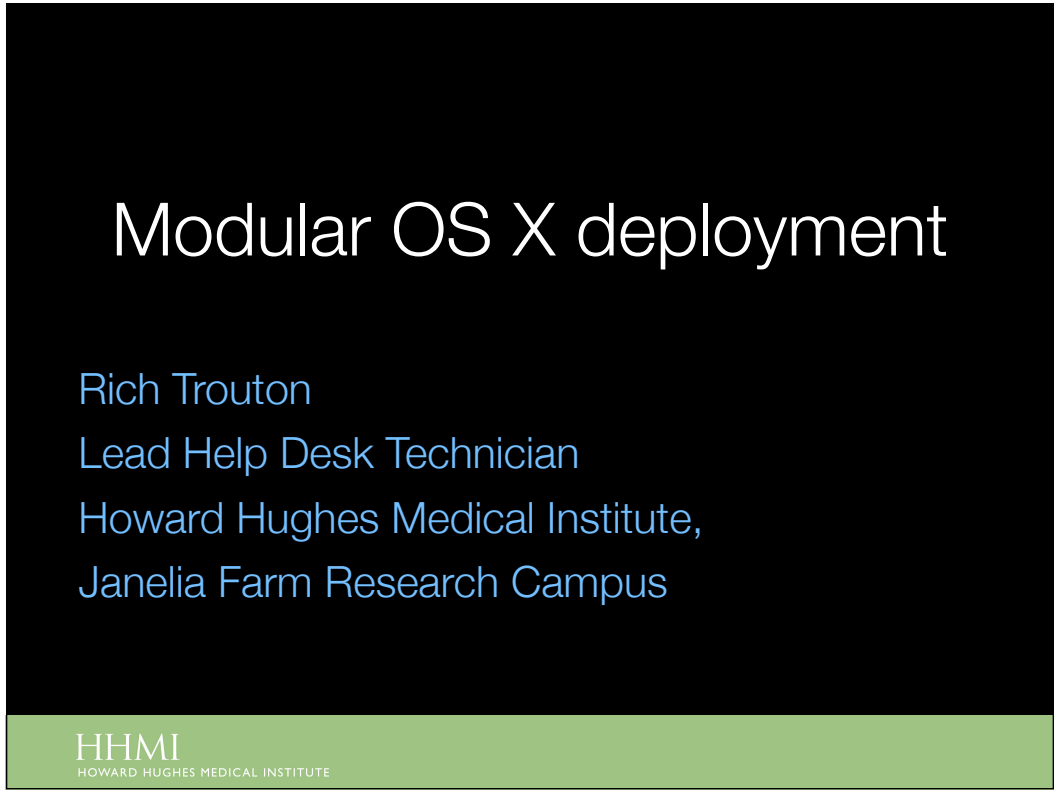
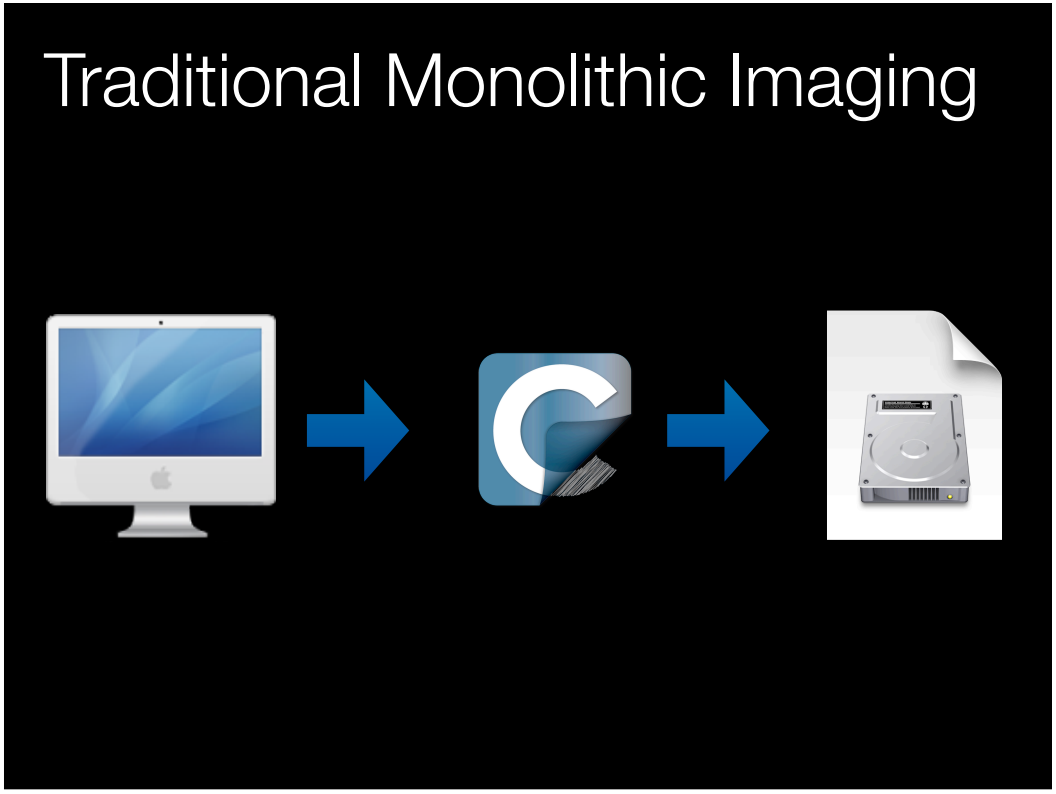


1. I'm going to start with a few questions for everyone, so please answer with a show of hands. How many have used imaging to set up new Macs? How many have used an imaging solution like InstaDMG? How about a deployment solution like DeployStudio? How many of you have built your own packages or wrote your own scripts?

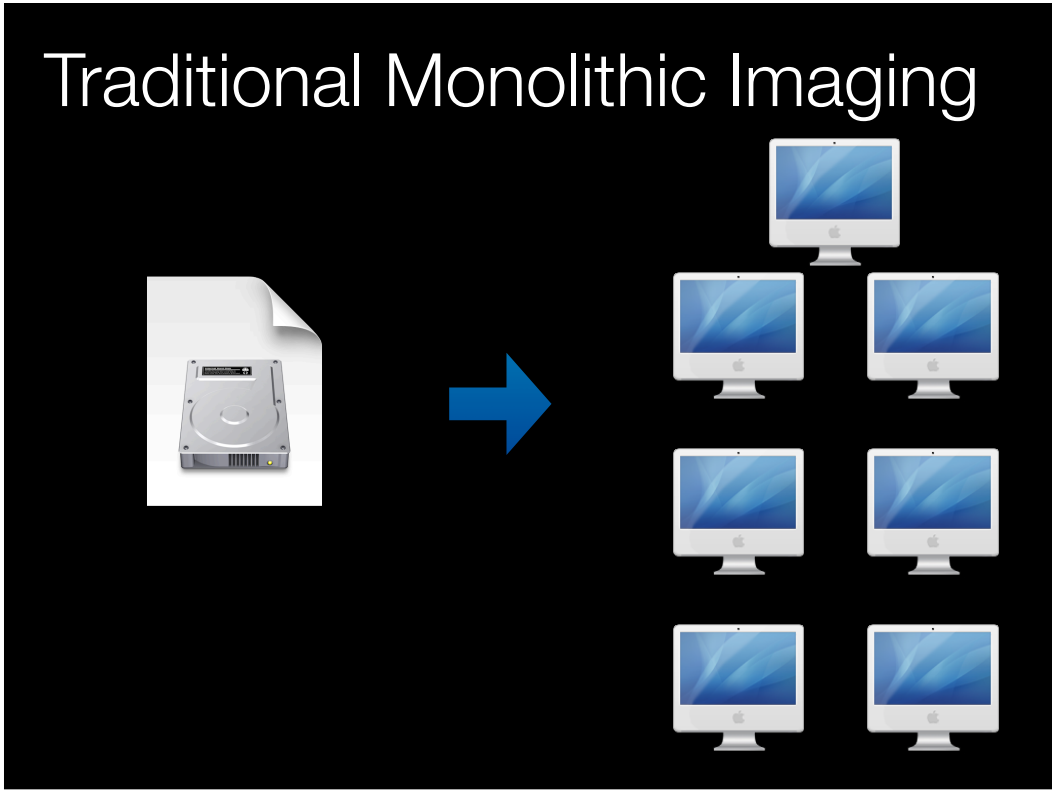
2. For those who have, great! You should hopefully be able to apply this session's information in useful ways. For those who haven't, hopefully what you learn today will encourage you to start looking how to do this in your own environments, as it will both save you time and make your life a lot simpler. I'm not going to go too in-depth about how to do this in today's session, as there are a number of sessions here at this conference that have or will, but I hope you'll see the advantages.



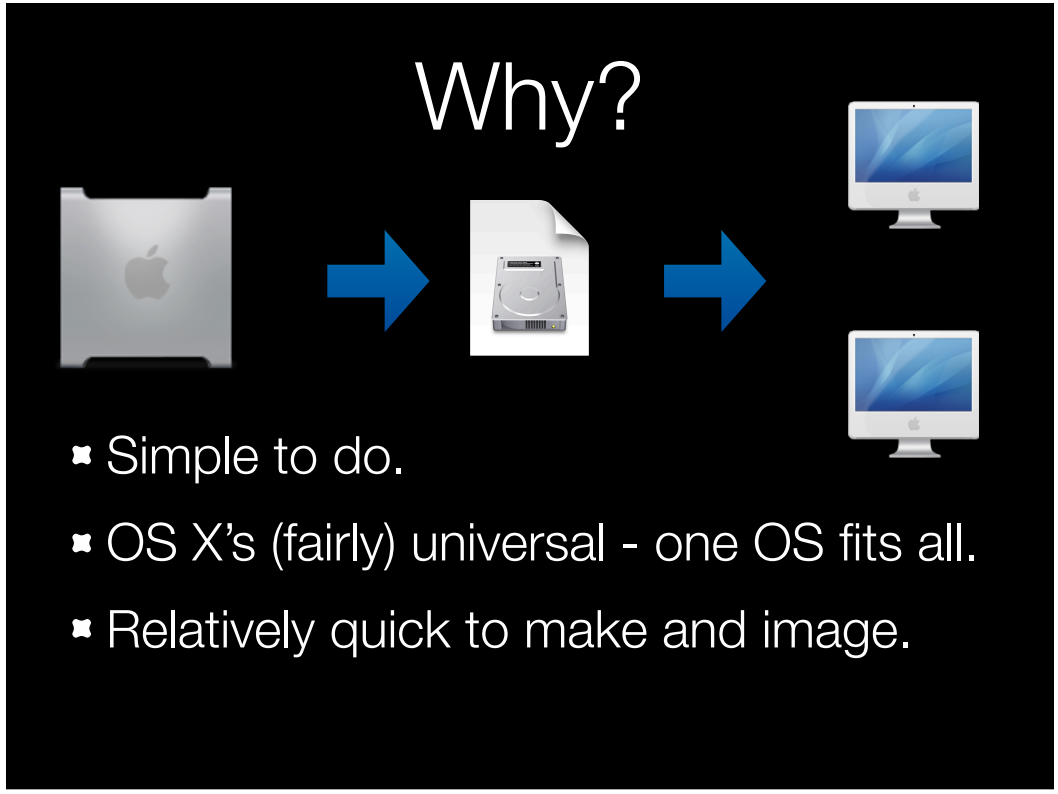
For a lot of us, here's how we started imaging:

A. Build a "perfect Mac", preferably using the newest Mac we've got. Get all of our site-licensed software onto it, make our tweaks, set the "apply to all Macs" settings.

B. Make an image of that Mac.



We then take that image and apply that image to the other Macs in our environment. Each one (regardless of model) is ultimately built from that Perfect Mac.



OK, so that's what we've been doing. Why are we doing it this way? To start with, it's simple. Make a "Perfect Mac", tweak it, grab an image. It's a task that you can give to an entry-level tech, with appropriate oversight, and be reasonably certain that it will be done correctly. The fact that OS X (within well-understood limits) gives you a universal build really helps here. It's also reasonably quick (depending on the complexity of your "Perfect Mac") to update your image. In this case, you would apply your image to a Mac, make the changes to your new "Perfect Mac" and then make a new image. Most experienced admins can probably make a Perfect Mac from scratch within a regular working day (assuming no distractions) and update one within a few hours.

What problems crop up?

- "We just got this new MacBook/iMac/Mac Mini in, and it doesn't work right with our image."
- Email comes in at 3:30 - "We need to add this one software package for a 50-person lab that's starting at 9:00 tomorrow morning, can you do that for us, kthanx."
- "Wow, this image is really behind on patches."
- "How'd we wind up with all of these images?"

So you've got your Perfect Mac-based image, everything seems accounted for, and it works. Right? So how many of you have had these thoughts or gotten these emails?

Improving the imaging process

- Break your image down
  - Base OS
  - Software
  - Settings

Here's how you can improve the process to address the issues. First of all, you need to think of how you can take that "Perfect Mac" and break it down into its separable parts: OS, Software, Settings.

Improving the imaging process

- Break your image down
  - Base OS - Use InstaDMG
  - Software
  - Settings

A. Image – build your image using a tool like InstaDMG. InstaDMG creates clean, never-booted ASR images, which can be as simple or as complex as you need it to be. You can use associated tools like InstaUp2Date to make sure that the image is up to date with Apple's software updates.

You can also use Apple's System Image Utility to build your base OS image, as 10.6's System Image Utility now using a similar methodology to InstaDMG. However, I've had much better results using InstaDMG.

Modularize your InstaDMG Image

- Use separate InstaUp2Date catalogs for your image's major parts
  - Base OS - Use "vanilla" catalog
  - Other software - build separate catalogs

InstaUp2Date allows you to create and use lists (catalog files) of installer packages and disk images from which InstaDMG would then build its images. You can leverage this functionality to break down your image into its major parts. The advantage to doing this breakdown is that you can make the various parts of your image more modular and be able to update just those parts that need to be changed without having to redo everything.

Modularize your InstaDMG Image

- Create separate InstaUp2Date catalogs for the various software installs
  - Build one (or more) for your third-party software.
  - Build others for parts that may not change often.

How you can do this is to create separate catalogs for the various parts. For my third-party software, like Office or Firefox, you can create a "customSoftware" catalog file and have it reference the software that you've checksummed and put into the "InstaUp2DatePackages" folder. If I also want to add XCode, I'll add the XCode disk image into the "InstaUp2DatePackages" folder and create another catalog called "xcode".

Modularize your InstaDMG Image

- Create a "master.catalog" file with the following include statements:
  - include-file: 10.6_vanilla.catalog
  - include-file: xcode.catalog
  - include-file: customSoftware.catalog
- When building, use the “master” catalog to run: **sudo /path/to/ instaUp2Date.py --process master.catalog**

Next you can create a "master.catalog" file with the following include statements:

include-file: 10.6_vanilla.catalog
include-file: xcode.catalog
include-file: customSoftware.catalog

The "vanilla" catalog is one that's updated and maintained by the InstaDMG developer team and defines all the current updates to the OS in question, so you'll be able to tap into that by including that in your master catalog. XCode will install your XCode installation, and customSoftware will install everything else. Once you've got everything you need configured, run the script (as root) with the "--process" flag to have InstaUp2date build the necessary InstaDMG files and folders, and then have InstaDMG run to build your image.

Improving the imaging process

- Break your image down
  - Base OS - Use InstaDMG
  - Software - Use installer packages
  - Settings

B. Software - make sure that all of your software can be installed with Apple installer packages. For some software vendors, this means you'll need to learn to re-package. For others, you may need to tweak their installers so that it's not asking for user input as part of the installation process, or doing something "helpful" that's not so helpful when nobody's actually logged into that particular machine.

Historically bad offenders in this category have been Adobe and Microsoft. Both have been cleaning up their act in recent years though. Microsoft released Office 2011 as a standard metapackage. Adobe has released Adobe Application Manager Enterprise Edition to facilitate re-packaging, as well as their increasing use of standard installer packages to install their newest software, has made deploying their software much easier.

For your packaging needs, there are a number of free packaging tools available. I personally use a combination of Iceberg and InstallEase, but try them out and use the one that works best for you. Both PackageMaker and InstallEase can be used to take "before" and "after" snapshots of your system, where you make the initial snapshot, install your software, then take a second snapshot once the installation is finished. The packaging software will then generate a list of the files and directories that changed and use those changes to generate a package. Stéphane Sudre's Iceberg and Packages, while they don't offer the snapshotting ability are fantastic for creating custom installer packages and metapackages. In fact, one of InstallEase's package creation options is to create an Iceberg project for a snapshotted package that needs additional scripting and customization.

If you need to use source control for your packaging, I recommend using The Luggage. It's an open-source project that uses code-checkable makefiles and Apple's PackageMaker to create installer packages. The Luggage was originally created by Joe Block, who had written a tool when he worked at Google to allow the other members of his group to easily review installer package changes before they were put into production. When Joe left Google, he wanted to have a similar tool available, so he wrote and open-sourced The Luggage.

Jamf's Composer is also a great tool for creating packages, and it can also take "before" and "after" snapshots to help generate packages. The main reason it's not up on this slide is that it isn't free. It is available for purchase from Jamf and is also included as part of Jamf's Casper Suite.

Improving the imaging process

- Break your image down
  - Base OS - Use InstaDMG
  - Software - Use installer packages
  - Settings - Script/install your settings.

C. Settings – learning how to script (or finding someone else who can script) can help you a lot by allowing you to create scripts to automate either all or parts of the post-imaging setup process. You can then bundle those scripts into installer packages so that they can also be installed onto your image. You can also capture settings that should be fairly universal and install them as-is with an installer package. A good example of this would be creating a package to install a customized user template.

Energy Saver Settings Script

```
#!/bin/sh

#
# Energy Saver Settings Example for Mac OS X 10.5.x - 10.6.x
# Rich Trouton, created September 18 2009
# Last modified March 5, 2011
#

# Set separate power management settings for desktops and laptops
# If it's a laptop, the power management settings for "Battery" are set to have the computer sleep in 15 minutes, disk will spin down
# in 10 minutes, the display will sleep in 5 minutes and the display itslef will dim to half-brightness before sleeping. While plugged
# into the AC adapter, the power management settings for "Charger" are set to have the computer never sleep, the disk doesn't spin down,
# the display sleeps after 30 minutes and the display dims before sleeping.
#
# If it's not a laptop (i.e. a desktop), the power management settings are set to have the computer never sleep, the disk doesn't spin down,
# the display sleeps after 30 minutes and the display dims before sleeping.
#

# Detects if this Mac is a laptop or not by checking the model ID for the word "Book" in the name.
IS_LAPTOP=`/usr/sbin/system_profiler SPHardwareDataType | grep "Model Identifier" | grep "Book"`

if [ "$IS_LAPTOP" != "" ]; then
    pmset -b sleep 15 disksleep 10 displaysleep 5 halfdim 1
    pmset -c sleep 0 disksleep 0 displaysleep 30 halfdim 1
else
    pmset sleep 0 disksleep 0 displaysleep 30 halfdim 1
fi

# Remove setup LaunchDaemon item

srm /Library/LaunchDaemons/com.company.powersettings.plist

# Make script self-destruct

srm $0
```

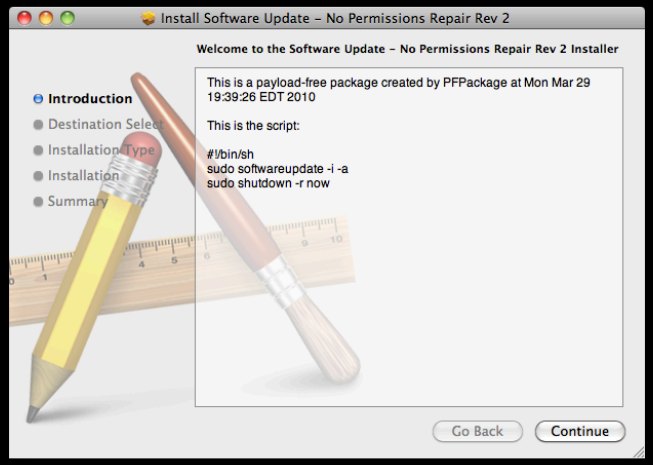
Here's an example of how you can set system settings using shell scripts. In this example, an installer package installed two files, with this script installed in /var and an accompanying LaunchDaemon installed in /Library/LaunchDaemons. On first boot, the LaunchDaemon triggers the script to run. The script runs, detects whether the Mac is a desktop or laptop, sets the appropriate Energy Saver settings, then deletes the LaunchDaemon and itself.

“Check the ‘Require password to unlock each System Preferences pane’ checkbox in System Preferences: Security” Script

```
#!/bin/sh

#
# Security Settings Example for Mac OS X 10.5.x - 10.6.x
# Rich Trouton, created March 5, 2011
#

# Check the "Require password to unlock each System Preferences pane" checkbox in System Preferences: Security

/usr/libexec/PlistBuddy -c 'set rights:system.preferences:shared bool false' '/etc/authorization'

# Remove setup LaunchDaemon item

srm /Library/LaunchDaemons/com.company.lock_preferences.plist

# Make script self-destruct

srm $0
```

Here’s another example, using the same installation methodology, where you’re setting the “Require Password” setting in the Security preferences pane.

Script-Only Installer Packages (AKA Payload-Free Packages)

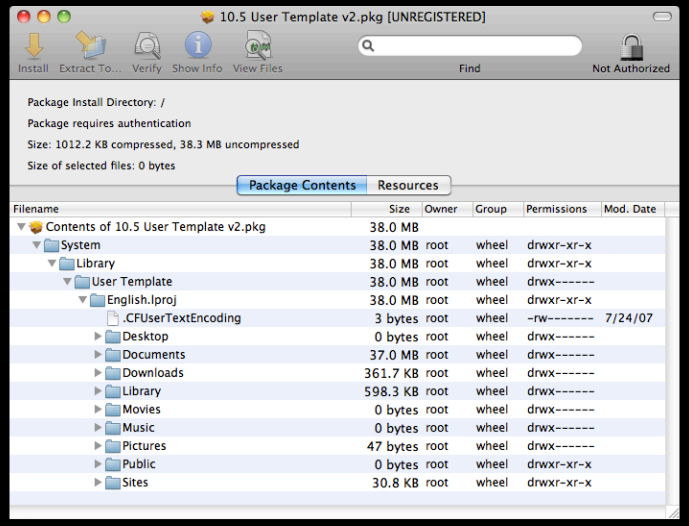


There's a way to combine packaging and scripting, using script-only installer packages. Using an installer package as part of a deployment workflow gives you a couple of advantages:

1. You only need to build it once –With a script-only installer package, you can build it once, then re-use the installer over and over to apply your script in a consistent fashion to multiple deployments.

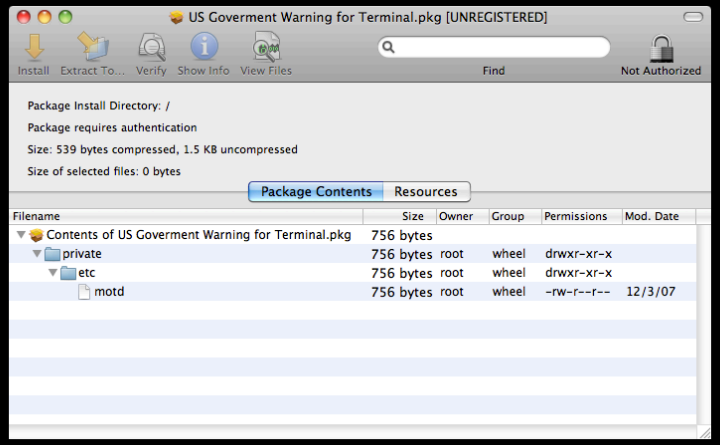
2. You can hand it off to anyone to run - Since the script is built and can't be altered, you can give this to anyone you need to. They don't need to know how the script works, they just need to be able to run the installer.

User Template Installation



You can also package up settings that can be set as a template or otherwise universally applied. In this case, you can set up a customized user template on one machine and build an installer package of it to install as part of your workflow.

Setting “Message of the Day” in Terminal



In this example, you can set a message to appear in new Terminal windows by setting the contents of the motd file in /etc. Here, this installer package was built to install a standard US Government warning that appears when you open a new Terminal window.

How to implement?

- InstaDMG - Add software as part of image build train.
- DeployStudio - Install software as pre- or post-boot actions
- Post-boot setup - Build metapackages and install after initial setup
- System Management tools - install the agent, let your system management tool install the rest.

A. InstaDMG - As part of the image building process, you can add third-party installer packages and have InstaDMG add that software and/or scripts as part of its build process.

B. DeployStudio - For those using DeployStudio, you can have it install packages as part of a pre-boot or post-boot install process.

C. Post-boot installation of metapackages - You can bundle your installer packages into one large metapackage (or a couple of metapackages) and install them after the Mac's first boot.

D. Using your system management tools - If you're using a systems management tool like Casper, Absolute Manage, KBox, Puppet, Munki, etc, you can have installer packages set to install as soon as the imaged Mac's agent checks in for the first time.

Making multiple tools work together

- Combination approach
  - Image building
  - Deployment
  - System Management

Ideally, you can make several tools work together to achieve your goal rather than trying to make one tool do everything. My recommended approach is to build your image so that it's only got the lowest common denominators. Everything else can be added via scripts and packages. That makes an image that's both easier to update, and one that takes less time to build. It also allows you to reduce how many images you're using (hopefully down to *one*), as all of the customization is handled via the installer packages and scripts.

For example, the base image that I'm using at HHMI has the following:

10.6.7 (with all available updates that InstaUp2Date can provide.)
XCode 4.0.2
The clearReg installer package, which tells the Apple Setup Assistant to not run on boot.

Everything else I'm installing is a software package that's installed by DeployStudio once the Mac's hard drive has the base image laid down. Patching is then handled by Kace's KBox.

New Mac image problem

- Imaging problem - Apple sometimes releases customized versions of the OS to accomodate just-released hardware.
- The OS that comes with the Mac is fine, but it takes a while to customize.

Another imaging problem that was touched on earlier is “What do you do about brand-new Macs that have a customized OS?” Chances are you’ve had this happen before, where a new Mac is released after the last major update to the OS came out, but before the next version. So Apple ships a customized version of the OS, which includes the drivers for this particular machine. However, your image doesn’t work right (or just doesn’t work) and you wind up installing everything from scratch. Maybe you decide to make this a new “Perfect Mac” and grab an image from it while you’re at it.

Thin Imaging*

- Install On Top of the new Mac's OS
  - Use custom-built and retail scripts and packages.
  - End Goal - Make that Mac look like it was imaged without actually erasing the Mac's boot drive.

*Credit to Penn State's Rusty Myers for "Thin Imaging" term. Thanks, Rusty!

How can we apply modularization to the new Mac problem? Don't image, thin image! Use the OS that comes with the Mac and lay your setup scripts and software packages down on that Mac without wiping and re-imaging. Added bonus - No more figuring out what Apple's changed in the customized OS that came with that new laptop.

What can be installed using a thin imaging process? Almost anything that you would normally put into your InstaDMG or DeployStudio imaging workflow. You can turn off the setup assistant, add users, bind to AD or OD, install software or apply settings. If you can script it or package it, you should be able to apply it to your thin imaging workflow.

Once you've got your process built and working, updating it is as easy as updating your other imaging workflows. You should be able to use the same scripts and packages.

Thin Imaging with DeployStudio

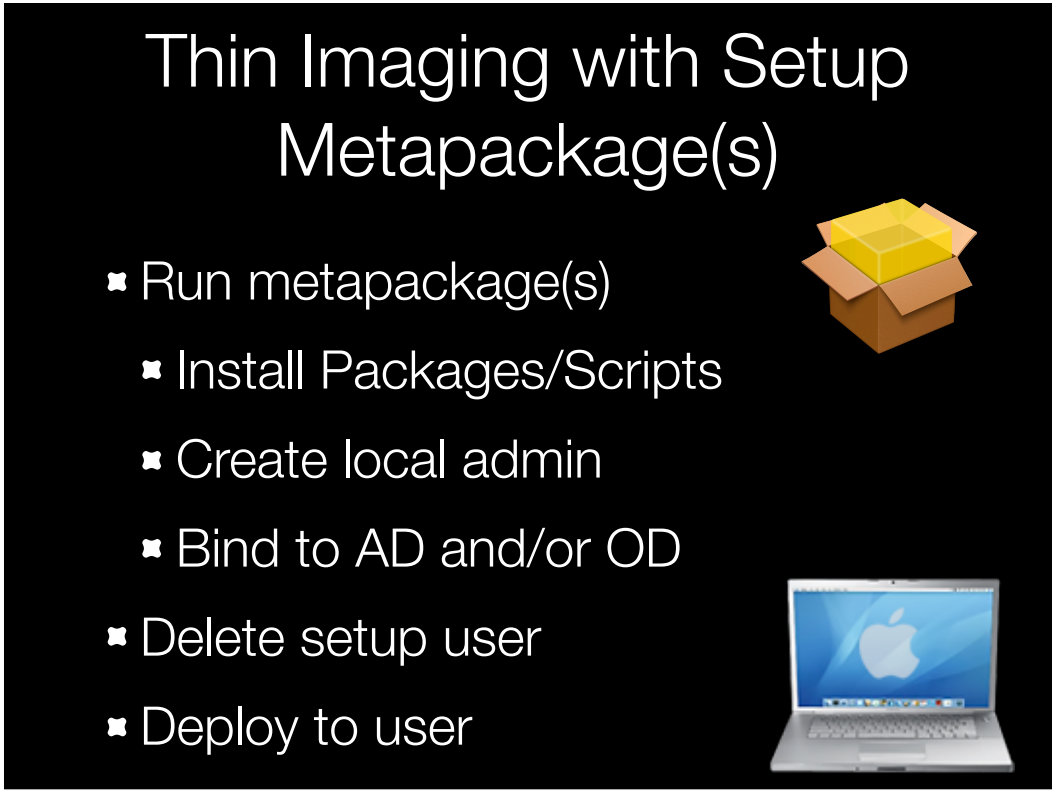
- Workflow Example
  - Remove Computer from Box.
  - Boot to DeployStudio, run workflow
    - Install Packages/Scripts
    - Create local admin
    - Bind to AD and/or OD
  - Deploy to user

Here's an example of how we can do this with DeployStudio. In this example, you've got the ability to NetBoot to a DeployStudio server or you have a USB/Firewire drive setup to boot to DeployStudio. You'd then have a workflow set up to target a drive named "Macintosh HD" and lay down a series of packages and scripts, create the users you need, and bind the Mac to Active Directory or Open Directory. You can also leverage DeployStudio's ability to run Apple software updates to automate the process of getting the Mac fully up to date before deployment.

Thin Imaging with Setup Metapackage

- Remove Computer from Box.
- Boot Mac up and run through Setup Assistant to create setup user
- Copy setup metapackage(s) to Mac

Not everyone is going to have access to server resources, so here's an example of how we can do this with setup metapackages. This process is a little more hands-on, because you need to be in a position where you can log in and run the setup metapackages. Rather than skipping Setup Assistant, run through it and use the user created to make an "install" admin user account and copy the setup metapackage or metapackages to the Mac being configured.



Once you've got the metapackages copied down, run them to install your packages and scripts, create the users you need, and bind the Mac to Active Directory or Open Directory. Once you're done, log into your just-created admin account and delete the install user account. After that, you should just need to run your Apple software updates to get it ready to deploy.

Real World Examples

8. Real world deployment examples

A. A medical research institute within a larger unnamed US Government agency. Has a population of about 500 - 600 Macs.

1. Don't have access to NetBoot.
2. Not using DeployStudio (largely because of the lack of NetBoot)
3. Labs buy their own equipment, the LAN Support folks are setting them up after they arrived (often times LAN Support learns about the new equipment from the "Can you set this up?" ticket.)

Tools used:

InstaDMG
Metapackages (one for initial OS setup, another which contains the latest versions of the site licensed software.)
Web server for hosting InstaDMG-built images, to leverage Disk Utility's ability to restore images that are available via HTTP.
Absolute Manage

B. A medical research non-profit. Has a population of 300 - 400 Macs.

1. Can use NetBoot (second greatest thing since sliced bread)
2. Using DeployStudio (greatest thing since sliced bread)
3. Labs buy their own equipment, but equipment delivery and setup come via IT.

Tools used:

InstaDMG
DeployStudio
Kace's KBox


Demo

Looking Ahead - What about 10.7?

- Imaging and packaging tools will probably need to be updated for 10.7
- In my own testing with the Lion Developer Preview, the majority of my own packages and setup scripts worked fine in 10.7 without any changes.
- Test, test, test.

So, looking ahead to 10.7, what changes to this methodology can be expected? Well, InstaDMG, Deploy Studio and the other tools that have been discussed will likely need to be updated but that's pretty much expected for a major OS upgrade. Otherwise, based on my testing with my own workflow and the Lion Developer Preview, the packages and scripts that are working today with your 10.6 build process should also work with 10.7.

However, your mileage may vary so test, test, test.

Tips

- Don’t reinvent the wheel if you don’t have to.
- Where possible, use the vendor-provided installer package instead of repackaging.
- Making script-only installer packages allows scripts to be easily integrated into your workflow by wrapping them in an installer.
- Design your process so that others can run your deployment tools easily.

Here are some tips I’ve learned. One, don’t reinvent the wheel if you don’t have to. Look and ask around first, see if someone’s already solved the problem you’re seeing. Two, save work and use the vendor-provided installer package whenever possible. Three, making script-only installer packages allow you to add custom scripting to just about any workflow. Four, make sure to design your tools so that others can run them easily. I don’t know about you, but I like not being disturbed when I’m out sick or when I’m on vacation. Building your tools so that they’re easy to use, then documenting how to use them properly, will cut down on “how do I do this?” phone calls considerably.

In fact, I’m going to repeat myself on one point: Documenting them properly is key. I’ve had numerous occasions where, six months down the line, I’ve wanted to adapt something I’d previously developed to another purpose. Most of time, I didn’t remember exactly what I did previously and had to go back to the documentation I’d written six months ago. I know my memory is fallible, so I make sure to write things down in as much detail as I can and put in screenshots and code wherever possible.

As an extra bonus, writing things down at that level of detail means they can be easily turned into documentation for the other non-Mac members of your IT department or for your users.

Useful Links

- InstaDMG - http://code.google.com/p/instadmg/
- InstaDMG forum - http://www.afp548.com/forum/index.php?forum=45
- DeployStudio - http://www.deploystudio.com/
- DeployStudio Forum - http://www.deploystudio.com/Forums/
- OS X Deployment Wiki - http://www.osxdeployment.com/

Here's some useful links for both InstaDMG and DeployStudio. The OS X Deployment wiki is meant as a general resource for anyone who deploys Mac images, manages patches, settings and/or security on their end-user machines. Wikis thrive on added content, so check it out and add your own information as appropriate.

Useful Links

- Iceberg - http://s.sudre.free.fr/Software/Iceberg.html
- Packages - http://s.sudre.free.fr/Software/Packages/about.html
- InstallEase - http://www.lanrev.com/solutions/package-building.html
- PackageMaker - http://developer.apple.com/
- The Luggage - http://luggage.apesseekingknowledge.net/ , https://github.com/unixorn/luggage

Here's some useful links for the package tools discussed today.

Questions?

Thank You For Attending

Enjoy the rest of the conference

@rtrouton