



A sweet Conference focused on  
Windows Automation (WinOps)

# killing golden images in your infrastructure setups

---

a software dev talking about system management



# Manfred Wallner

software developer (in automotive industry)

hobby photographer

travel enthusiast



Chocolatey user since summer 2016

volunteer package maintainer

**THE STORY BEGINS**  
**18 YEARS AGO**

golden images



... were sweet and promising ...



for quite some time



... the future looked bright and exciting ...

but after a while



... something changed ...

golden images were not nice anymore



... they were rotting from the inside out ...



I've been through this ...

but for real - they're not all that bad

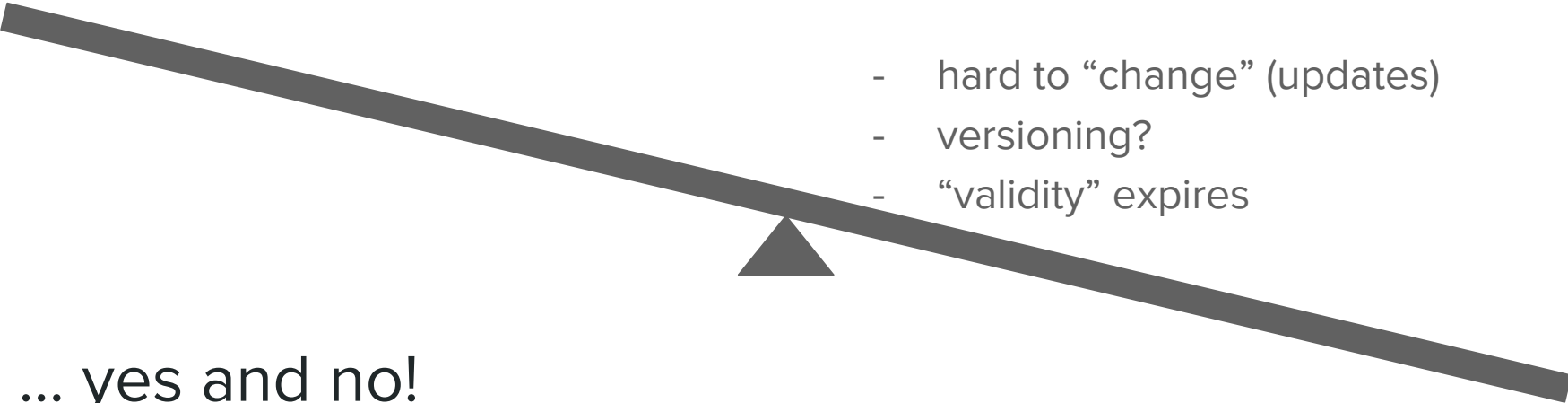
- + easy to understand
- + quick “rollout” of complex setups
- + “no dependencies”

- hard to “change” (updates)
- versioning?
- “validity” expires



# but for real - they're not all that bad?

- + easy to understand
- + quick “rollout” of complex setups
- + “no dependencies”

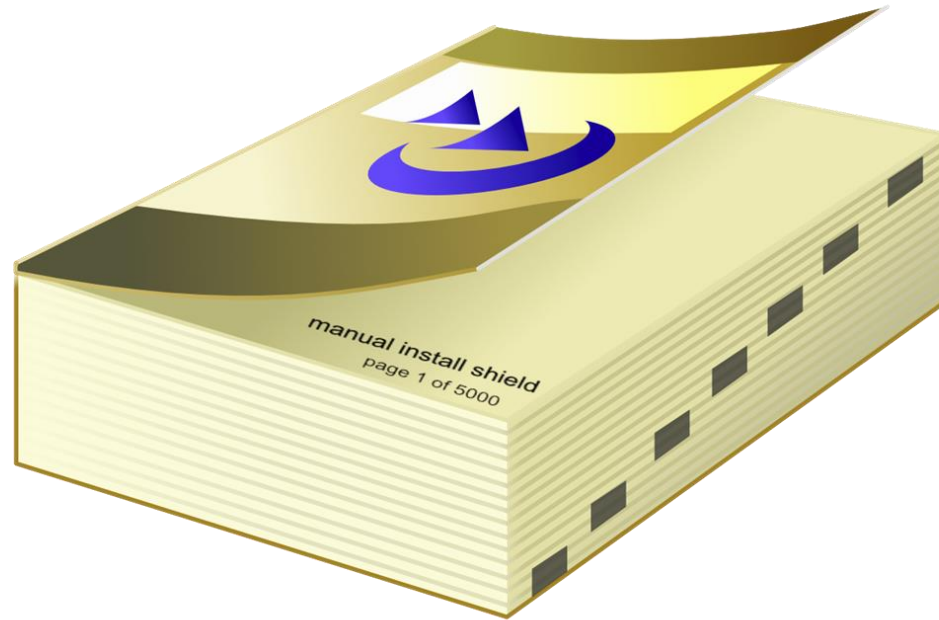
- 
- hard to “change” (updates)
  - versioning?
  - “validity” expires

... yes and no!

golden images ...



another extreme ...



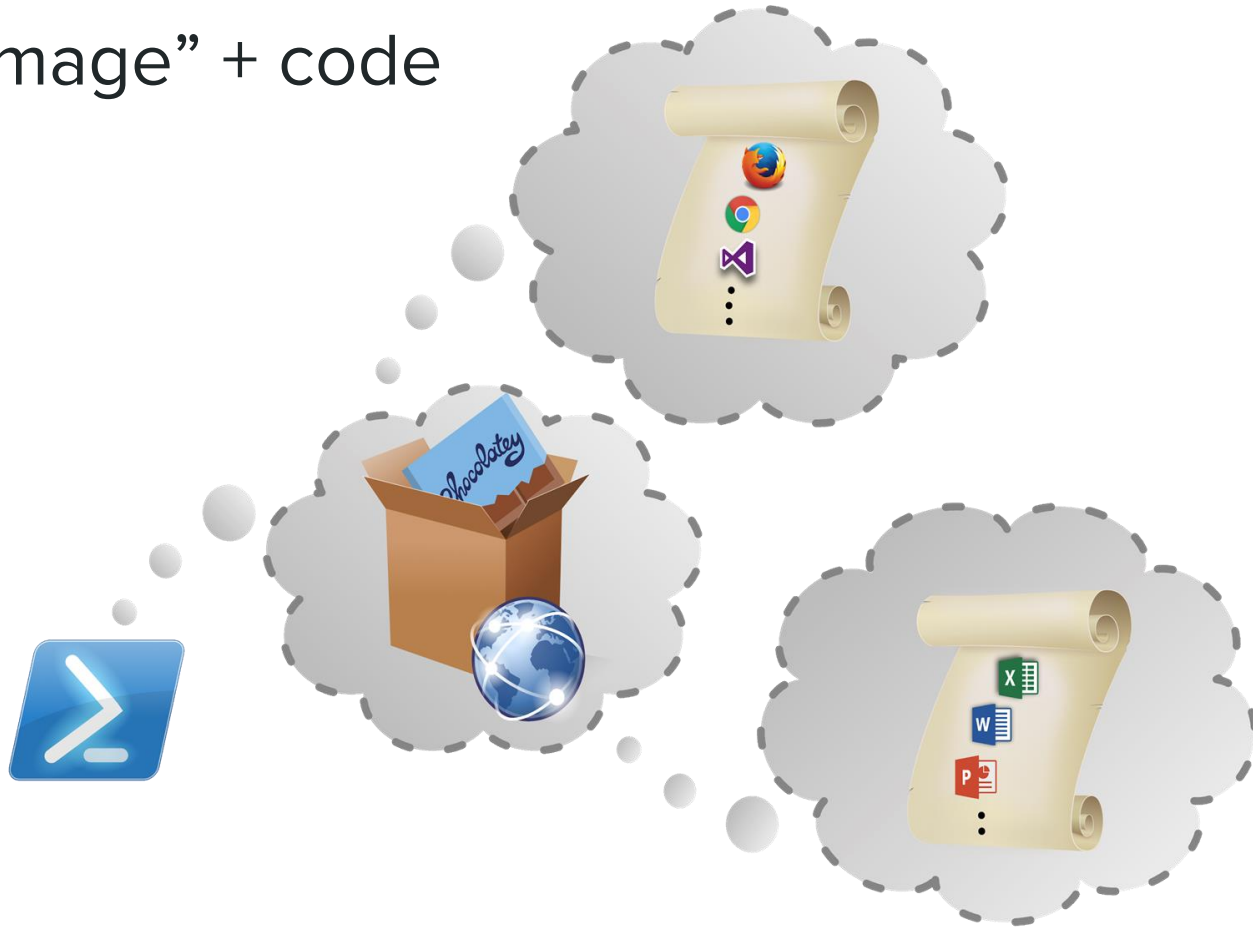


# base “image” + code

- *base Windows install*
- Packer
- Vagrant
- Docker
- ...



base “image” + code



# Chocolatey!

**repeatable**

**testable**

**version controlled**

**automated**

**deployment**

repeatable

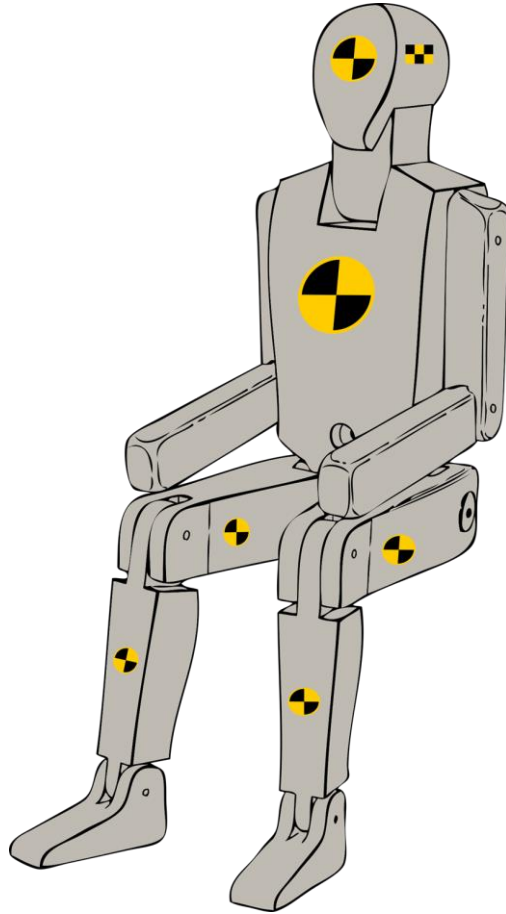


version controlled





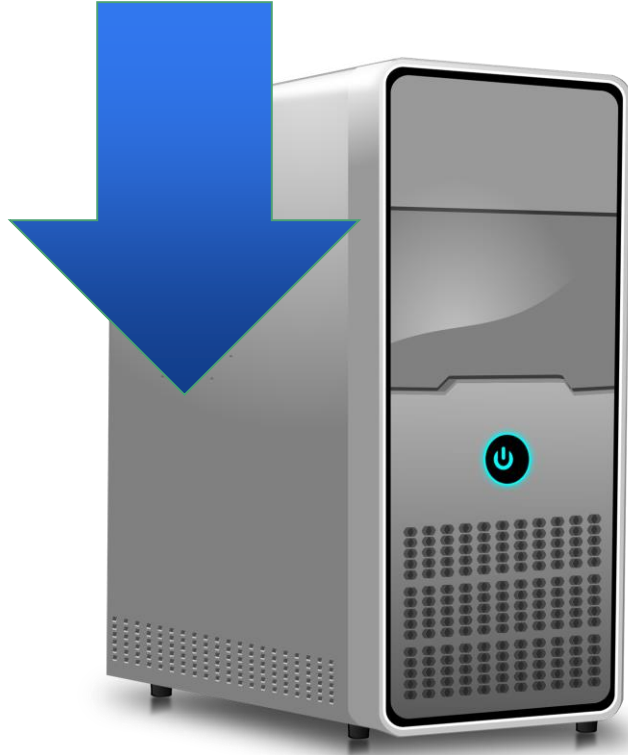
testable



automated

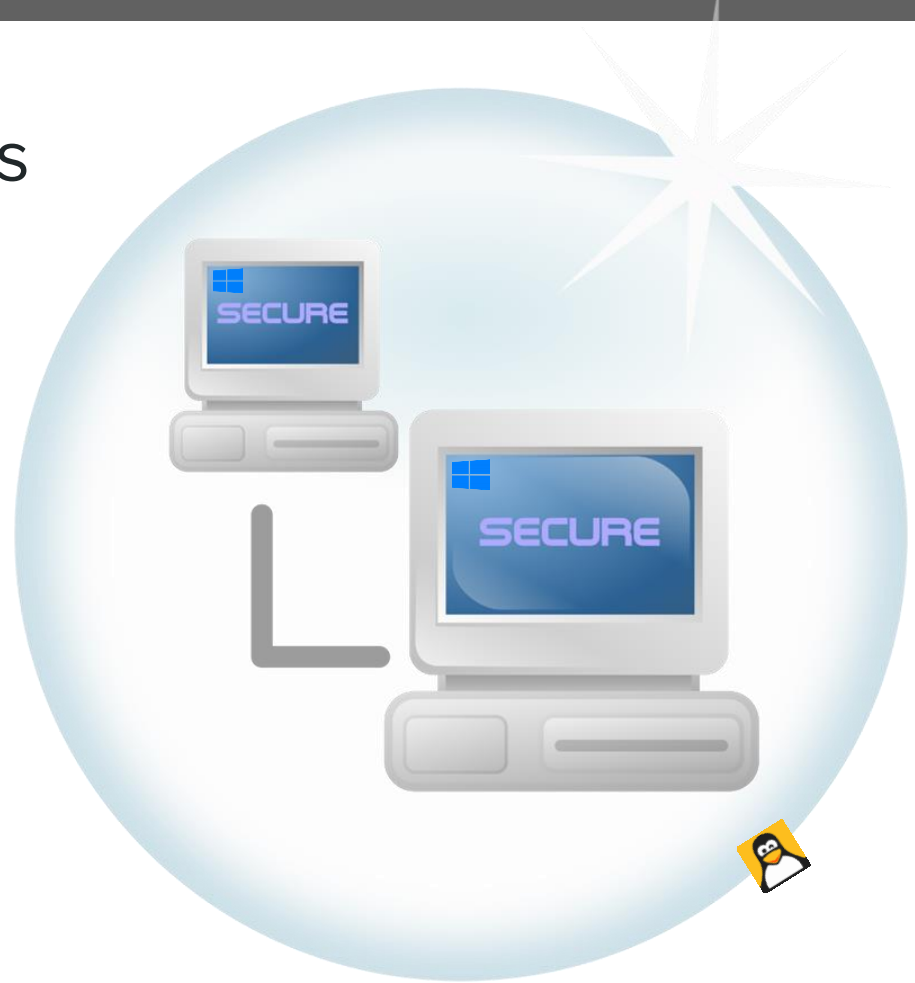


deployment



# Chocolatey 4 organizations

- internal package repository
- offline Chocolatey installation
- internalizing packages
- custom packages



# internal repository

- Chocolatey.Server
- ProGet
- Artifactory
- Samba share
- ...





# offline Chocolatey install

<https://chocolatey.org/install.ps1>

```
...  
Write-Output "Getting Chocolatey from $url."  
Download-File $url $file  
...  
Download-File 'https://chocolatey.org/7za.exe' "$7zaExe"  
...
```

<https://chocolatey.org/docs/how-to-setup-offline-installation>

<https://github.com/mwallner/killing-golden-images-with-chocolatey>

[https://github.com/pauby/SoftwareAutomation/blob/master/Chocolatey/corporate\\_install.ps1](https://github.com/pauby/SoftwareAutomation/blob/master/Chocolatey/corporate_install.ps1)

package internalization

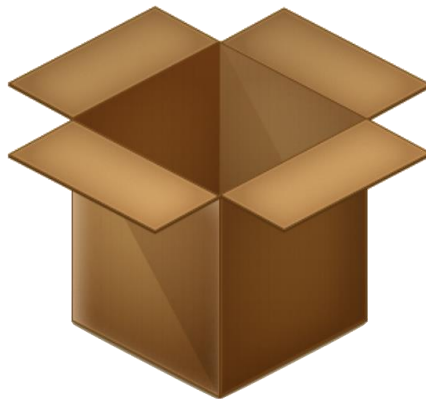
=> later!

custom packages

**Internal applications**  
**anything**  
**configuration**  
**licensing**

... there's more ...

# Boxstarter



<https://boxstarter.org/>



# Boxstarter



# Boxstarter - remote install

```
PS> Install-BoxstarterPackage -ComputerName
starkillermain01 -PackageName blowup -Credential
$secretcreds
Boxstarter Version 2.11.0
(c) 2018 Chocolatey Software, Inc, 2012 - 2018 Matt Wrock. https://boxstarter.org

Boxstarter: Configuring local PowerShell Remoting settings...
Boxstarter: Configuring CredSSP settings...
Boxstarter: Testing remoting access on starkillermain01.....

. . .
```



This PC



Control Panel

Visual Studio  
Code

Recycle Bin



WeirdPkg

Boxstarter  
Shell Windows Server 2012 R2

Windows Server 2012 R2 Datacenter

Build 9600



12:50 PM

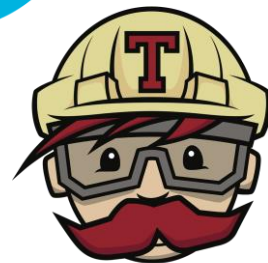


Strg Rechts

... isn't this talk supposed to be about golden images?

# CI joining the party

- Powershell / CLI - integrates everywhere
  - AppVeyor
  - Travis
  - Win Task Scheduler
  - **Jenkins**
    - Plugin [build-with-parameters](#)
    - Plugin [PowerShell](#)
  - ...



Travis CI



# Jenkins

## getting notified of possible updates (manual)

```
PS> choco outdated -r
AzurePowerShell|6.4.0|6.6.0|false
jdk10|10.0.1|10.0.2|false
notepadplusplus|7.5.7|7.5.8|false
...
```

# getting notified of possible updates

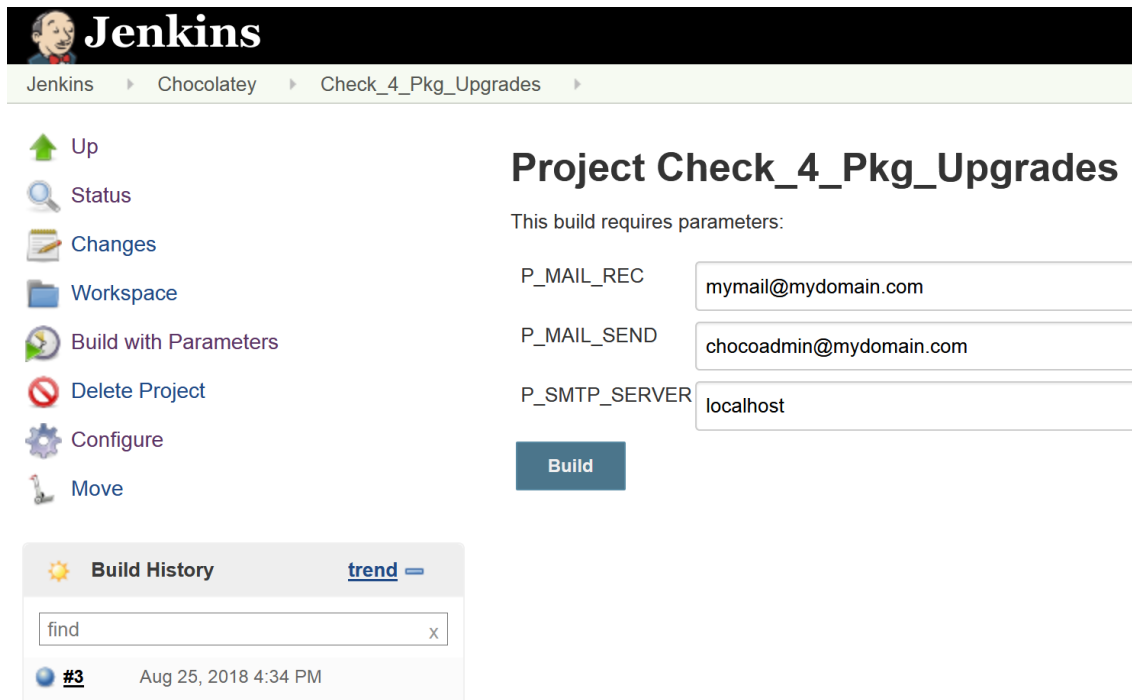
*-r, --limitoutput, --limit-output*

LimitOutput - Limit the output to essential information

*--noop, --whatif, --what-if*

NoOp / WhatIf - Don't actually do anything.

# getting notified of possible updates (with Jenkins CI)

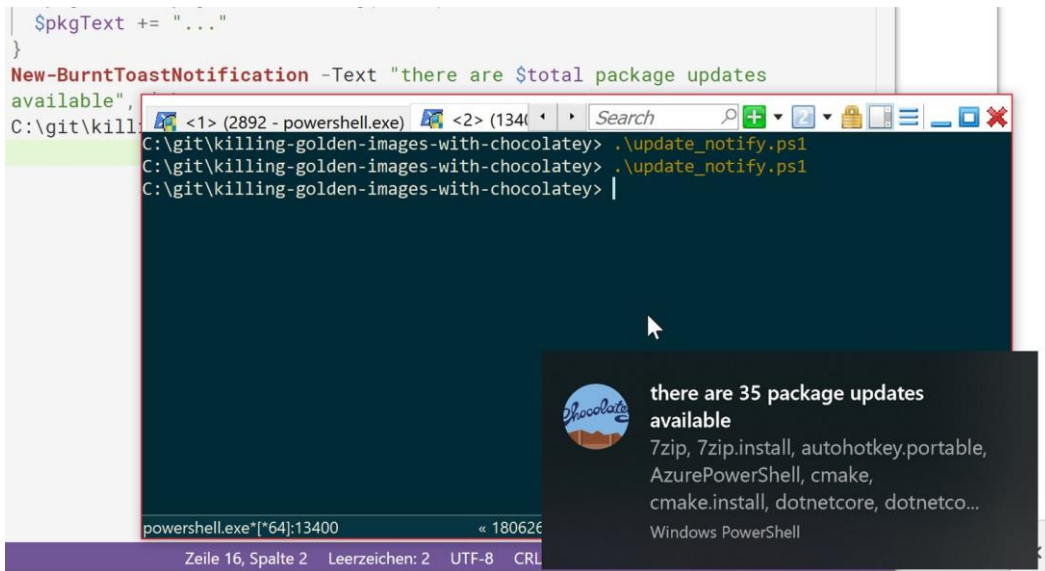


The screenshot shows the Jenkins web interface. At the top, there's a black header with the Jenkins logo and name. Below it, a breadcrumb trail shows 'Jenkins' > 'Chocolatey' > 'Check\_4\_Pkg\_Upgrades'. On the left sidebar, there are several icons and links: 'Up' (green arrow), 'Status' (magnifying glass), 'Changes' (notepad), 'Workspace' (folder), 'Build with Parameters' (play button), 'Delete Project' (red circle with slash), 'Configure' (gear), and 'Move' (key). The main content area is titled 'Project Check\_4\_Pkg\_Upgrades' and states 'This build requires parameters:'. Below this, there are three input fields: 'P\_MAIL\_REC' with the value 'mymail@mydomain.com', 'P\_MAIL\_SEND' with the value 'chocoadmin@mydomain.com', and 'P\_SMTP\_SERVER' with the value 'localhost'. A blue 'Build' button is positioned below these fields. At the bottom, there's a 'Build History' section with a 'trend' link and a search box containing the text 'find'. Below the search box, there's a list of builds, with the first one being '#3' dated 'Aug 25, 2018 4:34 PM'.



# getting notified of possible updates (Win tasks)

```
$pkgText += "..."  
}  
New-BurntToastNotification -Text "there are $total package updates  
available",  
C:\git\kill
```



The screenshot shows a PowerShell terminal window with the following commands and output:

```
C:\git\killing-golden-images-with-chocolatey> .\update_notify.ps1  
C:\git\killing-golden-images-with-chocolatey> .\update_notify.ps1  
C:\git\killing-golden-images-with-chocolatey> |
```

A Windows Burnt Toast notification is displayed over the terminal. The notification has a dark background and a light blue circular icon with the word "chocolatey" in white. The text of the notification reads:

there are 35 package updates available  
7zip, 7zip.install, autohotkey.portable,  
AzurePowerShell, cmake,  
cmake.install, dotnetcore, dotnetco...

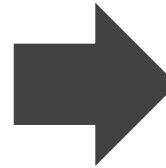
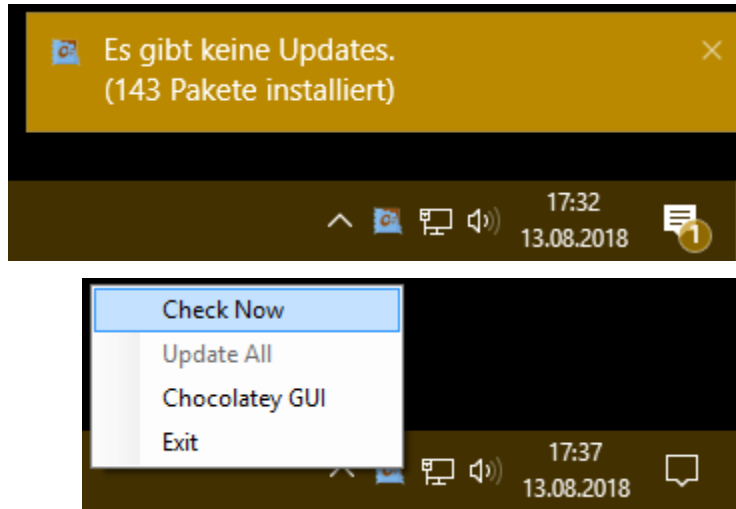
The notification is titled "Windows PowerShell" at the bottom.



# getting notified of possible updates

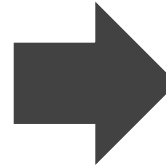
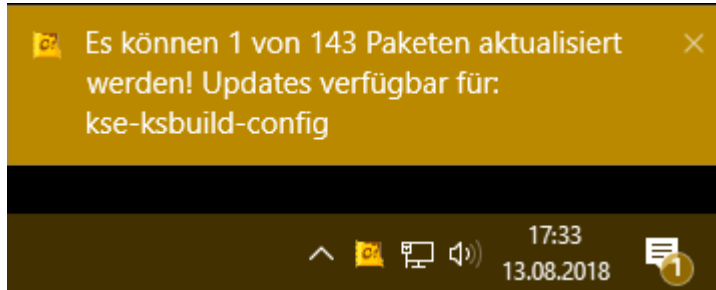
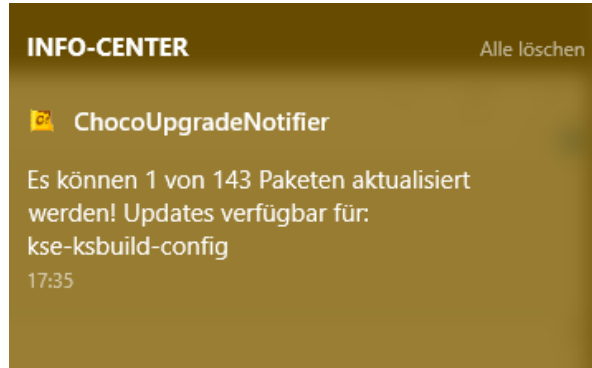


```
PS> choco install -y ChocoUpgradeNotifier
```



no updates available

# getting notified of possible updates



1 update available

# automated package internalization

```
PS> choco download --internalize $pkg `
      --resources-location="$uncshare\$pkg" `
      --source="$basefeed" --no-progress
```

...

## automated package internalization

...

```
PS> $genpkg = ((gci *.nupkg -recurse).FullName `
               | Select-String -Pattern $pkg)
```

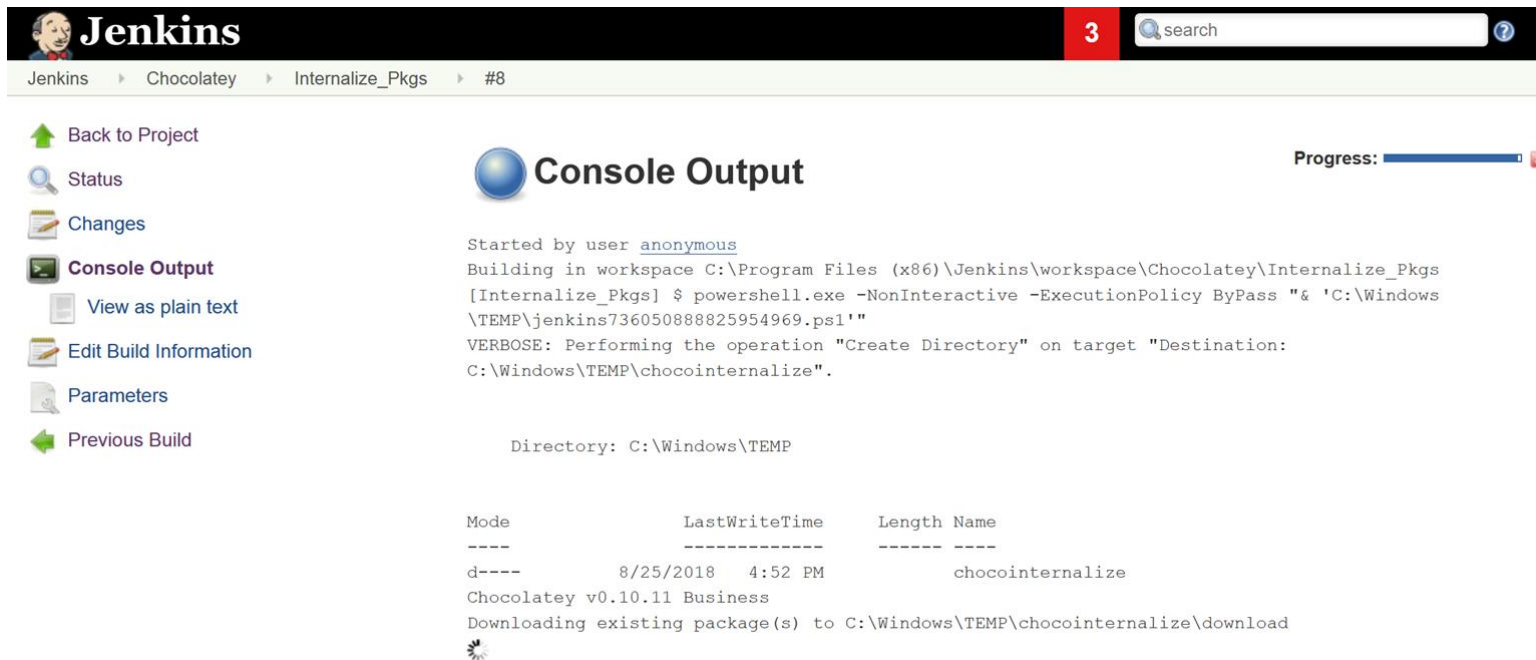
...

## automated package internalization

...

```
PS> $genpkg % { choco push $_.FullName `
                --source="$targetserver" `
                --api-key="$apikey" -Verbose }
```

# automated package internalization (with Jenkins #1)



The screenshot shows the Jenkins web interface. At the top, there's a black header with the Jenkins logo and name. Below it, a breadcrumb trail shows the path: Jenkins > Chocolatey > Internalize\_Pkgs > #8. On the left, a sidebar contains links: 'Back to Project', 'Status', 'Changes', 'Console Output' (which is highlighted), 'View as plain text', 'Edit Build Information', 'Parameters', and 'Previous Build'. The main area is titled 'Console Output' and shows the build's progress bar. The console output text is as follows:

```
Started by user anonymous
Building in workspace C:\Program Files (x86)\Jenkins\workspace\Chocolatey\Internalize_Pkgs
[Internalize_Pkgs] $ powershell.exe -NonInteractive -ExecutionPolicy Bypass "& 'C:\Windows\TEMP\jenkins736050888825954969.ps1'"
VERBOSE: Performing the operation "Create Directory" on target "Destination: C:\Windows\TEMP\chocointernalize".
```

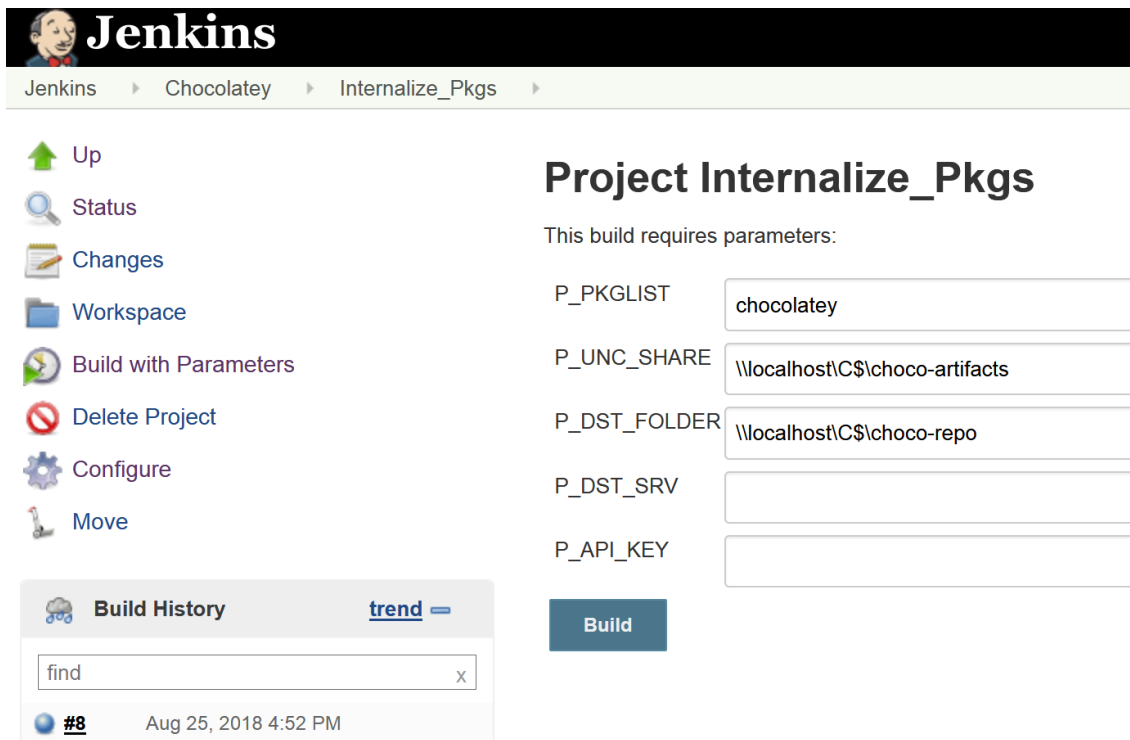
Below the text, there's a section for the file system:

Directory: C:\Windows\TEMP

Mode	LastWriteTime	Length	Name
d----	8/25/2018 4:52 PM		chocointernalize

Chocolatey v0.10.11 Business  
Downloading existing package(s) to C:\Windows\TEMP\chocointernalize\download

# automated package internalization (with Jenkins #2)



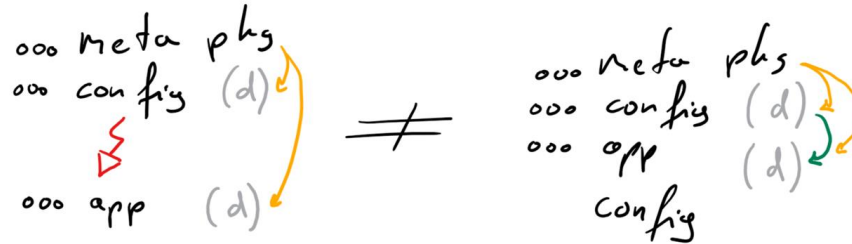
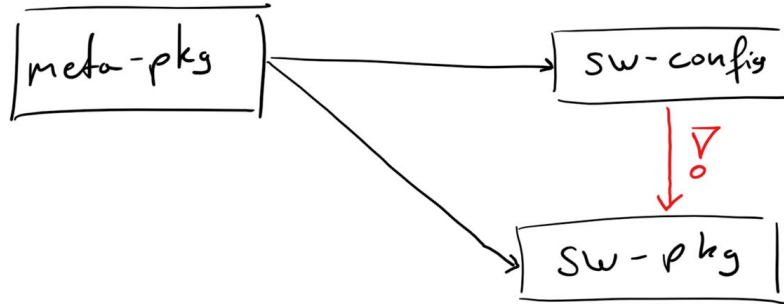
The screenshot displays the Jenkins web interface. At the top, the Jenkins logo and name are visible. Below the navigation bar, the breadcrumb path is 'Jenkins > Chocolatey > Internalize\_Pkgs'. On the left sidebar, there are links for 'Up', 'Status', 'Changes', 'Workspace', 'Build with Parameters', 'Delete Project', 'Configure', and 'Move'. The main content area is titled 'Project Internalize\_Pkgs' and states 'This build requires parameters:'. Below this, there are input fields for 'P\_PKGLIST' (containing 'chocolatey'), 'P\_UNC\_SHARE' (containing '\\localhost\C\$\choco-artifacts'), 'P\_DST\_FOLDER' (containing '\\localhost\C\$\choco-repo'), 'P\_DST\_SRV' (empty), and 'P\_API\_KEY' (empty). A 'Build' button is located below these fields. At the bottom left, there is a 'Build History' section with a search bar containing 'find' and a list of builds, with the most recent being '#8' dated 'Aug 25, 2018 4:52 PM'.



Package Name	Internalize / Manual	Internalize / Jenkins
VirtualBox	~ 15 minutes	~ 1 minute
Firefox	~ 5 minutes	~ 30 seconds
Wireshark	~ 10 minutes	~ 30 seconds
WinPcap	~ 5 minutes	~ 20 seconds

# package structure / dependency design

- beware wrong dependency chains!



tl;dr - meta packages are great!

if setup/used correctly





# Chocolatey @ ksengineers



100 internalized packages

300 custom packages

3.000 packages total

100.000 package downloads

custom notifications

logical application groups

(meta packages)

# conclusion

- we don't need golden images anymore
- we are better off without them
- transition is NOT hard
- Boxstarter is AWESOME



more...

let's talk!

I'm on the internets!

[github.com/mwallner](https://github.com/mwallner)

[@schusterfredl](https://twitter.com/schusterfredl)

[chocolatey@mwallner.net](mailto:chocolatey@mwallner.net)





