Autopkg, Munki & You: Automating 3rd party application updates.

•••

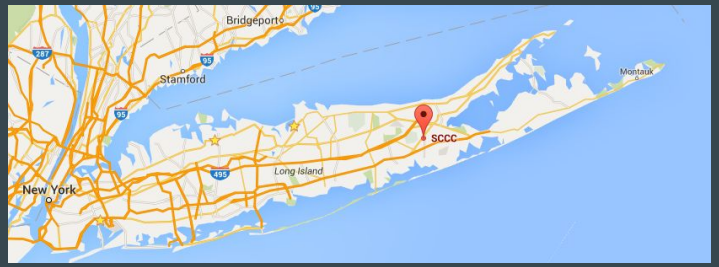
Damon Vogel
SUNY - Suffolk County Community College

MACADMINS CONFERENCE
AT PENN STATE

Who am I?

Damon Vogel - Professional Assistant II

- Suffolk County Community College on Long Island, New York
- I am the only full-time Apple Technician on The Eastern Campus of Suffolk County Community College.
- I support the Graphic Design, Computer Art, Photography, & Interior Design programs.
- I’m a big fan of the Graphic User Interface.



Google Maps

Where am I from?

Suffolk County Community College - Long Island, New York

With approximately 27,000 students enrolled at three campuses in Selden, Brentwood and Riverhead, Suffolk County Community College is the largest community college in New York State.

We offer Associate in Arts (A.A.), Associate in Science (A.S.), Associate in Applied Science (A.A.S.) degrees and professional certificates in 100 programs of study.

The Educational Technology Unit

The Educational Technology Unit manages all computers on campus that do not reside in Faculty offices, especially Computer Labs & Classrooms.

My two co-workers and I share the duties to manage approximately 600 PC workstations and 10 servers (6 virtual) on the campus.

I manage 100 Macintosh workstations, 5 Macintosh servers, and 2 Linux servers.

30 of the Macintosh workstations are dual-boot Apple OS X and Windows OS.

Why are we talking about this?

There are a lot of applications that we use on a daily basis....

- Google Chrome
- Mozilla Firefox
- Adobe Acrobat
- Java
- And More....

Automate that workflow….

Most of the time, application updates are solid and relatively bug-free.

Even if they have bugs, they are minor ones.

Or not.

Occasionally, when an application update goes bad, it goes really bad.

> Failures took down United Airlines, the Wall Street Journal's website and even halted the New York Stock Exchange.

http://money.cnn.com/2015/07/08/technology/united-nyse-wsj-down/

Convenience ...

As a one-man show, I needed to minimize the time that I spend on repetitive tasks that I can automate.

Downloading, testing and deploying applications manually takes time that can be better spent.

Vs Functionality

I also wanted to balance my time-saving with the ability to make sure that I wasn't creating more problems than I was fixing.

So I developed a workflow that allowed me to automate most of the steps, while not committing me to an install until I had the chance to check out the applications I felt I needed to.

The Cast of Characters

The Heavy-Lifter

Munki - https://www.munki.org/munki/

Made by Greg Neagle

Munki is a set of tools that, used together with a web-server based repository of packages and package metadata, can be used by OS X administrators to manage software installs (and in many cases removals) on OS X client machines.

Additionally, Munki can be configured to install Apple Software Updates, either from Apple's server, or yours.

What do I use it for?

Munki is the central repository for all of the applications that I store on a local server to be deployed to my workstations.

Munki allows deployment based off Manifests, or lists of items written in an XML format.

I have multiple Manifests, each based upon the type of user primarily on the computer.

The Heavy-Lifter's Older Brother

MunkiAdmin - https://github.com/hjuutilainen/munkiadmin

Made by Hannes Juutilainen.

MunkiAdmin is a GUI for managing munki repositories.

The Heavy Lifter's Little Brother (The Tattle-Tale)

Munki Reports PHP - https://munkireport.github.io/munkireport-php/

Made by Arjen van Bochoven

Features:

- Quick overview of your Apple fleet with a dashboard
- Get reports on many features (hardware types, disk usage, etc)

The Collector

Autopkg - https://autopkg.github.io/autopkg/

Made by Greg Neagle

AutoPkg is a system for automatically preparing software for distribution to managed clients.

Recipes allow you to specify a series of simple actions which combined together can perform complex tasks, similar to Automator workflows or Unix pipes.

What do I use it for?

Autopkg is an automated downloader for applications using Recipes.

There are many different repositories out there on GitHub where you can acquire recipes.

There are multiple different types of Recipes that allow for different types of outputted files.

Recipe Types

Download - Downloads the application in the format provided by the site.

Install - Automatically installs the application on the machine with Autopkg.

Pkg - Creates a pkg file with the application.

Munki - Downloads the application and sets it up in a Munki repository.

JSS - Takes the Pkg Recipe and imports it into a Casper Server.

Plus More....

The Collector's Older Brother

Autopkgr - http://www.lindegroup.com/autopkgr/

James Barclay, Elliot Jordan, and Josh Senick originally created AutoPkgr in June 2014, and ongoing development is led by Eldon Ahrold, Elliot Jordan, and James Barclay of the Linde Group.

AutoPkgr allows you to easily install all the components that AutoPkg requires so you can start downloading recipes right away.

It also allows the easy locating and adding of recipes to Autopkg.

The Fixer

Recipe-Robot - https://github.com/homebysix/recipe-robot

Made by Elliot Jordan of The Linde Group.

Recipe Robot is the easiest way to create new AutoPkg recipes for simple Mac apps.

It is a GUI that works via drag and drop.

Earlier today, Shea Craig and Elliot Jordan presented “Writing better AutoPkg with the help of Recipe Robot”

What do I use it for?

Recipe-Robot is my first stop to create a recipe for Autopkg and a package for Munki when I am unable to locate a pre-made recipe on GitHub.

The Day-to-Day Worker

Launchd - https://opensource.apple.com/source/launchd/

Made by Dave Zarzycki at Apple;

Launchd is a unified, open-source service-management framework, starts, stops and manages daemons, applications, processes, and scripts in Apple OS X environments. (Wikipedia)

What do I use it for?

I use launchd to set up my workstations to:

- Download packages from the Munki repository at a set time.
- Install packages from the Munki repository at a set time.
- Restart the machine, after the install.

I created a Munki package for each of these tasks and in my imaging procedure install all three tasks on my lab workstations.

The Day-to-Day Worker's Older Brother

LingonX - https://www.peterborgapps.com/lingon/

Made by Peter Borg

Lingon can start an app, a script or run a command automatically whenever you want it to. You can schedule it to run at a specific time, regularly or when something special happens.

It's essentially a GUI for Launchd.

How do they work for me?

The Collector & The Collector's Older Brother

Autopkg & Autopgkr

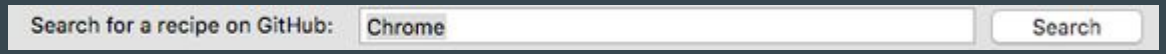
Autopkg is command line only, so it does all of the work, but Autopkgr has the GUI which allows all of the settings to be easily configured and adjusted.

Because GUI.

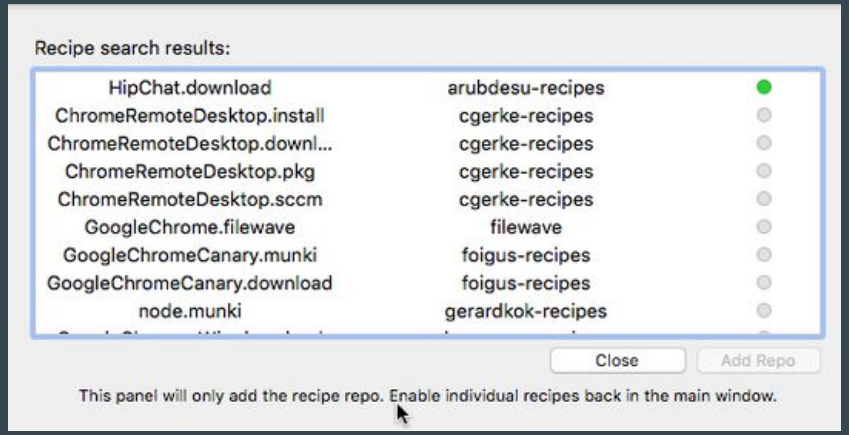
I currently use 23-24 Recipes. (Ugh... Eclipse)

Acquiring Recipes

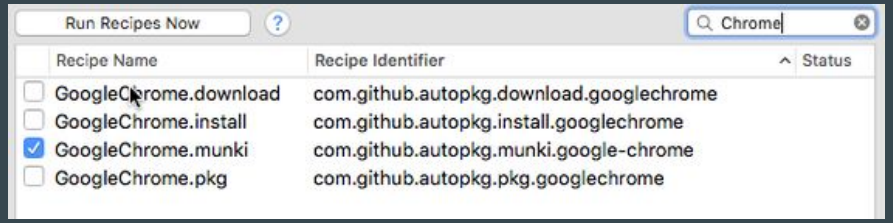
Easy as 1...

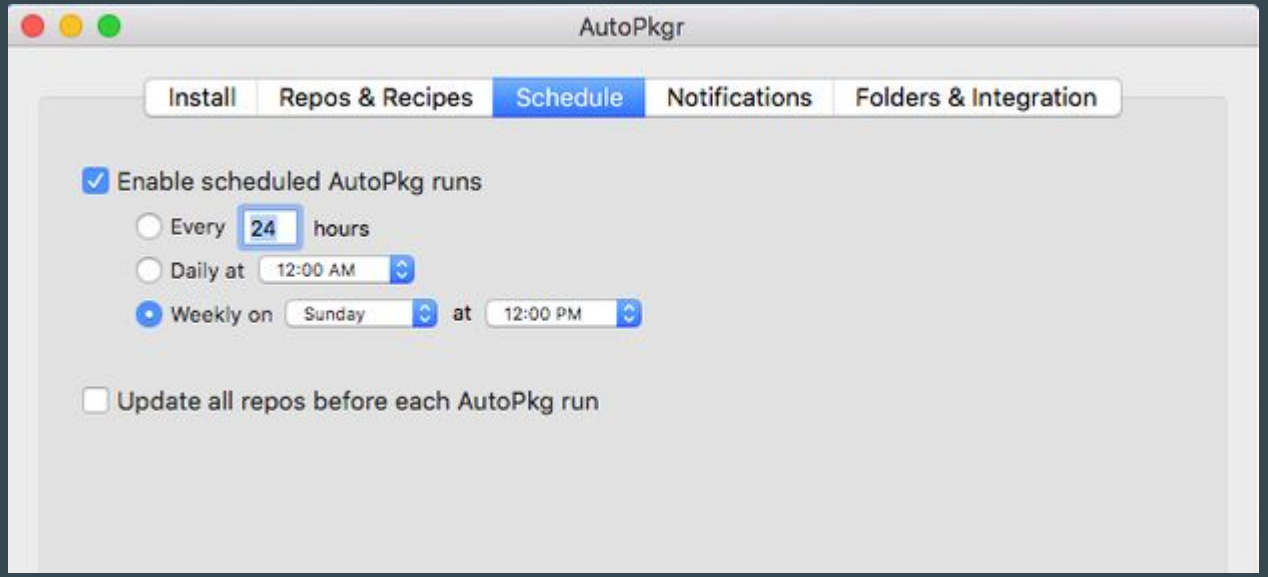


2...

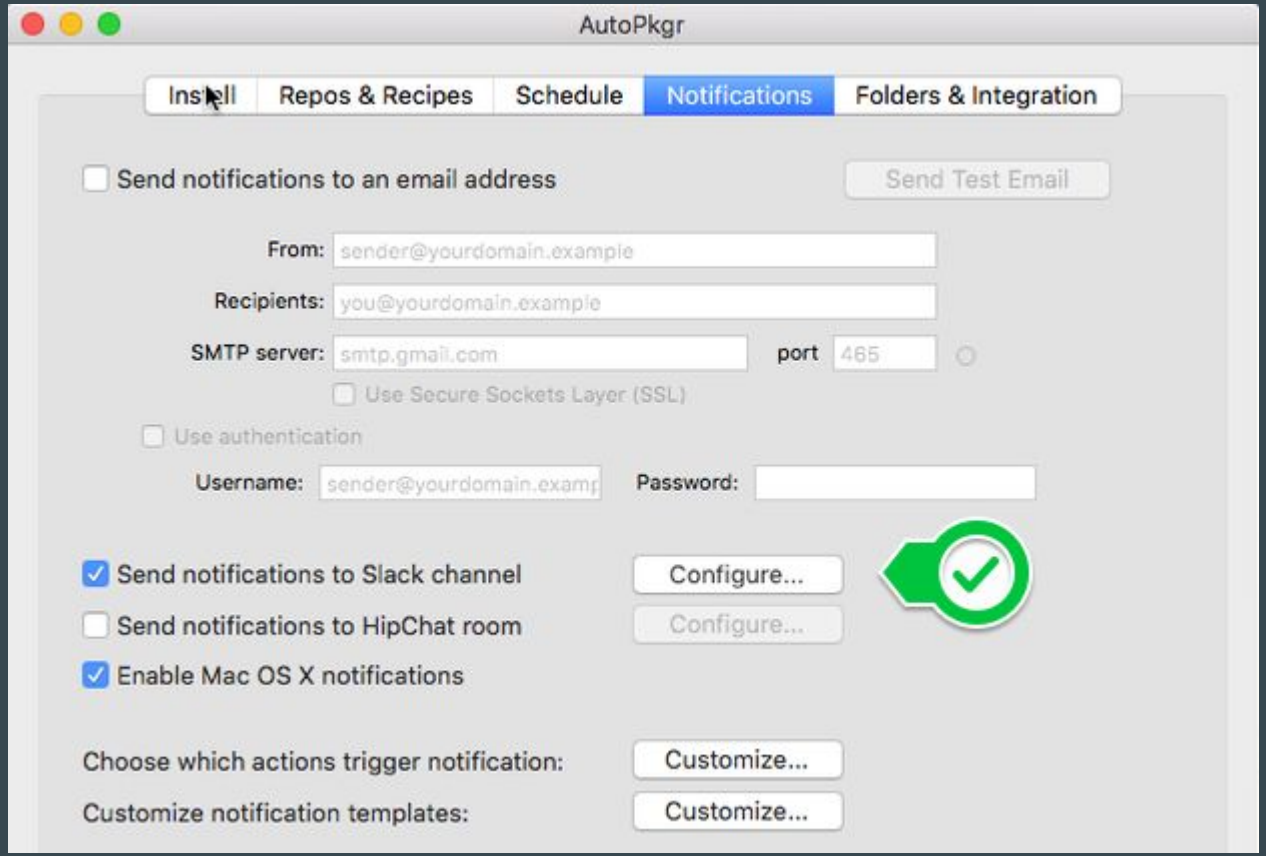


3...



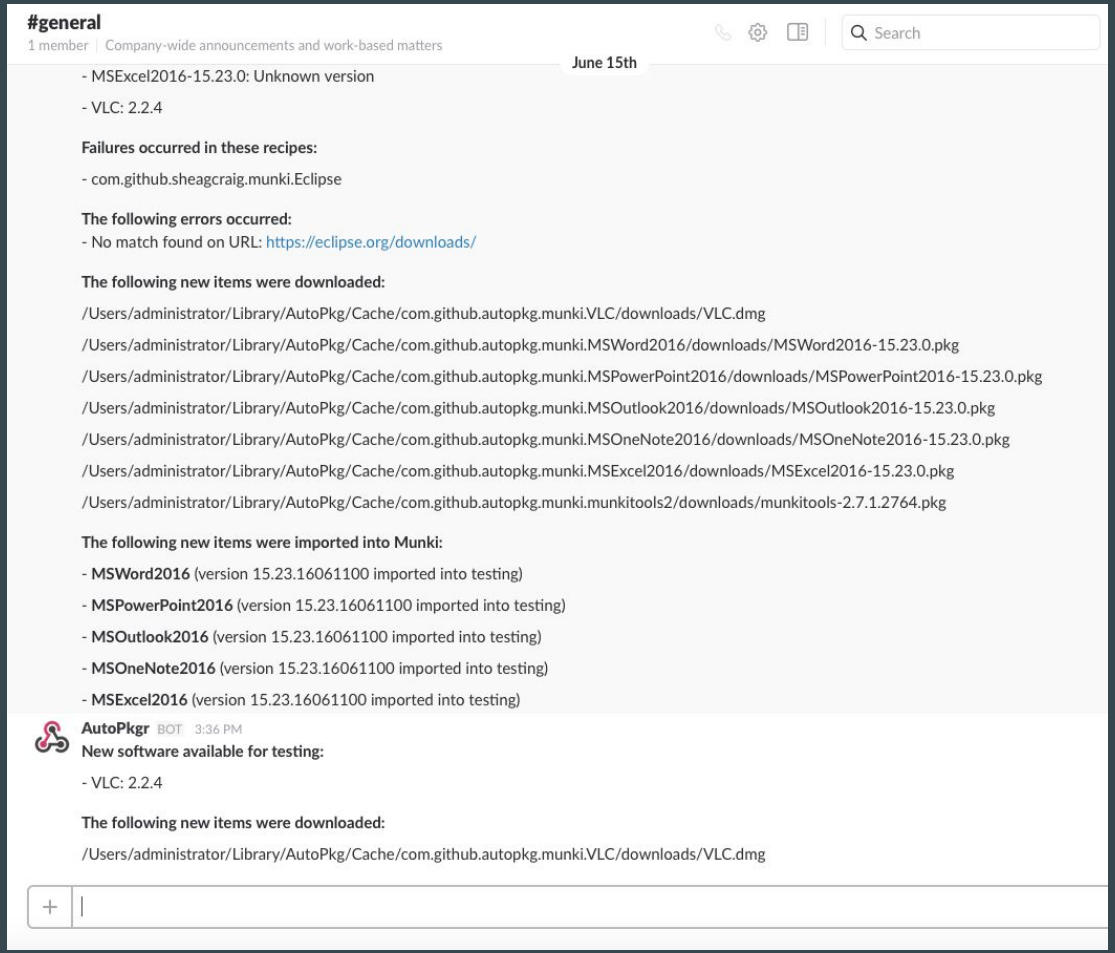


The Schedule Page



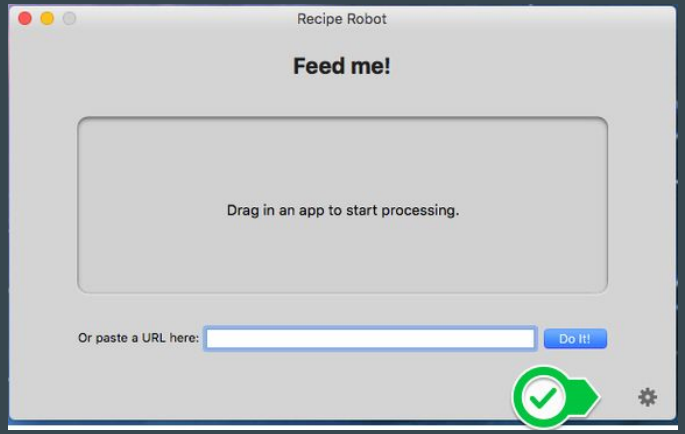
The Notifications Page

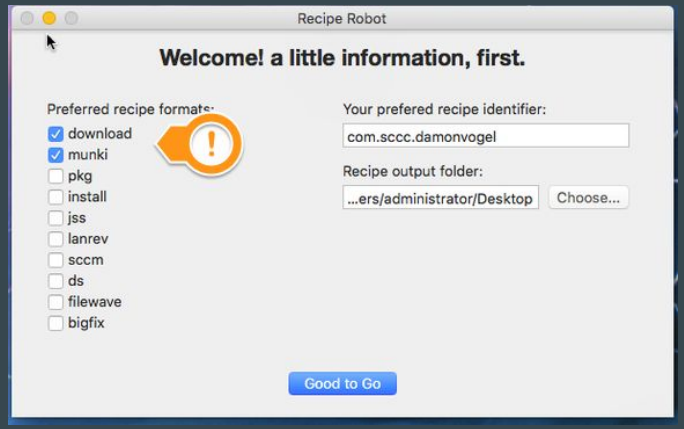
Slack Notifications



The Fixer

Recipe-Robot: The Recipe Robot app currently works best with apps that have a Sparkle feed, and for which there are no existing AutoPkg recipes. (From the RR Site)

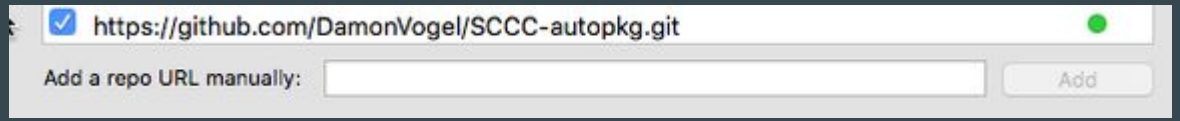




Once you've created recipes you can add them two ways:

1) Locally in the Overrides directory in Autopkg:

2) Remotely in a GitHub Repository



The Heavy Lifter & The Heavy Lifter's Older Brother

Munki & Munki Admin

Munki is command line only, so I add Munki Admin as a GUI, because I get a better idea of how Munki is set up.

And because GUI.

Munki Stats

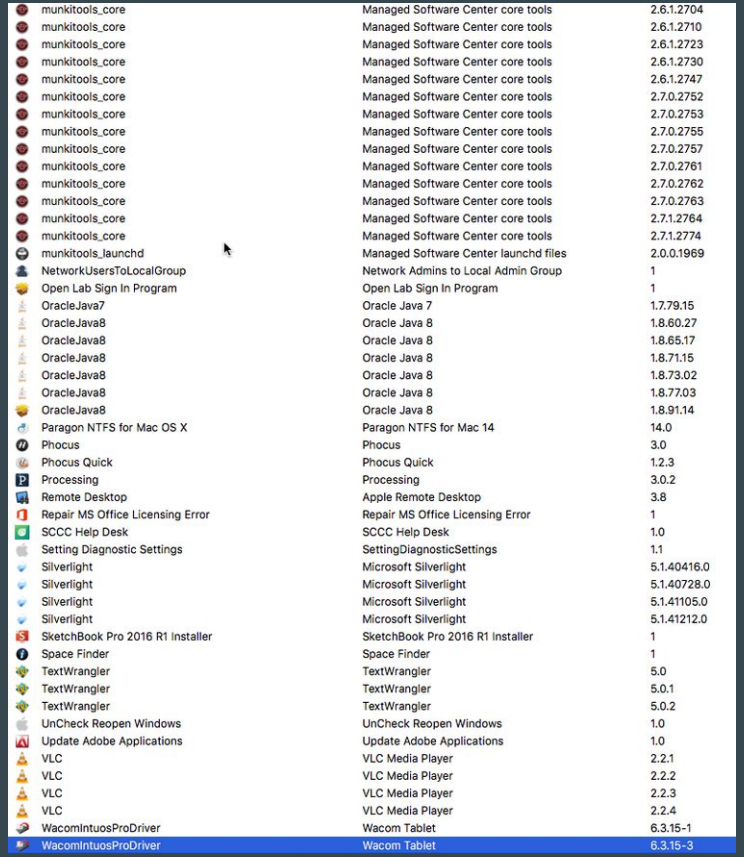
My current Munki Repository has:

- 248 Packages (Applications)
- 3 Catalogs (Development, Production, & Testing)
- 10 Manifests

248 Packages

248 packages = 71 Individual Applications

- Applications with Recipes display each version.
  - munkitools_core
  - OracleJava8
  - Silverlight
- Single Versions denote hand-made packages
  - SketchBook Pro 2016 R1 Installer
  - Paragon NTFS for Mac OS X
- Or Custom Scripts
  - Update Adobe Applications
  - Repair MS Office Licensing Error



munkitools_core	Managed Software Center core tools	2.6.1.2704
munkitools_core	Managed Software Center core tools	2.6.1.2710
munkitools_core	Managed Software Center core tools	2.6.1.2723
munkitools_core	Managed Software Center core tools	2.6.1.2730
munkitools_core	Managed Software Center core tools	2.6.1.2747
munkitools_core	Managed Software Center core tools	2.7.0.2752
munkitools_core	Managed Software Center core tools	2.7.0.2753
munkitools_core	Managed Software Center core tools	2.7.0.2755
munkitools_core	Managed Software Center core tools	2.7.0.2757
munkitools_core	Managed Software Center core tools	2.7.0.2761
munkitools_core	Managed Software Center core tools	2.7.0.2762
munkitools_core	Managed Software Center core tools	2.7.0.2763
munkitools_core	Managed Software Center core tools	2.7.1.2764
munkitools_core	Managed Software Center core tools	2.7.1.2774
munkitools_launchd	Managed Software Center launchd files	2.0.0.1969
NetworkUsersToLocalGroup	Network Admins to Local Admin Group	1
Open Lab Sign In Program	Open Lab Sign In Program	1
OracleJava7	Oracle Java 7	1.7.79.15
OracleJava8	Oracle Java 8	1.8.60.27
OracleJava8	Oracle Java 8	1.8.65.17
OracleJava8	Oracle Java 8	1.8.71.15
OracleJava8	Oracle Java 8	1.8.73.02
OracleJava8	Oracle Java 8	1.8.77.03
OracleJava8	Oracle Java 8	1.8.91.14
Paragon NTFS for Mac OS X	Paragon NTFS for Mac 14	14.0
Phocus	Phocus	3.0
Phocus Quick	Phocus Quick	1.2.3
Processing	Processing	3.0.2
Remote Desktop	Apple Remote Desktop	3.8
Repair MS Office Licensing Error	Repair MS Office Licensing Error	1
SCCC Help Desk	SCCC Help Desk	1.0
Setting Diagnostic Settings	SettingDiagnosticSettings	1.1
Silverlight	Microsoft Silverlight	5.1.40416.0
Silverlight	Microsoft Silverlight	5.1.40728.0
Silverlight	Microsoft Silverlight	5.1.41105.0
Silverlight	Microsoft Silverlight	5.1.41212.0
SketchBook Pro 2016 R1 Installer	SketchBook Pro 2016 R1 Installer	1
Space Finder	Space Finder	1
TextWrangler	TextWrangler	5.0
TextWrangler	TextWrangler	5.0.1
TextWrangler	TextWrangler	5.0.2
UnCheck Reopen Windows	UnCheck Reopen Windows	1.0
Update Adobe Applications	Update Adobe Applications	1.0
VLC	VLC Media Player	2.2.1
VLC	VLC Media Player	2.2.2
VLC	VLC Media Player	2.2.3
VLC	VLC Media Player	2.2.4
WacomIntuosProDriver	Wacom Tablet	6.3.15-1
WacomIntuosProDriver	Wacom Tablet	6.3.15-3

Munki Manifests

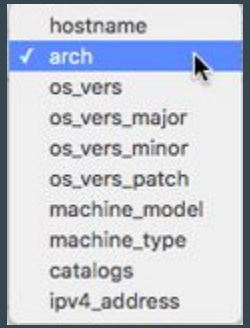
- 1 Faculty Manifest
- 1 Admin Manifest
- 1 Testing Manifest
- 1 Base Manifest with 6 Conditional manifests.

Conditions are perfect. It's a beautiful day.

I use Munki's Conditions to determine which computers various Manifests will apply.

You can use:

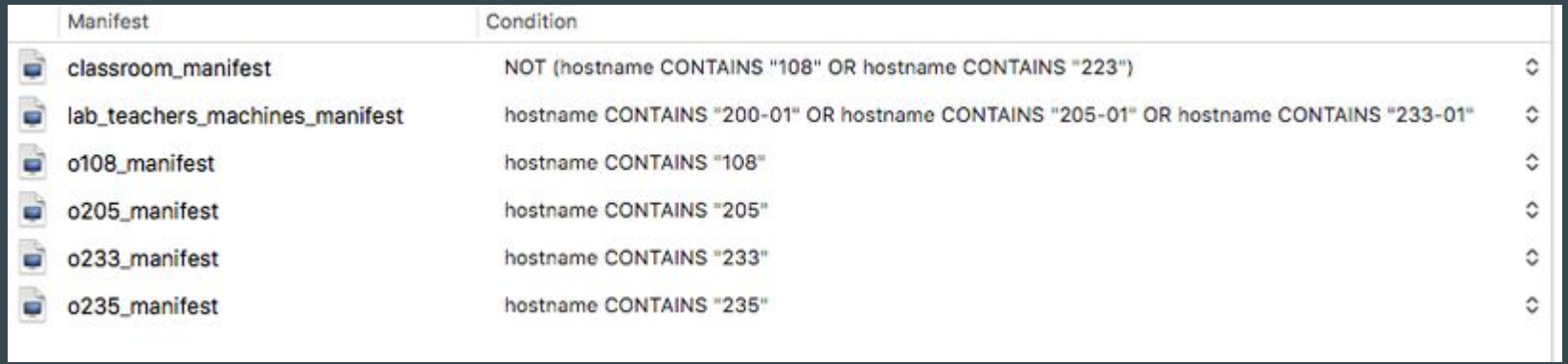
- Hostname
- Architecture
- OS Version (Name)
- OS Version (Number)
- Machine Model
- Machine Type
- Catalogs
- IP v4 Address



What condition am I in?

As you can see, I prefer hostnames.

As a college standard, the room number is a part of our hostnames.

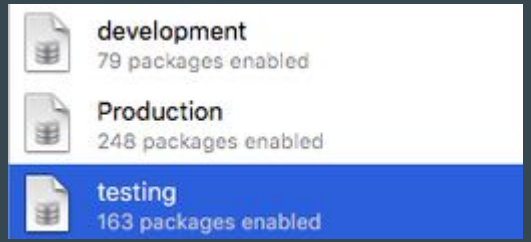


Manifest	Condition
classroom_manifest	NOT (hostname CONTAINS "108" OR hostname CONTAINS "223")
lab_teachers_machines_manifest	hostname CONTAINS "200-01" OR hostname CONTAINS "205-01" OR hostname CONTAINS "233-01"
o108_manifest	hostname CONTAINS "108"
o205_manifest	hostname CONTAINS "205"
o233_manifest	hostname CONTAINS "233"
o235_manifest	hostname CONTAINS "235"

For Safety's Sake!

When Autopkg downloads an application update and adds it to Munki, it automatically adds the application to the Testing catalog. This will push the application to my testing machines.

Once I have confirmed the application will not cause a problem, I add the application to the Production catalog. This will push the application to all machines on my schedule.



Slow It Down

Munki Client Settings - To prevent Munki from automatically installing any applications to my clients, I have these settings in the ManagedInstalls.plist that is copied onto clients during the imaging process.

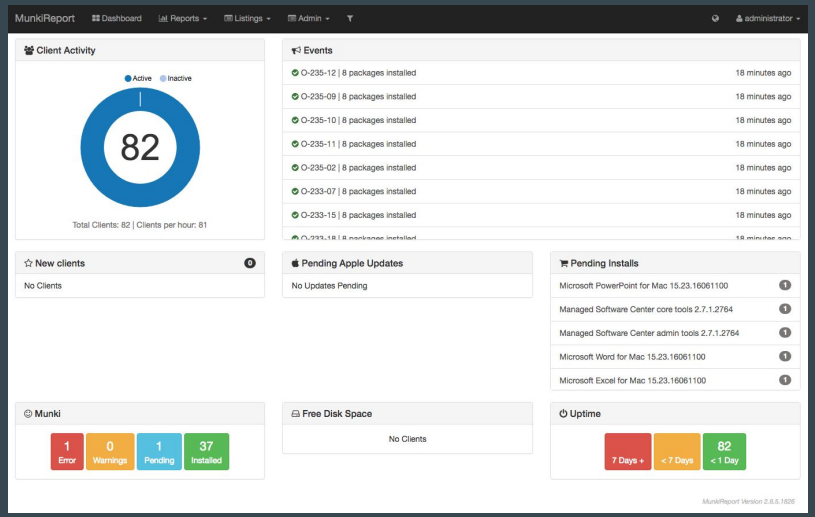
- SuppressAutoInstall - True
- SuppressLoginwindowInstall - True
- SuppressUserNotification - True (False on Testing Machines)

The Heavy Lifter's Little Brother (The Tattle-Tale)

Munki-Report-PHP

As of right now, I'm a little late to the party with Munki-Report-Php.

I have a very basic install set up, and I'm starting to work with it as I have time.



The Day-to-Day Worker's & The Worker's Older Brother

Launchd & LingonX

Writing plist files by hand is my kryptonite!

I can throw a typo in a plist with three lines.

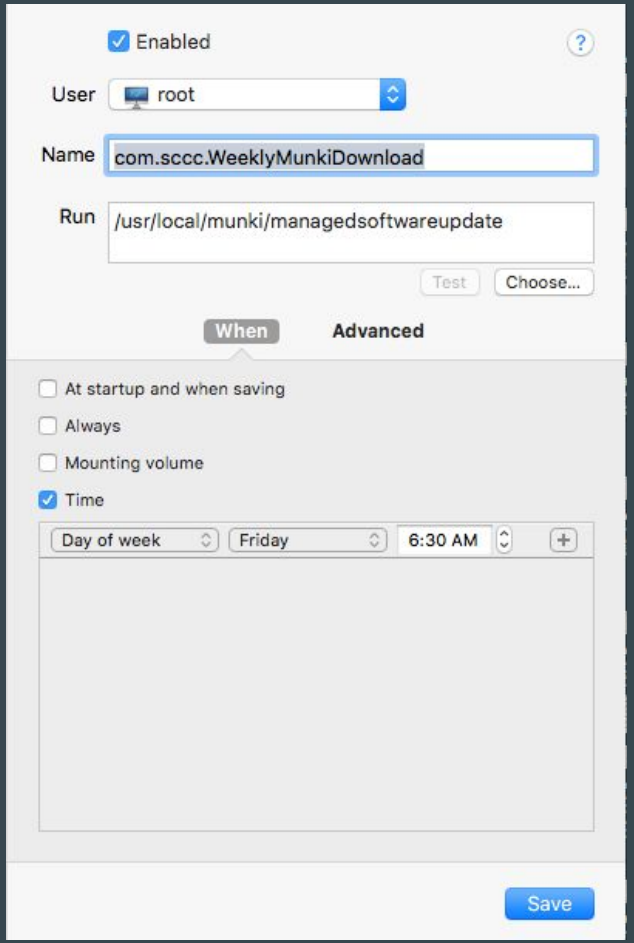
LingonX cost me $10 and it is well worth it.

Created plists

I used LingonX to create three plists:

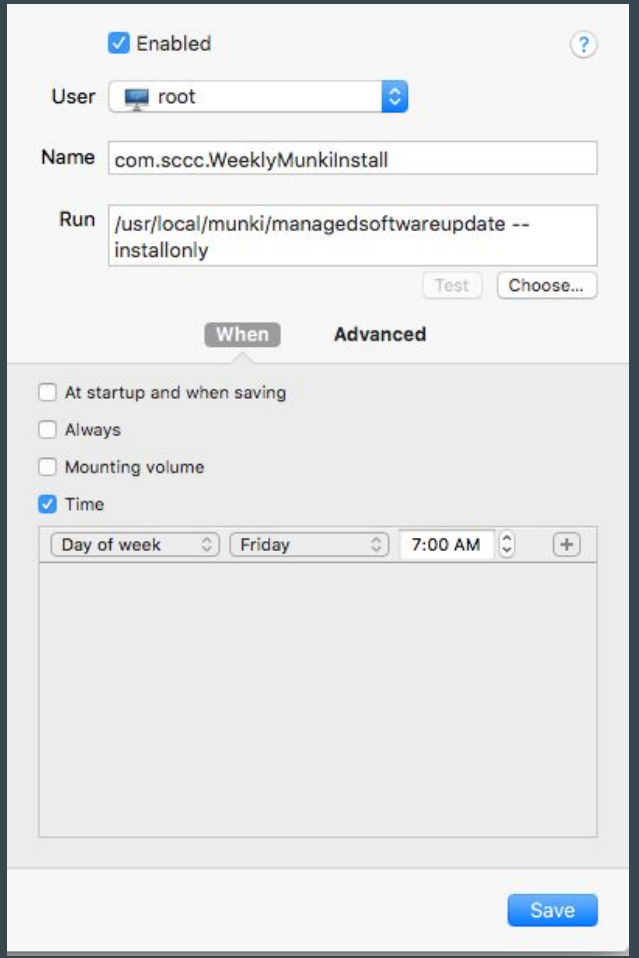
- WeeklyMunkiDownload
- WeeklyMunkyInstall
- WeeklyUpdateRestart

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>Disabled</key>
    <false/>
    <key>EnvironmentVariables</key>
    <dict>
        <key>PATH</key>

<string>/usr/local/bin:/usr/bin:/bin:/usr/sbin:/sbin:/opt/X11/bin:/usr/local/git/bin:/usr/
local/MacGPG2/bin:/usr/local/munki:/usr/local/sbin</string>
    </dict>
    <key>Label</key>
    <string>com.sccc.WeeklyMunkiDownload</string>
    <key>ProgramArguments</key>
    <array>
        <string>/usr/local/munki/managedsoftwareupdate</string>
    </array>
    <key>RunAtLoad</key>
    <false/>
    <key>StandardErrorPath</key>
    <string>/Library/Logs/WeeklyMunkiDownloadError.log</string>
    <key>StandardOutPath</key>
    <string>/Library/Logs/WeeklyMunkiDownload.log</string>
    <key>StartCalendarInterval</key>
    <array>
        <dict>
            <key>Hour</key>
            <integer>6</integer>
            <key>Minute</key>
            <integer>30</integer>
            <key>Weekday</key>
            <integer>5</integer>
        </dict>
    </array>
</dict>
</plist>
```



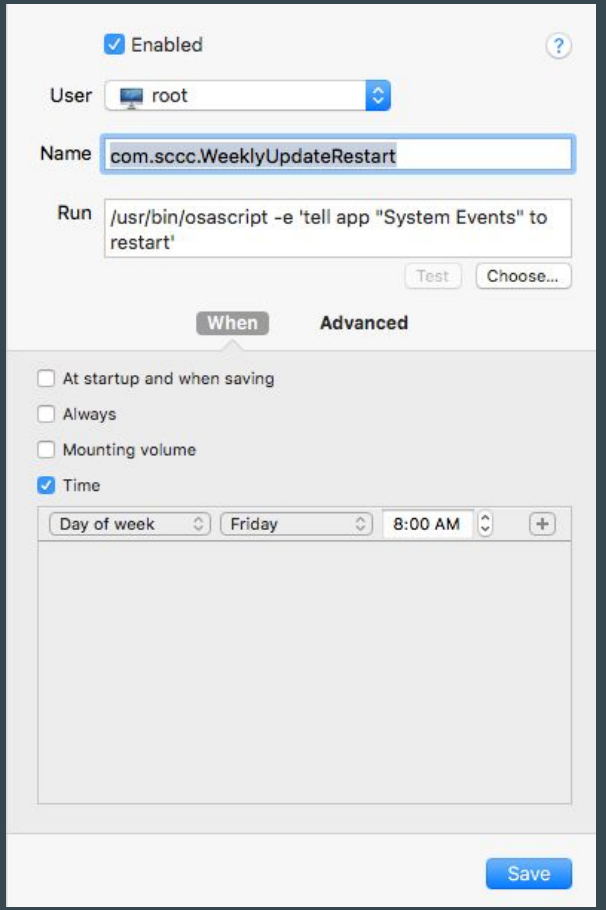
WeeklyMunkiDownload - Fridays @ 6:30am

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>Disabled</key>
    <false/>
    <key>EnvironmentVariables</key>
    <dict>
        <key>PATH</key>

<string>/usr/local/bin:/usr/bin:/bin:/usr/sbin:/sbin:/opt/X11/bin:/usr/local/git/bin:/usr/
local/MacGPG2/bin:/usr/local/munki:/usr/local/sbin</string>
    </dict>
    <key>Label</key>
    <string>com.sccc.WeeklyMunkiInstall</string>
    <key>ProgramArguments</key>
    <array>
        <string>/usr/local/munki/managedsoftwareupdate</string>
        <string>--installonly</string>
    </array>
    <key>RunAtLoad</key>
    <false/>
    <key>StandardErrorPath</key>
    <string>/Library/Logs/WeeklyMunkiInstallError.log</string>
    <key>StandardOutPath</key>
    <string>/Library/Logs/WeeklyMunkiInstall.log</string>
    <key>StartCalendarInterval</key>
    <array>
        <dict>
            <key>Hour</key>
            <integer>7</integer>
            <key>Minute</key>
            <integer>0</integer>
            <key>Weekday</key>
            <integer>5</integer>
        </dict>
    </array>
</dict>
</plist>
```



WeeklyMunkiInstall - Fridays @ 7:00am

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>Disabled</key>
    <false/>
    <key>EnvironmentVariables</key>
    <dict>
        <key>PATH</key>

<string>/usr/local/bin:/usr/bin:/bin:/usr/sbin:/sbin:/opt/X11/bin:/usr/local/git/bin:/usr/
local/MacGPG2/bin:/usr/local/munki:/usr/local/sbin</string>
    </dict>
    <key>Label</key>
    <string>com.sccc.WeeklyUpdateRestart</string>
    <key>ProgramArguments</key>
    <array>
        <string>/usr/bin/osascript</string>
        <string>-e</string>
        <string>tell app "System Events" to restart</string>
    </array>
    <key>RunAtLoad</key>
    <false/>
    <key>StartCalendarInterval</key>
    <array>
        <dict>
            <key>Hour</key>
            <integer>8</integer>
            <key>Minute</key>
            <integer>0</integer>
            <key>Weekday</key>
            <integer>5</integer>
        </dict>
    </array>
</dict>
</plist>
```



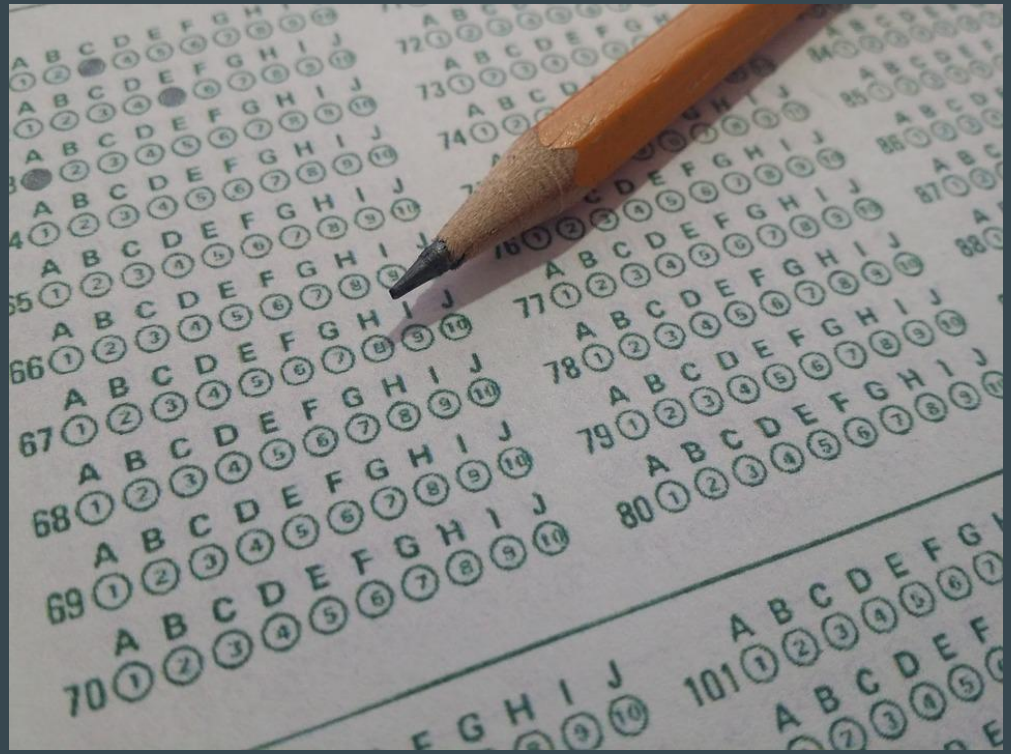
WeeklyUpdateRestart - Fridays @ 8:00am

The Regular Workflow, How It Goes..

1) I get a Slack message. The software has been downloaded and added to the Testing Catalog.

2) I am prompted on my Testing Machine to install the new software. I do that with:

a. Locally with the Managed Software Center (Munki Client Gui)
b. Or Remote Desktop via UNIX commands.



3) Testing.... Testing.... 1.... 2.... 3.....

4) When the software has been tested thoroughly, I go to MunkiAdmin and add the software to the Production Catalog.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>Disabled</key>
    <false/>
    <key>EnvironmentVariables</key>
    <dict>
        <key>PATH</key>

<string>/usr/local/bin:/usr/bin:/bin:/usr/sbin:/sbin:/opt/X11/bin:/usr/local/git/bin:/usr/
local/MacGPG2/bin:/usr/local/munki:/usr/local/sbin</string>
    </dict>
    <key>Label</key>
    <string>com.sccc.WeeklyUpdateRestart</string>
    <key>ProgramArguments</key>
    <array>
        <string>/usr/bin/osascript</string>
        <string>-e</string>
        <string>tell app "System Events" to restart</string>
    </array>
    <key>RunAtLoad</key>
    <false/>
    <key>StartCalendarInterval</key>
    <array>
        <dict>
            <key>Hour</key>
            <integer>8</integer>
            <key>Minute</key>
            <integer>0</integer>
            <key>Weekday</key>
            <integer>5</integer>
        </dict>
    </array>
</dict>
</plist>
```

5) Launchd runs the UNIX commands to automatically install the software to all of the workstations.

6) Move on to something else.

As Needed...

There are many different ways to add packages to Munki.

If I am unable to add an application to Munki via Autopkgr, I can:

1. Use Recipe-Robot to attempt to create a recipe.

2. Create a static installation of the application.

There are many different methods to create static installs:

- Casper Composer
- Munki's munkiimport

A website that I have found to be an amazing resource, is https://derflounder.wordpress.com/ run by our very own Richard Trouton.

Links

- Autopkg - https://autopkg.github.io/autopkg/
- Autopkgr - http://www.lindegroup.com/autopkgr/
- Munki - https://www.munki.org/munki/
- MunkiAdmin - https://github.com/hjuutilainen/munkiadmin
- Munki Reports PHP - https://munkireport.github.io/munkireport-php/
- Launchd - https://opensource.apple.com/source/launchd/
- LingonX - https://www.peterborgapps.com/lingon/

Thanks for listening. What did you think?

Feedback - https://bit.ly/psumac2016-68