Syracuse University

Penn State MacAdmins Conference, 2018

Enrollment, Bootstrapping, and the Munki Barrel. A University-wide approach.

Timothy Schutt

Computer Consultant and
Mac Environments Tech Team lead.

July 12, 2018

Who am I...

...and how did I end up here?

Timothy Schutt (pronounced 'skut')
Music major in college (early '90s)
Switched to IT immediately upon leaving college
Independent consultant in Rochester, NY - came to Syracuse University in 2005

@binkleybloom most places - Twitter, GitHub, Keybase, Reddit, Untappd

Presentation details & related code at https://github.com/binkleybloom/...

Presentation goals

This is a case study

Topics I plan on covering:

- First repo, grass-roots effort, and lessons learned
- Expansion to a campus-wide framework
- How we accommodate Apple's new "Recommendations"

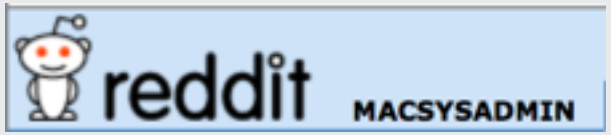
Not presentation goals...

Resources for further learning

I'm not going in to great detail on individual elements.
Others already have (and done a better job than I could).

Find them on...

MacAdmins.slack.com

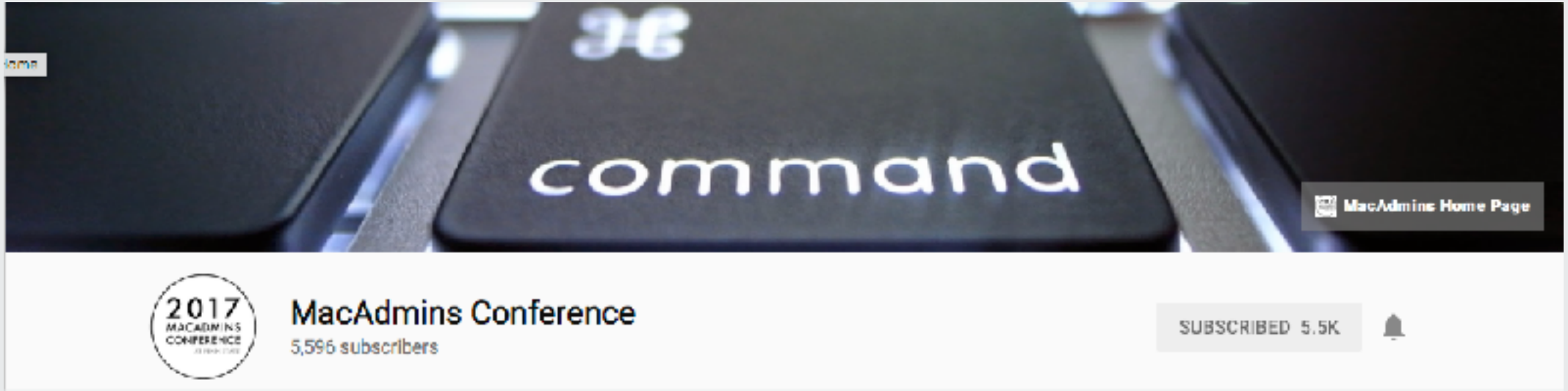


Not presentation goals...

Resources for further learning

I'm not going in to great detail on individual elements.
Others already have (and done a better job than I could).

Find them on...



Back to the start…

Careful what you put online, kids...

Back to the start…

Early Deployment Workflow

Building the foundation.

- Boot into DeployStudio
- Enter machine name & initial account information
- Block-copy vanilla OS image
- Install Munki tools & generic ManagedInstalls prefs
- Create the machine's manifest in Munki Web Admin
- Upon reboot, set ClientIdentifier by hand

dat command line interface

Time to grease the skids...

It's rather awkward, y'know...

```
$ defaults write /Library/Preferences/ManagedInstalls.plist ClientIdentifier <manifestName>
```

... because not everyone is comfortable in the terminal.

So, let's make it better.

First optimization

Set the ClientID for the IT Support person who is deploying the machine.

```bash
#! /bin/bash

# Checks for clientid value in three places:
#   1 - Custom preference: "MUNKIID"
#   2 - ARD Text4 field
#   3 - Computer name field
#
# setting it upon first value found.
# Tim Schutt, Syracuse University, 2013
# taschutt@syr.edu

echo -e " \n============= Setting Munki Client ID ==================="


TEXTID=$(defaults read "/Volumes/${DS_LAST_SELECTED_TARGET}/Library/Preferences/com.apple.RemoteDesktop" Text4)
echo -e " \n \
MUNKIID Custom Pref Value: ${CLIENTID}\n\
          ARD Text4 value: $TEXTID\n\n\
            Computer Name: $DS_COMPUTERNAME \n"

if [ ${MUNKIID} ]
then
    echo -e "\n     CLIENTID preference exists  in Deploystudio Workflow. \n"
    VALUE="${CLIENTID}"
elif [ $TEXTID ]
then
    echo -e "\n     No CLIENTID value found. ARD Text4 Field exists. \n"
    VALUE=$TEXTID
else
    echo -e "\n     No CLIENTID value or ARD Text4 value found. Setting ClientID to Computer Name. \n"
    VALUE=$DS_COMPUTERNAME
fi

defaults write "/Volumes/${DS_LAST_SELECTED_TARGET}/Library/Preferences/ManagedInstalls" ClientIdentifier "$VALUE"

echo "     Munki ClientIdentifier has been successfully set to '$VALUE'"
echo -e " \n========= Finished Setting Munki Client ID ================="
echo -e " \n "
exit 0
```

Look in three places for a value.
Set ClientID to the first one found.

1. DeployStudio Pref: "CLIENTID"
2. Value set in ARD Field #4
3. Computer Name

Hint #1:

Eat your own dogfood. Season it to taste.

Adoption is driven first by ease-of-use

Eat your own dog food. Season it to taste.

An unused management framework is of no value. More than simple technical challenges, you must streamline and optimize the overall experience for the IT Support people who will use this.

Step through the process repeatedly. The things that irritate you will irritate others. Get rid of as many as possible.

Test was successful.

Time to scale to campus.

Expanding the system

From "Shared between friends" to "Standard in the cul-de-sac"

To expand campus wide, several key pieces had to fall in to place.

- Separate each "IT group" to their own interface, resulting in:
  - Smaller number of "scoped" manifests, particular to their org.
- Org specific config (MunkiReport machine group settings).
- Better bootstrapping support for "Best Practices".

Introducing the Munki Barrel

What is a Munki Barrel?

A blend of the common and the unique.

A Munki Barrel is a Munki repository that combines common, centrally supplied and managed resources with unique, departmentally exclusive and manageable elements within a distributed support group's macOS management stack.

What makes a Munki Barrel?

The common elements

- pkgs / pkgsinfo / catalogs / icons / client_resources
- Key manifests
  - site_default_UniversitySource
  - Specific bundle manifests
    - BND-SelfServe / BND-ADBoundMacs / BND-InternetPlugins

The unique

- All other manifests
- An exclusive Munki Web Admin instance

No, seriously though... what MAKES a Munki Barrel?

tl; dr.

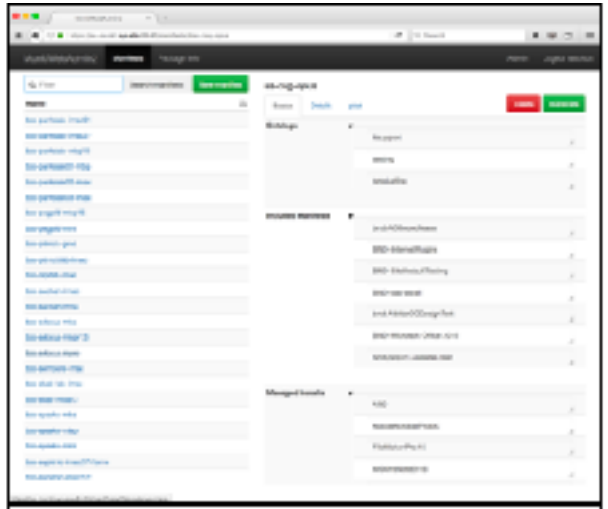
Create a virtual host for the repo & MunkiWebAdmin. Then...

```
$ ln -s <common repo element> <barrel element>
```

Security much? I mean, MunkiWebAdmin2 can muck about with pkginfo files...

Sure - let's put the central stuff on a separate server.
In a private subnet.
Available via a read-only nfs link.

Munki Web Admin 2
Port 8443



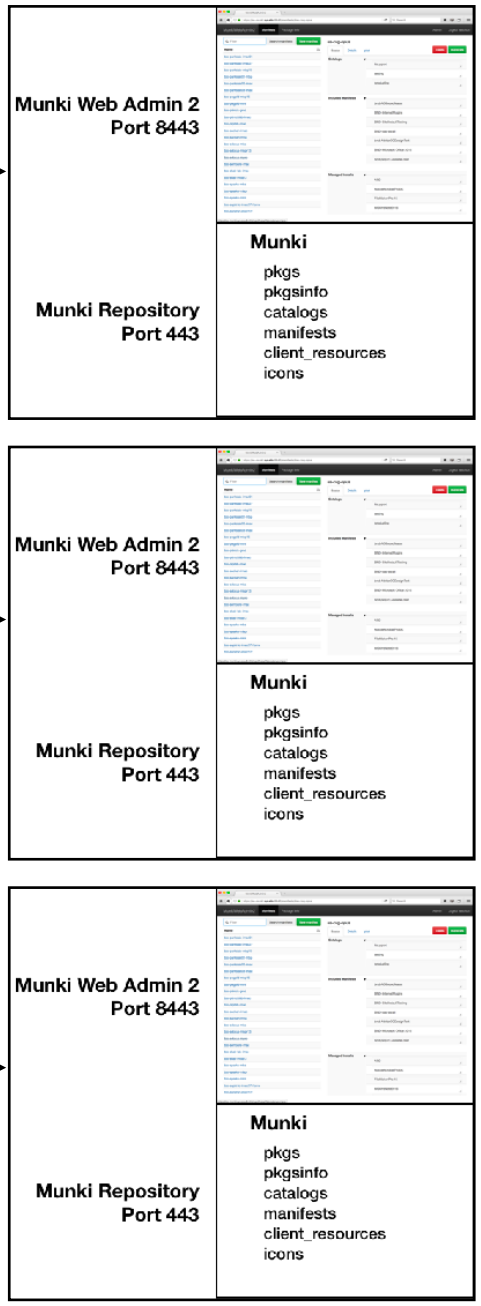
Munki Repository
Port 443

Munki

pkgs
pkgsinfo
catalogs
manifests
client_resources
icons

University-Wide Resources

pkgs
pkgsinfo
catalogs
icons
client_resources



University-Wide Resources

pkgs
pkgsinfo
catalogs
icons
client_resources

Munki Web Admin 2
Port 8443

Munki Repository
Port 443

Munki
pkgs
pkgsinfo
catalogs
manifests
client_resources
icons

Munki Web Admin 2
Port 8443

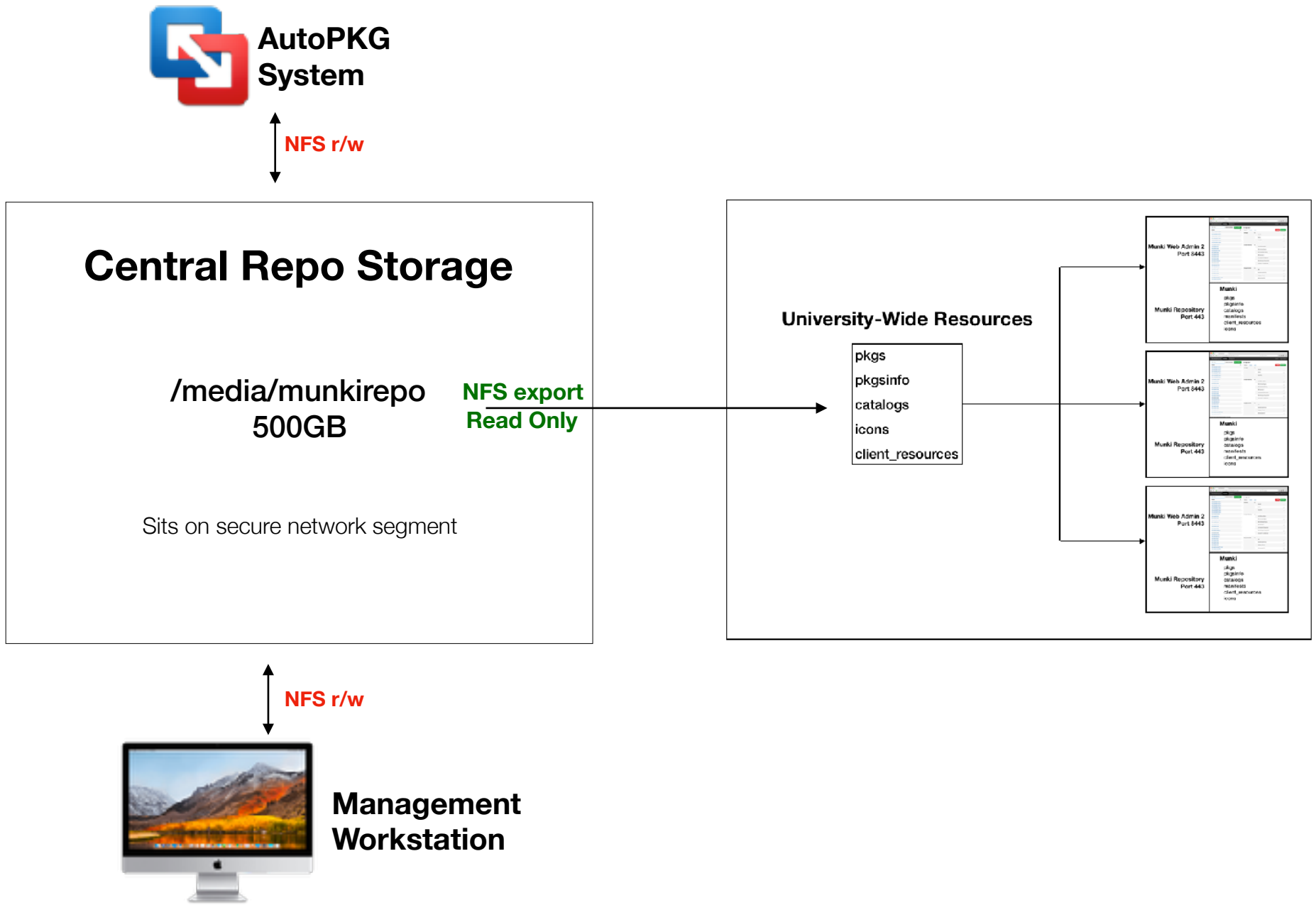
Munki Repository
Port 443

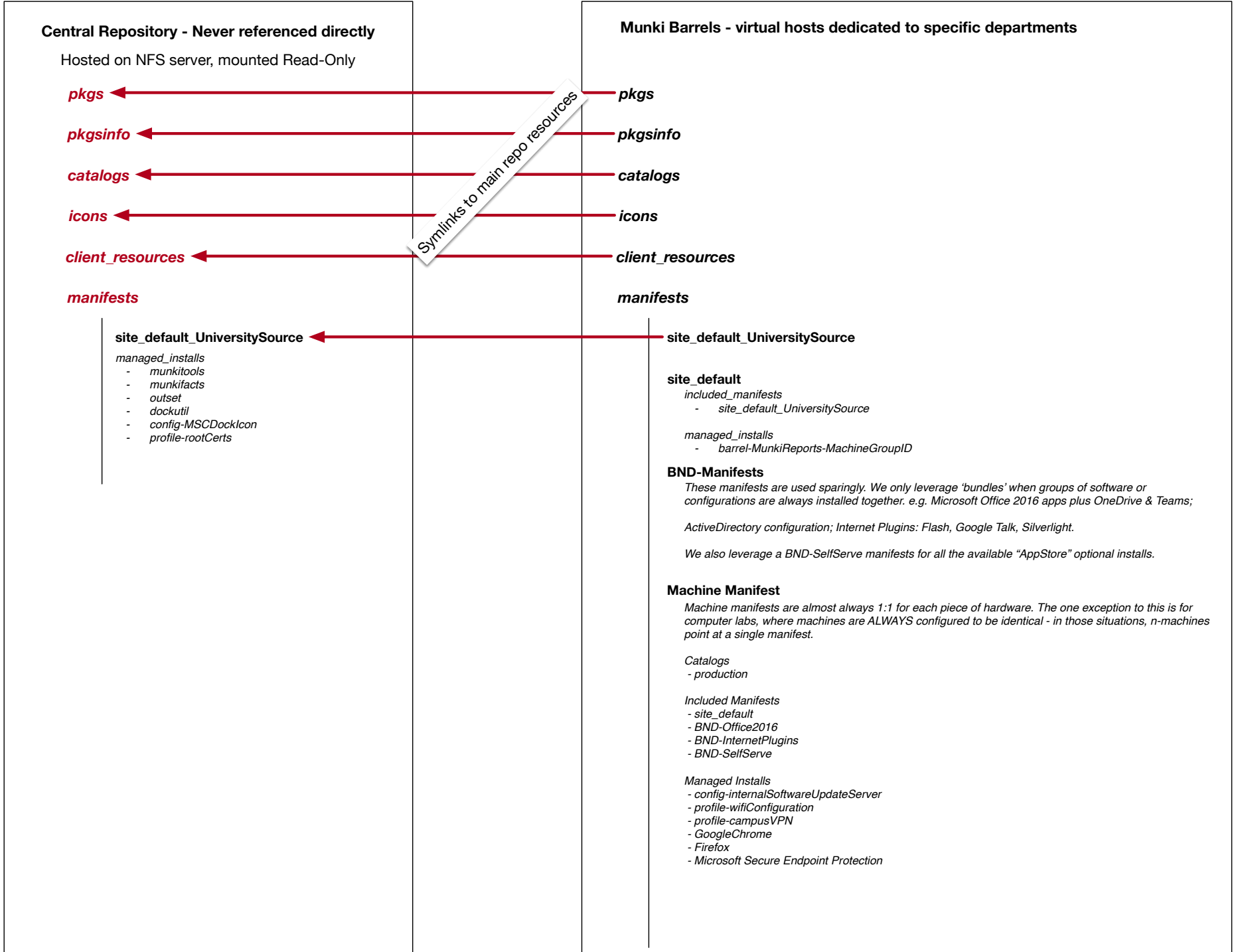
Munki
pkgs
pkgsinfo
catalogs
manifests
client_resources
icons

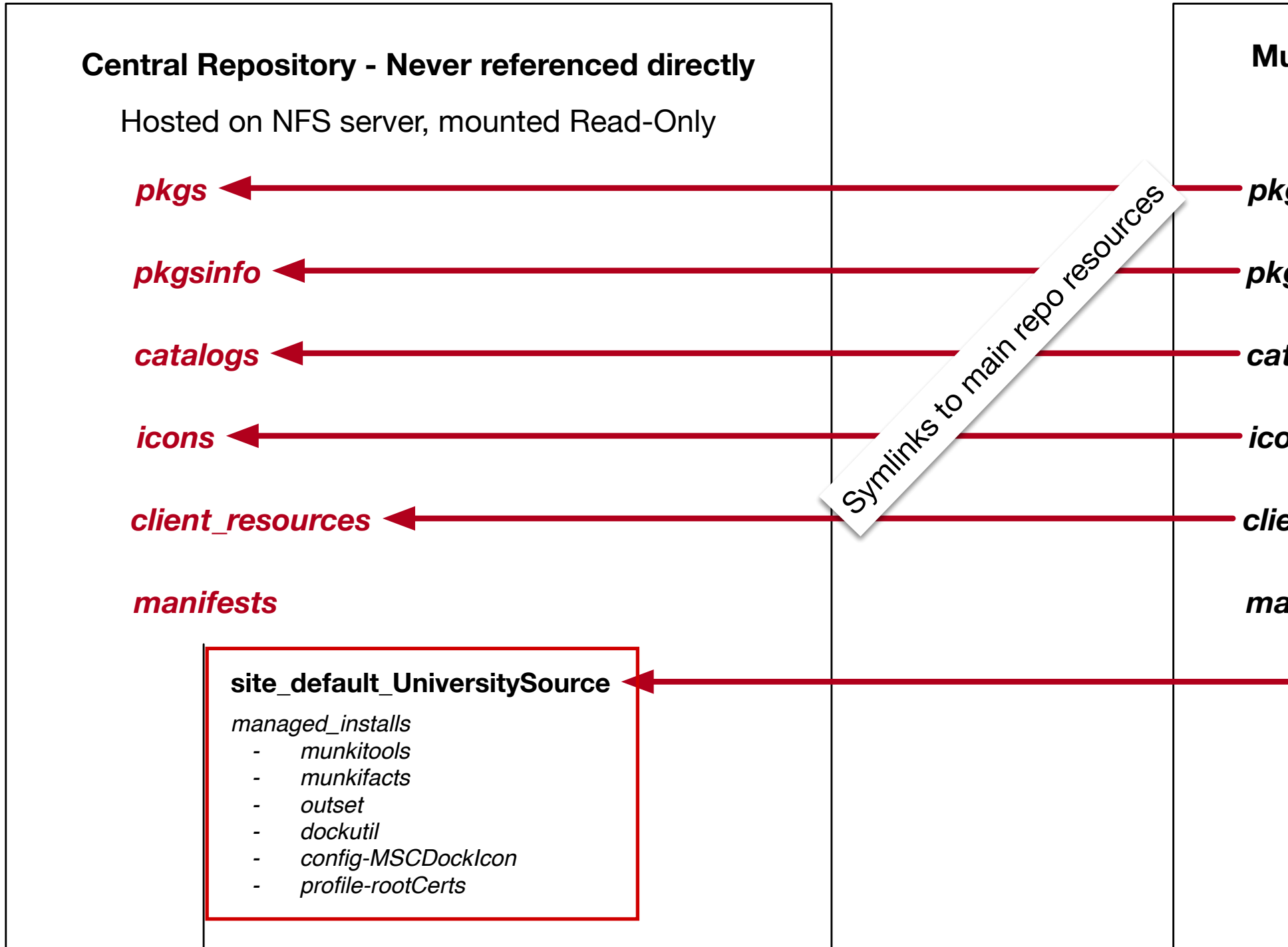
Munki Web Admin 2
Port 8443

Munki Repository
Port 443

Munki
pkgs
pkgsinfo
catalogs
manifests
client_resources
icons


AutoPKG System
NFS r/w
Central Repo Storage
/media/munkirepo
500GB
NFS export
Read Only
Sits on secure network segment
University-Wide Resources
pkgs
pkgsinfo
catalogs
icons
client_resources
NFS r/w
Management Workstation


Central Repository - Never referenced directly
Hosted on NFS server, mounted Read-Only
pkgs
pkgsinfo
catalogs
icons
client_resources
manifests
site_default_UniversitySource
managed_installs
- munkitools
- munkifacts
- outset
- dockutil
- config-MSCDockIcon
- profile-rootCerts
Symlinks to main repo resources
Munki Barrels - virtual hosts dedicated to specific departments
pkgs
pkgsinfo
catalogs
icons
client_resources
manifests
site_default_UniversitySource
site_default
included_manifests
- site_default_UniversitySource
managed_installs
- barrel-MunkiReports-MachineGroupID
BND-Manifests
These manifests are used sparingly. We only leverage 'bundles' when groups of software or configurations are always installed together. e.g. Microsoft Office 2016 apps plus OneDrive & Teams;
ActiveDirectory configuration; Internet Plugins: Flash, Google Talk, Silverlight.
We also leverage a BND-SelfServe manifests for all the available "AppStore" optional installs.
Machine Manifest
Machine manifests are almost always 1:1 for each piece of hardware. The one exception to this is for computer labs, where machines are ALWAYS configured to be identical - in those situations, n-machines point at a single manifest.
Catalogs
- production
Included Manifests
- site_default
- BND-Office2016
- BND-InternetPlugins
- BND-SelfServe
Managed Installs
- config-internalSoftwareUpdateServer
- profile-wifiConfiguration
- profile-campusVPN
- GoogleChrome
- Firefox
- Microsoft Secure Endpoint Protection


Central Repository - Never referenced directly
Hosted on NFS server, mounted Read-Only
pkgs
pkgsinfo
catalogs
icons
client_resources
manifests
site_default_UniversitySource
managed_installs
- munkitools
- munkifacts
- outset
- dockutil
- config-MSCDockIcon
- profile-rootCerts
Symlinks to main repo resources

Munki Barrels - virtual hosts dedicated to specific departments

n repo resources

pkgs

pkgsinfo

catalogs

icons

client_resources

manifests

site_default_UniversitySource

site_default

included_manifests

- *site_default_UniversitySource*

managed_installs

- *barrel-MunkiReports-MachineGroupID*

BND-Manifests

manifests

site_default_UniversitySource

site_default

included_manifests
- *site_default_UniversitySource*

managed_installs
- *barrel-MunkiReports-MachineGroupID*

BND-Manifests

These manifests are used sparingly. We only leverage 'bundles' when groups of software or configurations are always installed together. e.g. Microsoft Office 2016 apps plus OneDrive & Teams;

ActiveDirectory configuration; Internet Plugins: Flash, Google Talk, Silverlight.

We also leverage a BND-SelfServe manifests for all the available "AppStore" optional installs.

Machine Manifest

Machine manifests are almost always 1:1 for each piece of hardware. The one exception to this is for computer labs, where machines are ALWAYS configured to be identical - in those situations, n-machines point at a single manifest.

manifests

site_default_UniversitySource

site_default

included_manifests

- *site_default_UniversitySource*

managed_installs

- *barrel-MunkiReports-MachineGroupID*

BND-Manifests

These manifests are used sparingly. We only leverage 'bundles' when groups of software or configurations are always installed together. e.g. Microsoft Office 2016 apps plus OneDrive & Teams;

ActiveDirectory configuration; Internet Plugins: Flash, Google Talk, Silverlight.

We also leverage a BND-SelfServe manifests for all the available "AppStore" optional installs.

Machine Manifest

Machine manifests are almost always 1:1 for each piece of hardware. The one exception to this is for computer labs, where machines are ALWAYS configured to be identical - in those situations, n-machines point at a single manifest.

manifests

site_default_UniversitySource

site_default

included_manifests
- *site_default_UniversitySource*

managed_installs
- *barrel-MunkiReports-MachineGroupID*

BND-Manifests

These manifests are used sparingly. We only leverage 'bundles' when groups of software or configurations are always installed together. e.g. Microsoft Office 2016 apps plus OneDrive & Teams;

ActiveDirectory configuration; Internet Plugins: Flash, Google Talk, Silverlight.

We also leverage a BND-SelfServe manifests for all the available "AppStore" optional installs.

Machine Manifest

Machine manifests are almost always 1:1 for each piece of hardware. The one exception to this is for computer labs, where machines are ALWAYS configured to be identical - in those situations, n-machines point at a single manifest.

Machine Manifest

Machine manifests are almost always 1:1 for each piece of hardware. The one exception to this is for computer labs, where machines are ALWAYS configured to be identical - in those situations, n-machines point at a single manifest.

Catalogs
- *production*

Included Manifests
- *site_default*
- *BND-Office2016*
- *BND-InternetPlugins*
- *BND-SelfServe*

Managed Installs
- *config-internalSoftwareUpdateServer*
- *profile-wifiConfiguration*
- *profile-campusVPN*
- *GoogleChrome*
- *Firefox*
- *Microsoft Secure Endpoint Protection*

Hint #2:

You can paint yourself in to a corner with included manifests. Use them sparingly.

Let's talk about those machine manifests

If you make me create that every time... so help me

This was pre-MunkiWebAdmin2, so we did not have a method to easily duplicate manifests.

As I tested this system, I became increasingly irritated with having to recreate complex manifests.

Plus add human error to the equation.

wow... it really kinda sucked.

Machine Manifest

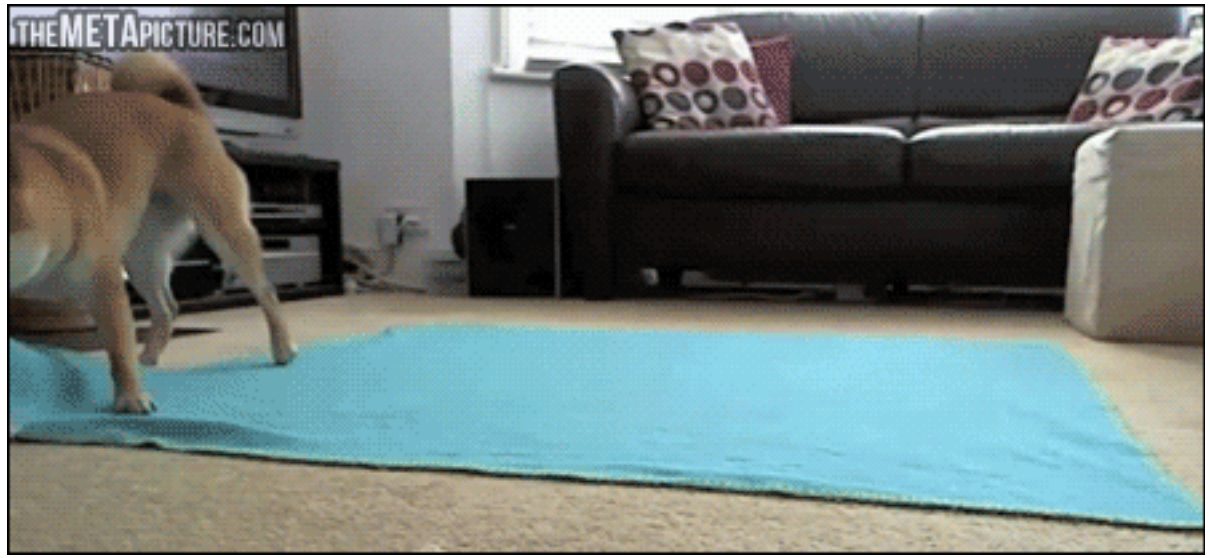
Machine manifests are almost always 1:1 for each piece of hardware. The o computer labs, where machines are ALWAYS configured to be identical - in point at a single manifest.

Catalogs
- production

Included Manifests
- site_default
- BND-Office2016
- BND-InternetPlugins
- BND-SelfServe

Managed Installs
- config-internalSoftwareUpdateServer
- profile-wifiConfiguration
- profile-campusVPN
- GoogleChrome
- Firefox
- Microsoft Secure Endpoint Protection

munki-enroller



Hint #3:

Any simple task you do repeatedly, scripts can do faster & more accurately.

Munki-Enroller

Any simple task you do repeatedly, scripts can probably do faster & more accurately.

Custom web API, written in python.

Receives POST data, sent via curl command in DeployStudio.

1. Client Identifier
2. Barrel ID

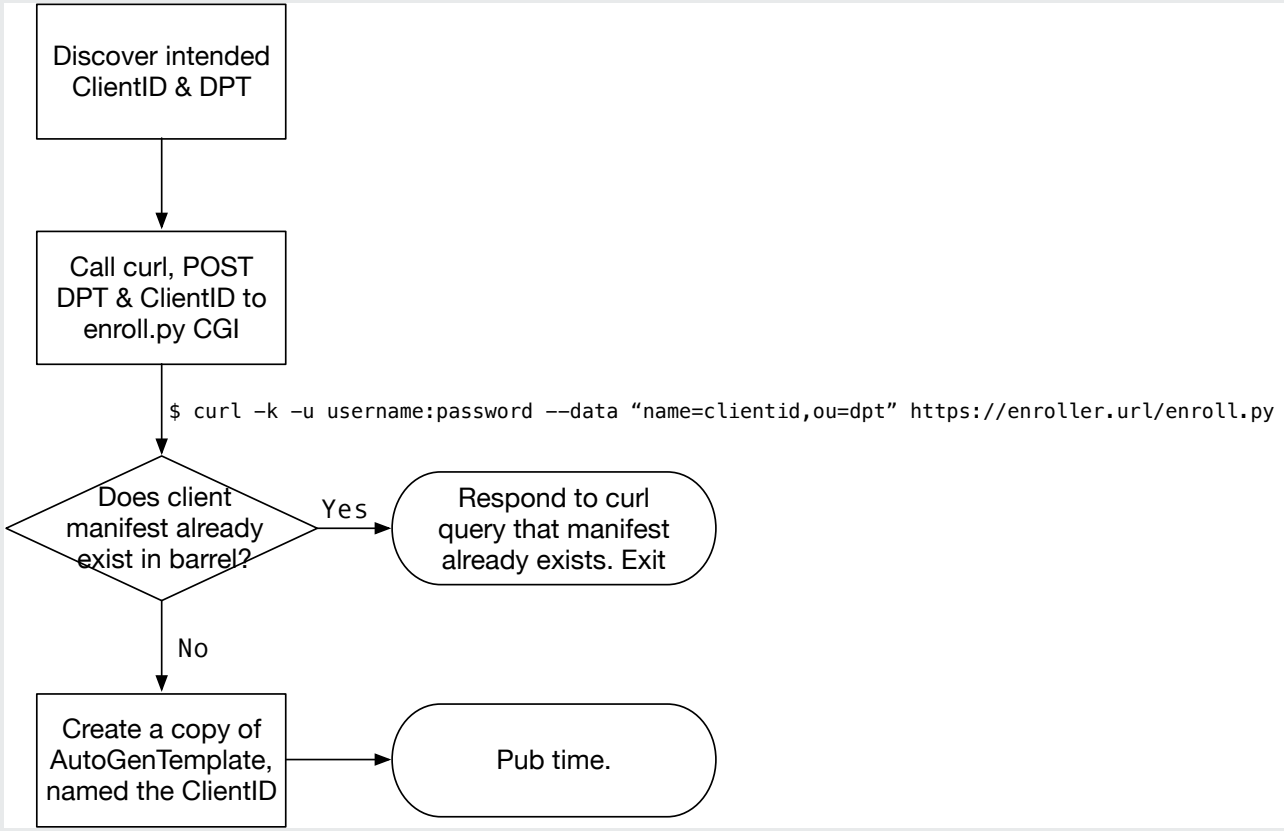
Creates a manifest in the correct barrel based on the ClientID that is communicated to it.

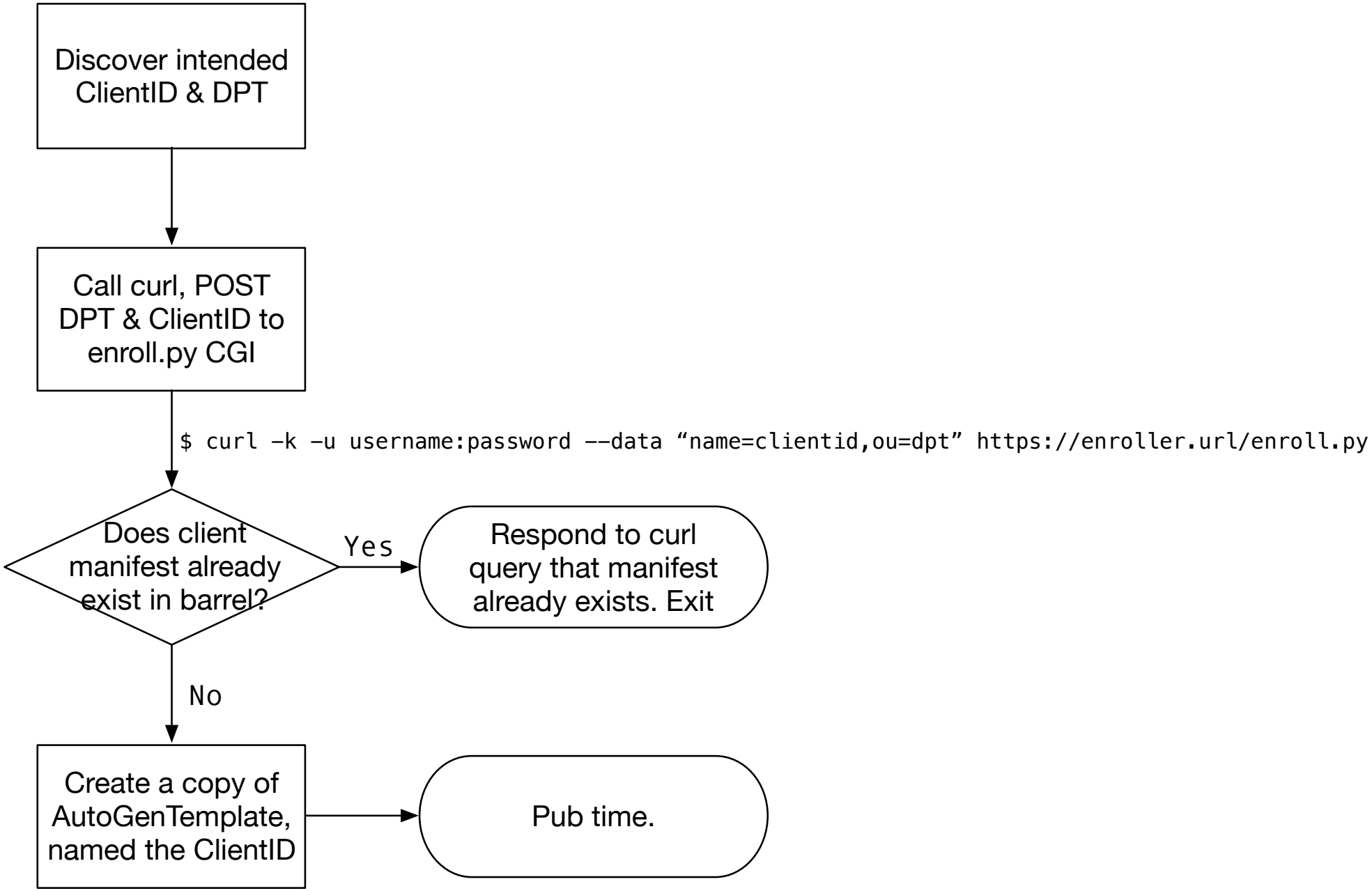
The enrollment process details

bubblegum & bailing twine

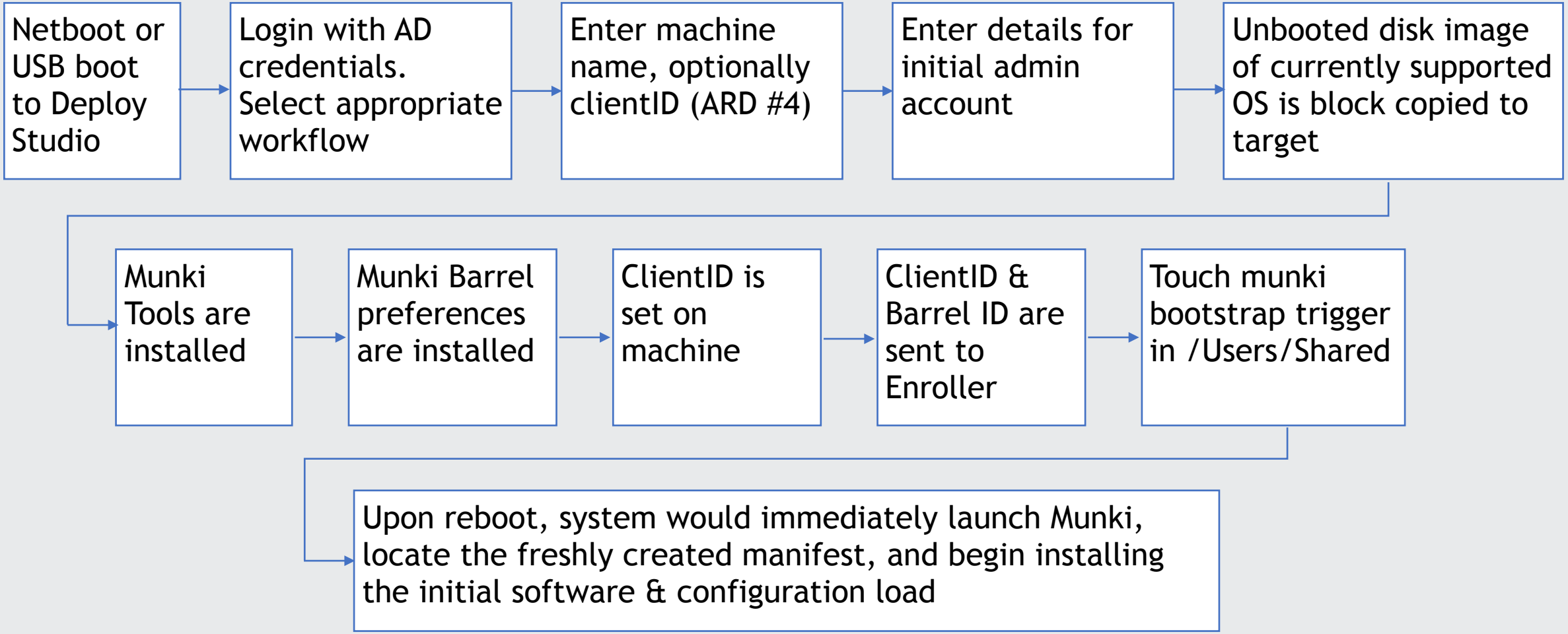
The Deploy Studio script takes the ClientID discovered earlier in workflow, combined with a barrelID, and sends the data to a web CGI script.

The python CGI endpoint takes the data, checks the specified munki barrel for the manifest (dies if it finds it), and creates a copy of the barrel's template manifest named for the ClientID.




Discover intended ClientID & DPT
Call curl, POST DPT & ClientID to enroll.py CGI
$ curl -k -u username:password --data “name=clientid,ou=dpt” https://enroller.url/enroll.py
Does client manifest already exist in barrel?
Yes
Respond to curl query that manifest already exists. Exit
No
Create a copy of AutoGenTemplate, named the ClientID
Pub time.

Complete thin-image workflow



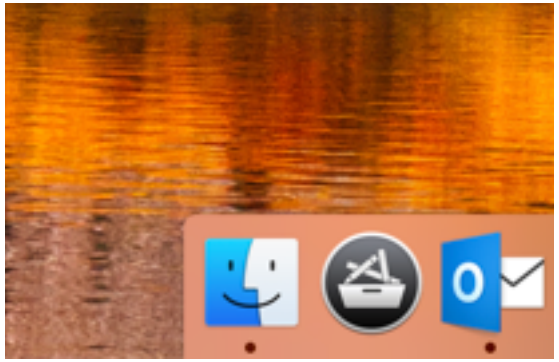
Great Success!

With our central DeployStudio instance, linked in to munki-enroller & the munki barrels, we were ready to recruit across campus.

The process for building a Mac had been streamlined, reduced the amount of work for distributed IT staff, and provided a compelling reason for colleges & admin departments to jump on board.

We were doing all right,
Gettin' good grades.
Future's so bright....

Meanwhile…

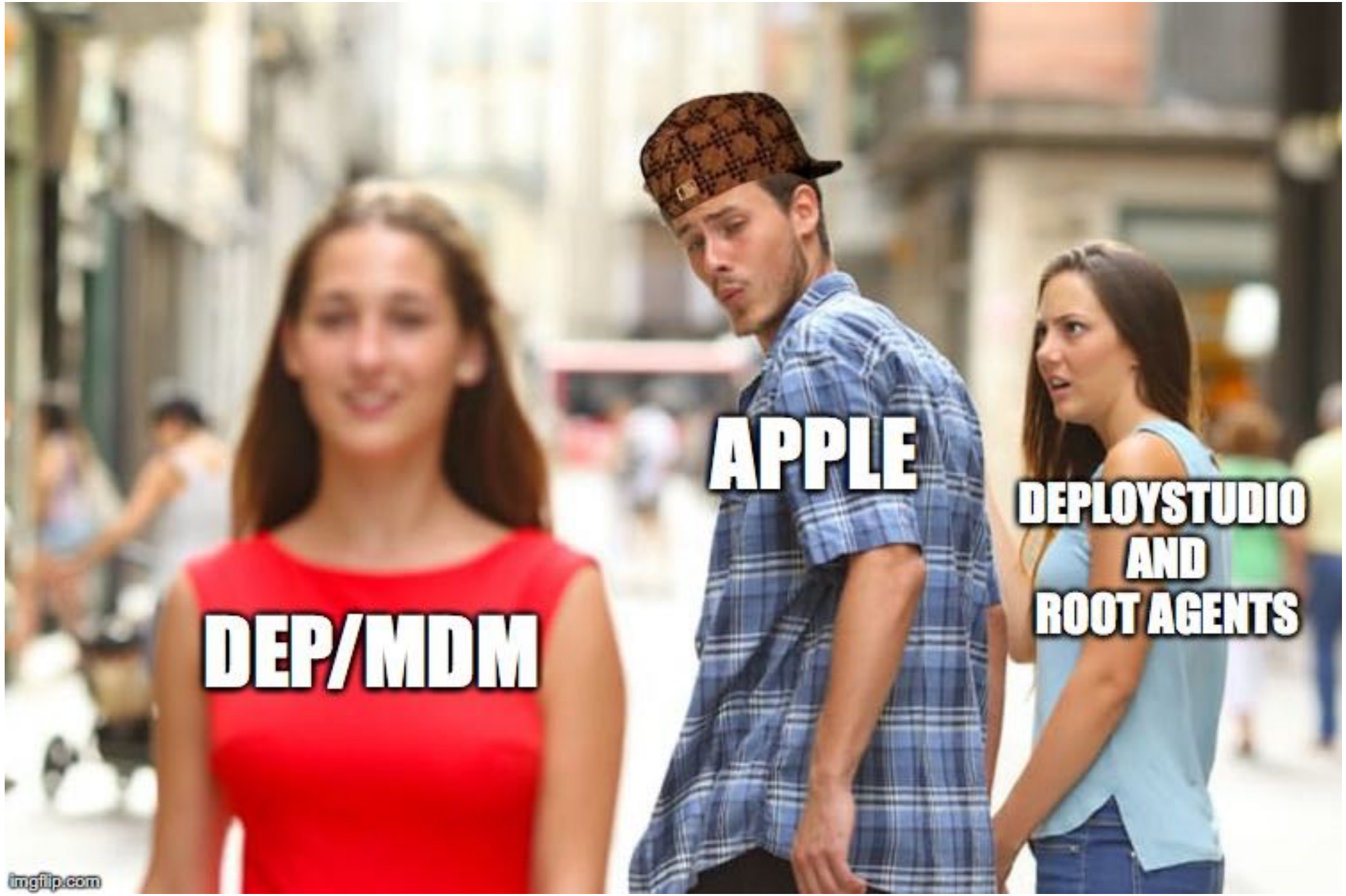


To: taschutt@syr.edu

From: ourAppleSE@apple.com

Hey Tim.

I think we need to have a phone conversation...


DEP/MDM
APPLE
DEPLOYSTUDIO
AND
ROOT AGENTS
imgflip.com

Wait... wha?

Double-Wait... wha?

We're losing all our tools to manage Macs?!?

Aw hell naw.

So, the MacAdmin community gave some feedback, and for once...


CAN WE KEEP ROOT AGENTS?
OK. BUT IMAGING IS STILL DEAD
imgflip.com

The switch to an MDM bootstrap workflow

Same steps, different tune.

We have to let go of block copy for deployment. That's it.

Everything else can be handled nicely another way.

Let me explain how I solved this...

DEP, MDM, Pkgs and Postinstall Scripts

DEP / MDM

- Initial user configuration (DEP only)
- Site-wide configuration profiles
  - SKEL profile
- Consistently install one signed pkg at enrollment
  - Let's get around that limitation with 'InstallApplications' from Eric Gomez.

Install pkg & scripts

- Install pkgs
  - Munki Tools
  - Munki Preferences
- Scripts
  - Set localization
  - Set Machine Name(?)
  - Set ClientID(?)

- Big questions
  - How do we communicate ClientID & Name
  - How do we choose the Munki barrel?

Conceptual workflow

Join machine to MDM (possibly via DEP)

Install pkgs - Munki Tools and base configuration.

If I run a post install script, I can discover & enroll a machine by serial number... now what?

?????

Machine enrolls in departmental barrel, creating the manifest if needed. Software deploys.

BFQ (Big Fat Question)

How do we route a Mac to the correct Munki Barrel with the correct name & ClientID?

What can we leverage?

Useful Munki behaviors:

- When a ClientID isn't set, Munki falls back to pre-determined IDs, like hostname & serial number.
- Munki will ignore XML keys in a manifest that it doesn't recognize.
- Munki caches stuff locally.
  - Machine manifest is located at /Library/Managed Installs/manifests/<manifest name>

Munki-Router

Munki-router

One barrel to route them all

A Munki Router is a Munki barrel with only serial-number based manifests.

The Router uses a customized version of Munki Web Admin & AutoGenTemplate that contain xml keys for Machine Name & ClientID.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PL
<plist version="1.0">
<dict>
    <key>catalogs</key>
    <array>
    </array>
    <key>included_manifests</key>
    <array>
    </array>
    <key>managed_installs</key>
    <array>
    </array>
    <key>managed_uninstalls</key>
    <array>
    </array>
    <key>managed_updates</key>
    <array>
    </array>
    <key>optional_installs</key>
    <array>
    </array>
</dict>
</plist>
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PL
<plist version="1.0">
<dict>
    <key>ClientIdentifier</key>
    <string></string>
    <key>ComputerName</key>
    <string></string>
    <key>catalogs</key>
    <array>
    </array>
    <key>included_manifests</key>
    <array>
    </array>
    <key>managed_installs</key>
    <array>
    </array>
    <key>managed_uninstalls</key>
    <array>
    </array>
    <key>managed_updates</key>
    <array>
    </array>
    <key>optional_installs</key>
    <array>
    </array>
</dict>
</plist>
```

MunkiWebAdmin2

Search

MunkiWebAdmin2 | Manifests | Package Info | Admin | Logout taschutt

Filter | Search manifests | New manifest

Name

C02FFFFFFF01

C02FFFFFFF01

Basics | Details | plist | Delete | Duplicate

Client Identifier as-demoSystem

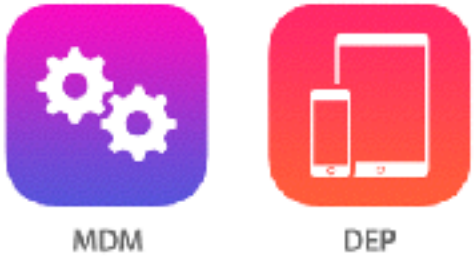
Computer Name as-demoSystem

Managed Installs

munkitools

AS-BarrelEnroll

Conceptual workflow



Join machine to MDM (possibly via DEP)

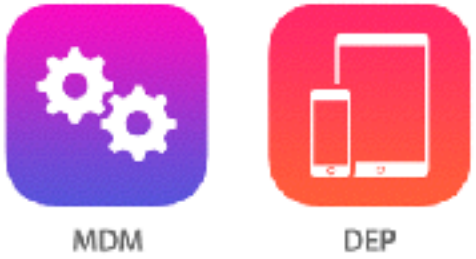
Install pkgs - Munki Tools and base configuration.

If I run a post install script, I can discover & enroll a machine by serial number... now what?

?????

Machine enrolls in departmental barrel, creating the manifest if needed. Software deploys.

Conceptual workflow



Join machine to MDM (possibly via DEP).

Install pkgs - Munki Tools and base configuration.

If I run a post install script, I can discover & enroll a machine by serial number... now what?

System is selected in Router by s/n. ClientID & Computer Name are set in Manifest.

OU Enrollment pkg is added.

Machine enrolls in departmental barrel, creating the manifest if needed. Software deploys.

Let's talk about that barrel enrollment pkg

Scripting some of those convenient DeployStudio workflow steps...

.pkg installs:

- Most barrel preferences in /Library/Preferences/...
- Barrel auth details go in /var/root/Library/Preferences/...

Postinstall script:

- Reads ComputerName & ClientID from cached manifest.
- Sets CompName & ClientID, plus timezone & NTP servers.
- Bootstraps 2nd munki run, once barrel prefs are in place.

What does this all look like?

We've come to the movie portion of our program

Demonstration time...

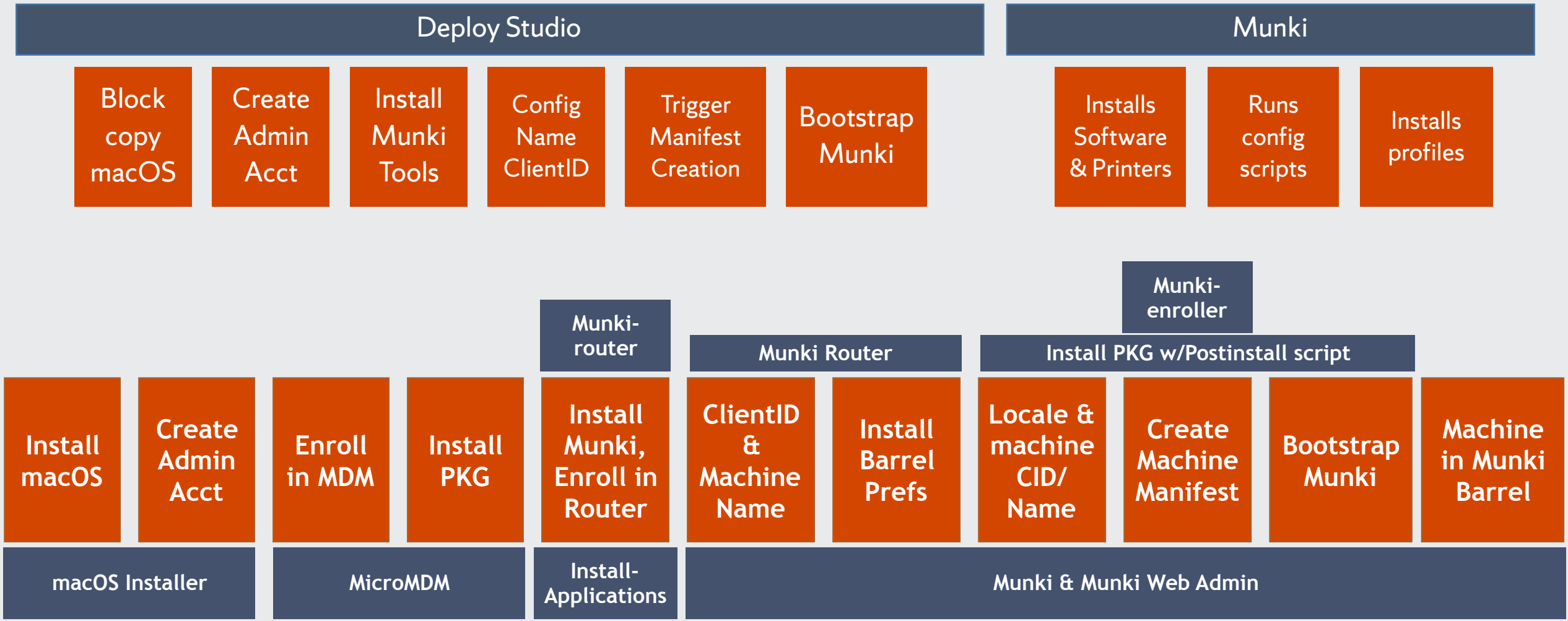
Subsequent machine builds

These data are sticky.

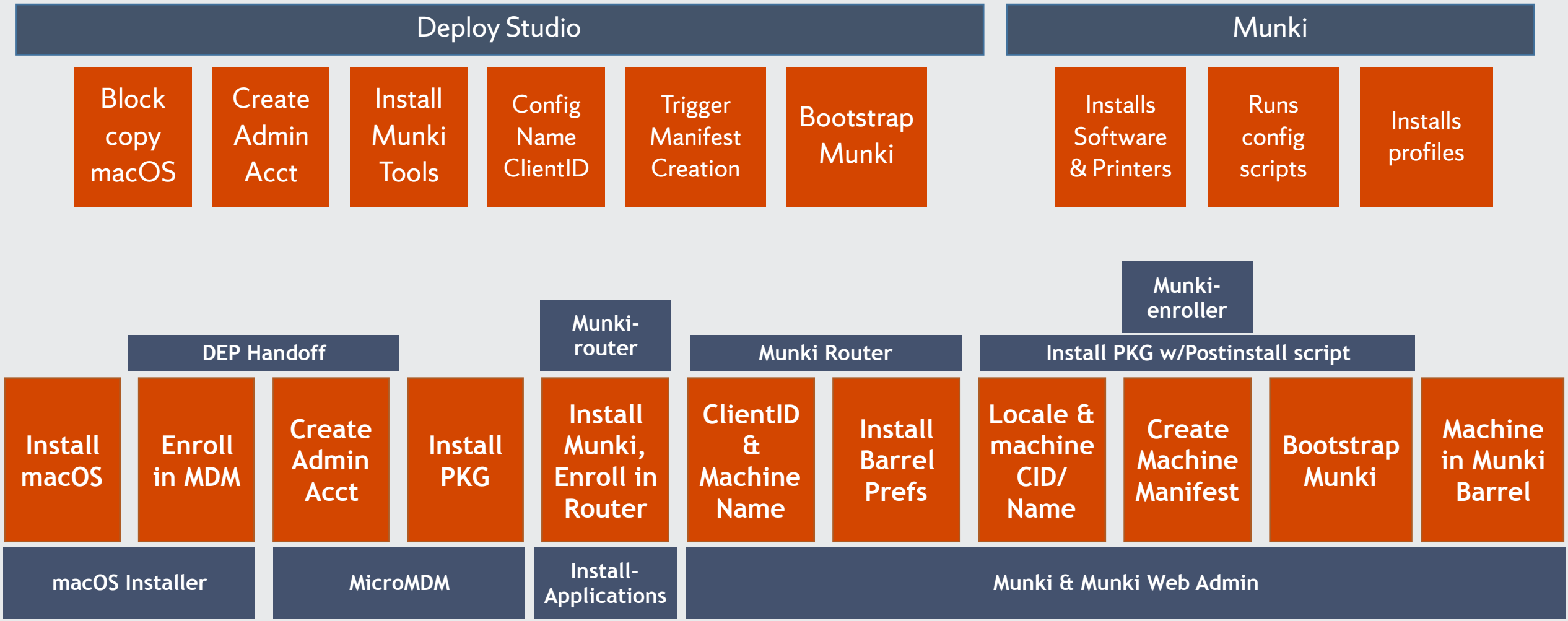
Munki Router serial number manifests stay put.

- Future builds involve
  - OS Install
  - Admin creation (if not using DEP)
  - MDM enroll & restart.

Each step, and associated tools - Manual MDM Enrollment

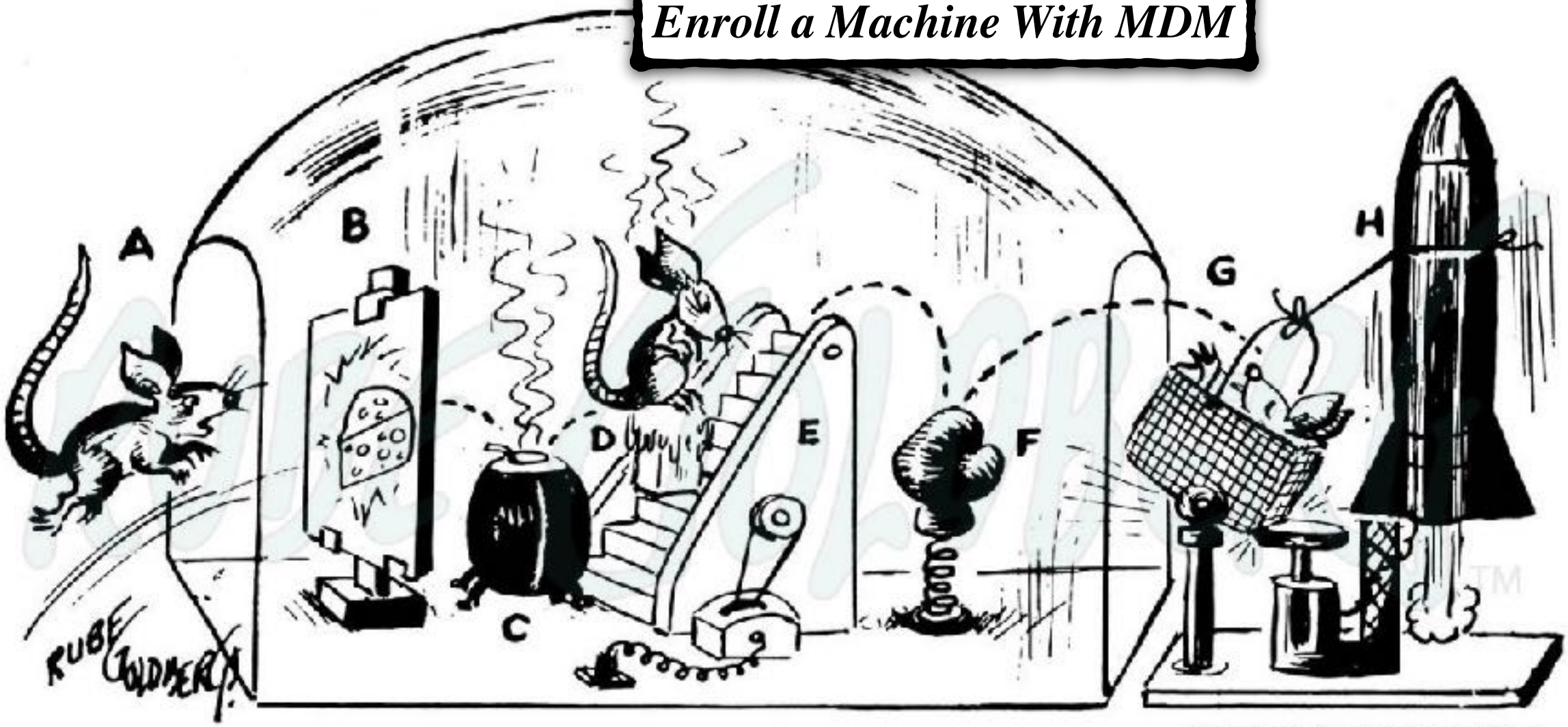


Each step, and associated tools - DEP to MDM Enrollment



How to ~~Get Rid of a Mouse~~

Enroll a Machine With MDM



Drawn for **Newsweek** by Rube Goldberg

The best mousetrap by Rube Goldberg: Mouse (A) dives for painting of cheese (B), goes through canvas and lands on hot stove (C). He jumps on cake of ice (D) to cool off. Moving escalator (E) drops him on boxing glove (F) which knocks him into basket (G) setting off miniature rocket (H) which takes him to the moon.

Results

When we started.

- One Munki Repository
- Three colleges
- 8 Distributed IT staff
- ~ 400 machines managed

Today.

- 18 Munki Barrels
- 11 Colleges, 7 Admin IT support groups
- ~ 75 Distributed IT support staff
- ~ 1600 machines managed
- 150+ Machines enrolled via MDM (Rolled out on June 14th)

One final thought

"Necessity may be the mother of invention, but adaptation is the milkman knocking on the side door."

Thank you.

Any questions?