**How to Create a NuGet Package with Cross Platform Tools**

Source: <https://docs.microsoft.com/en-us/dotnet/articles/core/deploying/creating-nuget-packages>

[Phillip Carter](https://github.com/cartermp)

9/1/2016 1 min to read Contributors

**Note**

The following shows command-line samples using Unix. The dotnet pack command as shown here works the same way on Windows.

For .NET Core 1.0, libraries are expected to be distributed as NuGet packages. This is in fact how all of the .NET Standard libraries are distributed and consumed. This is most easily done with the dotnet pack command.

Imagine that you just wrote an awesome new library that you would like to distribute over NuGet. You can create a NuGet package with cross platform tools to do exactly that! The following example assumes a library called **SuperAwesomeLibrary** which targets netstandard1.0.

If you have transitive dependencies; that is, a project which depends on another project, you'll need to make sure to restore packages for your entire solution with the dotnet restore command before creating a NuGet package. Failing to do so will result in the dotnet pack command to not work properly.

After ensuring packages are restored, you can navigate to the directory where a library lives:

$ cd src/SuperAwesomeLibrary

Then it's just a single command from the command line:

$ dotnet pack

Your /bin/Debug folder will now look like this:

$ ls bin/Debug

netstandard1.0/

SuperAwesomeLibrary.1.0.0.nupkg

SuperAwesomeLibrary.1.0.0.symbols.nupkg

Note that this will produce a package which is capable of being debugged. If you want to build a NuGet package with release binaries, all you need to do is add the -c/--configuration switch and use release as the argument.

$ dotnet pack --configuration release

Your /bin folder will now have a release folder containing your NuGet package with release binaries:

$ ls bin/release

netstandard1.0/

SuperAwesomeLibrary.1.0.0.nupkg

SuperAwesomeLibrary.1.0.0.symbols.nupkg

And now you have the necessary files to publish a NuGet package!

**Don't confuse dotnet pack with dotnet publish**

It is important to note that at no point is the dotnet publish command involved. The dotnet publish command is for deploying applications with all of their dependencies in the same bundle - not for generating a NuGet package to be distributed and consumed via NuGet.