The ultimate guide to chocolatey

2014

Zusammenfassung

Diese Dokumentation behandelt die Packet-Verwaltung Chocolatey. Chocolatey ist ein Paket Manager für Windows der apt-get nachempfunden ist. Behandelt werden alle normalen Arbeitsabläufe mit Chocolatey, die nötig sind um Computer mit Chocolatey zu konfigurieren und zu warten. Außerdem wird das erstellen von Paketen und das Verwalten FGBS spezifischer Pakete behandelt.

Inhaltsverzeichnis

1 Was ist Chocolaty?

Chocolatey ist ein Open-Source Paket-Manager ähnlich apt-get. Dass heißt, es dient zum verwalten von Software auf einem Computer. Dafür bringt es befehle mit um Software zu installieren, ein update zu machen, zu suchen und zu entfernen. Im Gegensatz zu apt-get ist Chocolatey noch recht jung. Daher funktionieren nicht immer alle Pakete, oder installieren sich anders als man es erwartet. Trotzdem hat Chocolatey eine recht aktive Community und liefert für viele Open-Source-Projekte neue Versionen binnen Minuten.

Defacto ist Chocolatey nur eine NuGet Erweiterung. Das merkt man unter anderem daran, dass der Chocolatey-Feed ein NuGet-Feed ist und dass die Pakete im Grunde NuGet Pakete sind. Dies hat den Vorteil, dass alle NuGet Pakete und einige weitere installiert werden können (z.B. Ruby On Rails).

1.1 Ich habe Problem XY und es wird hier nicht erwähnt

Ein genereller Anfangspunkt ist die Website von Chocolatey.
Falls ein Paket nicht zu funktionieren scheint, kann man es entweder über den Link bei dem Paket auf der Website herunterladen und versuchen es selbst zu reparieren oder die Paketersteller informieren. Manchmal sind die Paketersteller allerdings selber nicht mehr aktiv, dann kann man die Website-Admins informieren.
Falls es um Chocolatey Befehle oder Parameter geht, ist die Chocolatey Dokumentation sehr hilfreich. Diese ist meiner Erfahrung nach auch sehr gut gepflegt.
Außerdem findet man hier eine Übersicht über alle wichtigen Befehle von Chocolatey.
Seit Version 0.9.9 von Chocolatey ist die doku die man mit /? als parameter bekommt sehr gut und umfangreich. Dort findet man oft erste Anhaltspunkte. Außerdem gibt es ein aktualisiertes wiki unter `https://github.com/chocolatey/choco/wiki`
Falls das alles nicht hilft, findet man meist in der Google Chocolatey Group hilfe (Englisch). Oder kann auf Gitter mit den machern von Chocolatey direkt chatten (Englisch).

1.2 Seit 0.9.9 muss ich alles mit yes bestätigen

Seit 0.9.9 wird man gefragt bevor ein skript ausgeführt wird. Dies wurde als weiterer Sicherheitsmechanismus eingeführt um unbeabsichtigte Missbedienung zu erschweren. Diese Abfragen können bei jedem Befehl mit $-y$ unterdrückt werden.

2 Computer-Verwaltung mit Chocolatey

2.1 Installation

Es gibt 2 Wege Chocolatey zu installieren. Über Powershell und über die cmd. Zu beachten ist hierbei, dass man bei beiden Admin-rechte zum installieren braucht. Da sich der Befehl zum installieren eventuell in Zukunft ändert, geht man am besten auf chocolatey.org und kopiert ihn von dort.

2.2 Installieren von einzelnen Paketen

Nachdem man Chocolatey installiert hat, stehen die chocolatey Befehle von der Kommandozeile aus zur Verfügung.

2.2.1 Pakete suchen

Mit dem Befehl *choco search* $<$ *paketname* $>$ kann man alle eingetragenen Repositorys nach Paketen durchsuchen. Der Befehl hat noch weitere Einstellungsmöglichkeiten, siehe Doku. Mit dem Befehl

```
choco search git
```

kann man zum Beispiel Pakete suchen deren Name oder Beschreibung das Suchwort enthält. Alternativ kann man auch im chocolatey Feed auf chocolatey.org suchen.

2.2.2 Pakete installieren

Mit dem Befehl *cinst* (kurz für *chocolatey install*) kann man eins oder mehrere Pakete installieren. Mit dem Befehl

```
cinst git eclipse-java-luna
```

installiert man zum Beispiel das git und das eclipse-java-luna Paket.
Recht nützlich ist der Parameter $-force$ weil er das neuinstallieren eines Paketes erzwingt. Was vor allem beim erstellen und debuggen von Paketen sehr nützlich ist. Mit $-y$ kann man die abfragen während der Installation alle standardmäßig mit yes beantworten. Weitere interessante Parameter sind $-noop$, $--source$, $--prerelease$, $--version$, $--x86$, $-installArguments$

2.2.3 GUI? GUI!

`https://chocolatey.org/packages/ChocolateyGUI`

2.2.4 Installationsort

Siehe hier.
Seit Chocolatey 0.9.9 ist Chocolatey von Powershell-Scripten in C-Sharp Code migriert worden, damit haben sich auch die Ursprünglichen Gründe des zuvor eigenartigen Installationsorts erledigt. Nun ist Chocolatey im Programmordner zu finden.

2.3 Installieren von Paketgruppen

Eine Paketgruppe kann man installieren indem man statt des Paketnamens den Pfad zur $package.config$ Datei angibt. Zum Beispiel

```
cinst dev-packages.config
```

würde alle in $dev-packages.config$ spezifizierten Pakete installieren. Für mehr Informationen über Paketgruppen siehe 'Erstellen von Paketgruppen'.

2.4 Updaten von Paketen

Der Befehl *cup* $<$ *paketname* $>$ updatet ein Paket auf die neuste Version. Der Befehl *cup* ohne Parameter updatet Chocolatey.

2.5 Entfernen von Paketen

Mit dem Befehl *cuninst* < *paketname* > kann man einige Pakete deinstallieren. (Stand: v.0.9.8.27) Bisher funktioniert dies allerdings nur bei Paketen die entweder als portable installiert wurden oder ein Uninstall skript mitliefern. Andersweitig installierte Pakete kann man meist recht gut über die Windowsfunktion wieder entfernen.

3 Erstellen eigener Pakete

Die Dokumentation hierzu findet sich hier. Im folgenden eine kurze Übersicht:

3.1 Was ist ein Paket?

Ein Paket ist eine *.nupkg* Datei und besteht aus:

1. name.nuspec: Die Paketbeschreibung. Hier sind unter anderem die PaketID, Link zum Logo, Lizenzlinks aufgeführt.
2. Tools/ChocolateyInstall.ps1: Das Skript das die Installation durchführt.
3. Tools/ChocolateyUninstall.ps1: Das Skript das die Deinstallation durchführt.

Bei den allermeisten Paketen war es das auch schon. Es gibt viele Pakete ohne das Uninstall Skript und einige die außerdem noch direkt einen Installer (z.B. eine MSI oder EXE) oder andere Dateien mitbringen.
Chocolatey Pakete können Abhängigkeiten(Dependencies) haben. Wenn ein Paket also das .NET Framework benötigt, setzt man dieses einfach als Abhängigkeit.

3.2 Erstellen normaler Pakete

Im folgenden gehe ich die Arbeitsschritte beim erstellen eines Paketes durch. Wir starten mit der Installation des nützlichen Tools warmup beim erstellen von Paketen. Dann beschreiben wir das Paket in der *.nuspec*-Datei und beschäftigen uns schließlich mit der eigentlichen Installation die das Paket durchführt.

3.2.1 warmup

In 0.9.9 wurde warmup durch choco new pkgname ersetzt.
Die Dokumentation zur Benutzung von Warmup findet man hier.

Warmup ist ein Tool, dass es einem ermöglicht Paket-templates einfach zu erzeugen. Dafür installiert man das Tool, setzt einige Parameter, wie z.B. den Autor und generiert dann ein template für sein Paket. Um sein Paket-Template zu erzeugen ruft man

```
warmup chocolatey <paketname>
```

auf. Es gibt auch noch andere Template-Vorlagen aber im Moment ist das normale Chocolatey-Template das einzig Interessante für uns.

3.2.2 Die .nuspec Datei

Wie unter 3.1 beschrieben ist, beschreibt die nuspec Datei das Paket.
Wenn man das warmup-Template verwendet hat, muss man die Felder in der nuspec-Datei nur noch ausfüllen. Für genauere Infos siehe die NuSpec-Dokumentation. Abhängigkeiten die über die *.nuspec* spezifiziert werden,

werden in der nächsten Version behandelt.
Es folgt eine Liste der wichtigsten Felder:

id	Der Name des Pakets ohne Leerzeichen. Wird z.B. verwendet beim installieren. Oft getrennt mit . oder mit -
title	Der Anzeigename in der Gallerie.
version	Die Version des Paketes.
authors	Der Ersteller der Software.
owners	Der Ersteller des Paketes.
description	Ausführliche Beschreibung des Paketes.
projectUrl	Url zur Projektseite.
tags	Schlüsselworte zum Paket
copyright	Die Copyright Lizenz.
licenseUrl	Url zur Lizenzvereinbarung.
requireLicenseAcceptance	Ob eine manuelle Akzeptanz der Lizenz erforderlich ist.
releaseNotes	Beschreibung der Veränderungen zum letzten Paket.
iconURL	URL zu dem Icon der Software
files	Dateien im Paket, muss meist nicht bearbeitet werden.

3.2.3 Abhängigkeiten in der .nuspec Datei

Zu Bemerken ist noch, dass in der *.nuspec* Abhängigkeiten z.B. zu anderen Paketen angegeben werden können. Wenn man zum Beispiel ein Feature von chocolatey benutzt, das erst in einer bestimmten Version dazu kam kann man eine minimale Version von chocolatey erzwingen mit:

```
<dependencies>
    <dependency id="Chocolatey" version="0.9.8.20" />
</dependencies>
```

Oder es wird das *.NET*-Framework benötigt:

```
<dependencies>
    <dependency id="DotNet4.5.1" />
</dependencies>
```

So kann man sehr einfach bestehende Pakete wiederverwenden.

3.2.4 /tools/chocolateyInstall.ps1

Diese Datei wird aufgerufen beim Installieren des Paketes und führt die Installation durch.
Im einfachsten Fall sieht das Skript so aus:

```
Install-ChocolateyPackage '<paketid>' '<exe oder msi>'
 '<silentparameter>' '<url>'
```

Also z.B.

```
Install-ChocolateyPackage 'mdprojecttimer' 'exe'
 '/s' 'http://[...]/mdPROJECTTIMER%20Standard%20Setup.exe'
```

Für alle weiteren Befehle sind gut Dokumentierte Beispiele in der Template Datei vorhanden.

3.2.5 Erstellen des Pakets

Um das Paket zu erstellen navigiert man einfach mit der Konsole in den Ordner in dem die *.nuspec*-Datei ist und führt das Kommando *cpack* aus. Damit erhält man eine *.nupkg* Datei im Ordner des Aufrufs. Diese Datei

repräsentiert das ganze Paket. Falls man Pakete aus irgend einem Grund wieder öffnen möchte, z.B. um die enthaltenen Dateien zu öffnen, kann man diese *.nupkg* Dateien einfach mit z.B. 7-Zip öffnen.

3.2.6 Hochladen des Pakets

In das Öffentliche Repository: Auf chocolatey.org einen Account erstellen und Hochladen
In das Private Repository: Einfach in den dafür vorgesehenen Ordner kopieren.
Wenn man in den Öffentlichen Feed hochlädt, ist es praktisch, die Pakete gleich auch auf github zu veröffentlichen, wie zum Beispiel hier: `https://github.com/thigg/chocolateypackages`. Hierüber kann man dann auch gleich stabile Links zu den Icons erzeugen.

3.3 Chocolatey Infrastruktur am FGBS

Es gibt die Möglichkeit einen eigenen Feed zu hosten oder zum Beispiel einen *myget* Feed zu verwenden. Allerdings hat sich dafür bisher keine Notwendigkeit ergeben. Ein eigener Feed ist vor allem dann interessant, wenn man auch ein Web-Interface dafür haben möchte. Daher ist die einfachste Lösung einen Ordner auf einem Networkshare, der für jeden erreichbar ist, als Feed zu verwenden.

3.3.1 Bereitstellen eigener Pakete

Wenn man einen Feed für Chocolatey durchsuchbar machen möchte (damit z.B. *cinst* und *chocosearch* mit dem eigenen Feed funktionieren) editiert man im Chocolatey Ordner (Normalerweise zu finden unter: C:/ProgramData/chocolatey/ die *chocolatey.config* Datei, wie im Beispiel unten gezeigt:

```
<?xml version="1.0"?>
<chocolatey>
    <useNuGetForSources>false</useNuGetForSources>
    <checksumFiles>true</checksumFiles>
    <virusCheck>false</virusCheck>
    <sources>
        <source id="chocolatey"
            value="https://chocolatey.org/api/v2/" />
        <source id="ourOwnFeed"
            value="X:\Networkshare\chocoPakete\" />
    </sources>
</chocolatey>
```

Wie man im Beispiel sieht, kann man, wenn man anstatt eines normalen Feeds ein Netzlaufwerk verwendet, dieses einfach als source eintragen.

3.4 Erstellung von Paketgruppen

Eine Paketgruppe ist eine *.config* Datei die in etwa so aussieht:

```
<?xml version="1.0" encoding="utf-8"?>
<packages>
<package id="apackage" />
<package id="anotherPackage" version="1.1" />
<package id="chocolateytestpackage" version="0.1" source="somelocation" />
<package id="alloptions" version="0.1.1"
source="https://somewhere/api/v2/" installArguments=""
packageParameters="" forceX86="false" allowMultipleVersions="false"
ignoreDependencies="false"
```

```
/>
</packages>
```

Das letzte Packet gibt mit *installerArguments* direkt Parameter an den Installer weiter. Da der übergebene String einfach an den Installer angehängt wird, kann man beliebige Parameter anhängen. Parameterformat und Parametertrennung funktioniert wie bei einem normalen Aufruf aus der Konsole.
Die möglichen Parameter in einer Config-Datei sind:

1. *id* die ID des zu installierenden Pakets.
2. *version* die zu installierende Version.
3. *source* von wo das Paket installiert werden soll
4. *installerArguments* Argumente die an den Installer-Aufruf angehängt werden sollen.

Achtung: bisher gibt es noch kein Release (Stand 0.9.8.28) was den Parameter *installerArguments* unterstützt.

4 Anmerkungen

4.1 Weitere Informationen zum Erstellen von Paketen

Lesenswert ist diese Seite. Außerdem wurde Package-Moderation eingeführt. Das heißt das Pakete erst freigegeben werden müssen, bevor sie ohne Warnung auf dem Öffentlichen Feed erscheinen. Für mehr Informationen siehe hier.

4.2 Bekannte Probleme

4.2.1 Deinstallieren von Paketen

Funktioniert nicht immer richtig, geht oft gar nicht direkt über Chocolatey. Bekanntes Beispiel dafür, dass das Entfernen nicht richtig funktioniert, ist MinGW.

4.3 Weitere Möglichkeiten

4.3.1 Hosten eines Internen Chocolatey Feeds

Eine sehr gute anleitung zum Hosten von Chocolatey Feeds findet man hier.

4.3.2 Boxstarter

Zu erwähnen ist noch das auf Chocolatey basierende Projekt Boxstarter. Das noch einige Features mitbringt um neuinstallationen auf PC's durchzuführen.

Das folgende Script konfiguriert zum Beispiel den Windows Explorer, dass Versteckte Dateien und Dateierweiterungen angezeigt werden. Außerdem werden eine handvoll Pakete installiert.

```
Set-WindowsExplorerOptions -EnableShowHiddenFilesFoldersDrives\
      -EnableShowProtectedOSFiles -EnableShowFileExtensions
Enable-RemoteDesktop

cinst fiddler4
```

```
cinst git-credential-winstore
cinst console-devel
cinst sublimetext2
cinst poshgit
cinst dotpeek

cinst Microsoft-Hyper-V-All -source windowsFeatures
cinst IIS-WebServerRole -source windowsfeatures
```

Man kann dieses Skript zum Beispiel mit einer Zeile ausführen:

```
START
https://boxstarter.org/package/nr/url?
            https://server.org/gistfile1.txt
```

wobei das *nr* in der URL für *no-reboot* steht. Man ist allerdings nicht auf den Offiziellen Boxstarter Server angewiesen.
Der Vorteil an Boxstarter für das FGBS ist derzeit, dass *installArgs* für Pakete in einer Gruppe angegeben werden können obwohl dies derzeit nicht von *package.config* Dateien Unterstüzt wird. Man kann mit Boxstarter sogar einen PC über das Netzwerk konfigurieren, auch wenn weder Chocolatey noch Boxstarter installiert ist.