



BUILDING A SOE / MOE

Adam Reed

The Australian National University



Hashtag : **#xw13**

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

Agenda

First Session

- Introduction
- Definition of Terms
- Planning a MOE
- OS X File System
- Tracking Changes
- Packaging

Agenda

Second Session

- Deployment
- Scripting and the CLI
- Remote Access
- Extension Ideas
- Conclusion and Questions



Introduction

Why are we here and what are we going to cover?

Who Am I?

Adam Reed

Manager - Systems and Information Technology,
Facilities and Services Division

Previously

Team Leader - Managed Operating Environments,
Information Technology Services
The Australian National University

Email: adam.reed@anu.edu.au

Ph: (02) 6125 1479

What Did I Do?

For the MOE Team

- I was responsible for the MOE Team who manage the desktop images provided by Central IT
- 5.6 Staff - 2 Windows Admins, 1 Mac Admin, 1 Support Tech, 0.6 Liaison Officer and myself
- Primarily Student Images (~1700 machines)
 - Roughly 2/3 (1100) Windows, 1/3 (600) Mac
 - Multiple Mac images - 3 main images

This Session Is About

- Sharing knowledge, tips and tricks on building a SOE / MOE
- Showing you some of the tools that you can use to assist you in delivering managed environments
- Giving you the foundations to build your own MOE that suits your environment
- Showing you ways to extend your MOE

This Session Is Not About

- Providing a “cookbook” of steps to building a MOE
- Comprehensive coverage of all of the facets of building a MOE
- The only way to do things - its based on ideas, implement them as you see fit

Questions

- Feel free to ask questions at any time
- If you have any area you are particularly interested in let me know - time permitting I'll answer what I can
- I may not be able to answer all questions but can hopefully point you in the right direction



Definition of Terms

SOE / MOEs have a language of their own, so...

SOE

Standard Operating Environment(s)

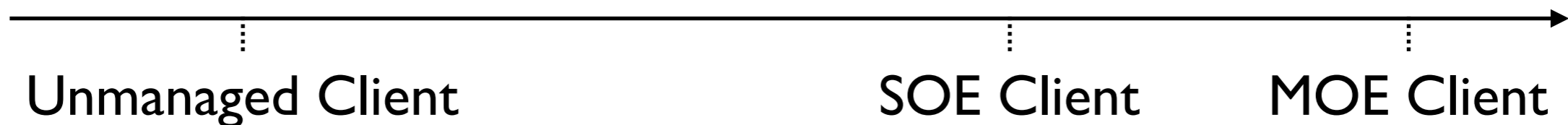
"The Standard Operating Environment (SOE) is a specification for standards for computer hardware, operating system, security and applications software."

<http://www.dundee.ac.uk/ics/services/soe/>

MOE

Managed Operating Environment

Unlike an image-based SOE, a MOE is an adaptable and dynamic environment able to grow and change with an organisation's hardware and software needs. It allows user-level customisation without affecting the integrity of the environment.



Image

- Term given to the software set of a managed computer.
Each SOE / MOE will have an image
- Can refer to either a currently running machine, or the files that are deployed to a machine
- 4 Types of images (in terms of deployment)
 - Monolithic, incremental, hybrid and thin
 - Thin Imaging is the current best practice

Deployment

- The process of making changes to and installing / removing software from MOE machines. Typically achieved remotely in a MOE environment
- Various tools to assist like:-
Munki, Radmin, Apple Remote Desktop, Puppet, Casper, Absolute Manage, etc.

Packaging

- The **art** and **science** of collecting all of the required items together into a container that can then be deployed to machines
- Typically a complete application, but can also be isolated elements such as preferences, resources, scripts etc.



Planning a MOE

MOEs start away from the keyboard

Know your needs

Different environments mean different MOEs

- Who is your target user group?
- What are your users' needs? What are your needs?
- What are your organisational needs?
- How often are changes going to be needed?
- What are your software licensing models?
- BYOD vs Uni owned machines



*Solve technical problems
technically and political
problems politically*

Environment

- Will your machines always be on and connected to your network?
- What is their network connection?
- Desktops vs laptops?
- Energy saving profile?
- Administrator access?



*MOEs are built one
component at a time. You
don't need an über image
from day one!*

Deployment Options

- What technologies are you going to use?
- What are its needs for things like packages?
- How are you going to interact with your image?
- How and how often are you going to update it?
- Modularity and reuse are **vital**
 - Plan for major OS upgrades



*Remember that
more \neq better
&
If it isn't broken
don't fix it - "KISS"*

Policies and Procedures

- Have defined policies for things like change management and requests - you **can** drive these, regardless of your position
- Documentation is **really** important! Use it to cover your backside and to make what you do repeatable
- Testing is also vital. Make sure you, and particularly your users, do testing. If possible make them sign off on changes



OS X - The File System

Where to look to bend it to your will

File System

Primary Folders - User Perspective

- /
- /Applications and ~/Applications
- /Library and ~/Library
- /System
- /Users and ~/



In Terminal
`cd /path/to/
folder`

File System

Primary Folders - Unix Perspective

- /etc - configuration items
- /tmp - temporary files
- /var
- /usr - binaries and libraries
- /Volumes - external mounts



File System

Permissions

- Two forms of permissions
- Standard POSIX
 - Based on owner, group and other
- ACLs (Access Control Lists)
 - ACLs are the same as available in Windows
 - Used in clean OS installs

File System

Permissions (ls -la /path/to/dir)

```
d rwx rwx r-x 2 fred admin 68 Jul 1 10:37 dir
```

Entry Type (d = directory, l = symlink, - = regular file)

Permissions for the owner (in this case Fred)

Permissions for the group (in this case Admin)

Permissions for other (used if the user isn't the owner or a member of the assigned group)

In this case, Fred and members of the Admin group can do everything. Other users can only read and execute

File System

Permissions (`ls -la /path/to/dir`)

```
drwxr-x--- 2 fred admin 68 Jul 1 10:37 dir
```

Item	Description	Display
r	Read	r = on - = off
w	Write	w = on - = off
x	Execute (needs to be on for directories)	x = on - = off

In this case, Fred can do everything, members of the Admin group can only read and execute and other users have no access rights

File System

Permissions - Unix Commands

Command	Description	Example
ls	List directory contents	ls -lae
chmod	Change file modes or ACLs	chmod 644 file
chown	Change file owner and group	chown root:wheel file
chgrp	Change group	chgrp admin file
chflags	Change file flags	chflags nouchg file

File System

Permissions - Unknown User (99)

- UID 99 and / or GID 99
- Means that the file inherits the current user's UID and / or GID
- Particularly handy in multi-user machines as you can set generic permissions and have them correctly applied for any user on the system

File System

Hidden Files

- You can “hide” files from your users if you wish, and many (particularly unix) apps do
- `.[filename]` - Add a dot at the beginning of the file, or
- `SetFile -a V /path/to/file`

Note: hidden \neq inaccessible or un-findable. If a user shouldn't access a file, change its permissions, don't hide it.

File System

Symbolic Links (`ln -s source name_of_link`)

- Symlink allow you to put a file or folder in one location, and then have a reference that points to it another location
- The system will automatically traverse the link
- Using for lots of things - e.g. if you are using Network Home Directories, `symlink ~/Library/Caches to /tmp` (which is a symlink) so that cache info isn't written to your fileserver
- As a general rule, link parent directories, not individual files for the best results

File System

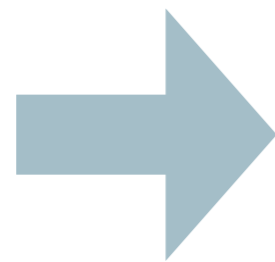
Domains

- Files can be placed in one of 4 domains:
 - User - Applicable to only the current user
 - Computer - Applicable to all users on the machine
 - Network - Applicable to appropriate machines on the network
 - System - Reserved for the system. Don't modify.

File System

Domains - Example (Safari preferences)

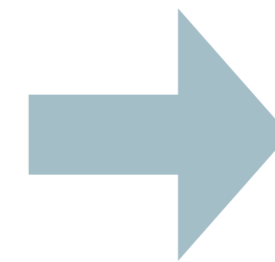
Search Precedence



User



Computer



Network

com.apple.Safari.plist

~/Library/Preferences/com.apple.Safari.plist

/Library/Preferences/com.apple.Safari.plist

/Network/Library/Preferences/com.apple.Safari.plist

File System

Domains - Why they are important in a MOE

- Some installers put items in the User domain when the Machine domain would be more appropriate
- Some items you could consider moving include:-
Spotlight Importers, Widgets, Plug-ins, Preference Panes, Screen Savers, Quicklook Plugins, etc
- Move resources, not user configuration files

Disclaimer: It *should* work but it depends on developers using the relevant Apple APIs. **Test** any changes you make.



Tracking Changes

How to find out what has happened...

Tracking Changes

What's changed?

- The ability to track changes made to the file system is vital for maintaining a MOE
- If you can determine what changes, you can deploy those changes in a repeatable and exact manner
- Also a good troubleshooting tool

Tracking Changes

Tools - Live as it happens

- fseventer - GUI App
- Subscribe to the same mechanisms Spotlight and Time Machine uses
- Doesn't require any pre-configuration
 - Very handy tool in your arsenal

Tracking Changes

Tools - Pre and Post “Snapshotting”

- Mix of GUI and CLI Tools
 - InstallEase, Casper, PackageMaker, Radmind, etc
- These apps take a before and after snapshot then show the difference
- I use InstallEase and Radmind (phasing out) in conjunction with fseventer. Different tasks have different needs



Hands On

Let's watch some live changes

Tracking Changes

- Start fseventer
- Configure prefs (“Events Expire” to “Never”)
- Start by clicking on the black “play” arrow
- Enter username and password - only needed on first run to give the app permission to view what is going on
- Watch what happens when you open some random apps, change prefs and quit

Tracking Changes

Troubleshooting

- If you have moved items, or changed permissions, you may see weird behaviour and errors
- Run the app on a “clean” machine and track it, then run it on a MOE machine and look for similar items
- Any differences maybe the cause of your problems

Tracking Changes

Difference Tools

- Once you know what changes, you can compare a pre-change to a post-change file and determine what actually changed
- Tools like diff, twdiff, TextWrangler and FileMerge will show you changes in text-based file - binary is harder.
- To convert plists from binary to xml
`plutil -convert xml1 /path/to/plist.plist`



Packaging

Installing and creating installable packages



Packaging

Three Sub Topics

- Types of installer packages
- Installing software
- Creating packages

Packaging

Types

- Drag and drop
- Custom installers (scripts, VISE, etc)
- Installer packages and metapackages
- Distribution and flat packages
- Mac App Store
- Built-in auto updating mechanisms (Sparkle framework and others - e.g. Adium)

Packaging

Installing - Drag and drop

- Drag and drop is common for a lot of smaller applications, and typically involves dragging the application from a disk image into /Applications
e.g. Firefox
- Some applications will do an “installation” on first run
 - Even when sandboxed

Packaging

Installing - Drag and drop

- Drag and drop installation is bad for a MOE
 - Too manual a process
 - Potentially error prone - you need to remember where you put the app last time
- ARD can do a copy file operation to install a drag and drop app
- Watch what happens on first run as it may setup its environment which you may need to replicate



Hands On

Install and packaging of “TextWrangler”

Install TextWrangler

1. Create the initial snapshot
 - 1.1. Start Absolute Manage InstallEase from
/Applications/Utilities
 - 1.2. Leave “Automatic” selected
 - 1.3. Click Continue
 - 1.4. Accept defaults and click “Take Snapshot”
 - 1.5. Enter admin password
 - 1.6. Wait for snapshot to complete

2. Start fseventer and observe while completing the rest of the steps
3. Mount “TextWrangler 4.5.2.dmg”
 - 3.1. Drag TextWrangler to the Applications folder
 - 3.2. Unmount “TextWrangler 4.5.2”
4. Run TextWrangler
 - 4.1. Ensure “Install the current command line tools” is enabled then click “Skip Registration”
 - 4.2. Enter admin password
 - 4.3. Quit TextWrangler

5. Back in InstallEase

- 5.1. Click “Take Snapshot”
- 5.2. Enter admin password if prompted
- 5.3. Review added files, removing items not needed
(i.e. Users folder). Click “Continue”
- 5.4. Check “Iceberg project”
- 5.5. Click “Create”
- 5.6. Save to Desktop as “TextWrangler”
- 5.7. Enter admin password if prompted

Installing TextWrangler

What happened?

- You will have noticed a couple of things about the install
 - XAttr (quarantine flag) was removed
 - Initial install was completed when you dragged and dropped the app
 - Additional components were installed on first run
 - Preferences were written on exit

Installing TextWrangler

What happened?

- From a simple drag and drop, files are now in:-

/Applications

/Library/LaunchDaemons

/Library/PrivilegedHelperTools

/usr/local/bin

/usr/local/share/man/man1

~/Library/Application Support

~/Library/Preferences

Packaging

Installing - Installer

- Installer installs Apple Packages, using the same technology regardless of vendor - like MSIs for Windows.
- Can run pre and post action scripts and check the machine matches set requirements
- Can be installed via a GUI or CLI tool
- Changes can be examined before they are made
- Repeatable

Packaging

Installing - Installer

- You really should look at “packaging” custom changes you make
- Allows for automation
- If you use Apple’s Package Format you can use tools like Munki, ARD, or InstaDMG
- We have a metapackage that will configure a generic OS X install to an known good ANU base configuration



Hands On

Install “Iceberg”

Installing Iceberg

The long but educational way...

- Mount Iceberg
- Right click on Iceberg.pkg and select show package contents, double click on Contents
- Start a terminal window and type `lsbom` and drag Archive.bom onto the window. Click enter.
 - It should read `lsbom /path/to/Archive.bom`

Installing Iceberg

The long but educational way...

- Leave terminal open but double click on package.
- Go Files → Show Files (⌘I)
- Both show the bill of materials which is what will be installed - note that scripts may make additional changes
- Hit space on the package to inspect with Suspicious Package
- Again see what is happening. Have a look at resources - particularly post* scripts.

Installing Iceberg

The long but educational way...

- Now that we know what is going to happen. Install Iceberg via the command line with:-
`sudo installer -verbose -pkg /path/to/pack -target /`

Installing Iceberg

What did we learn?

- Most of the steps were designed to show you how to look at the bill of materials
- Don't forget that scripts can also make changes
- The command line installer is the same as running the GUI in most cases

Creating a Package

PackageMaker vs Iceberg

- Apple provide PackageMaker for making packages.
- PackageMaker continues to improve but has a number of quirks (much better since Leopard - was useless in Tiger)
 - It's part of the Axillary Dev Tools download
- That said I still prefer Iceberg (a third party tool) or Packages (from the same vendor)



Hands On

Package SSH Settings

Creating a Package

Using Iceberg

1. Start Iceberg
2. File → Preferences
 - 2.1. Default Reference Style: Project Relative
3. File → New
4. Select “Package” and click “Next”
5. Project Name: “SSH”
6. Project Directory: “~/Desktop”
7. Click “Finish”

Creating a Package

Packaging SSH settings

- Copy my SSH Source folder into the SSH folder on your desktop
- Absolute vs Relative
 - I use relative so that a package templates can be passed around and is repeatable
 - Absolute is easier but not as repeatable

Creating a Package

Packaging SSH settings

- Expand the SSH item
- Settings
 - Version: 10.8
 - Identifier: `au.edu.exampleuni.pkg.SSH`
 - Get Info: SSH 10.8
 - Short Version: 10.8
 - Version: Major 10, Minor 8

Creating a Package

Packaging SSH settings

- Settings
 - Options
 - Authorization - Root Authorization
 - Flags
 - Allow Revert to Previous Version
 - Follow Symlinks

Creating a Package

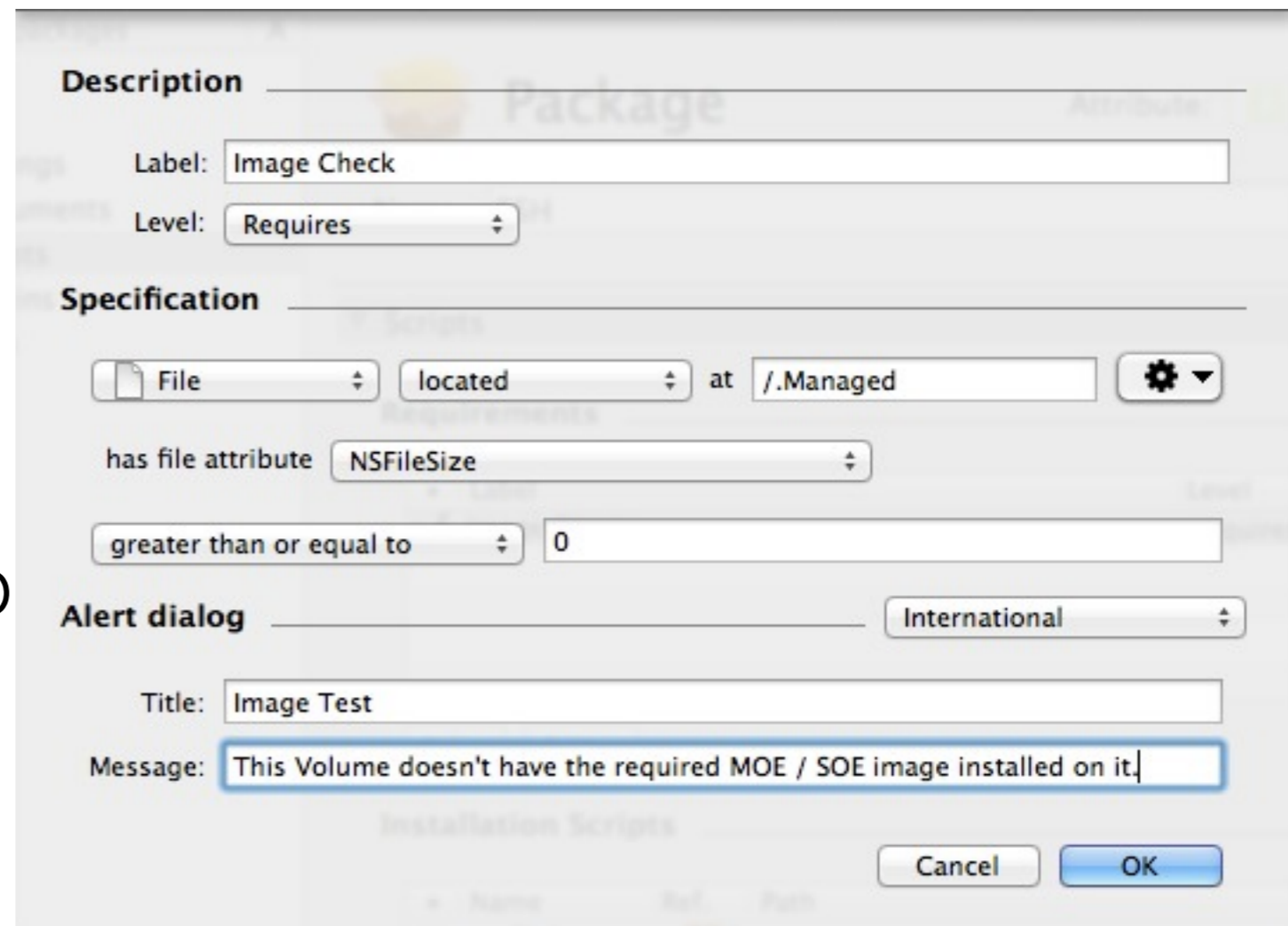
Packaging SSH settings

- Documents
 - Add read me and select path
 - Add a background image, no scaling with left bottom alignment, ensure path is selected
 - Make sure both are set to “R”, not “A”

Creating a Package

Packaging SSH settings

- Scripts
 - Add a postflight scripts from the provided resources
 - Add InstallationCheck to Additional Resources
 - Add this requirement

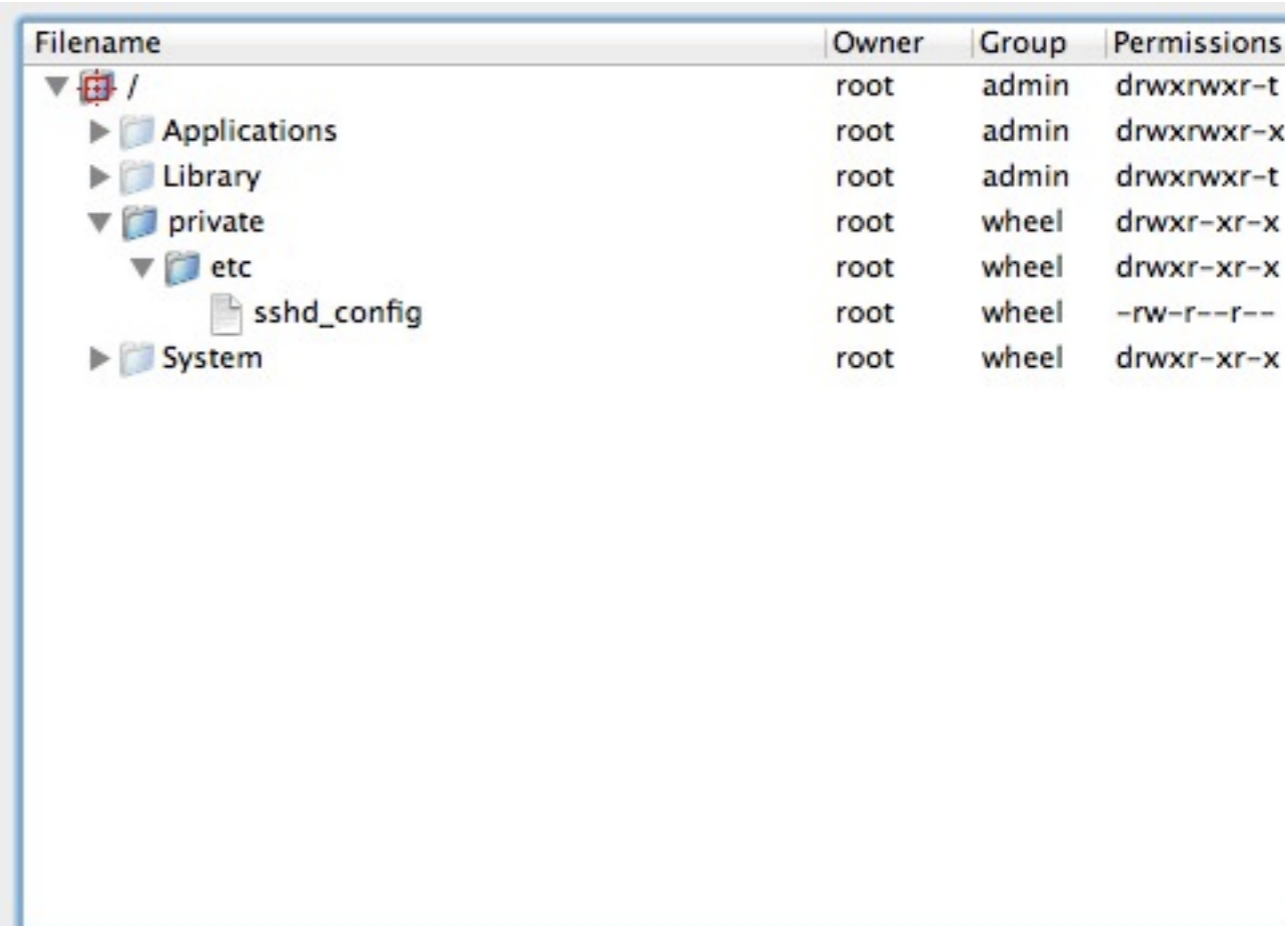


The screenshot shows the 'Package' configuration dialog box. The 'Description' section has a 'Label' field containing 'Image Check' and a 'Level' dropdown set to 'Requires'. The 'Specification' section is configured with 'File' located at '/.Managed', which 'has file attribute' 'NSFileSize' that is 'greater than or equal to' '0'. The 'Alert dialog' section is set to 'International', with a 'Title' of 'Image Test' and a 'Message' of 'This Volume doesn't have the required MOE / SOE image installed on it.'. 'Cancel' and 'OK' buttons are visible at the bottom right.

Creating a Package

Packaging SSH settings

- Files
 - Create the private and etc folders
 - Add the sshd_config file
 - It should look like this



Filename	Owner	Group	Permissions
/	root	admin	drwxrwxr-t
▶ Applications	root	admin	drwxrwxr-x
▶ Library	root	admin	drwxrwxr-t
▼ private	root	wheel	drwxr-xr-x
▼ etc	root	wheel	drwxr-xr-x
sshd_config	root	wheel	-rw-r--r--
▶ System	root	wheel	drwxr-xr-x

Creating a Package

Packaging SSH settings

- Build → Build and Run (⌘R)
- See that it installs as expected (it should fail)
 - Run: `sudo touch /.Managed` and try again
- Open the package up and have a look at the Info.plist file



Deployment

A brief look at deployment. It is a topic that we could spend weeks on.

Deployment

Thick vs thin images

- Accepted practice has change over the years, and thin imaging is now considered best practice
- Thin imaging is basically only deploying the bare minimum to get the machine to boot, and then bootstrapping with your deployment tool (like Munki)
- That said, thick images are still acceptable, if you're smart in how you build them
 - Hint: Build a thick image from a thin image

Deployment

Thick vs thin images

- Thin images
 - Very reusable and adaptable
 - Minimal amount of work to support new hardware
 - More agile for changing business needs
- Thick images
 - Quicker to deploy
 - All software already installed and configured

Deployment

Creating a thick image - the smart way

- Consider using InstaDMG
 - Automates the work for you
 - Highly flexible
 - Reusable
- It's built around the principles we discussed earlier of modularity, consistency, and repeatability
 - Gives you most of the benefits of Thin imaging

Deployment

Creating a thick image - from an existing machine

- Ensure machine is fully updated, and that you have emptied the trash, clear browser histories etc.
- Create the Apple Software Restore (asr) image by using DeployStudio
 - System Image Utility is ok and improving, but DeployStudio is a far better option
 - NetRestore has been discontinued

Deployment

Updating an Imaged Machine

- Once an image is deployed, how do you update it?
- You could re-image it later but this is destructive to any local data on the volume
- Use products like Munki, Radmin, Apple Remote Desktop, Puppet, Casper, Absolute Manage etc.
 - If you don't have a product already, seriously consider Munki

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

Mountain Lion

No DVD version

- Mountain Lion is only available from the App Store
 - Apple has a guide for how to deploy it in a managed environment
 - Basically get a code from Edu sales rep, redeem via AppStore, then run the installer on any machine
- NetInstall, NetBoot still supported in the same manner as with previous version

AppStore

- Tied to Apple ID - Consider whether you use University accounts, or individuals private accounts
- Work with vendors to acquire apps outside of store
- Apps in and out of store are not necessarily the same (TextWrangler)
- Consider volume purchasing, and MDM management of apps
- Look for improvements coming in Mavericks



Scripting and the CLI

Automating common tasks and saving you time while giving you more power

Scripting

Learn to love it!

- Provides a method of automation
- Saves you time and energy
- Saves you needing to remember what to do
- Repeatable
- Extremely powerful
- Plenty of help and pre-existing scripts available

Scripting

Learn to love it!

- OS X provides a lot of the functionality via the GUI but it is extended or in some cases only available via the CLI
- You can string commands together and manipulate the output
- You can run scripts on boot, login, logout, set intervals, and user driven
- There are endless possibilities...

Running Scripts on Boot

- LaunchD

 - /Library/LaunchDaemons

 - /Library/LaunchAgents

- SystemStarter

 - /Library/StartupItems

Login Hooks

Run Scripts on Login and Logout

- Login Hook

defaults write /var/root/Library/Preferences/

com.apple.loginwindow LoginHook /path/to/script

- Logout Hook

defaults write /var/root/Library/Preferences/

com.apple.loginwindow LogoutHook /path/to/script

Note: These are run as Root, not the user

Scripting

Notifying Users what is going on

- Scripts have no GUI - but at times, particularly if they are delaying the system during boot, login and logout, you may want to let the user know what is going on
- iHook is a way of providing a UI for a script
- Growl is also useful for providing notifications
 - Terminal-notifier does a similar task with notification centre on Mountain Lion or later



Hands On

Scripts with iHook - try `iHook Test.command`



Hands On

Scripts with Growl - try growl.sh



Hands On

Scripts with Terminal-notifier - try `notifier.sh`

CLI Commands

Running Commands

- There are multiple shells available, but I recommend using bash, which is the default shell
- Most command line tools will be installed in:-
`/usr/bin`, `/usr/sbin`, `/usr/local/bin`, and `/usr/local/sbin` but can be anywhere
- If the location is on your path you can Tab complete.
Type the first few characters and hit Tab

CLI Commands

Path Environment Variable

- To modify your path type

```
export PATH=$PATH:/new/path
```
- Or create `~/.bash_profile` and add the above line to it. It is searched in order of items
- To print current path use `echo $PATH`
- The `/usr/local/bin` and `/usr/local/sbin` aren't added by default so I recommend at least having

```
export PATH=$PATH:/usr/local/bin:/usr/local/sbin
```

CLI Commands

Getting Help

- The first step should always be to read the manual page
`man command` or `man -k term`
- Additionally running the command with `-h` or `--help` will normally print usage information
`command -h` or `command --help`
- To get a plain text version try
`man command | col -b > ~/command.txt`

CLI Commands

Commands

- `nano -w /path/to/file` - Text Editor
(if you use nano you **must** use the `-w` option)
- `defaults` and `plutil` - Manipulates Plists
- `system_profile` - Returns system information
- `touch` - creates an empty file
- `grep` - searches for a pattern
- `awk` - pattern scanning
- `rsync` - file synchronisation

CLI Commands

Some useful commands

- ssh, scp, sftp - Secure methods for working on remote machines
- hostname - Get hostname on machine
- top - show info on running processes
- ps - show currently running processes
- cp and mv - copy and move files
- open - open a file

CLI Commands

Some useful commands

- `sudo` - run a command as root
- `mount_*` - mount a remote file system
- `hdiutil` - work with disk images
- `update_dyld_shared_cache` - update caches
- list goes on and on....

CLI Commands

touch

- Touch will create a file if it doesn't exist, or update its modified time to the current time
- Useful for creating “flags” - little files that reflect a state of some sort
- I create flags for instructing scripts on what to do, and to reflect information like the fact that it's a managed machine
- We used a flag in the Packaging Example



Remote Access

Saves you time and money and lets you get home earlier

Remote Access

Your life blood. Don't leave home without it

- You **must** be able to access your managed machines remotely. Doesn't need to be publicly accessible but at least on the local subnet.
- It is too costly to visit each machine, and users have a tendency of turning a 5 minute trip into an hour
- Remote access leads to automation

Apple Remote Desktop

More powerful than just the Screen Sharing

- Apple Remote Desktop (ARD) is an awesome tool. It can collect system information, make changes, install software, send UNIX commands and much more to multiple machines.
- It also has VNC capabilities, allowing you to share and view screen sessions to assist a user over and above the built-in screen sharing

Apple Remote Desktop

Enabling

- System Preferences → Sharing → Remote Management

Configure the Access Privileges (Tip: Option Click next to a user will automatically select all options)

- Or via the CLI

```
sudo /System/Library/CoreServices/RemoteManagement/  
ARDAgent.app/Contents/Resources/kickstart -h
```

(for options and usage)

SSH

CLI Remote Access

- SSH allows you to run commands on a remote system
- Encrypted protocol so it is secure over the wire
- You can also do file (scp) and ftp (sftp) operations over the ssh protocol
- Can be configured for private / public key authentication
- Can be automated, particularly with keys

SSH

Enabling

- System Preferences → Sharing → Remote Logon
- Or via the CLI

```
sudo /usr/sbin/systemsetup -setremotelogin on
```

- Be aware that this will enable anybody that can log on to the machine via the login window to be able to login via ssh (including people in the AD or OD if configured)
 - Limit access as appropriate

SSH

Configuring and Securing

- Edit `/etc/sshd_conf`
- Recommend changes:
 - Protocol 2 - forces use of newer protocol
 - AllowUser <user>
 - If you have setup public / private keys disable password-based authentication
 - PasswordAuthentication no & UsePAM no



Hands On

Setting up SSH Keys

SSH

Creating the Public and Private Keys

- `ssh-keygen -t dsa`
- Hit enter to save it in the default location(`~/.ssh`)
- Enter a passphrase twice. Make it secure.
- This will create two files in `~/.ssh`. The public key is called `id_dsa.pub`, this is the key that you put onto the remote hosts. The private key is called `id_dsa`. Make sure that the private key is kept secure, it is now your "password" for accessing remote systems.

SSH

Deploying the Key

- Copy Public Key to Remote Machine

```
cd ~/.ssh; scp id_dsa.pub username@remotehost:~/id_dsa.pub
```

- Login to Remote Machine

```
ssh username@remotehost
```

- Activate Key

```
cd ~/.ssh (If .ssh doesn't exist then mkdir ~/.ssh; chmod 700 ~/.ssh)
```

```
touch authorized_keys; chmod 600 authorized_keys
```

```
cat ~/id_dsa.pub >> authorized_keys
```

```
rm ~/id_dsa.pub
```

SSH

Testing Key Deployment

- Log in to remote machine
`ssh username@remotehost`
- You should have logged in without being asked for the password. Keychain will manage the unlocking the keys for you in Snow Leopard or later

Conclusion and Questions

Recapping what we have covered and opening the floor to any outstanding questions

Conclusion

We have covered a lot...

- Terminology of MOEs
- Things to consider when planing a MOE
- The OS X file system and how to adapt it to your needs
- How to track changes to your system

Conclusion

We have covered a lot...

- Investigating and creating packages
- Briefly touched upon deployment
- Looked at scripting and CLI tools
- Covered remote access and ssh

Key Points

- When working with a MOE things need to be repeatable
- Document what you do, as you will need to refer to it later
- The command line is your friend
- Take MOEs one step at a time

Good Resources

- MacEnterprise and its mailing list
<http://macenterprise.org>
- AFP548
<http://afp548.com>
- Apple and some of their mailing lists
<http://www.apple.com> particularly the developer documentation (where the sysadmin stuff is)
- UniMacTech - AUC mailing list
<http://www.auc.edu.au/mailman/listinfo/unimactech>

Good Resources

- Your colleagues - every environment is different, but the problems are usually similar
- Bug Reporter - If you think you find a bug with OS X or any Apple product report it at <http://bugreporter.apple.com>
- Google (or the search engine of your choice) <http://www.google.com.au>



Questions

Additional Links

Tools that might be useful

- Pacifist

<http://www.charlessoft.com/>

- Iceberg

<http://s.sudre.free.fr/Software/Iceberg.html>

- Suspicious Package

<http://www.mothersruin.com/software/SuspiciousPackage/>

- fseventer

<http://www.fernlightning.com/doku.php?id=software:fseventer:start>

- Roaring Apps - Mountain Lion App Compatibility Crowd Sourced DB

<http://roaringapps.com/>

Additional Links

Tools that might be useful

- TextWrangler

<http://www.barebones.com/products/textwrangler/>

- Growl (fork)

<https://bitbucket.org/pmetzger/growl/downloads>

- iHook

<http://sourceforge.net/projects/ihook/>

- BatChmod

<http://www.lagentesoft.com/batchmod/index.html>

- MacTracker

<http://mactracker.ca/>

Additional Links

Tools that might be useful

- Munki

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

- Radmind

<http://radmind.org>

- InstaDMG

<http://afp548.com/forums/forum/software/instadmig/>

<http://code.google.com/p/instadmig/>

- DeployStudio

<http://www.deploystudio.com/Home.html>

- Absolute Manage InstallEase

<https://amrc.absolute.com/Evaluation.aspx?ie=1>