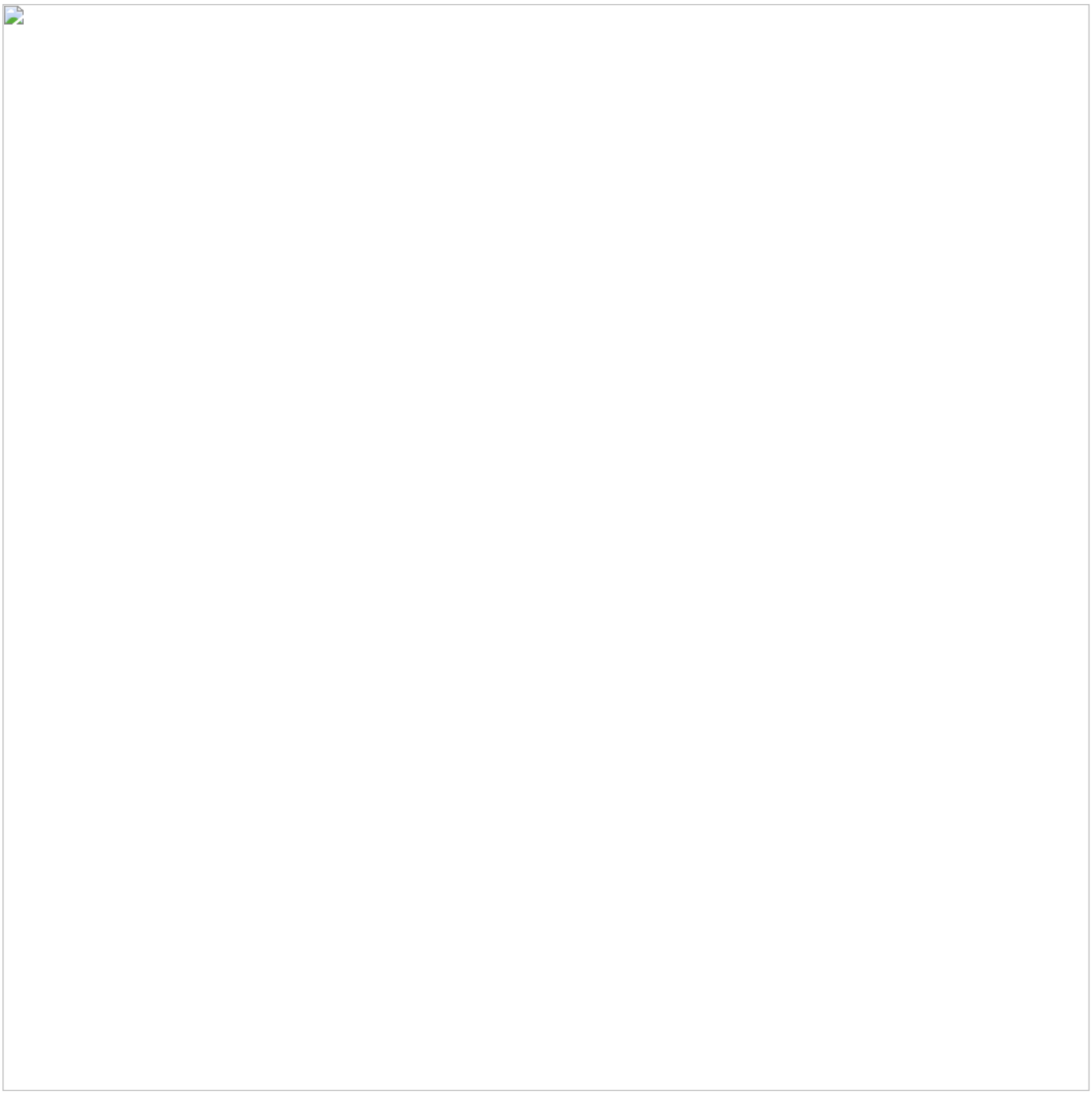
# Powershell: Your first internal PSScript repository


powershellexplained.com/2017-05-30-Powershell-your-first-PSScript-repository



Setting up a basic internal repository for scripts and modules is surprisingly simple. I had no idea how easy this was for the longest time. If you are looking for ways to distribute your modules to others in your team, then you should consider this approach.

## Index

## Pre-requirements

We need the `PowerShellGet` module and a folder on a network share.

–Update 2018-03-03– I have a follow up post where I use a NuGet Docker container instead of a network share: Using a NuGet server for a PSRepository

## PowerShellGet

> “PowerShellGet module contains cmdlets for discovering, installing, updating and publishing the PowerShell artifacts like Modules, DSC Resources, Role Capabilities and Scripts from the https://www.PowerShellGallery.com and other private repositories.” -JuanPablo Jofre: The PowerShell Gallery

If you are running Windows 10 or PowerShell 5.0, then you already have this module. If you need to support PowerShell 3.0, then see the readme for more details.

## Network share

The other thing we should have is an empty folder on a network share. This will be the location of our repository. Your users will need to have access to this location if they are going to be loading content from it.

```
$Path = '\\Server\Share\MyRepository'
```

If you just want to experiment with these commands, you can use a local folder for your repository. PowerShellGet does not care where the folder lives.

## Creating the repository

The first thing we should do is tell PowerShellGet that our `$Path` is a script repository.

```
Import-Module PowerShellGet

$repo = @{
    Name = 'MyRepository'
    SourceLocation = $Path
    PublishLocation = $Path
    InstallationPolicy = 'Trusted'
}
Register-PSRepository @repo
```

And we are done.

```
PS:> Get-PSRepository

Name         InstallationPolicy SourceLocation
----         ------------------ --------------
MyRepository Trusted            \\Server\Share\MyRepository
PSGallery    Untrusted          https://www.powershellgallery.com/api/v2/
```

Other than creating a folder, there is no complicated setup to creating this repository. Just by telling PowerShellGet that the folder is a repository, it will consider it to be one. The one catch is that you need to run this command on each machine to register the repository.

## Modules

I recently covered how to create a module as a way to package your functions. We can use this repository to distribute and share your modules.

## Publishing modules

We need to first make sure the module is installed in your `$env:PSModulePath` and then we can publish it.

```
Publish-Module -Name MyModule -Repository MyRepository -Verbose
```

This will package your module up as a NuGet package and deploy it to the `MyRepository`.

## Listing modules

You can list all the modules in your repository (or search for a specific one) with `Find-Module`.

```
PS:> Find-Module -Repository MyRepository

Version Name      Repository   Description
------- ----      ----------   -----------
0.5.3   MyModule  MyRepository Sample module for blog demos
```

## Installing modules

The last thing we need to know is how to install a module from our repository.

```
Install-Module -Name MyModule -Repository MyRepository
```

## Publishing updated modules

If you want to update a module that you have published, make sure you update the version number in your module manifest. I personally try to use semantic versioning.

You also need to make sure you place this in the `$env:PSModulePath` and don't have any other conflicting installs of your module.

## Installing updated modules

You can run `Update-Module` to update your module. But I find that it leaves the old version on the system. I tend to remove and then install fresh when I need to update.

```
Uninstall-Module -Name MyModule
Install-Module -Name MyModule -Repository MyRepository -Scope CurrentUser
```

I also use the `CurrentUser` scope so that I don't have to be at an administrator command prompt.

## Scripts

Personally, I have never published a script to any repository before now. With that said, it certainly is a valid option. There are separate Cmdlets for publishing scripts.

```
Find-Script
Install-Script
Publish-Script
Save-Script
Update-Script
```

## In closing

I mostly wanted to bring more visibility to using `Register-PSRepository` to create this basic repository. This could be a good stepping stone on your way to setting up a proper NuGet Server.

–Update 2018-03-03– I have a follow up post where I use a NuGet Docker container instead of a network share: Using a NuGet server for a PSRepository

Tags: Powershell Modules PowerShellGet

- ← Previous Post
- Next Post →

Kevin Marquette