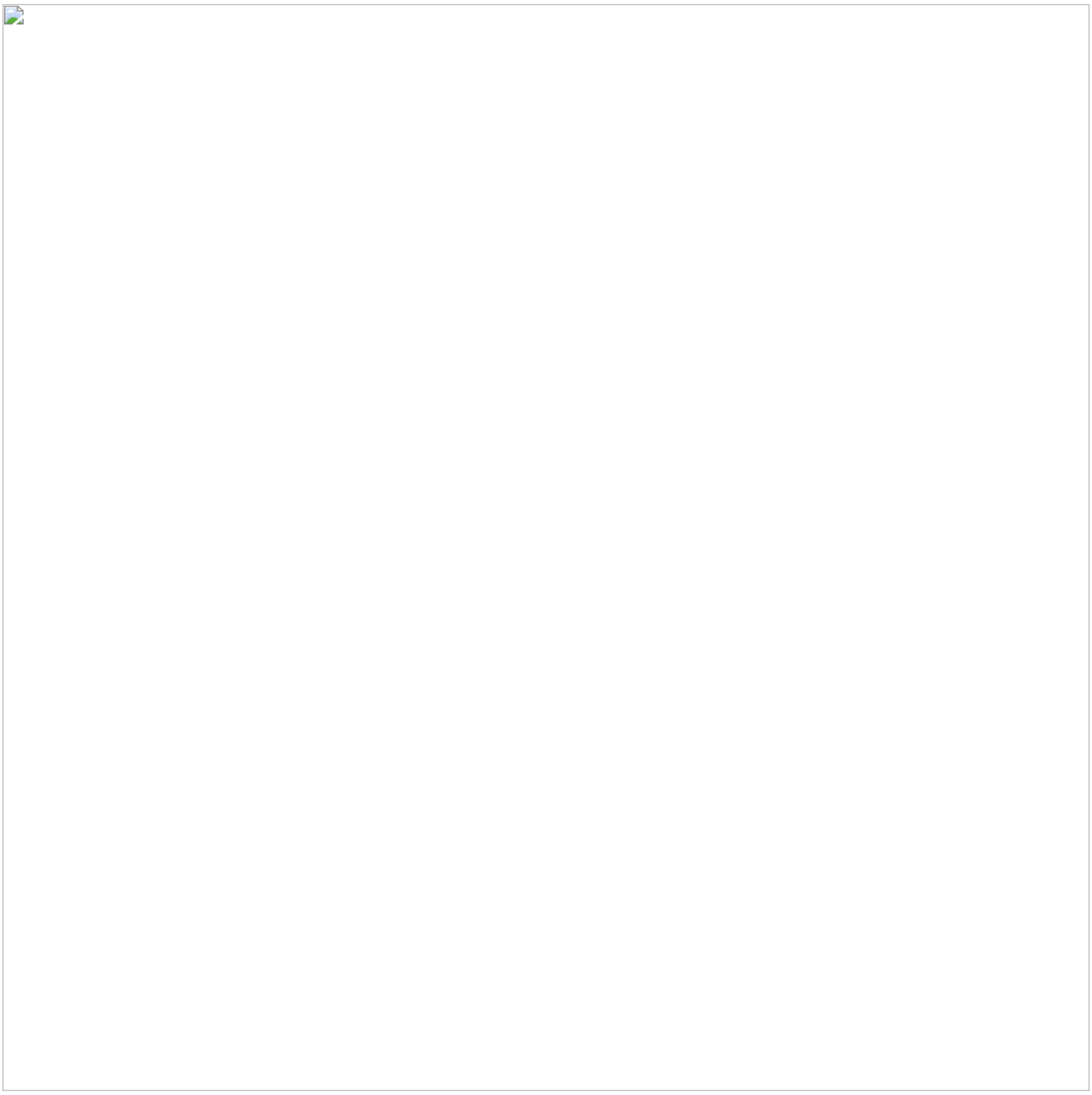
Powershell: Using a NuGet server for a PSRepository



In my post talking about Your first internal PSScript repository, I used a network share to show how easy it is to get started. I am going to take that idea to the next step and publish a module to a NuGet server. This is how we manage our modules internally.

Index

Why use a NuGet server?

A NuGet server or feed is the reccomended way to distribute your modules. This is basically what the PSGallery is. The NuGet feed will scale much better because it indexes the packages as they are uploaded.

A NuGet feed also makes it easy to distribute modules to systems that are not part of your domain. This is just a website when it comes down to it. If you open access up to your DMZ, those systems can pull modules off this feed.

A file share is fine for local testing but for production you want a NuGet feed. With Docker, you will see it is just as easy to test with a NuGet feed.

Getting Started

The first thing we need is a NuGet server. There many options for creating your own site to host feeds and there are a few pre-built servers available. The Microsoft docs site has a good list here: Hosting your own NuGet feeds. If you are already using VSTS (Visual Studio Team Services), they have a NuGet Package Management offering that you can use. For todays post, we are going to spin up a local instance in Docker.

docker run sunside/simple-nuget-server

The sunside/simple-nuget-server was the first container that I could find. Here is the command that I ran to start this instance.

```
$apiKey = New-Guid
$arguments = @(
    'run'
    '--detach=true'
    '--publish 5000:80'
    '--env', "NUGET_API_KEY=$apiKey"
    '--volume', 'C:\Images\nuget\database:/var/www/db'
    '--volume', 'C:\Images\nuget\packages:/var/www/packagefiles'
    '--name', 'nuget-server'
    'sunside/simple-nuget-server'
)

Start-Process Docker -ArgumentList $arguments -Wait -NoNewWindow
```

Make sure you save that `$apiKey`. We will need it in later examples.

I'm running this on my Windows 10 system so I had to make sure my local Docker was set to run linux containers.

Register this repository

Now that our server is running, we need to register it as a repository. The `docker run` command mapped port 80 to port 5000 on our localhost.

```
Import-Module PowerShellGet

$uri = 'http://localhost:5000'
$repo = @{
    Name = 'MyRepository'
    SourceLocation = $uri
    PublishLocation = $uri
    InstallationPolicy = 'Trusted'
}
Register-PSRepository @repo
```

This is the same example that I used in my last article except I am now using the uri to my server.

```
PS> Get-PSRepository

Name         InstallationPolicy SourceLocation
----         ------------------ --------------
MyRepository Trusted            http://localhost:5000/
PSGallery    Untrusted          https://www.powershellgallery.com/api/v2/
```

You may notice that `MyRepository` is trusted. If you take a close look at the parameters that were used to register it, I set `InstallationPolicy = 'Trusted'`.

Using your repository

You use this repository just like any other repository. I already covered how to use `Publish-Module`, `Find-Module`, and `Install-Module` from a repository in my Your first internal PSScript repository post. All those commands still apply to this repository so I am not going to cover all of them again.

One important difference is that we need to provide a NuGet API key when publishing.

```
Publish-Module -Name MyModule -Repository MyRepository -NuGetApiKey $apiKey
```

This is the same API key we specified when setting up the server.

Regular packages

The focus of this article was on Modules, but you can use NuGet to distribute other packages too. We package all of our software installs as NuGet packages and distribute them with `Install-Package`. But I do recommend that you do not mix modules and regular packages.

I found that having them in the same feed creates a lot of clutter and makes your feed more confusing. Running `Find-Package` would list your modules. If you called `Install-Package` on a module, it does not install it like `Install-Module` would. Create different feeds for different package types.

This Docker container is the simplest NuGet server possible so it only supports the one feed. If you use any of the pre-packaged NuGet servers, then you should have the option to configure more feeds.

In my next post, I cover publishing community modules to an internal Repository.

Tags: PowerShell NuGet PowerShellGet Modules

Kevin Marquette