USING GIT WITH, AND AUTOMATING MUNKI

Adam Reed

The Australian National University

Hashtag : **#xw13**

Please leave comments on this talk at **auc.edu.au/xworld/sessions**

Australian National University

Git

Powerful Version Control System

Version Control

What is it and why should you use it?

- Manage changes to any file (not just source code)
  - Text files (plists, Apache config, scripts, etc) and Binary files (images, etc)
- Track changes over time
  - Revert to previous version (NOT a backup system!)
- Self documenting
- Enables collaboration

What is Git?

- Distributed Version Control System
  - Can be entirely local, or have multiple remote (distributed) repositories
  - Most operations are local however
- Fast, flexible and efficient - doesn’t get in your way
- Easy to set up and use, and is well supported on OS X
- Extensive and simple documentation is available

Why use Git

Git isn't just for programmers!

- **Branching and merging**
  - Try ideas with ease
  - Keep or discard them as required
  - Frictionless switching between branches
- **Tracked changes**
  - Revert to known good version
  - See what has changed (and by whom) over time

- **Distributed**
  - Work locally, but collaborate with co-workers with ease
  - Collaboration workflow not defined or enforced, use distributed repositories any way your team likes
- **Staging**
  - Can commit whole or just parts of files
- **Automation**
  - Verify files before committing
  - Do actions after committing

Where to start

Just start playing with dummy files

- Download and install Git
  - http://git-scm.com
- Follow a tutorial, or read the book
  - http://try.github.io - Online Git tutorial
  - http://git-scm.com/book - Free book
- Try a GUI client
  - http://www.sourcetreeapp.com - SourceTree

Australian National University

Munki

Managed Software Installation for OS X

Munki

It’s awesome - you should use it!

“Munki is a set of tools that, used together with a webserver-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.”

http://code.google.com/p/munki/

https://groups.google.com/group/munki-dev

Munki

Quick overview

- Install or uninstall (most) software and Apple updates
  - End user doesn't require admin privileges
- Upgrade software
  - Whether Munki installed it or not
- Optional installs
- Handles dependencies (apps, hardware, OS, etc)
- Free - with a vibrant community providing support

Australian National University

Munki's Data

All Munki metadata is saved in plists

- **Catalogs**
  - Typically used to create a Dev, Test, Prod workflow
  - Clients will search configured catalogs (in provided order) to find a matching item
- **Manifests**
  - Lists what's to be installed, updated, removed, or optionally available

- **Manifests (cont.)**
  - Can include other manifests
  - Can optionally specify catalogs
- **PkgInfos**
  - Information about the “package” to install
    - Name, version, description, type, etc
    - http://code.google.com/p/munki/wiki/SupportedPkginfoKeys
- **Packages**
  - The actual installer (disk image of files, or flat pkg)

Australian National University

Munki + Git

Enhance your workflow with minimal effort

Munki + Git

How to make these two tools work together

- Munki's metadata is stored as plists
  - Perfect candidate for version control

    (plutil --convert xml1 /path/to/plist if binary plist)
- Version control
  - Manifests, Pkginfos and possibly Catalogs
    - Catalogs are automatically generated
    - I don't track them, but there is no reason you can't

- Packages are typically large binaries
  - Not really suited to Git
  - Typically large, and as git stores a copy locally, not really practical for sysadmin machines
- Just ignore them (from the git repo)
  - Most people use something like rsync
    - Valid, but a manual solution
  - I recommend automating the handling of the packages via Git hooks
    - Validate via “installer_item_hash”

Demo

Set up Munki Repo with Git

Demo

Basic Git with an existing Munki repo

- git init (via SourceTree)
- Set up ignored files and directories
- Import existing Munki contents
- Add a new package
- Change a package
- Modify a manifest

Automation with Git Hooks

Why use Git hooks?

- Automates mundane tasks
  - Checks and balances done automatically
- Catch errors before they impact clients
- Enforce corporate standards
  - Naming, commit messages, etc
- Add the ability to manage packages as part of the Git workflow

Demo

Git hooks

Demo

Enabling Git hooks

- Set up Git pre and post hooks
  - munki_repo/.git/hooks/pre-commit
  - munki_repo/.git/hooks/post-commit
- Make a change that's rejected by pre-commit
- Make a valid change that passes pre-commit, and is then actioned in post-commit

Australian National University

Munki Graph

Sneak peak at Munki Graph

Munki Graph

Visualisation tool for Munki relationships

- Creates directed graphs showing relationships between items
  - Manifests
    - included_manifests
  - Pkginfos
    - update_for **and** requires
- Show or hide detail, and filter on particular item

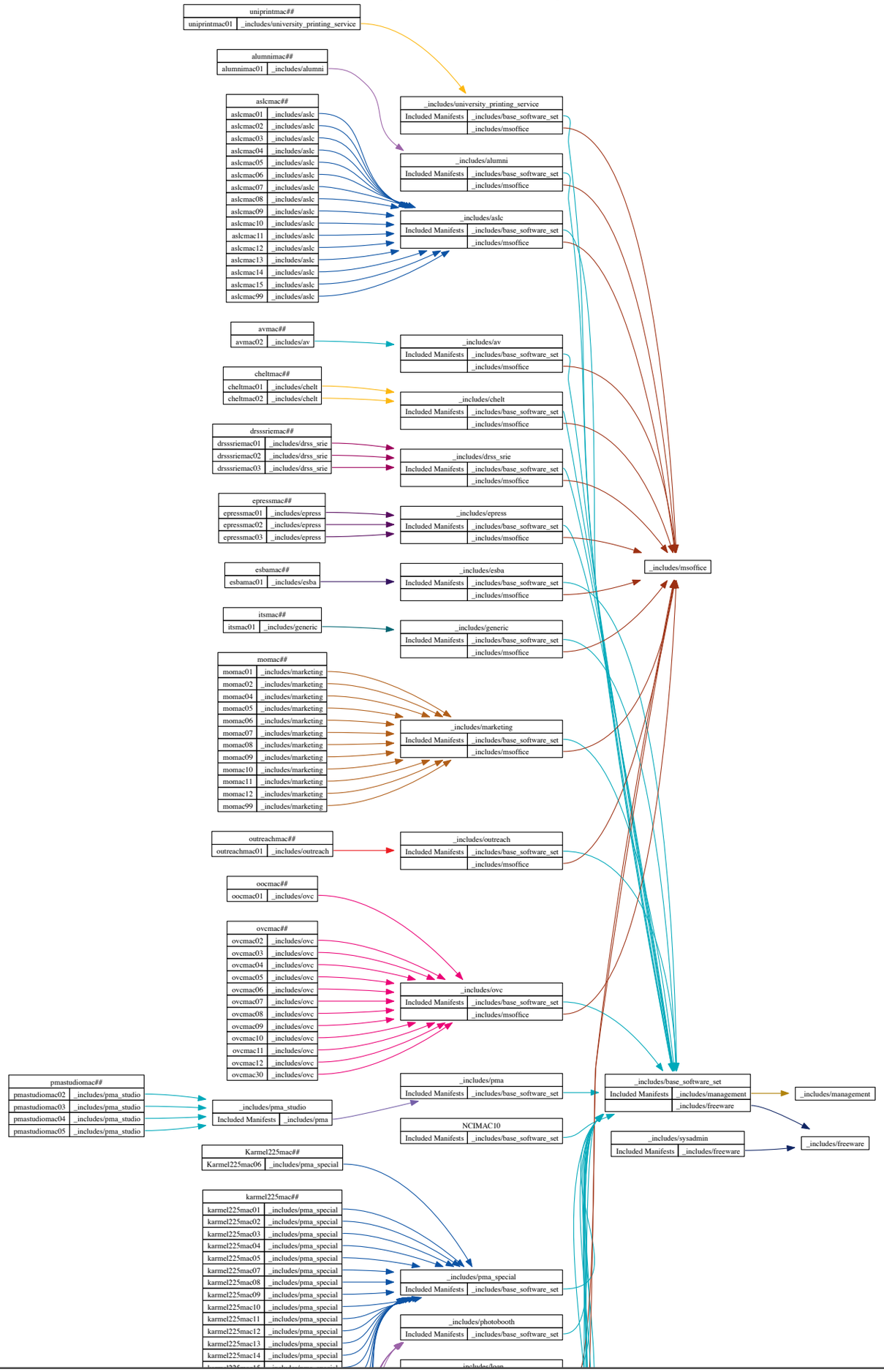
Munki Graph

- Multiple output formats
- With Git
  - Go back in time and compare relationships as they were X months ago
  - Work on a branch and see what the relationship would be before checking the changes in

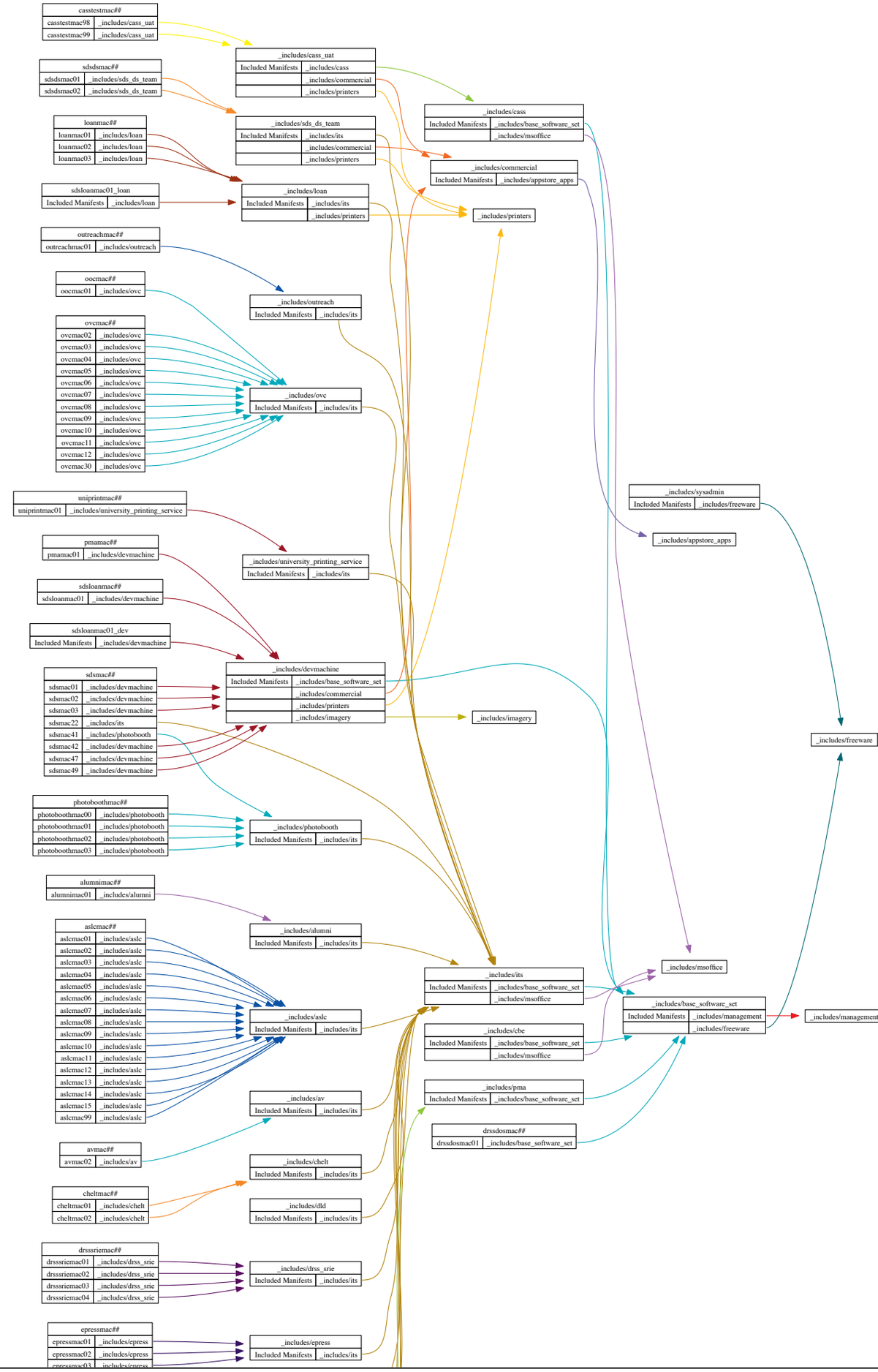
Coming soon to Github

Look for announcement on the Munki mailing list

Munki Manifest Dependencies as of 2013-04-12 at 9:44pm



Munki Manifest Dependencies as of 2013-04-13 at 10:28pm



Auto Munki Importer

Make the computer do the boring / time-consuming work for you

Australian National University

Automating Packaging

How to keep up to date the easy way

- Auto Munki Importer
  - Monitors user-defined websites for new versions
    - Or Sparkle framework RSS feeds
  - When found, downloads and imports the app
  - Can handle most websites, and the various ways vendors handle downloads

http://neographophobic.github.io/autoMunkiImporter

Australian National University

Automating Packaging

Saving time and effort with Auto Munki Importer

- Automated process
  - Set and forget*
- Optionally reports new versions via email
- Git aware
- Testing and deployment is still up to the SysAdmin
- Enables you to keep current with minimal effort
  - Proactive, instead of reactive packaging

Installation

- Install XCode Command Line Tools

  https://developer.apple.com/downloads/index.action

- Download and install Auto Munki Importer

  http://neographophobic.github.io/autoMunkiImporter/download.html

  - Installer is a standard Apple installer package
  - Installer handles all other dependencies

Configuration

- Edit config plist (/Library/Application Support/autoMunkiImporter/_DefaultConfig.plist), and set up default values particularly for:-
  - smtpServer, fromAddress and toAddress
- Test configuration with autoMunkiImporter.pl --test
- Set a Launch Daemon to run it on a schedule
  - http://neographophobic.github.io/autoMunkiImporter/configuration.html#launchd

Australian National University

Import Data Plist

Monitor anything you want

- Import data plists tell Auto Munki Importer
  - Where to find the download to monitor
  - How it should be imported into Munki
  - Other settings specific to the app
- Samples and template provided with the install
- Easy to create your own
  - http://neographophobic.github.io/autoMunkiImporter/dataplists.html

Example

Automating Google Earth and TextWrangler packaging

Static Download Link

Google Earth

- http://www.google.com/earth/download/ge/agree.html
- Select options that you want (unselect all extras)
- Click “Agree and Download”
- Copy “click here” link
  - No version info in URL, most likely a static URL where the URL doesn’t change, but the version does

- Import data plist items
  - URLToMonitor: http://dl.google.com/earth/client/advanced/current/GoogleEarthMacNoUpdate-Intel.dmg
  - itemToImport: Google Earth.app
  - name: GoogleEarth
  - type: static
  - displayName: Google Earth
- Save as /Library/Application Support/autoMunkiImporter/GoogleEarth.plist

Dynamic Download Link

TextWrangler

- http://www.barebones.com/products/textwrangler/
- Find download link
  - Link has version information in it which will most likely will change with each new version
  - Need to handle via “dynamic” mechanism
- Find unique link text for download link
  - “Download Now”

- Import data plist items
  - URLToMonitor: http://www.barebones.com/products/textwrangler/
  - downloadLinkRegex: Download Now
  - itemToImport: TextWrangler.app
  - name: TextWrangler
  - type: dynamic
  - description: Powerful Text Editor
- Save as /Library/Application Support/autoMunkiImporter/TextWrangler.plist

Process the data files

Verify that it has worked

- autoMunkiImporter.pl --data /Library/Application\ Support/autoMunkiImporter/GoogleEarth.plist
- autoMunkiImporter.pl --data /Library/Application\ Support/autoMunkiImporter/TextWrangler.plist
- Check that the apps were imported
  - Use --verbose to troubleshoot issues

Questions