Managed Software Installation with Munki

Jon Rhoades
St Vincent's Institute & University of Melbourne

jrhoades@svi.edu.au

Managed Installation

- Why?
- What are we using now?
- Needs
  - Installs
  - Updates
  - Apple Updates

Munki

- Collection of Python scripts and Client GUI
- Apache 2 License
- Created by Greg Neagle
  - Disney Animation Studios
  - Enterprise Mac Managed Preferences (Apress 2010)
  - Mac Enterprise mailing list
- http://code.google.com/p/munki/

Requirements

- Any webserver or AFP share (not recommended)
- OS X (standard or server) for creation of packages
- Client to be installed on each Mac
- Works with 10.4/10.5/10.6

Installation

- Run munkitools-xxx.mpkg
- Have a look at the customize option, but don't deselect any
- Install
- Restart - Boo!
  - `defaults write com.apple.finder AppleShowAllFiles TRUE`
  - `killall Finder`
  - Copy .profile (adds /usr/local/munki to your path)

Setup

- Enable “Sharing->Web Sharing”
- Create or Copy the following folder structure in ~/Sites:
  - repo
    - catalogs
    - manifests
    - pkgs
    - pkgsinfo

Configuring munkiimport

- munkiimport --configure
  - Path to Repo: */Users/username/Sites/repo*
  - Repo Fileshare URL: *http://localhost/~username*
  - pkginfo extentsion: *plist*
  - pkginfo editor: Text Wrangler.app (if you like)

Import a “Drag to /Applications”

- munkiimport Firefox 3.6.17.dmg
  - Item name: *Firefox*
  - Display name: *Firefox 3.6*
  - Description: *Whatever*
  - Version: *Accept default*
  - Catalogs: *testing, production*
  - Review & Import: *y*
  - Upload Item to: *browsers & create it*
  - Rebuild Catalogs: *y*

pkginfo plist

- ~/Sites/repo/pkgsinfo/browsers/Firefox-3.6.17.plist
  - name - Identifies the package
  - display_name - “pretty” name for the updater GUI
  - CFBundleShortVersionString - version number
  - catalogs - to come
  - installer type
    - copy_from_dmg
    - package
    - adobe package

Import an updated version

- munkiimport Firefox 4.0.1.dmg
  - Use existing item as template: *y*
    - Accept defaults except for:
      - Display name: *Firefox 4*
      - Catalogs: *testing*

Add a package

- Mount the munkitools dmg - note it creates the dmg for us
- munkiimport munkitools-xxx.mpkg
- Upload to subdirectory path: *utils*
  - Catalogs: *testing*
- Have a peek at ~/Sites/repo/pkgsinfo/utils/munkitools-xxx.plist
  - receipts show the 4 sub packages
  - Require restart

“Bigger” Installers

- Office - many sub packages
- Adobe CS
  - Own installer type due to Adobe’s “unique” installer design
- http://code.google.com/p/munki/source/browse/ has many examples

Creating pkgs notes

- Try and import using Thuderbird 2
  - Munki uses *hdiutil*, it can't handle DMG licenses
- Sometimes may need to repackage using PackageMaker
- If Munki can't install it: munki-dev@googlegroups.com
  - Munki has special code to handle "obnoxious" installers

Advanced pkgsinfo options

- *blocking_applications* - don't install if an application is running.
- *requires* - useful for updates and apps not installed by munki. Also enforces install order eg don't install office update until office is installed!
- *update_for* - no need to add to a manifest, munki automatically installs it.
- *unistall_scripts* - embedded shell scripts
- *unattended_install* - installs immediately with no possibility of user interaction: no notification or option to cancel

Manifests

- See *basic.xml*
- Add to *managed_installs*
  - *firefox*
- Add to *optional_installs*
  - munkitools

Sample Manifests

- basic.xml

```
<key>included_manifests</key>
<array>
        <string>basic.xml</string>
</array>
<key>optional_installs</key>
<array>
        <string>munkitools</string>
</array>
```

Advanced Manifests

- *Catalogs* - can be a member of multiple catalogs
- *managed_uninstalls*
- *managed_updates* - for updating non munki installed software

Client Configuration

- /Library/ManagedInstalls.plist
  - plutil -convert xml1 /Library/ManagedInstalls.plist
  - ClientIdentifier*: basic.xml*
  - SoftwareRepoURL: *http://localhost/~username*
- Options
  - *InstallAppleSoftwareUpdates*
  - *AppleSoftwareUpdatesOnly*
  - *Logs*

Advanced Client Configuration

- *DaysBetweenNotifications*
- *Certificates*
- *AddtitonalHTTPHeaders*
- *SupressUserNotification* (can still be run manually)
- *SupressAutoInstall*
- *InstallrequiresLogout*
- *Change URLs & Directories*

Run the Client!

- /Applications/Utilities/Managed Software Update.app
- Check out (but don’t install) optional extras
- Run Update without logging out

Update an App

- Redo *munkiimport* Firefox 4
  - All the same, but add to productions as well as testing
  - Run managed updates again

What's missing?

- Scalable management of clients
- Reporting
- 'Friendly' admin interface
- Rollback

What's out there

- MunkiReport
  - http://code.google.com/p/munkireport/
- MunkiServer
  - Ruby on Rails
  - https://github.com/jraine/munkiserver
- Simian
  - Google App Engine 'Appliance'
  - http://code.google.com/p/simian/

Munki & 10.7

- Installations/Updates working
- Apple updates still to do
- Should be done by release of 10.7

But it's a web server…

- Manifest are just xml files
- Use PHP/ASP/Python/Ruby/Node.JS to serve the file
  - Most basic, read a static file according to the url
  - Better, create the manifest on the fly according to business logic/databases etc
- Quick example:
  - Uses php to serve the manifest, using a list of machines

php Example

- Need to use mod_rewrite as munki doesn't like Get query strings (eg manifest.php?id=123)
- .htacess:
  - `RewriteEngine on`
  - `RewriteRule ^id/([^/\.]+)/?$ mannifest.php?id=$1 [L]`
- Will rewrite /id/1234 to manifest.php?id=1

php Example

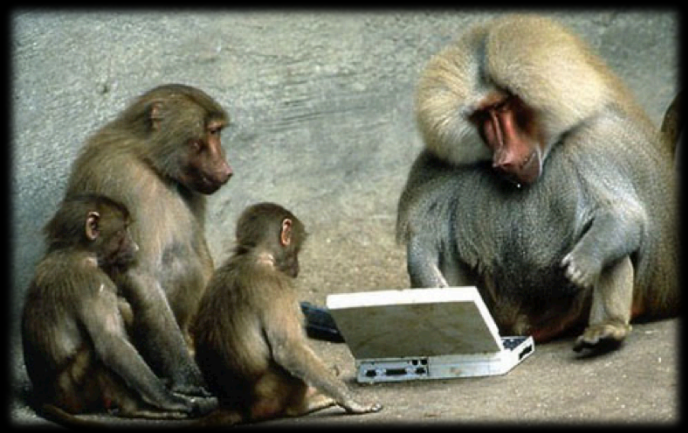
```
<?php
//Store the groups in an array for this example, in real life you would use config
//files or better still a database.
$h2g = array();
$h2g['12345'] = "standard.xml"
$h2g['12346'] = "standard.xml";
$h2g['12347'] = "advanced.xml";

// if no id is provided or the provided id doesn't match
$default = "basic.xml";

// If set assign the id else assign it to false
if(isset($_GET['id']){
   $id = $_GET['id'];
}else{
   $id = false;
}
// If the $id is not false and it's in the array, then assign the
// value from $h2g to $manifest, else assign the default manifest
if($id && array_key_exists($id, $h2g)){
   $manifest = $h2g[$id];
}else{
   $manifest = $default;
}
// Read (and print) the contents for the relevant manifest
readfile($manifest);
```

Munki in action

- Our setup at St Vincent's
  - Munki manifests and binaries served by nginx on Ubuntu
  - User accounts Open LDAP
  - Open Directory in "Golden Triangle" on Mac Mini "server"
  - 3 catalogs
    - Dev - Development & testing machine
    - Testing - IT Team
    - Production - Requires CAB approval
  - Group & individual manifests



Apple Updates

- What do we do now?
  - softwareupdate -l -a
  - COMMAND_LINE_INSTALL=1 export COMMAND_LINE_INSTALL; softwareupdate -l -a
  - Download every update and deploy…
- How do we deal with testing, groups etc?

Munki & Apple Updates

- ManagedInstalls.plist
  - SoftwareRepoURL - specify a local update server
  - InstallAppleSoftwareUpdates - True or False
  - AppleSoftwareUpdatesOnly - Only install Apple software updates, don't install anything else
- Munki will install updates for us

One More Thing - Reposado

- Apple Software Updates without OS X Server
  - Uses any webserver
  - Python scripts run on OS X/Linux/Windows
- Usage
  - Caching
  - En mass approval
  - Branch approval eg testing/production etc
- https://github.com/wdas/reposado