Recompiling is a great way to quickly get your organization up to speed on managing software with Chocolatey packages.

How to Internalize An Existing Package Automatically

See Package Internalizer - Automatically Internalize/Recompile Packages.

- Have Chocolatey for Business (or Chocolatey for MSP).
- Call a command like the following:
  - `choco download notepadplusplus.commandline --internalize`
  - `choco download git.install --internalize --resources-location \\unc\share`
  - `choco download nodejs.install --internalize --resources-location http://some/internal/url --internalize-all-urls .`
- That's it! It does all of the manual steps below in a fraction of the time.

Package Internalizer can be hooked up to continuous integration automation or scheduled tasks to really reduce manual work.

How To Internalize/Recompile An Existing Package Manually

Chocolatey's community feed has quite a few packages but they are geared towards community and use the internet for downloading from official distribution sites due to copyright law and a publicly offered repository. However, they are attractive as they have everything necessary to install a piece of software on your machine. Through the internalization process, by which you take a community package and bring all of the bits internal and/or embed them into the pacakge, you can convert an existing package to be 100% offline and reliable and host it on an internal Chocolatey repository. This gives you complete control over a package and removes the aforementioned production trust and control issues.

To make the existing package local, use these steps.

1. Download the package from Chocolatey's community feed by going to the package page and clicking the download link.

2. Rename the downloaded file to end with `.zip` and unpack the file as a regular archive.

3. Delete the `_rels` and `package` folders and the `[Content_Types].xml` file. These are created during `choco pack` and should not be included, as they will be regenerated (and their existence leads to issues).

4. Next, open `tools\chocolateyInstall.ps1`.

```
Install-ChocolateyZipPackage 'notepadplusplus.commandline' 'https://notepad-plus-plus.org/re
```

5. Download the zip file and place it in the tools folder of the package.

6. If the file ends in `.exe`, create an empty file called `*.exe.ignore`, where the name is an exact match with the exe with `.ignore` appended at the end (e.g. `bob.exe` would have a file next to it named `bob.exe.ignore`).

7. Next, edit `chocolateyInstall.ps1` to point to this embedded file instead of reaching out to the internet (if the size of the file is over 100MB, you might want to put it on a file share somewhere internally for better performance).

```
$toolsDir   = "$(Split-Path -parent $MyInvocation.MyCommand.Definition)"
Install-ChocolateyZipPackage 'notepadplusplus.commandline' "$toolsDir\npp.6.8.7.bin.zip" "$t
```

The double quotes allow for string interpolation (meaning variables get interpreted instead of taken literally).

8. Next, open the `*.nuspec` file to view its contents and make any necessary changes.

```
<?xml version="1.0"?>
<package xmlns="http://schemas.microsoft.com/packaging/2010/07/nuspec.xsd">
 <metadata>
   <id>notepadplusplus.commandline</id>
   <version>6.8.7</version>
   <title>Notepad++ (Portable, CommandLine)</title>
   <authors>Don Ho</authors>
   <owners>Rob Reynolds</owners>
   <projectUrl>https://notepad-plus-plus.org/</projectUrl>
   <iconUrl>https://cdn.rawgit.com/ferventcoder/chocolatey-packages/02c21bebe5abb495a56747cb
   <requireLicenseAcceptance>false</requireLicenseAcceptance>
   <description>Notepad++ is a ... </description>
   <summary>Notepad++ is a free (as in "free speech" and also as in "free beer") source code
   <tags>notepad notepadplusplus notepad-plus-plus</tags>
 </metadata>
</package>
```

Some organizations will change the version field to denote this is an edited internal package, for example changing `6.8.7` to `6.8.7.20151202`. For now, this is not necessary.

9. Now you can navigate via the command line to the folder with the `.nuspec` file (from a Windows machine unless you've installed Mono and built choco.exe from source) and use `choco pack`. You can also be more specific and type `choco pack path\to\notepadplusplus.commandline.nuspec`. The output should be similar to below.

```
Attempting to build package from 'notepadplusplus.commandline.nuspec'.
Successfully created package 'notepadplusplus.commandline.6.8.7.nupkg'
```

10. Normally you test on a system to ensure that the package you just built is good prior to pushing the package (just the *.nupkg) to your internal repository. This can be done by using `choco.exe` on a test system to install (`choco install notepadplusplus.commandline -source .`) and uninstall (`choco uninstall notepadplusplus.commandline`).

NOTE: Originally posted at https://puppet.com/blog/chocolatey-creating-recompiled-packages and https://docs.puppet.com/pe/latest/windows_modules.html