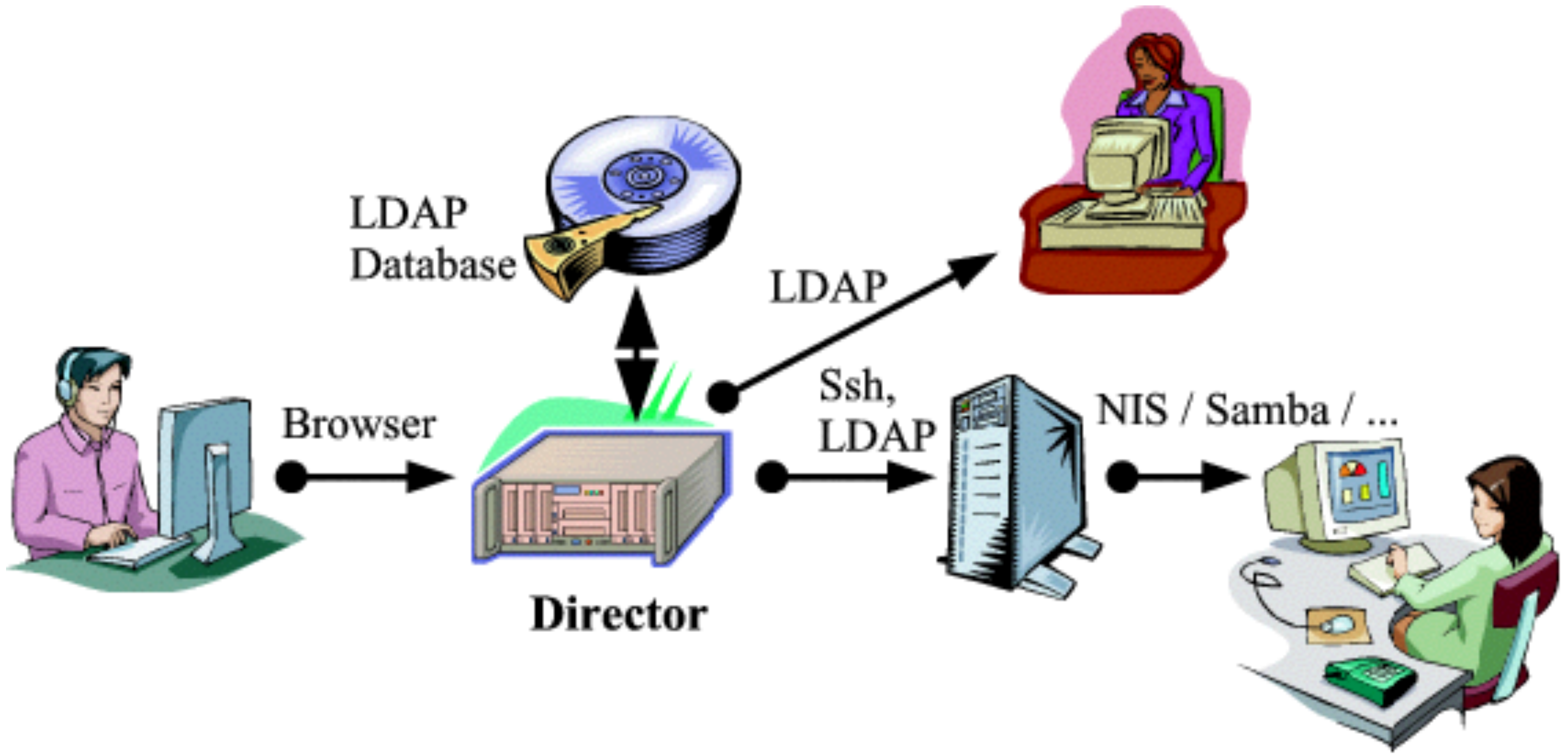
Tools and Process for Streamlining Mac Deployment

Tim Sutton
Concordia University, Faculty of Fine Arts
Montreal

Things change


LDAP Database
LDAP
Browser
Ssh, LDAP
NIS / Samba / ...
Director

Release cycle

- Annual releases of macOS, iOS
- Mid-cycle features added in iOS, Server (i.e. iOS 9.3 EDU features)
- Testing betas in summer

Process

Perspective

Tools

https://macops.ca/macsysadmin2016

~~OS X~~ macOS builds

~~OS X~~ macOS builds

- macOS (multiple versions, and choose your format)
- Techs with specific OS requirements, self service imaging
- Open source tools where possible
- Aim for repeatability, don’t repeat yourself

Products and ingredients

- macOS (start from the Installer app)
  - Restore image
  - Installer package
  - NetBoot image (System Image Utility, DeployStudio, Imagr, etc.)
- Bare minimum configuration (contained in packages)
  - Certs, basic config, binding
  - Local admin user, disabling setup assistant
  - Management tool / agent (Munki, Casper, etc.)

Single input, multiple outputs

AutoDMG
DeployStudio / Imagr
createOSXInstallPkg
PLIST
Munki
AutoDSNBI
PLIST
NetInstall / BSDPy

Jenkins

- Master server connected to multiple build machines (different OS / environments)
- Central configuration, credentials storage, role-based web access
- Publishing and post-processing of builds
- More in MacDevOps:YVR 2016 presentation:
    - http://www.macdevops.ca/presentations/

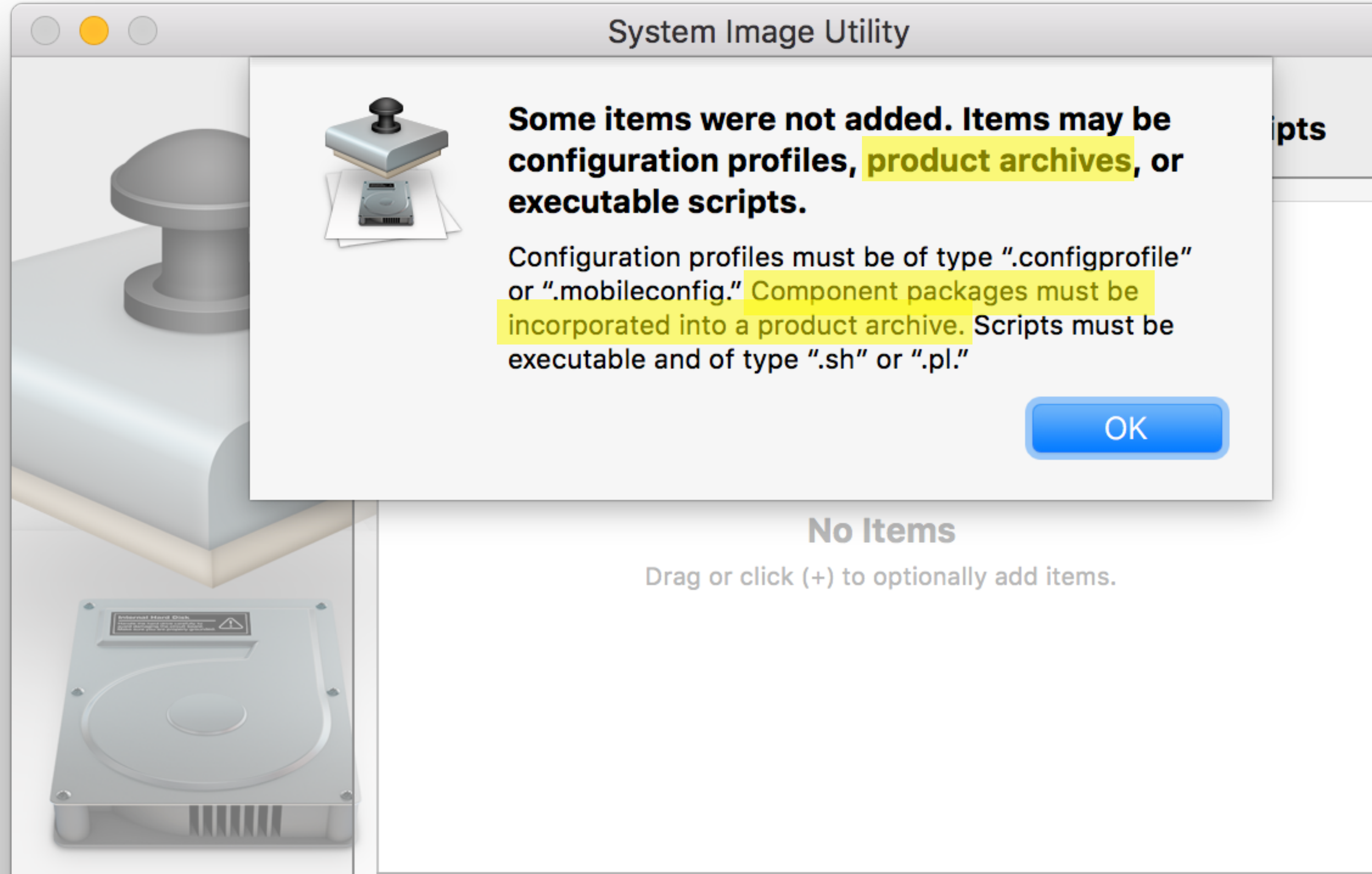
- pkgbuild and productbuild

```
pkgbuild \
  --identifier 'se.macsysadmin.DisableSetupAssistant' \
  --root /path/to/payload/directory/root \
  --version 2016.09.07 \
  'DisableSetupAssistant.pkg'
```

- Packaging tools / frameworks
  - munkipkg (CLI, Python script) - https://github.com/munki/munki-pkg
  - Luggage (CLI, make) - https://github.com/unixorn/luggage
  - Packages (Cocoa app and CLI) - http://s.sudre.free.fr

Product archive (distribution format)?

```
productbuild \
  --package component.pkg \
  output.pkg
```



System Image Utility

Some items were not added. Items may be configuration profiles, product archives, or executable scripts.

Configuration profiles must be of type ".configprofile" or ".mobileconfig." Component packages must be incorporated into a product archive. Scripts must be executable and of type ".sh" or ".pl."

Signed packages?

- Signed packages (`pkgutil --sign` or `productbuild --sign`)
  - Distribution outside your deployment tools
  - Installation via MDM + DEP (InstallApplication MDM command)
  - Possibly a requirement in the future?

ASR image, macOS package

```
/Applications/AutoDMG.app/Contents/MacOS/AutoDMG build \
  --output ~/somewhere \
  '/path/to/Install macOS.app' \
  disable_setup.pkg \
  create_user.pkg
```

```
createOSXInstallPkg \
  --source '/path/to/Install macOS.app' \
  --pkg disable_setup.pkg \
  --pkg munkitools.pkg \
  --pkg munki_config.pkg
```

NetBoot

- AutoNBI (creates a vanilla or custom NetInstall):
  - `AutoNBI.py -a -s /path/to/InstallESD.dmg -d . -n MyNBI`
- Imagr (vanilla NetInstall with Imagr and dependencies):
  - `make nbi`
- DeployStudio NBI:
  - https://github.com/MagerValp/AutoDSNBI (`autodsnbi.sh`)

Keep the code

- Keep the source together in a source repository (Git, etc.)
- Version your packages
- Share common logic across different scripts/projects
- Remove manual setup steps wherever possible (downloading, copying large required files, etc.)

Keep the code

```
➜ tree -L 2 .
.
├── jobs
|   ├── common.sh
|   ├── create_ds_nbi
|   ├── create_osx_install_pkg
|   └── create_restore_image
└── pkgs
    ├── build_all_pkgs.sh
    ├── create-itadmin
    ├── disable-diagnostics
    ├── disable-icloud-welcome
    ├── disable-setupassistant
    └── munki-kickstart
```

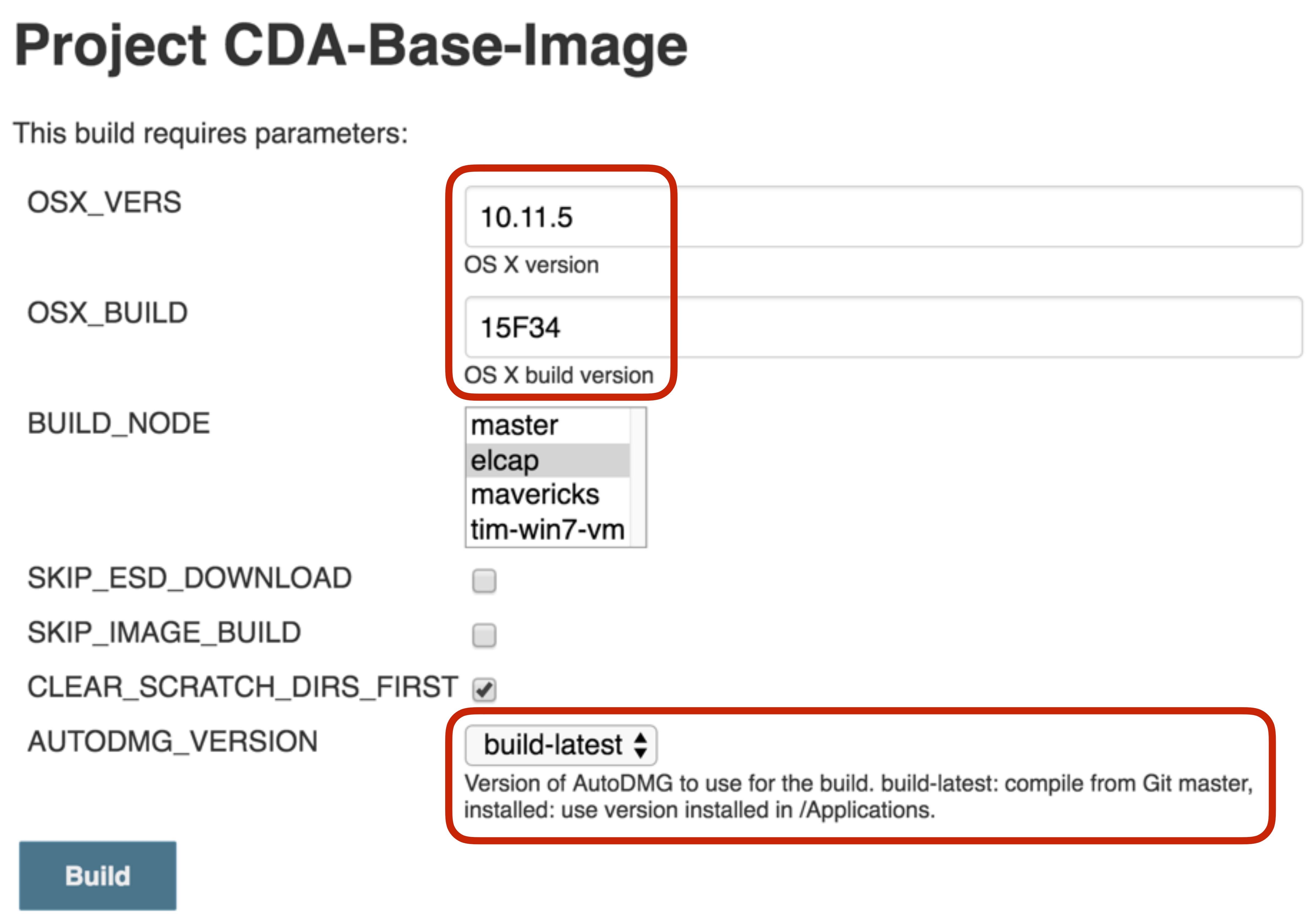
```
#!/bin/sh -e
# common.sh
# exports: CACHED_INSTALLOSX_DMG_PATH
#          PROPFILE

export CACHE_DIR="$HOME/Library/Caches/OSXBuilds"
if [ ! -d "${CACHE_DIR}" ]; then
  echo "Cache dir doesn't exist, creating at ${CACHE_DIR}"
  mkdir -p "${CACHE_DIR}"
fi

if [ -n "${OSX_VERS}" ] && [ -n "${OSX_BUILD}" ]; then
  # cache the installer app
  echo "Caching OS X installer app ${OSX_VERS} build ${OSX_BUILD}"
  mkdir -p "${CACHE_DIR}/InstallOSX"
  export CACHED_INSTALLOSX_DMG_PATH="${CACHE_DIR}/InstallOSX/${OSX_VERS}-${OSX_BUILD}.dmg"
  if [ ! -e "${CACHED_INSTALLOSX_DMG_PATH}" ]; then
    curl \
      --location \
      --output "${CACHED_INSTALLOSX_DMG_PATH}" \
      "http://macsysadmin.se/macos/${OSX_VERS}-${OSX_BUILD}.dmg"
  fi
fi

export PROPFILE="${WORKSPACE}/job.properties"
```

Jenkins pre-build screen



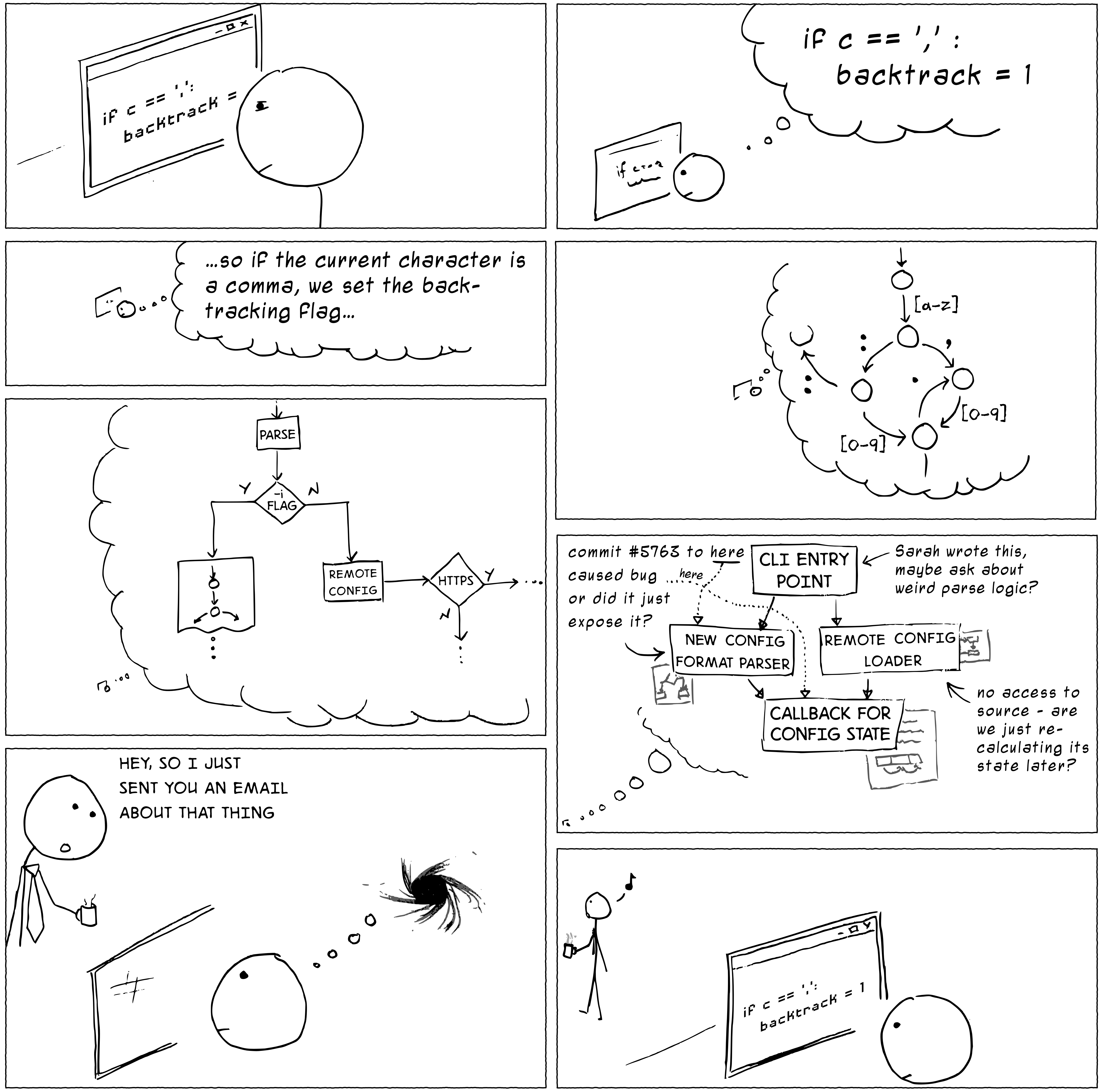
Project CDA-Base-Image

This build requires parameters:

OSX_VERS	10.11.5 OS X version
OSX_BUILD	15F34 OS X build version
BUILD_NODE	master elcap mavericks tim-win7-vm
SKIP_ESD_DOWNLOAD	☐
SKIP_IMAGE_BUILD	☐
CLEAR_SCRATCH_DIRS_FIRST	✓
AUTODMG_VERSION	build-latest Version of AutoDMG to use for the build. build-latest: compile from Git master, installed: use version installed in /Applications.

Save yourself the effort

- Details aren’t complicated but there are a lot of them – this is its own complexity
- We can rebuild any piece at any macOS version, with the latest config components
- Go on vacation – someone else can click “build”



if c == ',':
backtrack =
if c == ',' :
backtrack = 1
...so if the current character is a comma, we set the back-tracking flag...
[a-z]
,
.
:
[0-9]
[0-9]
PARSE
-i FLAG
Y
N
REMOTE CONFIG
HTTPS
Y
N
commit #5763 to here caused bug or did it just expose it?
here
CLI ENTRY POINT
Sarah wrote this, maybe ask about weird parse logic?
NEW CONFIG FORMAT PARSER
REMOTE CONFIG LOADER
CALLBACK FOR CONFIG STATE
no access to source - are we just re-calculating its state later?
HEY, SO I JUST SENT YOU AN EMAIL ABOUT THAT THING
if c == ',':
backtrack = 1

Making changes

Lab user environment

- Desktop Macs
- Domain users
- Not administrators
- Usually getting a “fresh” profile after logging in

User Template

```
rm -rf /System/Library/User\ Template/English.lproj/*

cp -R /Users/it_admin/* /System/Library/User\ Template/English.lproj/

chown -R root:wheel /System/Library/User\ Template/English.lproj

rm -f /System/Library/User\ Template/English.lproj/Library/Keychains/login.keychain
```

Directory Utility
File
Edit
Window
Help
Undo ⌘Z
Redo ⇧⌘Z
Cut ⌘X
Copy ⌘C
Paste ⌘V
Clear
Select All ⌘A
Change Root Password...
Enable Root User
Start Dictation fn fn
Emoji & Symbols ^⌘Space
Directory Utility
Services
Search Policy
Directory Editor
Apply

Select a service and click the pencil icon to edit settings.

Name
Active Directory
LDAPv3
NIS

Click the lock to prevent further changes.

```
#  Author:  Tim S
#  Created: 2011/04/23
include /usr/local/share/luggage/luggage.make
TITLE=Cyberduck
REVERSE_DOMAIN=ca.concordia.cda
PACKAGE_VERSION=4.0.2

CD_FILE=Cyberduck-${PACKAGE_VERSION}.zip
CD_URL=http://cyberduck.ch/${CD_FILE}
CD_PREF=ch.sudo.cyberduck.plist

PAYLOAD=\
    unpack-cd \
    pack-cd \
    pack-user-template-plist-${CD_PREF} \
    pack-user-template-appsupport-cd-bookmarks

unpack-cd: l_Applications
      curl -L ${CD_URL} -o ${CD_FILE}

pack-cd: l_Applications
      @sudo ${DITTO} -x -k --noqtn ${CD_FILE} ${WORK_D}/Applications/

pack-user-template-appsupport-cd-bookmarks: l_System_Library_User_Template_Application_Support_Cyberduck_Bookmarks
       # each .duck file is a bookmark - login user and UUID keys can be left
       # empty, login user will use the logged-in user's shortname and UUID will
       # self-generate and rename the .duck file in the user's profile
       @sudo ${CP} cdaftp-bookmark.duck ${USER_TEMPLATE_APPLICATION_SUPPORT}/Cyberduck/Bookmarks
```

Date: Fri, 06 May 2011 09:40:23
From: Rob Middleton
To: munki-dev@googlegroups.com
Subject: Re: [munki-dev] User Filler Postflight Script

[…],

The generic way this is solved for all possibilities is the use of a custom LaunchAgent (with plist at /Library/LaunchAgents/).

When any user logs in to the local system their home directory is first mounted then LaunchAgents run as configured in the security context of the logged in user with their home directory accessible. In this way you can script initial install of items or repair of missing items in the user context.

I no longer use methods which do not work in all circumstances.

Rob.

“In all circumstances”

- Context of one device: does it matter whether…
  - A user is already logged in?
  - The system or component you’re managing has been run before, or is running currently?
  - The user or device needs to also have some license / entitlement?
  - (iOS): Restrictions are set? Devices is supervised and/or DEP?

“In all circumstances”

- Context of the fleet of devices:
  - What do you know about how things worked up to now?
  - What change would be backwards compatible with every other element on the system which could be affected?
- Allows for smaller but more frequent, and less invasive, changes
- A change can be not perfect, but still an improvement and with a way forward

Admin user

- Packaged using CreateUserPkg (Per Olofsson, packaged user technique by Greg Neagle)
- …not all our systems had this user created by the package, though
- UID 501, 502, 503…

Admin user

- Problem #1: The user package seems like the best way to deploy and/or update the user, but it depends on the user having a consistent UID on all machines, which for us it wasn’t.
- Solution: Customize the package script so it can also handle the situation of the user existing already.

Admin user

```
#!/bin/bash
# preinstall

USER_SHORTNAME=itadmin

uid_stored="$3/private/tmp/.create_user_uid"
user_plist="$3/private/var/db/dslocal/nodes/Default/users/$USER_SHORTNAME.plist"
if [ -e "$user_plist" ]; then
    DISCOVERED_UID=$(/usr/libexec/PlistBuddy -c 'Print :uid:0' "$user_plist")
    echo "Discovered existing user $USER_SHORTNAME, uid $DISCOVERED_UID"
    echo "Storing uid in $uid_stored.."
    echo "$DISCOVERED_UID" > "$uid_stored"
fi
```

Admin user

```
#!/bin/bash
# postinstall

USER_SHORTNAME=itadmin

uid_stored="$3/private/tmp/.create_user_uid"
user_plist="$3/private/var/db/dslocal/nodes/Default/users/$USER_SHORTNAME.plist"

# (most of postinstall script generated by CreateUserPkg goes here)

# Check for our temporary uid that was created if the user already existed
if [ -e "$uid_stored" ]; then
    preexisting_uid=$(cat $uid_stored)
    /usr/libexec/PlistBuddy -c 'Delete :uid' "$user_plist"
    PlistArrayAdd "$user_plist" uid "$preexisting_uid"
    rm "$uid_stored"
fi
# (service restart section of postinstall script)
```

Admin user

- Problem #2: We don’t want Munki to install this user on every single machine, but we want to manage the user pkg if it is installed. How could Munki know when to install our user package?

- Solution: Give Munki the ability to know about local users so we can determine whether to install this package.

Admin user

```
<plist version="1.0">
<dict>
[………]
  <key>user_local_list</key>
  <array>
    <string>com.apple.calendarserver</string>
    <string>daemon</string>
    <string>Guest</string>
    <string>nobody</string>
    <string>root</string>
    <string>itadmin</string>
    <string>tim</string>
  </array>
[………]
</dict>
</plist>
```

ConditionalItems.plist

```
<plist version="1.0">
<dict>
<key>conditional_items</key>
  <array>
    <dict>
      <key>condition</key>
      <string>'itadmin' IN user_local_list</string>
      <key>managed_installs</key>
      <array>
        <string>itadmin_user</string>
      </array>
    </dict>
  </array>
</dict>
</plist>
```

Munki manifest

Reality

Toyota Production System (TPS)

- Jidoka (Autonomation)
  - “Automation with a human touch”
- Do not allow for defects to make it into the finished product
- The cost and effort to go out and fix a problem in a “product” (i.e. a change, update, etc.) is higher than that spent on improving quality control

Taiichi Ohno

Expecting failure

- Automation can help almost any process by communicating between different systems and databases
- Internal: LDAP/AD/identity, inventory, JSS, Code42, Zentral
- External: Creative Cloud user licenses, cloud-based ticket / support tracker

Expecting failure

- Great power, great responsibility
- External dependencies
- Internal edge cases
- Also, bugs

AutoPkg Recipe Repos

```
➜  ~ autopkg search malware

Name                                        Repo                        Path
----                                        ----                        ----
KnockKnock.munki.recipe                     andrewvalentine-recipes   KnockKnock/KnockKnock.munki.recipe
MalwarebytesAntiMalware.munki.recipe        aysiu-recipes             MalwarebytesAntiMalware/MalwarebytesAntiMalware.munki.r
VirusBarrier2013.munki.recipe               dankeller-recipes         VirusBarrier2013/VirusBarrier2013.munki.recipe
VirusBarrierX6.munki.recipe                 dankeller-recipes         VirusBarrierX6/VirusBarrierX6.munki.recipe
KnockKnock.munki.recipe                     jps3-recipes              Objective-See/KnockKnock.munki.recipe
KnockKnock.download.recipe                  jps3-recipes              Objective-See/KnockKnock.download.recipe
RansomWhere.munki.recipe                    jps3-recipes              Objective-See/RansomWhere.munki.recipe
BlockBlock.munki.recipe                     jps3-recipes              Objective-See/BlockBlock.munki.recipe
MalwarebytesAntiMalware.jss.recipe          rtrouton-recipes          JSS/MalwarebytesAntiMalware.jss.recipe
MalwarebytesAntiMalware.install.recipe      rtrouton-recipes          MalwarebytesAntiMalware/MalwarebytesAntiMalware.install
MalwarebytesAntiMalware.download.recipe     rtrouton-recipes          MalwarebytesAntiMalware/MalwarebytesAntiMalware.downloa
MalwarebytesAntiMalware.pkg.recipe          rtrouton-recipes          MalwarebytesAntiMalware/MalwarebytesAntiMalware.pkg.rec
DetectX.jss.recipe                          rtrouton-recipes          JSS/DetectX.jss.recipe
```

AutoPkg Recipe Repos

https://github.com/autopkg/autopkg/blob/master/Scripts/setup_new_recipe_repo.py

```
➜  ~ ./setup_new_recipe_repo.py timsutton/autopkg-recipes

Here's what's going to happen: the repo at 'timsutton/autopkg-recipes'
will be be cloned and pushed to a new repo at 'autopkg/timsutton-recipes'.
A new team, 'timsutton-recipes', will be created with access to this repo,
and the GitHub user 'timsutton' will be added to it with 'admin' rights.

Type 'yes' to proceed:
```

AutoPkg Recipe Repos

- Script options, GitHub token validity
- Additional user option required if submitting repo is owned by an org and not a user
- Existence of GitHub user given
- Existence of source GitHub repo
- Existence of proposed new GitHub repo
- Existence of proposed new team
- Check API call to create the repo
- Check API call to create new team
- Check API call to modify default team user assignment, add new team member
- Bare git clone the source repo, mirror push it to the new destination

Script errors

- Bash scripting?
    - set -u: exit if any $variables are not set
    - set -e: exit shell on any non-zero exit
    - set -o pipefail: non-zero exits within pipes return that exit code as well
- Python / Ruby / Swift / Go / etc?
    - Anticipate and handle errors

Tack!

@timsutton

@tvsutton

https://macops.ca/macsysadmin2016