* 5 minuten

Met PowerShell kunt u opdrachten schrijven en deze onmiddellijk uitvoeren. Dit staat bekend als **interactieve modus**.

Bedenk dat het algemene doel in het voorbeeld van Customer Relationship Management (CRM) is om drie testomgevingen te maken die virtuele machines bevatten. U gebruikt resource-groepen om ervoor te zorgen dat de VM’s in afzonderlijke omgevingen worden georganiseerd: één voor eenheidstests, één voor integratietests en één voor acceptatietests. U hoeft de resource-groepen slechts één keer te maken, dus het gebruik van de interactieve PowerShell-modus in deze use case is een goede keuze.

Wanneer u een opdracht invoert in PowerShell, koppelt PowerShell de opdracht aan een *cmdlet* en voert PowerShell vervolgens de gevraagde actie uit. We bekijken enkele algemene opdrachten die u kunt gebruiken en vervolgens kijken we naar het installeren van de Azure-ondersteuning voor PowerShell.

## Wat zijn PowerShell-cmdlets?

Een PowerShell-opdracht wordt een **cmdlet** genoemd (uitgesproken als ‘command-let’). Een cmdlet is een opdracht waarmee één functie wordt gemanipuleerd. De term **cmdlet** is bedoeld om “small command” te impliceren. Volgens afspraak worden cmdlet-auteurs aangemoedigd om cmdlets eenvoudig en voor één doel te houden.

Het PowerShell-basisproduct wordt geleverd met cmdlets die werken met functies zoals sessies en achtergrondtaken. U kunt modules toevoegen aan uw PowerShell-installatie om cmdlets op te halen die andere functies manipuleren. Er zijn bijvoorbeeld modules van derden om met ftp te werken, uw besturingssysteem te beheren, toegang te krijgen tot het bestandssysteem, enzovoort.

Cmdlets volgen een naamgevingsconventie voor werkwoord-zelfstandig naamwoorden; bijvoorbeeld , , en . Er is ook een conventie voor werkwoordskeuze: “get” om gegevens op te halen, “set” om gegevens in te voegen of bij te werken, “formatteren” om gegevens op te maken, “out” om uitvoer naar een bestemming te leiden, enzovoort.Get-Process``Format-Table``Start-Service

Auteurs van cmdlets wordt aangeraden voor elke cmdlet een Help-bestand op te nemen. De cmdlet geeft het Help-bestand voor elke cmdlet weer. Als u bijvoorbeeld hulp wilt krijgen voor de cmdlet, voert u de volgende instructie in een Windows PowerShell-sessie in:Get-Help``Get-ChildItem

Get-Help -Name Get-ChildItem -Detailed

## Wat is een PowerShell-module?

Cmdlets worden geleverd in *modules*. Een PowerShell-module is een DLL die de code bevat voor het verwerken van elke beschikbare cmdlet. U laadt cmdlets in PowerShell door de module te laden waarin ze zich bevinden. U kunt een lijst met geladen modules krijgen met behulp van de opdracht:Get-Module

Get-Module

Dit zal iets als:

ModuleType Version Name ExportedCommands  
---------- ------- ---- ----------------  
Manifest 3.1.0.0 Microsoft.PowerShell.Management {Add-Computer, Add-Content, Checkpoint-Computer, Clear-Con...  
Manifest 3.1.0.0 Microsoft.PowerShell.Utility {Add-Member, Add-Type, Clear-Variable, Compare-Object...}  
Binary 1.0.0.1 PackageManagement {Find-Package, Find-PackageProvider, Get-Package, Get-Pack...  
Script 1.0.0.1 PowerShellGet {Find-Command, Find-DscResource, Find-Module, Find-RoleCap...  
Script 2.0.0 PSReadline {Get-PSReadLineKeyHandler, Get-PSReadLineOption, Remove-PS...

## Wat is de Az PowerShell module?

**Az** is de formele naam voor de Azure PowerShell-module, die cmdlets bevat om met Azure-functies te werken. Het bevat honderden cmdlets waarmee u bijna elk aspect van elke Azure-resource kunt beheren. U kunt werken met resourcegroepen, opslag, virtuele machines, Azure Active Directory, containers, machine learning, enzovoort. Deze module is een open-source component [die beschikbaar is op GitHub](https://github.com/Azure/azure-powershell).

Notitie

Mogelijk hebt u Azure PowerShell-opdrachten gezien of gebruikt die een indeling gebruikten. Omdat Az PowerShell-modules nu alle mogelijkheden van AzureRM PowerShell-modules en meer hebben, zullen we AzureRM PowerShell-modules op 29 februari 2024 buiten gebruik stellen. Om serviceonderbrekingen te voorkomen, [werkt u uw scripts](https://aka.ms/azpsmigrate) die Gebruikmaken van AzureRM PowerShell-modules uiterlijk op 29 februari 2024 bij om Az PowerShell-modules te gebruiken. Als u uw scripts automatisch wilt bijwerken, volgt [u de snelstartgids](https://docs.microsoft.com/en-us/powershell/azure/quickstart-migrate-azurerm-to-az-automatically).-AzureRM

### Installeer de Az PowerShell module

De Az PowerShell-module is beschikbaar in een algemene opslagplaats die de PowerShell-galerie wordt genoemd. U kunt de module op uw lokale computer installeren via de cmdlet.Install-Module

Als u de nieuwste Azure Az PowerShell-module wilt installeren, voert u de volgende opdrachten uit:

1. Open het menu **Start** en voer **PowerShell** in.
2. Selecteer het **PowerShell-pictogram**.
3. Voer de volgende opdracht in en druk op .Enter

* Install-Module -Name Az -Scope CurrentUser -Repository PSGallery

Hiermee wordt de module voor uw huidige gebruiker geïnstalleerd (geregeld door de parameter).Scope

De opdracht is afhankelijk van NuGet om onderdelen op te halen, dus afhankelijk van de versie die u hebt geïnstalleerd, wordt u mogelijk gevraagd de nieuwste versie van NuGet te downloaden en te installeren.

NuGet provider is required to continue  
PowerShellGet requires NuGet provider version '2.8.5.201' or newer to interact with NuGet-based repositories. The NuGet  
 provider must be available in 'C:\Program Files\PackageManagement\ProviderAssemblies' or  
'C:\Users\<username>\AppData\Local\PackageManagement\ProviderAssemblies'. You can also install the NuGet provider by running  
'Install-PackageProvider -Name NuGet -MinimumVersion 2.8.5.201 -Force'. Do you want PowerShellGet to install and import  
 the NuGet provider now?  
 [Y] Yes [N] No [S] Suspend [?] Help (default is "Y"):

Enter **Y** and press .Enter

By default, the PowerShell Gallery isn’t configured as a trusted repository for PowerShellGet. Each time you perform an installation from an untrusted repository, you’ll be prompted to confirm you want to install the module with following output:

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"):

Enter **Y** or **A**, then press .Enter

#### Script execution failed

Depending on your security configuration, might fail with something like the following output:Import-Module

import-module : File C:\Program Files\PowerShell\Modules\az\6.3.0\Az.psm1 cannot be loaded  
because running scripts is disabled on this system. For more information, see about\_Execution\_Policies at  
https:/go.microsoft.com/fwlink/?LinkID=135170.  
At line:1 char:1  
+ import-module Az  
+ ~~~~~~~~~~~~~~~~~~~~~  
 + CategoryInfo : SecurityError: (:) [Import-Module], PSSecurityException  
 + FullyQualifiedErrorId : UnauthorizedAccess,Microsoft.PowerShell.Commands.ImportModuleCommand

It might also fail by not responding at all. In this case, press + to stop the program.CtrlC

Both behaviors typically indicate that the execution policy is “Restricted”, meaning you can’t run modules that you download from an external source, including the PowerShell Gallery. You can check this by running the cmdlet . If it returns “Restricted”, then do the following:Get-ExecutionPolicy

1. Use the cmdlet to change the policy to “RemoteSigned”:SetExecutionPolicy

* Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope CurrentUser
* This will prompt you for permission:
* 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

1. Enter **Y** or **A**, then press .Enter
2. At the command prompt, use the up arrow on your keyboard and rerun the command for Azure.Install-Module

You should be able to see the Az module loading. After it completes, you’ll be able to use to load the cmdlets.Import-Module

Als u Azure PowerShell installeert op Linux of macOS, worden dezelfde opdrachten gebruikt.

1. Voer in een terminal de volgende opdracht uit om PowerShell te starten.

* pwsh

1. Run the following command at the PowerShell prompt to install Azure PowerShell.

* Install-Module -Name Az -Scope CurrentUser -Repository PSGallery -Force

1. If you’re asked whether you trust modules from **PSGallery**, answer **Yes** or **Yes to All**.

### Update a PowerShell module

If you get a warning or error message indicating that a version of the Azure PowerShell module is already installed, you can update to the *latest* version by issuing the following command:

Update-Module -Name Az

As with the cmdlet, answer **Yes** or **Yes to All** when prompted to trust the module. You can also use the command to reinstall a module if you’re having trouble with it.Install-Module``Update-Module

## Example: How to create a resource group with Azure PowerShell

Once you’ve installed the Azure module, you can begin working with Azure. Let’s do a common task: creating a Resource Group. As you know, we use resource groups to administer related resources together. Creating a new resource group is one of the first tasks you’ll do when starting a new Azure solution.

There are four steps you need to perform:

1. Import the Azure cmdlets.
2. Connect to your Azure subscription.
3. Create the resource group.
4. Verify that creation was successful.

The following illustration shows an overview of these steps.

An illustration showing the steps to create a resource group.

Each step corresponds to a different cmdlet.

### Import the Azure cmdlets

Beginning with PowerShell 3.0, modules are loaded automatically when you use a cmdlet within the module. It’s no longer necessary to manually import PowerShell modules unless you’ve changed the default module autoloading settings.

### Connect

When you’re working with a local install of Azure PowerShell, you’ll need to authenticate before you can execute Azure commands. The cmdlet prompts for your Azure credentials, then connects to your Azure subscription. It has many optional parameters, but if all you need is an interactive prompt, you don’t need any parameters:Connect-AzAccount

Connect-AzAccount

### Work with subscriptions

If you’re new to Azure, you probably only have a single subscription. But if you’ve been using Azure for a while, you might have created multiple Azure subscriptions. You can configure Azure PowerShell to execute commands against a particular subscription.

You can only be in one subscription at a time. Use the cmdlet to determine which subscription is active. If it’s not the correct one, you can change subscriptions using another cmdlet.Get-AzContext

1. Get a list of all subscription names in your account with the command.Get-AzSubscription
2. Change the subscription by passing the name of the one to select.

Set-AzContext -Subscription '00000000-0000-0000-0000-000000000000'

If you need to look up the **Subscription ID**, open Azure and select **Subscriptions** on the home page.

### Get a list of all resource groups

You can retrieve a list of all Resource Groups in the active subscription.

Get-AzResourceGroup

To get a more concise view, you can send the output from the to the cmdlet using a pipe ‘|’.Get-AzResourceGroup``Format-Table

Get-AzResourceGroup | Format-Table

The output will look something like this:

ResourceGroupName Location ProvisioningState Tags TagsTable ResourceId  
----------------- -------- ----------------- ---- --------- ----------  
cloud-shell-storage-southcentralus southcentralus Succeeded /subscriptions/00000000-0000-0000...  
ExerciseResources eastus Succeeded /subscriptions/00000000-0000-0000...

### Create a resource group

As you know, when you’re creating resources in Azure, you’ll always place them into a resource group for management purposes. A resource group is often one of the first things you’ll create when starting a new application.

You can create resource groups by using the cmdlet. You must specify a name and location. The name must be unique within your subscription. The location determines where the metadata for your resource group will be stored (which may be important to you for compliance reasons). You use strings like “West US”, “North Europe”, or “West India” to specify the location. As with most of the Azure cmdlets, has many optional parameters. However, the core syntax is:New-AzResourceGroup``New-AzResourceGroup

New-AzResourceGroup -Name <name> -Location <location>

Note

Remember, we will be working in an active Azure sandbox, which creates the Resource Group for you. Use the command above if you prefer to work in your own subscription.

### Verify the resources

The lists your Azure resources, which is useful here to verify the resource group creation was successful.Get-AzResource

Get-AzResource

Like the command, you can get a more concise view through the cmdlet:Get-AzResourceGroup``Format-Table

Get-AzResource | Format-Table

You can also filter it to specific resource groups to only list resources associated with that group:

Get-AzResource -ResourceGroupName ExerciseResources

### Create an Azure Virtual Machine

Another common task you can do with PowerShell is to create VMs.

Azure PowerShell provides the cmdlet to create a virtual machine. The cmdlet has many parameters to let it handle the large number of VM configuration settings. Most of the parameters have reasonable default values, so we only need to specify five things:New-AzVm

* **ResourceGroupName**: The resource group into which the new VM will be placed.
* **Name**: The name of the VM in Azure.
* **Location**: Geographic location where the VM will be provisioned.
* **Credential**: An object containing the username and password for the VM admin account. We’ll use the cmdlet. This cmdlet will prompt for a username and password and package it into a credential object.Get-Credential
* **Image**: The operating system image to use for the VM, which is typically a Linux distribution or Windows Server.

New-AzVm  
 -ResourceGroupName <resource group name>  
 -Name <machine name>  
 -Credential <credentials object>  
 -Location <location>  
 -Image <image name>

You can supply these parameters directly to the cmdlet as shown above. Alternatively, you can use other cmdlets to configure the virtual machine, such as , , , and .Set-AzVMOperatingSystem``Set-AzVMSourceImage``Add-AzVMNetworkInterface``Set-AzVMOSDisk

Here’s an example that strings the cmdlet together with the parameter:Get-Credential``-Credential

New-AzVM -Name MyVm -ResourceGroupName ExerciseResources -Credential (Get-Credential) ...

The suffix is specific to VM-based commands in PowerShell. There are several others you can use:AzVM

| Command | Description |
| --- | --- |
| Remove-AzVM | Deletes an Azure VM. |
| Start-AzVM | Start a stopped VM. |
| Stop-AzVM | Stop a running VM. |
| Restart-AzVM | Restart a VM. |
| Update-AzVM | Updates the configuration for a VM. |

#### Example: Getting information for a VM

You can list the VMs in your subscription using the command. This command also supports entering a specific VM by including the property. Here, we’ll assign it to a PowerShell variable:Get-AzVM -Status``-Name

$vm = Get-AzVM -Name MyVM -ResourceGroupName ExerciseResources

The interesting thing is that now your VM is an *object* with which you can interact. For example, you can make changes to that object, then push changes back to Azure by using the command:Update-AzVM

$ResourceGroupName = "ExerciseResources"  
$vm = Get-AzVM -Name MyVM -ResourceGroupName $ResourceGroupName  
$vm.HardwareProfile.vmSize = "Standard\_DS3\_v2"  
  
Update-AzVM -ResourceGroupName $ResourceGroupName -VM $vm

The interactive mode in PowerShell is appropriate for one-off tasks. In our example, we’ll likely use the same resource group for the lifetime of the project, so creating it interactively is reasonable. Interactive mode is often quicker and easier for this task than writing a script and executing that script exactly once.

## Next unit: Exercise - Create an Azure Resource using scripts in Azure PowerShell

[Continue](https://docs.microsoft.com/en-us/learn/modules/automate-azure-tasks-with-powershell/6-exercise-create-resource-interactively/)