName : lab-azure
Location           : eastus
ProvisioningState   : Succeeded
Tags               :
ResourceId          : /subscriptions/c496a4db-c3af-485c-80a3-c0d4961071ff/resourceGroups/lab-azure

ResourceGroupName : NetworkWatcherRG
Location           : eastus
ProvisioningState   : Succeeded
Tags               :
ResourceId          : /subscriptions/c496a4db-c3af-485c-80a3-c0d4961071ff/resourceGroups/NetworkWatcherRG

ResourceGroupName : lab-RG
Location           : westus
ProvisioningState   : Succeeded
Tags               :
```

```
PS C:\Users\pc-test01> Get-AzResource -ResourceGroupName Dev-rg

Name                : vmss01autoscale
ResourceGroupName    : Dev-rg
ResourceType         : Microsoft.Insights/autoscalesettings
Location             : southcentralus
ResourceId            : /subscriptions/c496a4db-c3af-485c-80a3-c0d4961071ff/resourceGroups/Dev-rg/providers/Microsoft.Insights/autoscalesettings/vmss01autoscale
Tags                 :

Name                : websvr01autoscale
ResourceGroupName    : Dev-rg
ResourceType         : Microsoft.Insights/autoscalesettings
Location             : southcentralus
ResourceId            : /subscriptions/c496a4db-c3af-485c-80a3-c0d4961071ff/resourceGroups/Dev-rg/providers/Microsoft.Insights/autoscalesettings/websvr01autoscale
Tags                 :

PS C:\Users\pc-test01> Get-AzResource

Name                : vmss01autoscale
ResourceGroupName    : Dev-rg
ResourceType         : Microsoft.Insights/autoscalesettings
Location             : southcentralus
ResourceId            : /subscriptions/c496a4db-c3af-485c-80a3-c0d4961071ff/resourceGroups/Dev-rg/providers/Microsoft.Insights/autoscalesettings/vmss01autoscale
Tags                 :

Name                : websvr01autoscale
ResourceGroupName    : Dev-rg
ResourceType         : Microsoft.Insights/autoscalesettings
Location             : southcentralus
ResourceId            : /subscriptions/c496a4db-c3af-485c-80a3-c0d4961071ff/resourceGroups/Dev-rg/providers/Microsoft.Insights/autoscalesettings/websvr01autoscale
Tags                 :
```

- **Liệt kê tất cả các mạng ảo (Vnet) trong tài khoản Azure:**

Get-AzVirtualNetwork

- **Xem kết quả chi tiết hơn:** Get-AzVirtualNetwork | fl

- **Liệt kê tất cả các cân bằng tải (Load balancer) trong tài khoản Azure:**
Get-AzLoadBalancer
- **Liệt kê tất cả các máy ảo trong tài khoản Azure**

Get-AzVM

- **Liệt kê tất cả các máy ảo (VM) trong một Resource Group**
Get-AzVM -ResourceGroupName Your_ResourceGroup_name
 - **VD:** Get-AzVM -ResourceGroupName lab-azure
 - **Để lấy thông tin chi tiết hơn:** Get-AzVM -ResourceGroupName lab-azure | fl
- **Liệt kê tất cả các Storage Account (tài khoản lưu trữ dữ liệu) trong tài khoản Azure:**
Get-AzStorageAccount
- **Stop (Tắt) máy ảo (VM)**
Stop-AzVM
- **Start (bật) máy ảo (VM)**
Start-AzVM
- **Stop (Tắt) và Start (bật) máy ảo (VM) trong một Resource Group :**
Stop-AzVM -ResourceGroupName Your_ResourceGroup_name -Name Your_VM
 - **VD:** Stop-AzVM -ResourceGroupName lab-azure -Name testVM
 - Start-AzVM -ResourceGroupName Your_ResourceGroup_name -Name Your_VM
 - **VD:** Start-AzVM -ResourceGroupName lab-azure -Name testVM

```
PS C:\Users\pc-test01> Stop-AzVM -ResourceGroupName lab-azure -Name testVM

Virtual machine stopping operation
This cmdlet will stop the specified virtual machine. Do you want to continue?
[Y] Yes [N] No [S] Suspend [?] Help (default is "Y"): y

OperationId : dcda4ee3-d572-46b0-9c9a-18548d9587b7
Status      : Succeeded
StartTime   : 2/21/2024 9:55:36 AM
EndTime     : 2/21/2024 9:56:22 AM
Error       :

PS C:\Users\pc-test01> Stop-AzVM

cmdlet Stop-AzVM at command pipeline position 1
Supply values for the following parameters:
(Type !? for Help.)
ResourceGroupName:
PS C:\Users\pc-test01> Get-AzVM

ResourceGroupName  Name Location      VmSize OsType      NIC ProvisioningState Zone
-----
LAB-AZURE          testVM  eastus  Standard_B1s Linux testvm80_z1 Succeeded 1
```

- **Xóa một máy ảo (VM)**
Remove-AzVM
- **Xóa một máy ảo (VM) trong một Resource Group :**
Remove-AzVM -ResourceGroupName Your_RG_name -Name YourVM_name -Force
 - **VD:** Remove-AzVM -ResourceGroupName lab-azure -Name testVM -Force

```
Windows PowerShell
PS C:\Users\pc-test01> Remove-AzVM

cmdlet Remove-AzVM at command pipeline position 1
Supply values for the following parameters:
(Type !? for Help.)
Name:
PS C:\Users\pc-test01> Remove-AzVM -ResourceGroupName lab-azure -Name testVM -Force

OperationId : 3c8dfbfc-88dc-46f2-92cb-7373236de62f
Status      : Succeeded
StartTime   : 2/21/2024 10:14:06 AM
EndTime     : 2/21/2024 10:14:19 AM
Error       :
```

- **Tạo một Resource Group:**
New-AzResourceGroup
 - **Tạo một Storage Account:**
New-AzStorageAccount
 - **Tạo một mạng ảo Vnet:**
New-AzVirtualNetwork
 - **Tạo một địa chỉ Public IP:**
New-AzPublicIpAddress
 - **Login vào một tài khoản Azure khác:**
Login-AzAccount
 - **Thoát ra khỏi một tài khoản Azure:**
Logout-AzAccount
- **Tạo và quản lý một máy ảo VM**
1. **Tạo một ResourceGroup:**
New-AzResourceGroup -Name myResourceGroup -Location EastUS
 2. **Tạo một máy ảo VM chạy hệ điều hành Windows 2022:**
New-AzVm `
-ResourceGroupName 'myResourceGroup' `
-Name 'myVM' `
-Location 'EastUS' `
-Image 'MicrosoftWindowsServer:WindowsServer:2022-datacenter-azure-
edition:latest' `
-VirtualNetworkName 'myVnet' `
-SubnetName 'mySubnet' `
-SecurityGroupName 'myNetworkSecurityGroup' `
-PublicIpAddressName 'myPublicIpAddress' `
-OpenPorts 80,3389

```

PS C:\Users\pc-test01> New-AzVm
>> -ResourceGroupName 'myResourceGroup'
>> -Name 'myVM'
>> -Location 'EastUS'
>> -Image 'MicrosoftWindowsServer:WindowsServer:2022-datacenter-azure-edition:latest'
>> -VirtualNetworkName 'myVnet'
>> -SubnetName 'mySubnet'
>> -SecurityGroupName 'myNetworkSecurityGroup'
>> -PublicIpAddressName 'myPublicIpAddress'
>> -OpenPorts 80,3389
>> -Size 'Standard_DS1_v2'

cmdlet New-AzVm at command pipeline position 1
Supply values for the following parameters:
Credential

ResourceGroupName      : myResourceGroup
Id                     : /subscriptions/c496a4db-c3af-485c-80a3-c0d4961071ff/resourceGroups/myResourceGroup/providers/Microsoft.Compute/virtualMachines/myVM
VmId                   : b6934e52-cc1c-4338-9af7-6c167b197148
Name                   : myVM
Type                   : Microsoft.Compute/virtualMachines
Location               : eastus
Tags                   : {}
HardwareProfile         : {VmSize}
NetworkProfile          : {NetworkInterfaces}
OSProfile               : {ComputerName, AdminUsername, WindowsConfiguration, Secrets, AllowExtensionOperations, RequireGuestProvisionSignal}
ProvisioningState       : Succeeded
StorageProfile          : {ImageReference, OsDisk, DataDisks, DiskControllerType}
FullyQualifiedDomainName : myvm-a91a52.EastUS.cloudapp.azure.com
TimeCreated             : 2/21/2024 4:16:11 AM

PS C:\Users\pc-test01>

```

3. Lấy địa chỉ IP Public của VM vừa tạo:

Get-AzPublicIpAddress -ResourceGroupName myResourceGroup | select IPAddress

```

PS C:\Users\pc-test01> Get-AzPublicIpAddress -ResourceGroupName myResourceGroup | select IPAddress

IPAddress
-----
13.68.203.16

PS C:\Users\pc-test01>

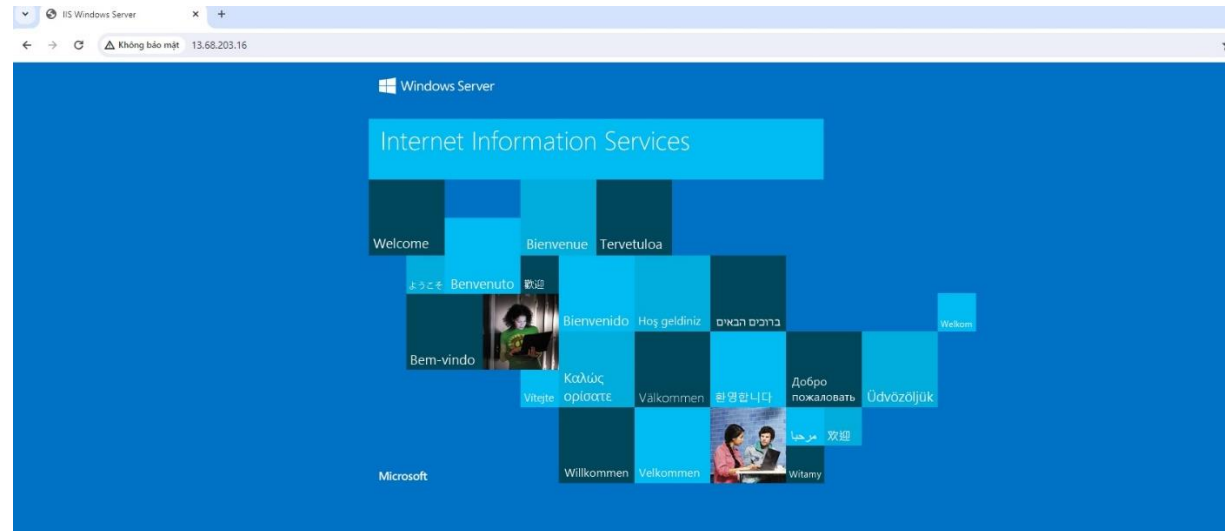
```

4. Bạn có thể sử dụng IP Public của VM để remote desktop vào server

+ Mở PowerShell của windows server 2022 với quyền administrator, sử dụng lệnh sau để setup webserver:

Install-WindowsFeature -name Web-Server -IncludeManagementTools

+ Mở trình duyệt web trên máy tính của bạn, gõ IP Public của VM vừa tạo để truy cập vào website của server.



➤ Tạo một Storage Account

1. Lấy tên của một Resource Group trong tài khoản của bạn:

Get-AzResourceGroup

```

PS C:\Users\pc-test01> Get-AzResourceGroup

ResourceGroupName : Dev-rg
Location           : southcentralus
ProvisioningState  : Succeeded
Tags              :
ResourceId         : /subscriptions/c496a4db-c3af-485c-80a3-c0d4961071ff/resourceGroups/Dev-rg

ResourceGroupName : lab-azure
Location           : eastus
ProvisioningState  : Succeeded
Tags              :
ResourceId         : /subscriptions/c496a4db-c3af-485c-80a3-c0d4961071ff/resourceGroups/lab-azure

ResourceGroupName : NetworkWatcherRG
Location           : eastus
ProvisioningState  : Succeeded
Tags              :
ResourceId         : /subscriptions/c496a4db-c3af-485c-80a3-c0d4961071ff/resourceGroups/NetworkWatcherRG

ResourceGroupName : myResourceGroup
Location           : eastus
ProvisioningState  : Succeeded
Tags              :
ResourceId         : /subscriptions/c496a4db-c3af-485c-80a3-c0d4961071ff/resourceGroups/myResourceGroup

ResourceGroupName : lab-RG
Location           : westus
ProvisioningState  : Succeeded
Tags              :
ResourceId         : /subscriptions/c496a4db-c3af-485c-80a3-c0d4961071ff/resourceGroups/lab-RG

PS C:\Users\pc-test01>

```

2. Chọn một Resource Group và copy tên của nó, sau đó tạo một biến để nhận giá trị của Resource Group của bạn:

`$resourcegroup='RESOURCE_GROUP_NAME'`

3. Set biến location nhận giá trị của location của Resource Group của bạn:

`$location='Location_Name'`

VD: `$location='westus'`

4. Tạo một Storage Account sử dụng lệnh `New-AzStorageAccount` với các tham số sau (lưu ý: tên của Storage Account phải là duy nhất):

```

New-AzStorageAccount -ResourceGroupName $resourcegroup `
-Name 'sto22323324' `
-Location $location `
-SkuName Standard_LRS `
-Kind StorageV2

```

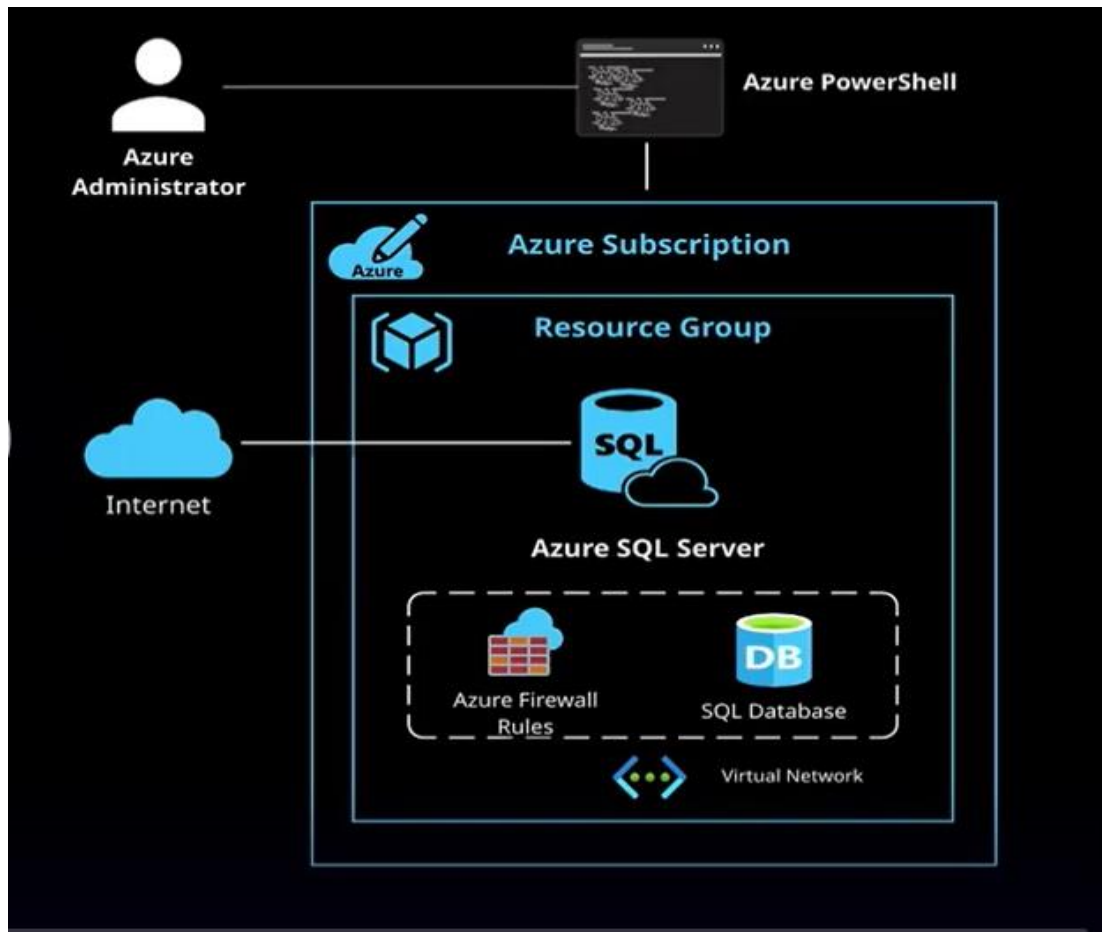
```

Windows PowerShell
PS C:\Users\pc-test01> $resourcegroup='lab-RG'
PS C:\Users\pc-test01> $location='westus'
PS C:\Users\pc-test01> New-AzStorageAccount -ResourceGroupName $resourcegroup `
>> -Name 'sto22323324' `
>> -Location $location `
>> -SkuName Standard_LRS `
>> -Kind StorageV2

StorageAccountName ResourceGroupName PrimaryLocation SkuName Kind AccessTier CreationTime ProvisioningState EnableHttpsTrafficOnly LargeFileShares
-----
sto22323324 lab-RG westus Standard_LRS StorageV2 Hot 2/21/2024 6:03:34 AM Succeeded True
PS C:\Users\pc-test01>

```

➤ **Tạo Azure SQL Databases**
SƠ ĐỒ NGUYÊN LÝ:

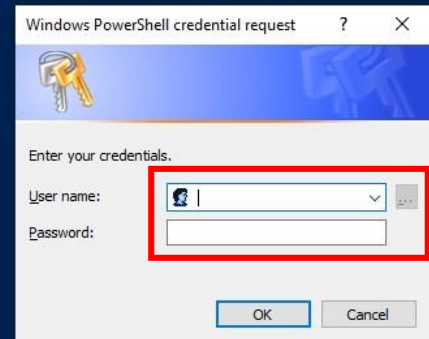


1. Lấy tên của một Resource Group trong tài khoản của bạn:
`Get-AzResourceGroup`
2. Chọn một Resource Group và copy tên của nó, sau đó tạo một biến để nhận giá trị của Resource Group của bạn:
`$resource='RESOURCE_GROUP_NAME'`
3. Set biến location nhận giá trị của location của Resource Group của bạn:
`$location='Location_Name'`
 VD: `$location='westus'`
4. Tạo một biến nhận giá trị của SQL Server:
`$name='sql01tes195433'`
5. Tạo Azure SQL Server:
`New-AzSqlServer`
 -ResourceGroupName $resource`
 -Location $location`
 -ServerName $name`
 -ServerVersion '12.0`
 -SqlAdministratorCredentials (Get-Credential)`

6. Nhập username và password quản trị SQL server:

```
MinimalTlsVersion      : None
PublicNetworkAccess     : Enabled
RestrictOutboundNetworkAccess : Disabled
Administrators           :
PrimaryUserAssignedIdentityId :
KeyId                   :
FederatedClientId       :
```

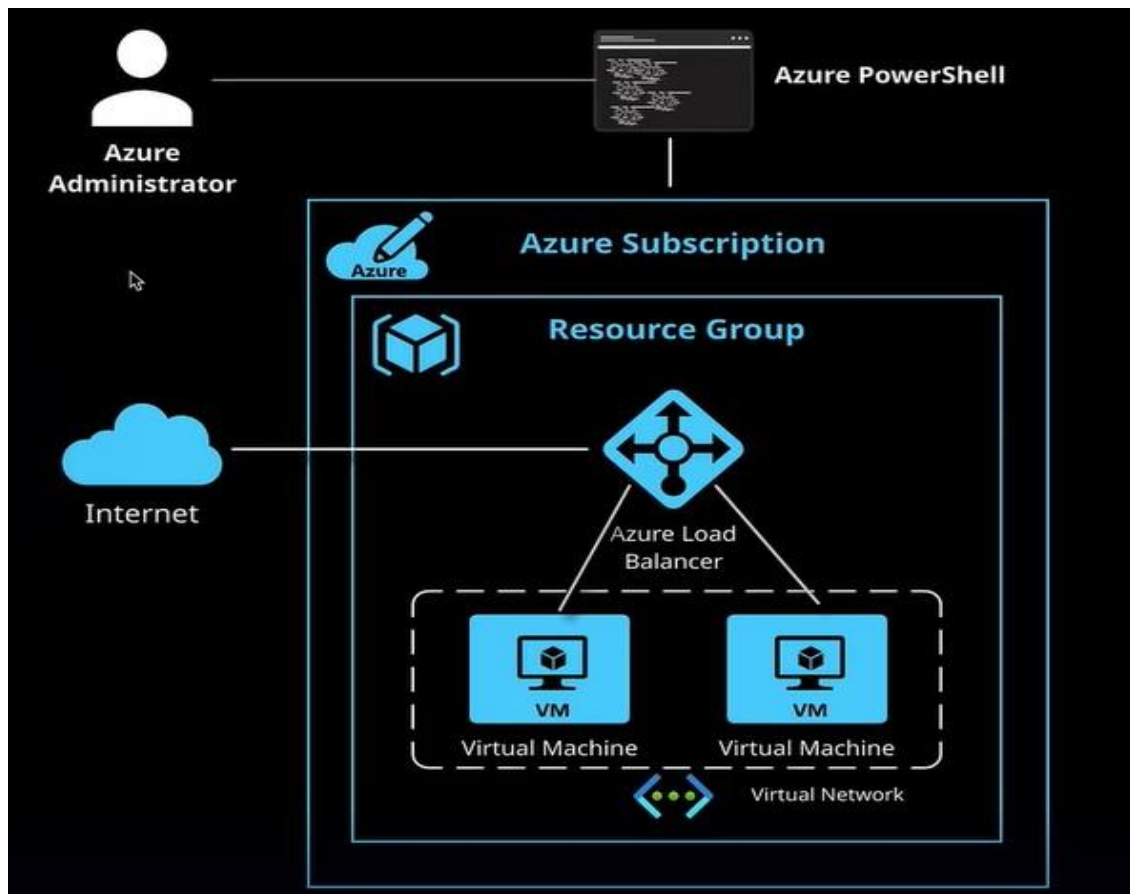
```
PS C:\Users\pc-test01> New-AzSqlServer `
>> -ResourceGroupName $resource `
>> -Location $location`
```



7. Tạo một SQL Database

```
New-AzSqlDatabase -ResourceGroupName $resource -ServerName $name -
DatabaseName 'Database01'
```

- Tạo cân bằng tải Azure Standard Load Balancers
Sơ đồ nguyên lý



1. Lấy tên của một Resource Group trong tài khoản của bạn:
`Get-AzResourceGroup`
2. Chọn một Resource Group và copy tên của nó, sau đó tạo một biến để nhận giá trị của Resource Group của bạn:
`$resource='RESOURCE_GROUP_NAME'`
3. Set biến location nhận giá trị của location của Resource Group của bạn:
`$location='Location_Name'`

VD: `$location='westus'`

4. Tạo địa chỉ public IP:

```
$publicIP = New-AzPublicIpAddress `
-ResourceGroupName $resource `
-Name 'myPublicIp' `
-Location $location `
-AllocationMethod static `
-Sku Standard
```

5. Tạo các biến cho NAT rules:

```
$natrule1 = New-AzLoadBalancerInboundNatRuleConfig `
-Name 'myLoadBalancerRDP1' `
-FrontendIpConfiguration $feip `
-Protocol tcp `
-FrontendPort 4221 `
-BackendPort 3389
```

```
$natrule2 = New-AzLoadBalancerInboundNatRuleConfig `
-Name 'myLoadBalancerRDP2' `
-FrontendIpConfiguration $feip `
-Protocol tcp `
-FrontendPort 4222 `
-BackendPort 3389
```

```
$natrule3 = New-AzLoadBalancerInboundNatRuleConfig `
-Name 'myLoadBalancerRDP3' `
-FrontendIpConfiguration $feip `
-Protocol tcp `
-FrontendPort 4223 `
-BackendPort 3389
```

6. Tạo Azure Standard Load Balancer:

```
New-AzLoadBalancer `
-ResourceGroupName $resource `
-Name 'myLoaBalancer' `
-Sku Standard `
-Location $location `
-FrontendIpConfiguration $feip `
-BackendAddressPool $bepool `
-Probe $probe `
-LoadBalancingRule $rule `
-InboundNatRule $natrule1,$natrule2,$natrule3
```

6.4 Quản lý danh tính Azure (Azure Identities)

6.4.1. Quản lý các subscriptions

- Các Subscriptions (thuê bao) là một hợp đồng hoặc thỏa thuận của bạn với Microsoft để sử dụng các dịch vụ đám mây (cloud services) bao gồm Azure
- Mọi tài nguyên (Resource) trong Azure đều được liên kết với một Subscription, một số tài nguyên bao gồm tenants và users. Tenants là các thực thể của Azure Active Directory bao gồm toàn bộ organization và users là các tài khoản đăng nhập vào Azure để tạo, quản lý và sử dụng các tài nguyên trong Azure.
- Hầu hết các users chỉ có một Subscription. Tuy nhiên đôi khi có trường hợp một user có thể thuộc về nhiều Subscriptions, nếu user này có nhiều hơn một organization, hoặc user này trong một organization đã được chia ra để truy cập vào các tài nguyên cụ thể trong các Resource Groups (nhóm tài nguyên).
- Xem danh sách của các Subscriptions trong tài khoản Azure:

```
Get-AzSubscription
```

- Chuyển tới một active Azure Subscription cụ thể bằng cách sử dụng Subscription ID :

```
Set-AzContext -SubscriptionId <ID Number>
```

6.4.2. Quản lý danh tính (Azure Identities)

6.2.1 Tạo Service Principals

- Service Principal thường được tạo ra để các ứng dụng và dịch vụ được triển khai và hoạt động trong môi trường đám mây Azure có thể truy cập và quản lý các tài nguyên Azure một cách tự động và an toàn.
- Có hai loại xác thực (authentication) cho Service Principals : Password-based (dựa trên mật khẩu) và Certificate-based (Dựa trên chứng chỉ).
- Đầu tiên cần đảm bảo có Az resources module đã được cài đặt, sử dụng lệnh:

```
Import-Module Az.Resources
```

- Tạo một Password-based authentication:

```
$sp = New-AzADServicePrincipal -DisplayName ServicePrincipalName
```


- Export mật khẩu (secret):

```
$sp.PasswordCredentials.SecretText
```

- Để lấy active tenant sau khi Service Principal đã được tạo ra:

```
(Get-AzContext).Tenant.Id
```

- Để lấy DisplayName của Service Principal:

```
$sp.DisplayName
```

- Đối tượng được trả về từ lệnh `New-AzADServicePrincipal` chứa các members là `Id` và `DisplayName`, một trong members này có thể được sử dụng để sign in (đăng nhập) bằng Service Principal.

6.2.2. Tạo Users và Groups trong Azure

- Azure Active Directory là dịch vụ quản lý danh tính (identity) và directory dựa trên đám mây đa người dùng của Microsoft. Azure AD kết hợp các core directory services, quản trị danh tính nâng cao và quản lý truy cập ứng dụng.
- Azure AD cũng cung cấp một nền tảng phong phú dựa trên các tiêu chuẩn cho phép các developers cung cấp quyền kiểm soát truy cập vào ứng dụng của họ dựa trên chính sách (policy) tập trung và quy tắc (rules).
- Azure AD có thể được tích hợp với Windows server active directory
- Vậy chúng ta sẽ làm việc với Azure AD để tạo users và groups.

- Đầu tiên chúng ta tạo một biến, biến này nhận giá trị của lệnh `New-Object`, lệnh này tạo một đối tượng mới trong Azure AD module:

```
$passwordprofile = New-Object -TypeName  
Microsoft.Open.AzureAD.Model.PasswordProfile
```

- Tạo một mật khẩu (password) cho user:

```
$passwordprofile.Password = 'Password123!'
```

- Tạo AzureAD user:

+ Kết nối vào AzureAD:

```
Connect-AzureAD
```

+ Sử dụng lệnh sau để tạo user mới, và đặt tên cho user là NewUser :

```
New-AzureADUser -DisplayName "New User" -PasswordProfile $passwordprofile -
UserPrincipalName "NewUser@Your_AzureAD_Domain" -AccountEnabled $true -
MailNickName "NewUser"
```

VD: New-AzureADUser -DisplayName "New User" -PasswordProfile
\$passwordprofile -UserPrincipalName "phuongluuhogmail.onmicrosoft.com" -
AccountEnabled \$true -MailNickName "newUser"

```
PS /home/study> New-AzureADUser -DisplayName "New User" -PasswordProfile $passwordprofile -UserPrincipalName "NewUser@phuongluuhogmail.onmicro
soft.com" -AccountEnabled $true -MailNickName "NewUser"

ObjectId                               DisplayName UserPrincipalName                UserType
-----                               -
69037e2e-3545-42ae-b696-c026e45f8d2c New User   NewUser@phuongluuhogmail.onmicro Member

PS /home/study> 
```

- Sử dụng lệnh sau để tạo group mới, và đặt tên cho group là NewGroup:

```
New-AzureADGroup -DisplayName "New Group" -MailEnabled $false -
SecurityEnabled $true -MailNickName "NotSet"
```

```
PS /home/study> New-AzureADGroup -DisplayName "New Group" -MailEnabled $false -SecurityEnabled $true -MailNickName "NotSet"

ObjectId                               DisplayName Description
-----                               -
a3200486-739b-49bb-bf6a-9f8a9f9c3e4e New Group
```

- Để add một user vào một group chúng ta sử dụng lệnh sau:

```
Add-AzureADGroupMember -ObjectId "ObjectId-Of-Group" -RefObjectId
"ObjectId-Of-User"
```

VD: Add-AzureADGroupMember -ObjectId "a3200486-739b-49bb-bf6a-
9f8a9f9c3e4e" -RefObjectId "69037e2e-3545-42ae-b696-c026e45f8d2c"

6.5 Scripting Azure PowerShell

- Sử dụng PowerShell ISE để Scripting Azure PowerShell trong môi trường Windows

- PowerShell ISE (Integrated Scripting Environment) là một công cụ tích hợp sẵn trong Windows để phát triển và thử nghiệm các kịch bản PowerShell.
- Azure PowerShell là một bộ công cụ dựa trên PowerShell cho phép tương tác và quản lý các tài nguyên trong môi trường đám mây Azure.

- **Khởi động PowerShell ISE và cài đặt Azure PowerShell:**

1. Mở PowerShell ISE: Bấm phím Windows, gõ "PowerShell ISE" và mở ứng dụng.
2. Scripting tạo một VM Windows 2022 trên Azure

- **Bước 1:** Kết nối vào tài khoản Azure: Sử dụng `Connect-AzAccount` để đăng nhập vào tài khoản Azure của bạn.
- **Bước 2:** Chọn subscription: Sử dụng `Select-AzSubscription` để chọn subscription mà bạn muốn làm việc.
- **Bước 3:** Tạo VM: Sử dụng `New-AzVM` để tạo một máy ảo trên Azure. Chúng ta một ví dụ sau, bạn hãy copy đoạn code dưới và dán vào trình soạn thảo>click File>Save script.ps1

```
# Đặt các thông số cho VM

$resourceGroup = "myResourceGroup"

$location = "EastUS"

$vmName = "myVM"

$adminUsername = "azureuser"

$adminPassword = "YourPassword123!"

$image = "MicrosoftWindowsServer:WindowsServer:2022-datacenter-azure-edition:latest"

# Tạo VM

New-AzVM `

  -ResourceGroupName $resourceGroup `

  -Name $vmName `

  -Location $location `

  -VirtualNetworkName "myVnet" `

  -SubnetName "mySubnet" `

  -SecurityGroupName "myNetworkSecurityGroup" `

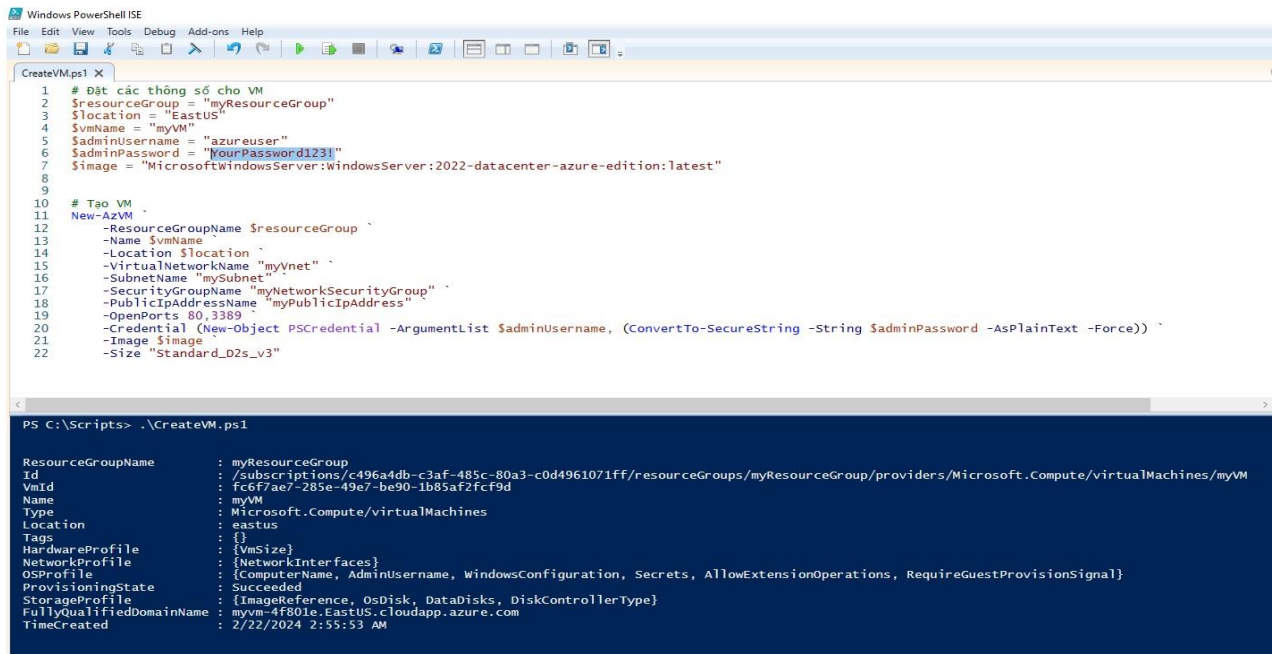
  -PublicIpAddressName "myPublicIpAddress" `

  -OpenPorts 80,3389 `

  -Credential (New-Object PSCredential -ArgumentList $adminUsername, (ConvertTo-
SecureString -String $adminPassword -AsPlainText -Force)) `
```

-Image \$Image`

-Size "Standard_D2s_v3"



```
1 # Đặt các thông số cho VM
2 $resourceGroup = "myResourceGroup"
3 $location = "EastUS"
4 $vmName = "myVM"
5 $adminUsername = "azureuser"
6 $adminPassword = "YourPassword123!"
7 $image = "MicrosoftWindowsServer:WindowsServer:2022-datacenter-azure-edition:latest"
8
9
10 # Tạo VM
11 New-AzVM `
12 -ResourceGroupName $resourceGroup `
13 -Name $vmName `
14 -Location $location `
15 -VirtualNetworkName "myVnet" `
16 -SubnetName "mySubnet" `
17 -SecurityGroupName "myNetworkSecurityGroup" `
18 -PublicIpAddressName "myPublicIpAddress" `
19 -OpenPorts 80,3389 `
20 -Credential (New-Object PSObject -ArgumentList $adminUsername, (ConvertTo-SecureString -String $adminPassword -AsPlainText -Force)) `
21 -Image $image `
22 -Size "Standard_D2s_v3"
```

```
PS C:\Scripts> .\CreateVM.ps1

ResourceGroupName : myResourceGroup
Id                : /subscriptions/c496a4db-c3af-485c-80a3-c0d4961071ff/resourceGroups/myResourceGroup/providers/Microsoft.Compute/virtualMachines/myVM
VmId              : fc6f7ae/-285e-49e/-be90-1b85af2fcf9d
Name              : myVM
Type              : Microsoft.Compute/virtualMachines
Location          : eastus
Tags              : {}
HardwareProfile   : {VmSize}
NetworkProfile    : {NetworkInterfaces}
OSProfile         : {ComputerName, AdminUsername, WindowsConfiguration, Secrets, AllowExtensionOperations, RequireGuestProvisionSignal}
ProvisioningState : Succeeded
StorageProfile    : {ImageReference, OsDisk, DataDisks, DiskControllerType}
FullyQualifiedDomainName : myvm-4f801e.eastus.cloudapp.azure.com
TimeCreated       : 2/22/2024 2:55:53 AM
```

- PowerShell ISE là một công cụ mạnh mẽ để phát triển và chạy các kịch bản PowerShell, kết hợp với Azure PowerShell, chúng ta có thể dễ dàng tạo và quản lý các tài nguyên trên nền tảng đám mây Azure.

- Và bạn cũng có thể download Visual Studio Code và setup vào máy tính của bạn, để scripting PowerShell:

<https://code.visualstudio.com/>

- **Inputs và biến (Variables):**

1. Một biến là một đơn vị bộ nhớ trong đó giá trị được lưu trữ vào biến.
2. Tên của biến trong PowerShell bắt đầu bằng dấu \$ (dollar) và có thể chứa chữ cái, số hoặc gạch dưới nào.
3. Để gán một giá trị cho biến, chúng ta sử dụng toán tử bằng (=) và để hiển thị giá trị của một biến, chúng ta đơn giản chỉ cần nhập tên biến.
4. Bạn có thể lưu trữ tất cả các loại giá trị trong các biến PowerShell. Ví dụ, bạn có thể lưu trữ kết quả của các lệnh.
5. Bạn có thể lưu trữ các phần tử được sử dụng trong các lệnh và biểu thức, như tên (names), đường dẫn (paths), cài đặt (settings) và giá trị (values).
6. Tên biến không phân biệt chữ hoa chữ thường và có thể bao gồm dấu cách và ký tự đặc biệt, nhưng tên biến chứa các ký tự trung gian và dấu cách là khó sử dụng và nên tránh việc đặt tên biến như vậy.

- Chúng ta thực hành vào trình soạn thảo PowerShell ISE, viết một script, tạo một biến để lấy ngày hiện tại bằng cách sử dụng lệnh **Get-Date**. Như sau:

```
# Welcome to my first script
Write-Host "Hello everyone!!!"

#Get the date

$date = Get-Date
Write-Host " Today is " $date

$name = "Robert"
# Get my name
Write-Host "My name is " $name

# Get all of Azure Resources

$Resource = Get-AzResource | Format-Table
$Resource
```

- Lưu nội dung code trên vào một script , VD : đặt tên cho script là: myscript.ps1. Và chạy script, như hình dưới.

The screenshot shows the Windows PowerShell ISE interface. The top pane displays a PowerShell script named `myscript.ps1` with the following content:

```
1 # Welcome to my first script
2 Write-Host "Hello everyone!!!"
3
4 #Get the date
5
6 $date = Get-Date
7 Write-Host " Today is " $date
8
9 $name = "Robert"
10 # Get my name
11 Write-Host "My name is " $name
12
13 # Get all of Azure Resources
14
15 $Resource = Get-AzResource | Format-Table
16 $Resource
17
```

The bottom pane shows the execution of the script in a terminal window. The output is as follows:

```
PS C:\Scripts> .\myscript.ps1
Hello everyone!!!
Today is 2/22/2024 12:01:48 PM
My name is Robert
```

Below the text output, a table of Azure resources is displayed:

Name	ResourceGroupName	ResourceType	Location
vmss01autoscale	Dev-rg	Microsoft.Insights/autoscalesettings	southcentralus
webssvr01autoscale	Dev-rg	Microsoft.Insights/autoscalesettings	southcentralus
autoscale-prod	lab-azure	Microsoft.Insights/autoscalesettings	westus
vmss001-Autoscale-47	lab-azure	Microsoft.Insights/autoscalesettings	westus
vmss01autoscale	lab-azure	Microsoft.Insights/autoscalesettings	eastus
vault1	lab-RG	Microsoft.RecoveryServices/vaults	westus
lifecyclemanagement001	lab-RG	Microsoft.Storage/storageAccounts	eastus
sto2232324	lab-RG	Microsoft.Storage/storageAccounts	westus
myVM_OsDisk_1_d2a5b6476b154b3d8fe3089150719303	MYRESOURCEGROUP	Microsoft.Compute/disks	eastus
myVM	myResourceGroup	Microsoft.Compute/virtualMachines	eastus
myVM	myResourceGroup	Microsoft.Network/networkInterfaces	eastus
myNetworkSecurityGroup	myResourceGroup	Microsoft.Network/networkSecurityGroups	eastus
myPublicIpAddress	myResourceGroup	Microsoft.Network/publicIPAddresses	eastus
myVnet	myResourceGroup	Microsoft.Network/virtualNetworks	eastus
NetworkWatcher_eastus	NetworkWatcherRG	Microsoft.Network/networkWatchers	eastus
NetworkWatcher_westus	NetworkWatcherRG	Microsoft.Network/networkWatchers	westus

The terminal window ends with the prompt `PS C:\Scripts>`.

- **Xuất (Export) các kết quả đầu ra của một lệnh sang các files định dạng có thể sử dụng được:**

- Trong PowerShell, để xuất kết quả của một lệnh ra định dạng của file CSV, sử dụng lệnh **Export-Csv**. Sau đây là cú pháp cơ bản:

<Command> | Export-Csv -Path <đường dẫn tới file CSV>

- **<Command>** : Là lệnh hoặc biểu thức PowerShell mà bạn muốn xuất kết quả của nó ra file CSV.
- **<đường dẫn tới file CSV>**: Là đường dẫn và tên của file CSV mà bạn muốn tạo.

VD: Để xuất kết quả của tất cả các tài nguyên có trong tài khoản Azure của bạn vào một file CSV, chúng ta sử dụng lệnh **Get-AzResource** để lấy thông tin của tất cả các tài nguyên, và tiếp theo là lệnh **Export-Csv**:

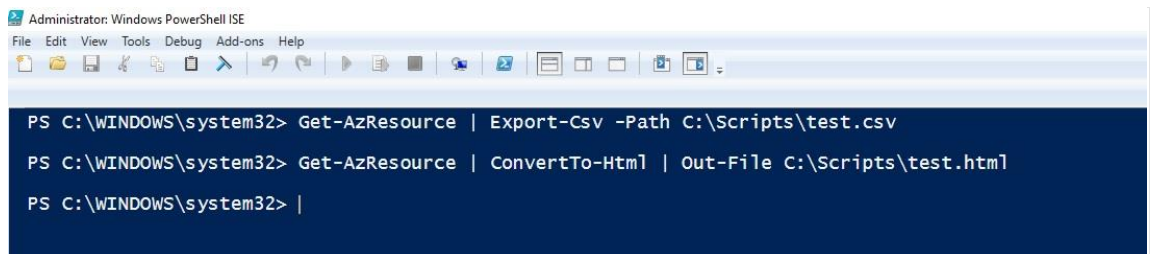
Get-AzResource | Export-Csv -Path C:\Scripts\test.csv

- * Để xuất kết quả của một lệnh ra định dạng của file HTML, sử dụng lệnh **ConvertTo-Html**. Sau đây là cú pháp cơ bản:

<Command> | ConvertTo-Html | Out-File <đường dẫn tới file HTML>

VD: Để xuất kết quả của tất cả các tài nguyên có trong tài khoản Azure của bạn vào một file HTML

Get-AzResource | ConvertTo-Html | Out-File C:\Scripts\test.html



```

Administrator: Windows PowerShell ISE
File Edit View Tools Debug Add-ons Help
PS C:\WINDOWS\system32> Get-AzResource | Export-Csv -Path C:\Scripts\test.csv
PS C:\WINDOWS\system32> Get-AzResource | ConvertTo-Html | Out-File C:\Scripts\test.html
PS C:\WINDOWS\system32> |
  
```

+ **Kết quả CSV file:**

```
test - Notepad
File Edit Format View Help

#TYPE Microsoft.Azure.Commands.ResourceManager.Cmdlets.SdkModels.PSResource
"ResourceId","Id","Identity","Kind","Location","ManagedBy","ResourceName","Name","ExtensionResourceName","ParentResource","Plan","Properties","ResourceGroupName","Type","ResourceType","ExtensionResourceType","Sku","Tags","TagsTable","Sub
"/subscriptions/c496a4db-c3af-485c-80a3-c0d4961071ff/resourceGroups/Dev-rg/providers/Microsoft.Insights/autoscalesettings/vms01autoscale","/subscriptions/c496a4db-c3af-485c-80a3-c0d4961071ff/resourceGroups/Dev-rg/providers/Microsoft.Ins
"/subscriptions/c496a4db-c3af-485c-80a3-c0d4961071ff/resourceGroups/Dev-rg/providers/Microsoft.Insights/autoscalesettings/webv01autoscale","/subscriptions/c496a4db-c3af-485c-80a3-c0d4961071ff/resourceGroups/Dev-rg/providers/Microsoft.I
"/subscriptions/c496a4db-c3af-485c-80a3-c0d4961071ff/resourceGroups/lab-azure/providers/microsoft.insights/autoscalesettings/autoscale-prod","/subscriptions/c496a4db-c3af-485c-80a3-c0d4961071ff/resourceGroups/lab-azure/providers/microsof
"/subscriptions/c496a4db-c3af-485c-80a3-c0d4961071ff/resourceGroups/lab-azure/providers/microsoft.insights/autoscalesettings/vms001-Autoscale-47","/subscriptions/c496a4db-c3af-485c-80a3-c0d4961071ff/resourceGroups/lab-azure/providers/mi
"/subscriptions/c496a4db-c3af-485c-80a3-c0d4961071ff/resourceGroups/lab-azure/providers/Microsoft.Insights/autoscalesettings/vms01autoscale","/subscriptions/c496a4db-c3af-485c-80a3-c0d4961071ff/resourceGroups/lab-azure/providers/Microso
"/subscriptions/c496a4db-c3af-485c-80a3-c0d4961071ff/resourceGroups/lab-RG/providers/Microsoft.RecoveryServices/vaults/vault1","/subscriptions/c496a4db-c3af-485c-80a3-c0d4961071ff/resourceGroups/lab-RG/providers/Microsoft.RecoveryService
"/subscriptions/c496a4db-c3af-485c-80a3-c0d4961071ff/resourceGroups/lab-RG/providers/Microsoft.Storage/storageAccounts/lifecyclemanagement001","/subscriptions/c496a4db-c3af-485c-80a3-c0d4961071ff/resourceGroups/lab-RG/providers/Microsoft
Name
Value
=====
ms-resource-usage azure-cloud-shell
","c496a4db-c3af-485c-80a3-c0d4961071ff",,,"
"/subscriptions/c496a4db-c3af-485c-80a3-c0d4961071ff/resourceGroups/lab-RG/providers/Microsoft.Storage/storageAccounts/sto2232324","/subscriptions/c496a4db-c3af-485c-80a3-c0d4961071ff/resourceGroups/lab-RG/providers/Microsoft.Storage/st
"/subscriptions/c496a4db-c3af-485c-80a3-c0d4961071ff/resourceGroups/NetworkWatcherRG/providers/Microsoft.Network/networkWatchers/NetworkWatcher_eastus","/subscriptions/c496a4db-c3af-485c-80a3-c0d4961071ff/resourceGroups/NetworkWatcherRG/
"/subscriptions/c496a4db-c3af-485c-80a3-c0d4961071ff/resourceGroups/NetworkWatcherRG/providers/Microsoft.Network/networkWatchers/NetworkWatcher_westus","/subscriptions/c496a4db-c3af-485c-80a3-c0d4961071ff/resourceGroups/NetworkWatcherRG/
```

+ Kết quả HTML file:

ResourceId	Id	Identity
/subscriptions/c496a4db-c3af-485c-80a3-c0d4961071ff/resourceGroups/Dev-rg/providers/Microsoft.Insights/autoscalesettings/vms01autoscale	/subscriptions/c496a4db-c3af-485c-80a3-c0d4961071ff/resourceGroups/Dev-rg/providers/Microsoft.Insights/autoscalesettings/vms01autoscale	
/subscriptions/c496a4db-c3af-485c-80a3-c0d4961071ff/resourceGroups/Dev-rg/providers/Microsoft.Insights/autoscalesettings/webv01autoscale	/subscriptions/c496a4db-c3af-485c-80a3-c0d4961071ff/resourceGroups/Dev-rg/providers/Microsoft.Insights/autoscalesettings/webv01autoscale	
/subscriptions/c496a4db-c3af-485c-80a3-c0d4961071ff/resourceGroups/lab-azure/providers/microsoft.insights/autoscalesettings/autoscale-prod	/subscriptions/c496a4db-c3af-485c-80a3-c0d4961071ff/resourceGroups/lab-azure/providers/microsoft.insights/autoscalesettings/autoscale-prod	
/subscriptions/c496a4db-c3af-485c-80a3-c0d4961071ff/resourceGroups/lab-azure/providers/microsoft.insights/autoscalesettings/vms001-Autoscale-47	/subscriptions/c496a4db-c3af-485c-80a3-c0d4961071ff/resourceGroups/lab-azure/providers/microsoft.insights/autoscalesettings/vms001-Autoscale-47	
/subscriptions/c496a4db-c3af-485c-80a3-c0d4961071ff/resourceGroups/lab-azure/providers/Microsoft.Insights/autoscalesettings/vms01autoscale	/subscriptions/c496a4db-c3af-485c-80a3-c0d4961071ff/resourceGroups/lab-azure/providers/Microsoft.Insights/autoscalesettings/vms01autoscale	
/subscriptions/c496a4db-c3af-485c-80a3-c0d4961071ff/resourceGroups/lab-RG/providers/Microsoft.RecoveryServices/vaults/vault1	/subscriptions/c496a4db-c3af-485c-80a3-c0d4961071ff/resourceGroups/lab-RG/providers/Microsoft.RecoveryServices/vaults/vault1	
/subscriptions/c496a4db-c3af-485c-80a3-c0d4961071ff/resourceGroups/lab-RG/providers/Microsoft.Storage/storageAccounts/lifecyclemanagement001	/subscriptions/c496a4db-c3af-485c-80a3-c0d4961071ff/resourceGroups/lab-RG/providers/Microsoft.Storage/storageAccounts/lifecyclemanagement001	Sto
/subscriptions/c496a4db-c3af-485c-80a3-c0d4961071ff/resourceGroups/lab-RG/providers/Microsoft.Storage/storageAccounts/sto2232324	/subscriptions/c496a4db-c3af-485c-80a3-c0d4961071ff/resourceGroups/lab-RG/providers/Microsoft.Storage/storageAccounts/sto2232324	Sto
/subscriptions/c496a4db-c3af-485c-80a3-c0d4961071ff/resourceGroups/NetworkWatcherRG/providers/Microsoft.Network/networkWatchers/NetworkWatcher_eastus	/subscriptions/c496a4db-c3af-485c-80a3-c0d4961071ff/resourceGroups/NetworkWatcherRG/providers/Microsoft.Network/networkWatchers/NetworkWatcher_eastus	
/subscriptions/c496a4db-c3af-485c-80a3-c0d4961071ff/resourceGroups/NetworkWatcherRG/providers/Microsoft.Network/networkWatchers/NetworkWatcher_westus	/subscriptions/c496a4db-c3af-485c-80a3-c0d4961071ff/resourceGroups/NetworkWatcherRG/providers/Microsoft.Network/networkWatchers/NetworkWatcher_westus	

Cảm ơn các bạn đã đọc tài liệu này!!!!

Chúc các bạn thành công trong công việc và tìm hiểu về PowerShell và đám mây Azure.

