Mac Management with Munki
Managed Software Installation and Configuration for Mac OS X

Patrick McNeal • J-PAL • mcneal@mit.edu
The Challenge
The Solution

munki
The Solution

- Manages software installs (and uninstalls) on Mac OS X clients
- User can control when updates are applied
- Doesn’t require administrator access
- Integrates with Apple's Software Update
- Budget friendly
The Solution....does not
So really, what does it do?

- Install, update and remove software software
- Works with user-installed software
- Allow non-admins to apply Apple Software Updates
- Anything else that can be scripted (printer setup, security settings, local user accounts, etc.)
- Collect software inventory
What do I need?

- An HTTP server
- A machine for adding software and testing
- Munki tools
Okay. Show me.
Demo of installing required and optional software.
How?
First, some terms

- **Catalog** - “A list of items available on the server.”

- **Installer Item** – “Piece of software to be installed. May be be either a flat Apple package (.pkg), or a disk image (.dmg).”

- **Info File (or PkgInfo File)** – “A plist file that describes an installer item.”

- **Manifest** – “Describes what software a client should have installed or uninstalled.”

- **Repo** – Collection of catalogs, manifests, info files, installer items on a Munki server
How?

1. Client asks server for what software should be installed or removed (manifest & catalog)
2. Client compares what manifest says it should have with what is actually installed (catalog)
3. Client downloads and caches software from server (installer item)
4. Install and remove necessary software, prompting user if they are currently logged in.
Munki tools

- munkiimport – imports pkgs, disk images and apps into Munki repo, and created related info file
- makepkginfo – creates a well crafted info file, or just specific parts
- makecatalogs – builds catalogs from info files
- manifestutil – modifies manifests through interactive shell
- managedsoftwareupdate – main client tool
Adding Software

1. Use munkiimport to import software
   - Supports Apple Packages, application bundles (.app) and files on a disk image

2. Review info file to make sure it looks correct
   - Add “installs” key

3. Add package to manifest using manifestutil

4. Rebuild catalogs
It can’t be that easy

Show me
Determining what to do

- Munki determines what manifest to use by

  1. Looking at the ClientIdentifier key in /Library/Preferences/ManagedInstalls.plist
  2. Fully-qualified hostname
  3. Unqualified hostname
  4. Hard-coded value ‘site-default’
Determining what to do

- Manifests tell Munki what to install and what to remove and are broken into for sections:
  - managed_installs – software to install
  - managed_updates – software that Munki doesn’t install but should be updated if found
  - managed_uninstalls – software to be removed
  - optional_installs – software that the user can choose to install or remove at any time.
Determining what’s installed

1. Install check script

```bash
#!/bin/sh

# Grab current version of installed python module
version="$(python -c 'import argparse;print argparse.__version__; 2>/dev/null')"

# Compare with the version we want to install
if [ $(version) \< 1.2.1 ]; then
  exit 0
else
  exit 1
fi
```
Determining what’s installed

1. Install check script
2. Installs Item

```xml
<key>installs</key>
<array>
  <dict>
    <key>CFBundleIdentifier</key>
    <string>org.mozilla.firefox</string>
    <key>CFBundleName</key>
    <string>Firefox</string>
    <key>CFBundleShortVersionString</key>
    <string>6.0</string>
    <key>minosversion</key>
    <string>10.5</string>
    <key>path</key>
    <string>/Applications/Firefox.app</string>
    <key>type</key>
    <string>application</string>
  </dict>
</array>
```
Determining what’s installed

1. Install check script
2. Installs Item
3. Receipts

```xml
<dict>
  <key>filename</key>
  <string>AvidCodecsLE.pkg</string>
  <key>installed_size</key>
  <integer>1188</integer>
  <key>name</key>
  <string>AvidCodecsLE</string>
  <key>packageid</key>
  <string>com.avid.avidcodecsle</string>
  <key>version</key>
  <string>2.3.4</string>
</dict>
```
Updating Software

- Info files can include the “update for” key
- Specifies that this package should not be installed on its own, but as an update for another package
- Don’t have to be listed in manifest – Munki will just find them
- To install multiple updates in order, use the “requires” key
Updating Software

<key>name</key>
<string>Office2011_sp1</string>
<key>update_for</key>
<array>
  <string>Office2011</string>
</array>
<key>requires</key>
<array>
  <string>Office2011-14.0.0</string>
</array>
MunkiWebAdmin

- Supplemental web application
- Incorporates the functionality of the MunkiReport project
- Provides the ability to browse catalogs, and browse and edit manifests of an existing Munki repo
Show me what it can do

Deploying Sophos
Tips

• Use virtualization for testing

• Install package on clean machine to see how it behaves. Does it require reboot? How is it removed?

• Run Managed Software Update twice to make sure it “sees” that software was installed

• Test software removal before you need it

• Don’t use ‘-’ in package name, it can confuse Munki

• Put your repo under version control
Did it work?
Links

- https://code.google.com/p/munki for software, documentation, examples and mailing lists

- http://managingosx.wordpress.com for Greg Neagle’s blog
Q & A
mcneal@mit.edu