Session 1

'Scripting in Bash', Armin Briegel scriptingosx.com

bash 3.2 on macOS

bash provides:

- Interactive shell - scripting osx terminal primer
- scripting

Can use different shells. Best to script in bash because ubiquitous.

When can scripts be run: Scheduled login startup file events Triggers: launchd tricky but outset good util

ssh-installer allow you to install a package via SSH.

PolicyBanner; how to create a pop up

using pkgbuild to build a pkg on command line

/bin/bash is fine for administering Macs

$() is better that ` because can be nested and not deprecated

Traps to avoid:

Consider what you are assuming, make code defensive: 1) What PATH variable is set? set it explicitly for the script export PATH=/usr/bin:usr/local/bin:etc

2) Get the working directory, eg to ensure output created there: projectfolder=$(dirname "$0")

3) Always quote substituted variables

4) Put interesting variables at the top, easy to update

5) Conditionals should used [[ ]] not [ ] bash only can string together tests with && and ll more resilient string comparison with wildcards regex using =~

6) Check for null arguments if using script options: if [[-n $1]]; then... -n means "non null" More compact: projectfolder=${1:-"build"} if $1 is null, set to "build"

projectfolder=${1:?"No value!"} if $1 is null, error and exit

shellcheck.net will identify potential problems with script

Strict mode, Trace mode (use for debug not production): set -e fail on non zero command set -u fail on unset variables set -o pipefail fail if any element of a pipechain fails set -x / +x (to disable) lists all actually executed commands can run from int shell: bash -x

Version control every script

Session 2

'50/50', Duncan McKracken

50 tips in 50 minutes

- Central source of truth - eg. wiki
- Use an abstraction layer and don't change original data
- Don't panic, no one is going to die.
- Be methodical in your approach to everything
- Document your actions.
- Remember that changing something and changing it back is two actions, not one.
- Put it down. Accept you can't work tired; you may crack it in minutes next day. Tired you can make it worse.
- learn a high level language eg python. bash cannot be codesigned.
- Commands: time script.sh basename / dirname return filenmame and path of a file file tells you what a file is; may be surprising pkgutil --expand / --flatten strings : use to inspect binaries, find hidden options but be careful with these Strip off codesig from a config profile to confirm the payload contents (security cmd); can fix then re-codesign. Can use profiles command to manually install a config profile (eg if APN not available) delivered via pkg

find -group or -user root

find -newerct '5 minutes ago' way to list all changes eg during an install

use xpath to manipulate xml in bash

check for java without generating a pop-up /usr/libexec/java_home 2>/dev/null

Session 3

How to Maintain a moving target

This talk dissected a solution to encrypting a Mac in a specific situation (no LDAP and the evolution of the sysadmin's approach and the script itself. The general lesson was find an existing solution, then make it work for your situation and keep in mind that Apple will inevitably update the OS, so future-proof where

possible by avoiding deprecated OS components and review regularly.

Another important point was that no tool defines the scope of the options available; only the OS. No tool is perfect for everything.

Session 4

Introducing Munki 3.0

Open source project munki is the main software deployment tool in Orchard. 3.0 and 3.1 brings a number of changes, the most useful to us being:

UI updates:

- Limit the apps on the front screen to highlight the essential ones (previously was one big list)
- Software update notifications can be more/less intrusive

Workflow improvements for FileVault encrypted Macs:

- Improved support for software/OS updates requiring restarts, as these updates may not complete on FileVaulted Macs unless the user logs in or out respectively. To avoid this passwords can be cached across boots ("authenticated restarts") or automatic login after an OS update can be prevented ("enhanced bootstrapping"). In future macOS will involve frequent firmware updates so this could be useful.

Software:

- Native support for OS installer means less steps to deploy these.
- Unused applications can be automatically identified and uninstalled, useful to save licensing costs if these are paid apps. This could be combined with a post-install to release the licence from a pool.
- Can override Gatekeeper preventing unsigned apps from running, useful if a certificate expires and we want to temporarily allow an application to run while the vendor is updating the certificate.
- Can tag software as requiring a certain OS update to be installed, incentivising users to update.

SESSION 4 – MUNKI 3.0 <REG NEAGLE.

NEW FEATURES

1. NATIVE SUPPORT FOR INSTALLER APP.
   PREVIOUSLY HAD TO CONV. TO PKG.
2. AUTHENTICATED RESTARTS FOR FILEVAULT (e.g. PROMPTS FOR CACHED PASSWORD)
   (P1) CFG DETAIL: ENV MAC → STORE OS INSTALLER → PASSWORD
   [SET IN PREFS] → RESTARTS & INSTALLER BEGINS w/o ASKING FOR PWD.
3. NOTIFICATIONS, THEN FULL SCREEN POP UP FOR BLOCKING APP UPDATES.
4. REPO PLUGINS TO SUPPORT DIFF. STORAGE
   EG. AMAZON.
5. FEATURED ITEMS ON MSC FRONT PAGE.
   (AVOIDS USER OVERLOAD).

6/7 EXPIRED CERTS? OS (NO) ALLOW_UNTRUSTED REQ.

8. AUTO IDENTIFY UNUSED S/W (EG. SL).
   USED BY: <unused_sw_removal_info>
   – PACKAGE RECPT.
   – DAYS.
   * MARK AS 'UNATTENDED INSTALL' SO

USER NOT NOTIFIED ✓

* TO RECOVER LICENCES (OR DO OTHER POST INSTALL ACTIONS)
  USE A PACKAGE POST-INSTALL.
  ADOBE CCS — MUNKI MUNKIRUN (API)

9.
3.1 SUPPORTS MACOS H/S.
10. ENHANCED BOOTSTRAPPING:
* MUNKI CAN NOW IMMEDIATELY AFTER OPTIONAL DEFER USER LOGIN, WHEN FIRST AUTHED.
- AUN - ONCE.
- UPGRADE → USER UNLOCKS DISK → MUNKI → USER LOGIN
  NORMAL PATH.

11. ~~SEE~~ SOFTWARE/UPDATES THAT REQUIRE THE USER TO UPGRADE OS.
- CHROOT (IS STICK)

12. SUDO MUNKIMAC S/W UPDATE --config
LISTS WHERE PREFERENCES COME FROM.

SESSION 5 - OPEN SOURCE MDM
JESSE PETERSON.

1) MDM + DEP

2) microMDM.

↓

MDM PROTOCOL NOW PUBLIC FROM APPLE.

WHAT CAN MDM DO THAT OTHER TOOLS CAN'T?
(EG. CENTRAL ETC).

1. DEP BOOTSTRAPPING.
2. REMOTE WIPE
3. VPP APP DISTRO.
4. SSL (JAMF) BYPASS.

DEP = 3 COMPONENTS (AS SEEN IN APPLE'S CONSOLE)

1. deploy.apple.com

2. DEP API ← MOST HTTP BODIES JSON
? (AI FIND OUT)

3.
"PROFILE FETCH" SERVICE

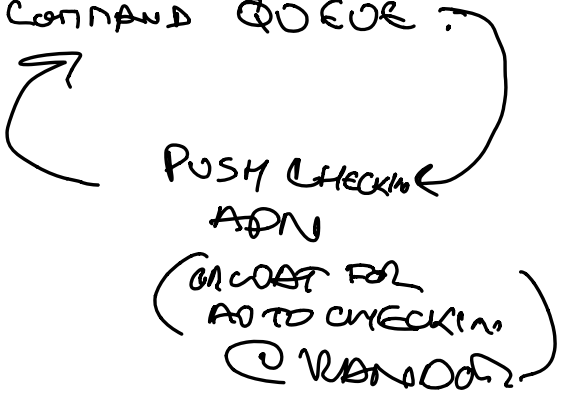
MDM PROTOCOL + SPEC

LOTS TECH DETAIL ; MAYBE USEFUL FOR DOC.

ENROLLMENT :

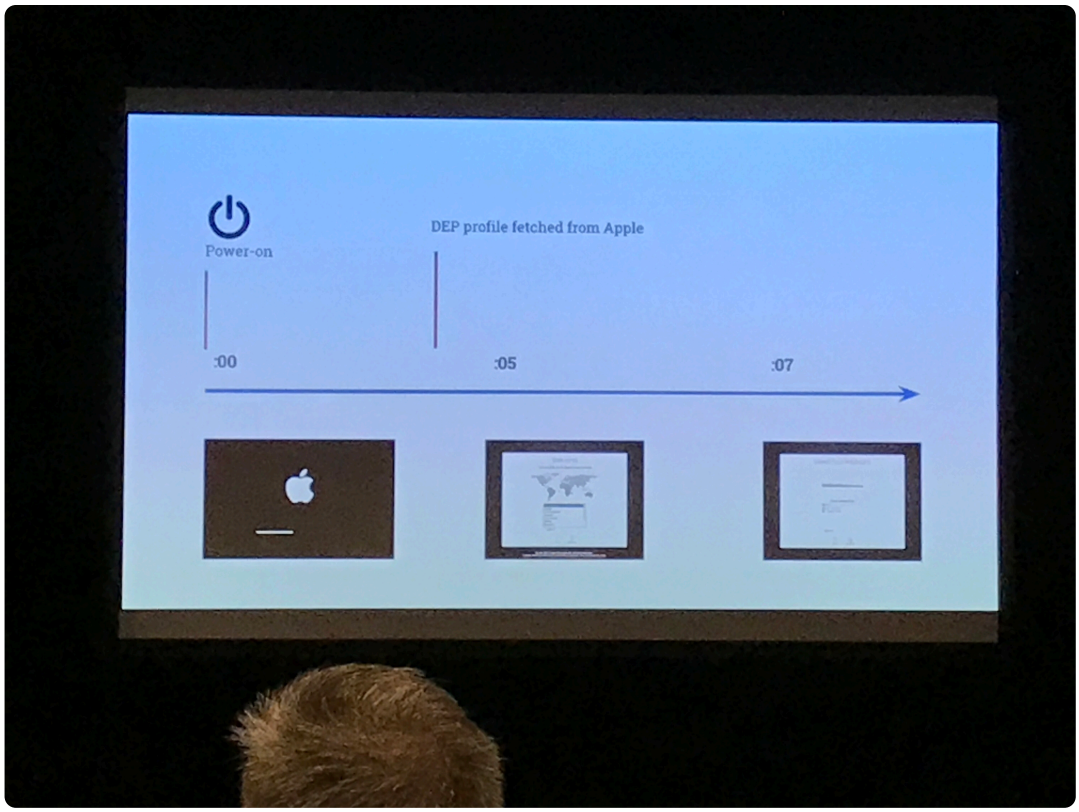
NEED A CONFIG PROFILE. - EXPLAINS WHAT 'DEVICE IDENTITY' IS (SCEP ETC).

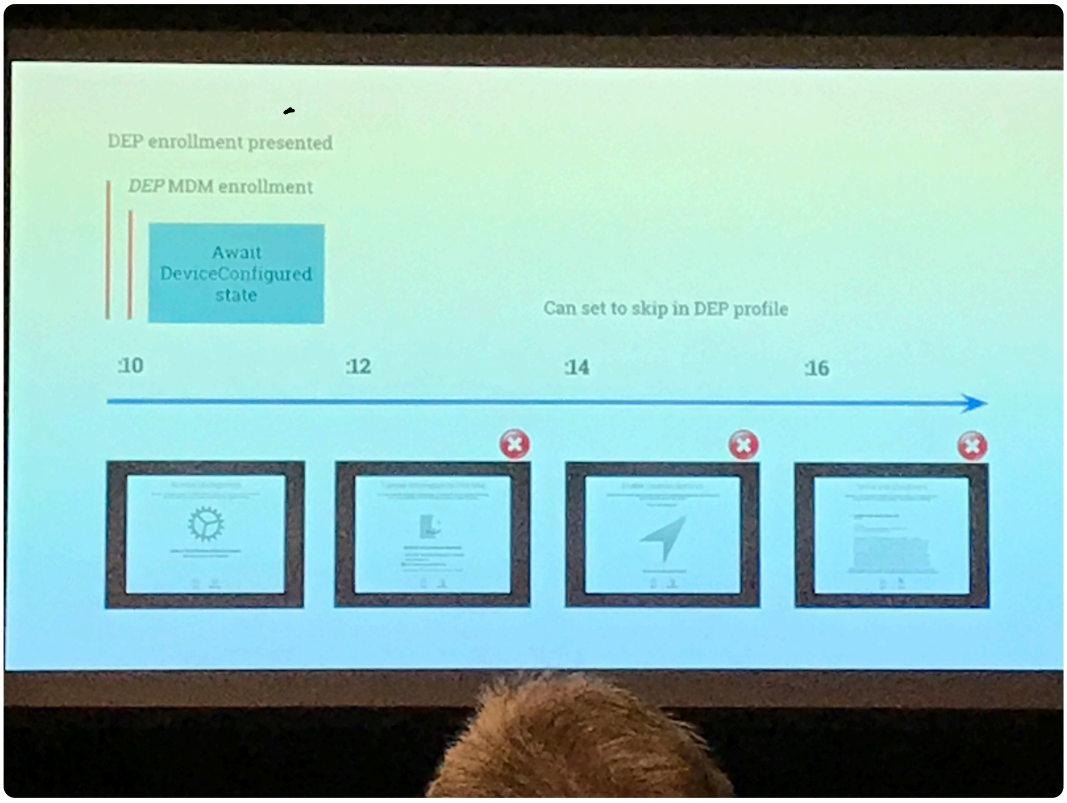
1. REQUEST ENROLLMENT
2. CHECKIN → ENROLLMENT COMPLETE.
3. PROCESSES COMMAND QUEUE :



COMMANDS : INSTALLAPPLICATION

- CURRENTLY ONLY USED TO INSTALL FOUNDATION'S PROPRIETARY AGENT.
- CAN BE USED TO INSTALL ANY PACKAGE.
- NEEDS MANIFEST


Power-on
DEP profile fetched from Apple
:00
:05
:07


DEP enrollment presented
DEP MDM enrollment
Await DeviceConfigured state
Can set to skip in DEP profile
:10
:12
:14
:16

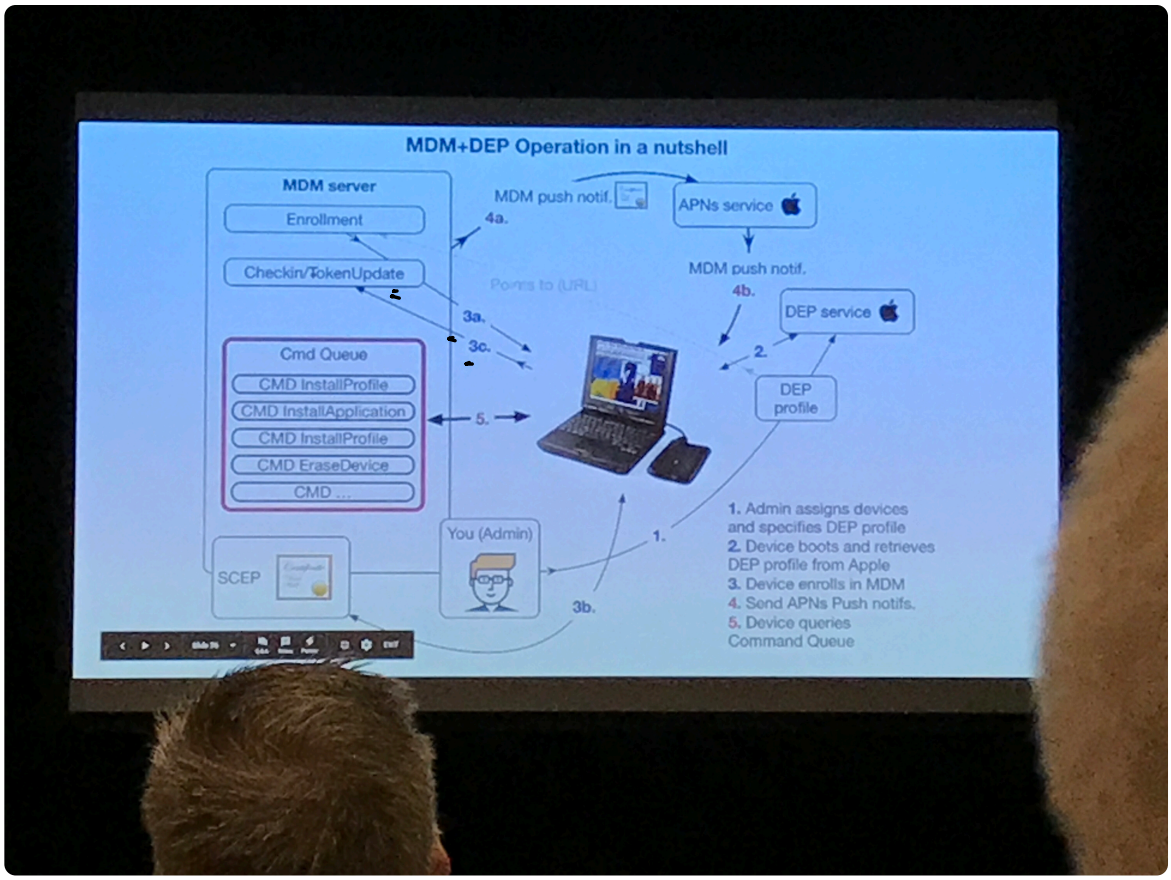
1. INSTALL APPLICATIONS (CASKING)
2. DEPNOTIFY (MACADMINS)

} ADD S/W INSTALL TO DEP WORKFLOW

INTRODUCE DEP TESTING.

@ rde

https://www.rderewianko.com/how-to-create-a-vm-thatll-work-with-dep-on-vmware-fusion/



Get started with installApplication from @erik http://blog.eriknicolasgomez.com/2017/03/08/Custom-DEP-Part-1-An-Introduction/

SESSION 6 — TIM SUTTON, SACCLETS.

UNIFIED LOGGING

"log" CMD "UNIFIED" = 1 TOOL ACROSS MACOS
IOS
TVOS
WATCHOS

PENNSTATE MACBRIDE 2017
'PEELING BACK THE DARK' CHILCOTE

https://www.youtube.com/watch?v=SQ_pWLgY6pU

WWDC 'LOG' VIDEO

https://developer.apple.com/videos/play/wwdc2016/721/

'Log' SPECIFICS

→ DATABASE, NO PLAIN TEXT. (ONE LOG + CONSOLE CAN READ)

→ NOT EVERYTHING PERSISTS ACROSS REBOOTS. (EG. DEBUG).

https://developer.apple.com/documentation/os/logging#overview

FUNS

log show --last 8h

" --predicate

GOOD INFOGRAPHIC ONE

CONSOLE IS 'LOG' TOOLS.

IMPORTANT QUOTING + CAPITALISATION NOTES.

'CONSOLIDATION' GUI FOR 'log' COMMAND.