Lab Nauseam - Dawn of the DEP

Hello! Can I have a quick show of hands? Who's managing lab environments?

And by that, I mean computers that are shared, by multiple users?

Who's still imaging? Me too - ish.

Are you concerned about the direction Apple is taking as it seems to be dropping support for this? Me too. Who's moving, or has moved their workflows to DEP? Me too - ish.

I'd like to share the journey I'm taking - and I say TAKING on purpose, because I don't think we're quite where we want to be just yet.

jamf NATION Roadshow

Neil Martin

Senior Desktop Support Specialist

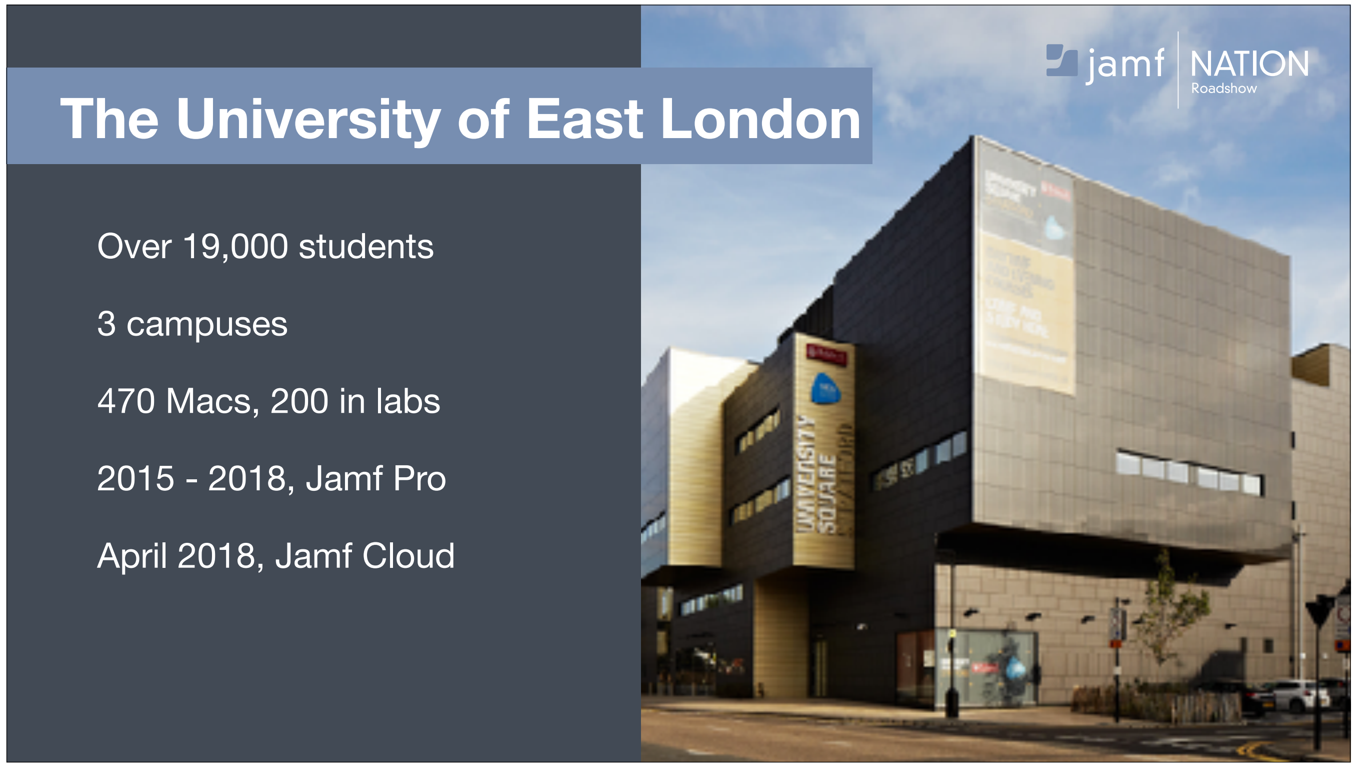
University of East London

@neilmartin83

I'm Neil and I joined UEL in 2004, working in one of its schools supporting specialist music facilities. Like lots of folks who do what we do, I didn't have an IT background - I had a music degree and I was a sound engineer.

Back then we had a couple of labs of eMacs and that's where I got bitten by the Apple bug. One thing led to another and now I'm with IT Services.

<C> I'm also honoured to be one of the administrators on the Slack, where you may know me as this person.



The University of East London

Over 19,000 students

3 campuses

470 Macs, 200 in labs

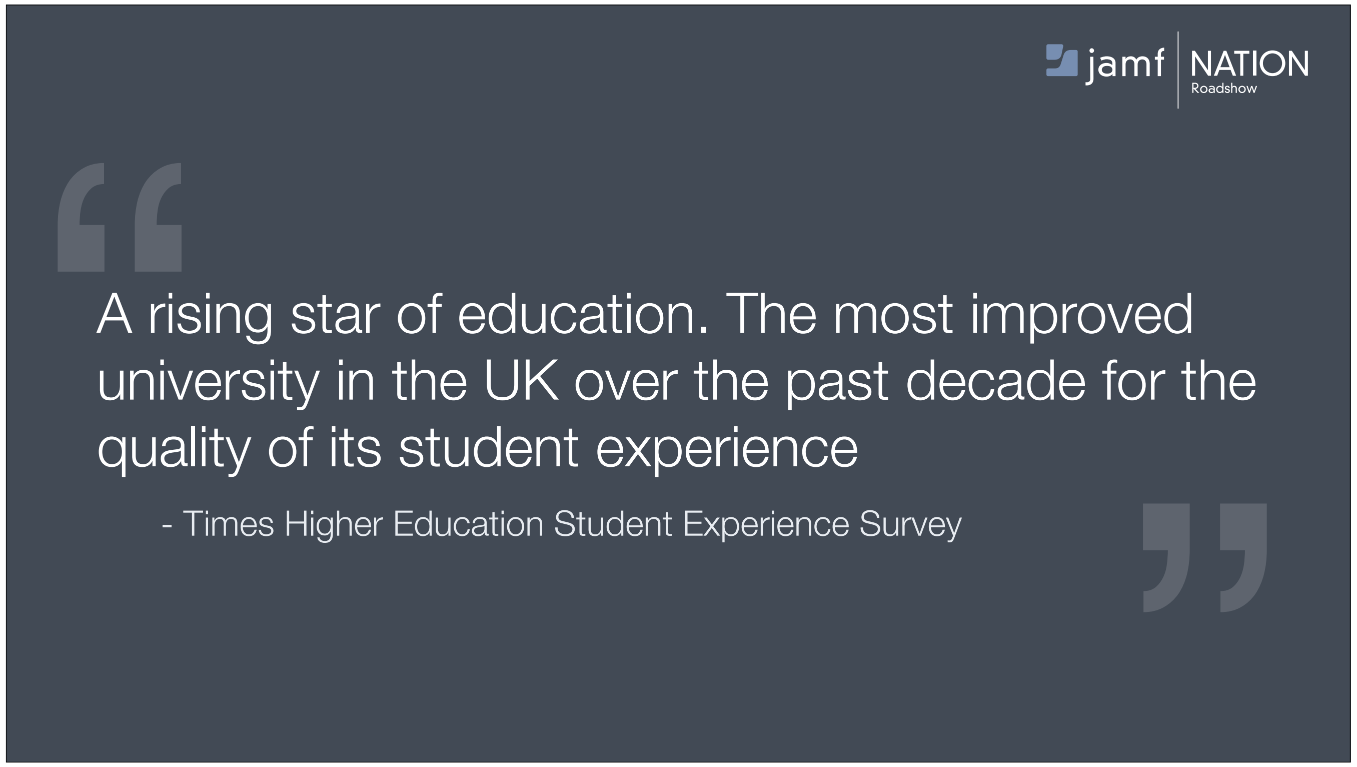
2015 - 2018, Jamf Pro

April 2018, Jamf Cloud

UEL has more than 19,000 students, spread over our Docklands, Stratford and University Square Stratford campuses.

We manage around 5,000 PCs, with Macs making up about 10% of the estate. On both platforms, we manage 2 key sorts of experience - a 1-1 model for staff, and a shared “Lab” model for students.

We’ve been managing our Macs with Jamf Pro since 2015 and we’ve just migrated to Jamf Cloud. The migration was really smooth - we worked with Moof-IT and Jamf, who were both great - and it meant we could get rid of 4 on-premise servers, saving my team and I time as well as resources.

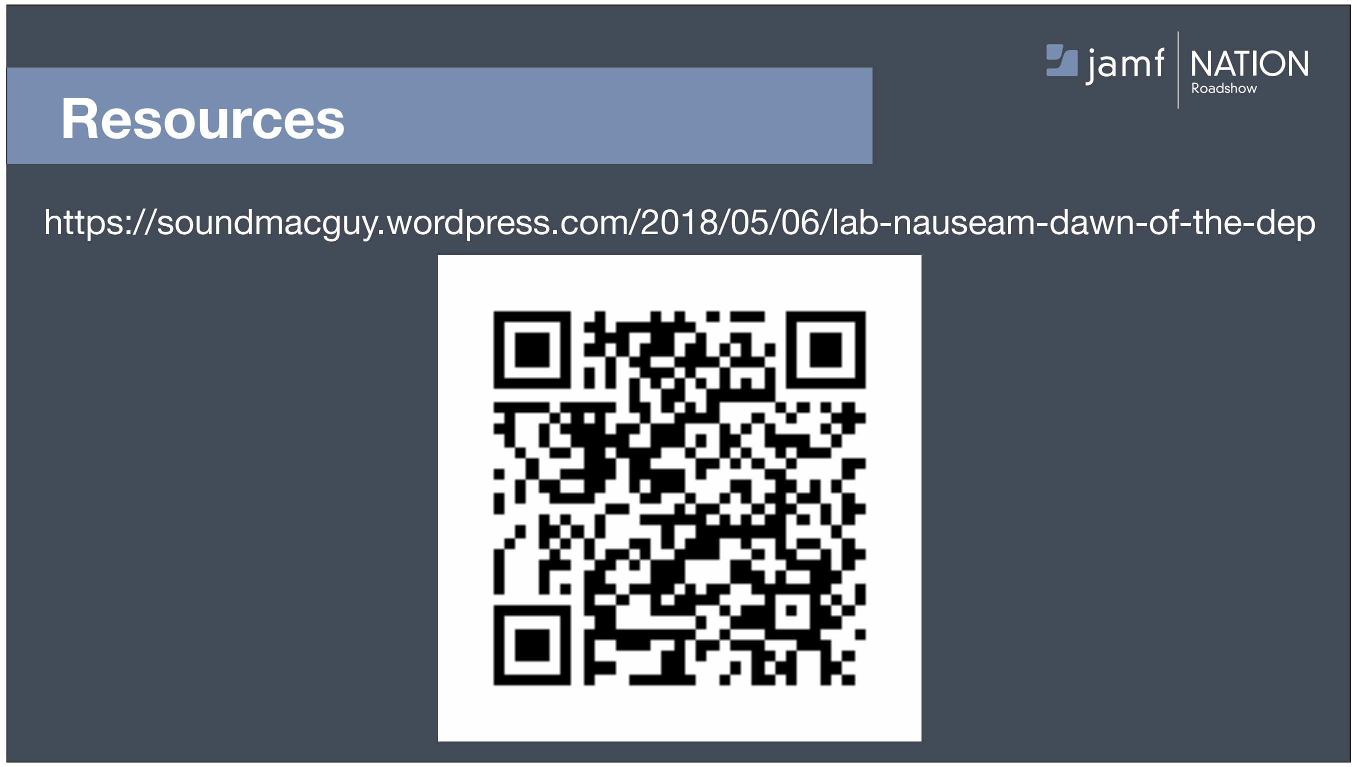


jamf | NATION Roadshow

A rising star of education. The most improved university in the UK over the past decade for the quality of its student experience

- Times Higher Education Student Experience Survey

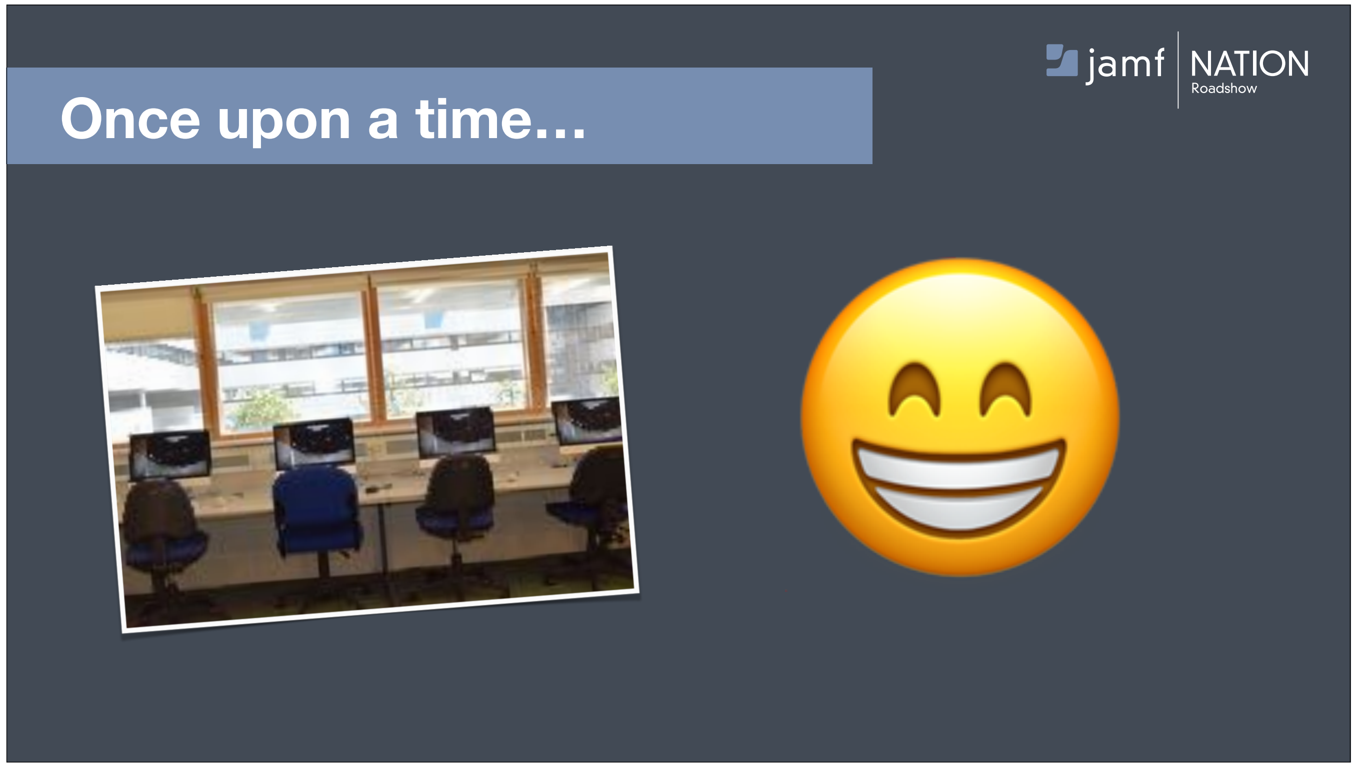
In this year’s Times Higher Education survey for Student Experience, we were called out for increasing our score by nearly 15 points since last year, a jump that was higher than any other UK university.



Resources

https://soundmacguy.wordpress.com/2018/05/06/lab-nauseam-dawn-of-the-dep

Before I begin, a copy of these slides, resources and links to all the other things I'll mention will all be on my blog, so don't worry about taking notes.



Once upon a time…

I'd like to tell you a story. Once upon a time, one summer, there was a lab.

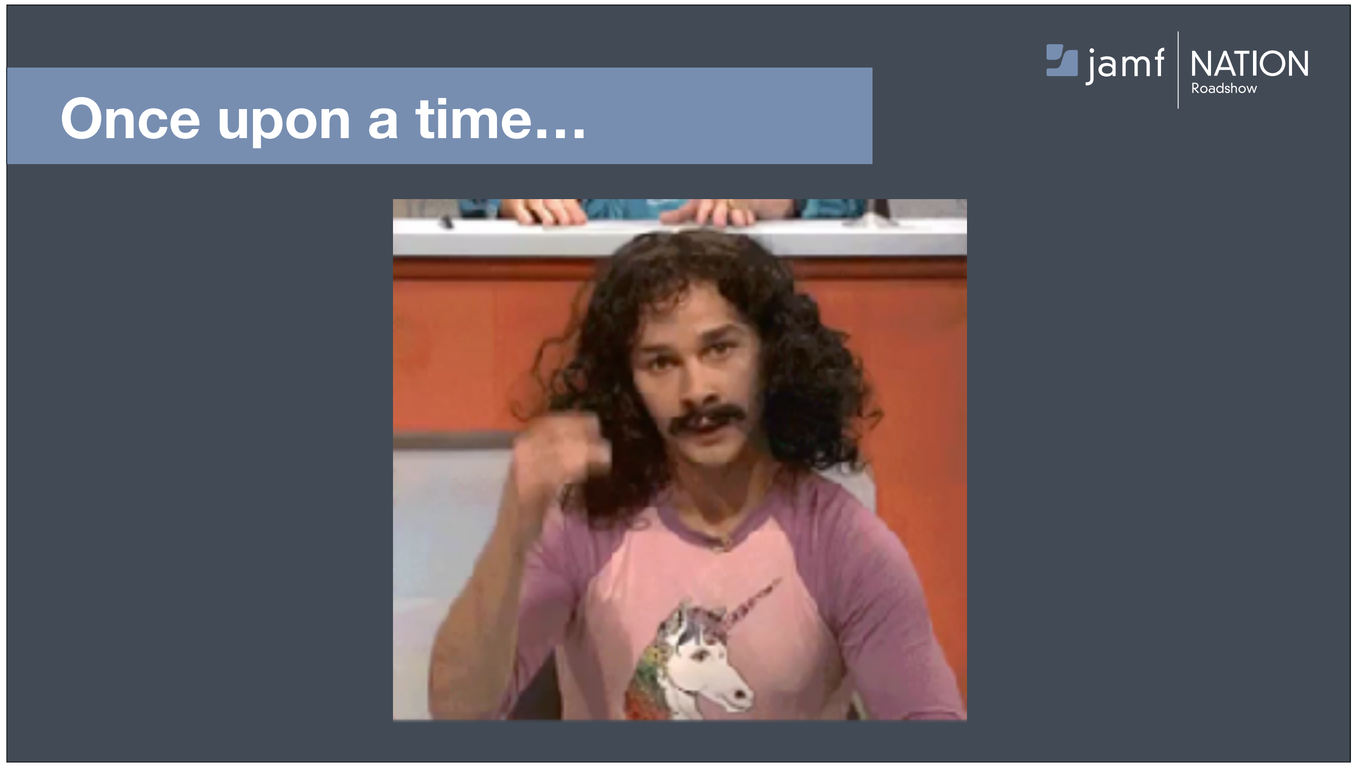
<C> and it was a happy lab.

It was time for the Macs to get their annual re-image up to the current version of macOS.

Once upon a time…

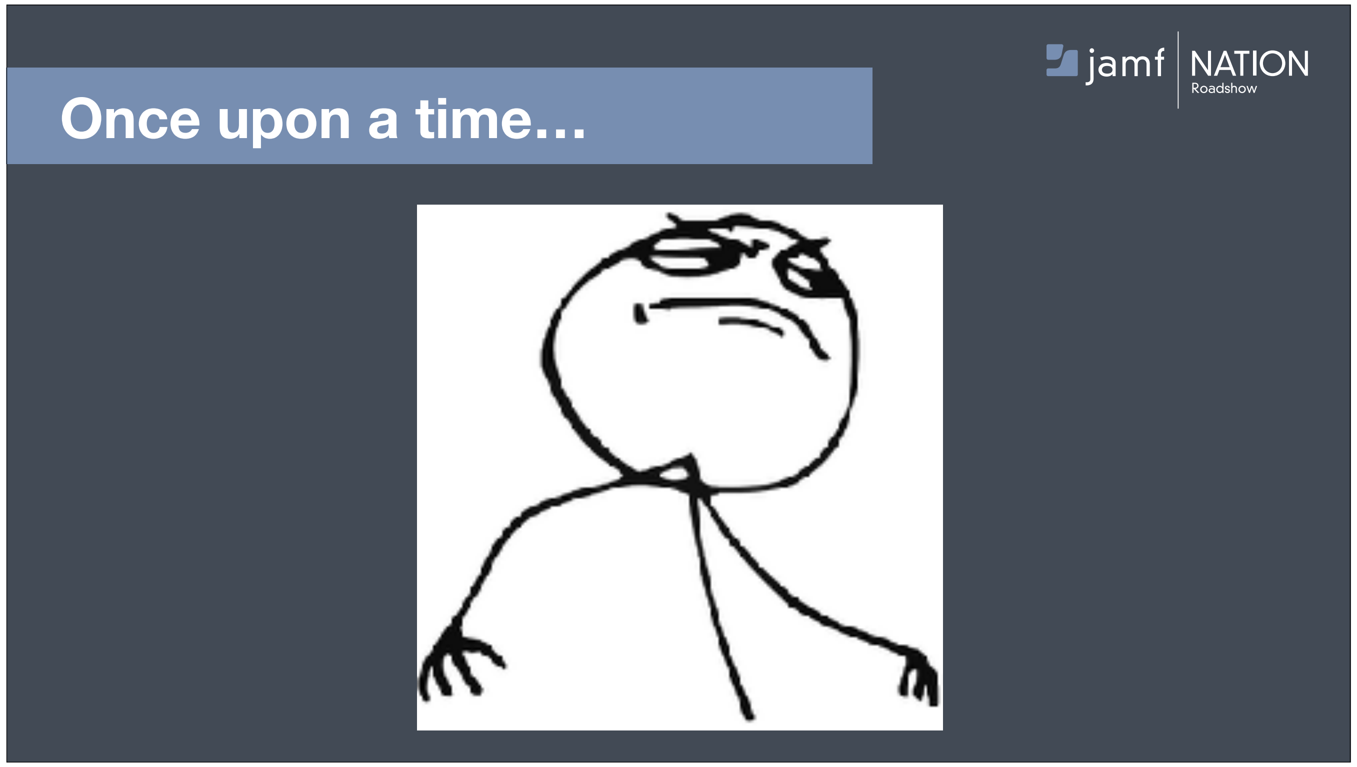
So, you’d NetBoot the lab, remotely.

<C> Jamf (or Casper) Imaging would open, then Autorun imaging would kick off.



Once upon a time…

Magic!

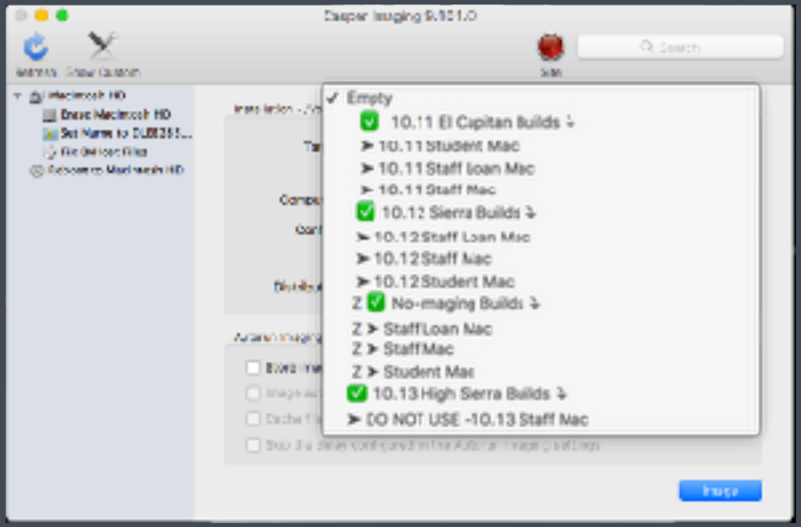


Once upon a time…

jamf NATION Roadshow

And you’d walk away, feeling pretty good about yourself.

Once upon a time…

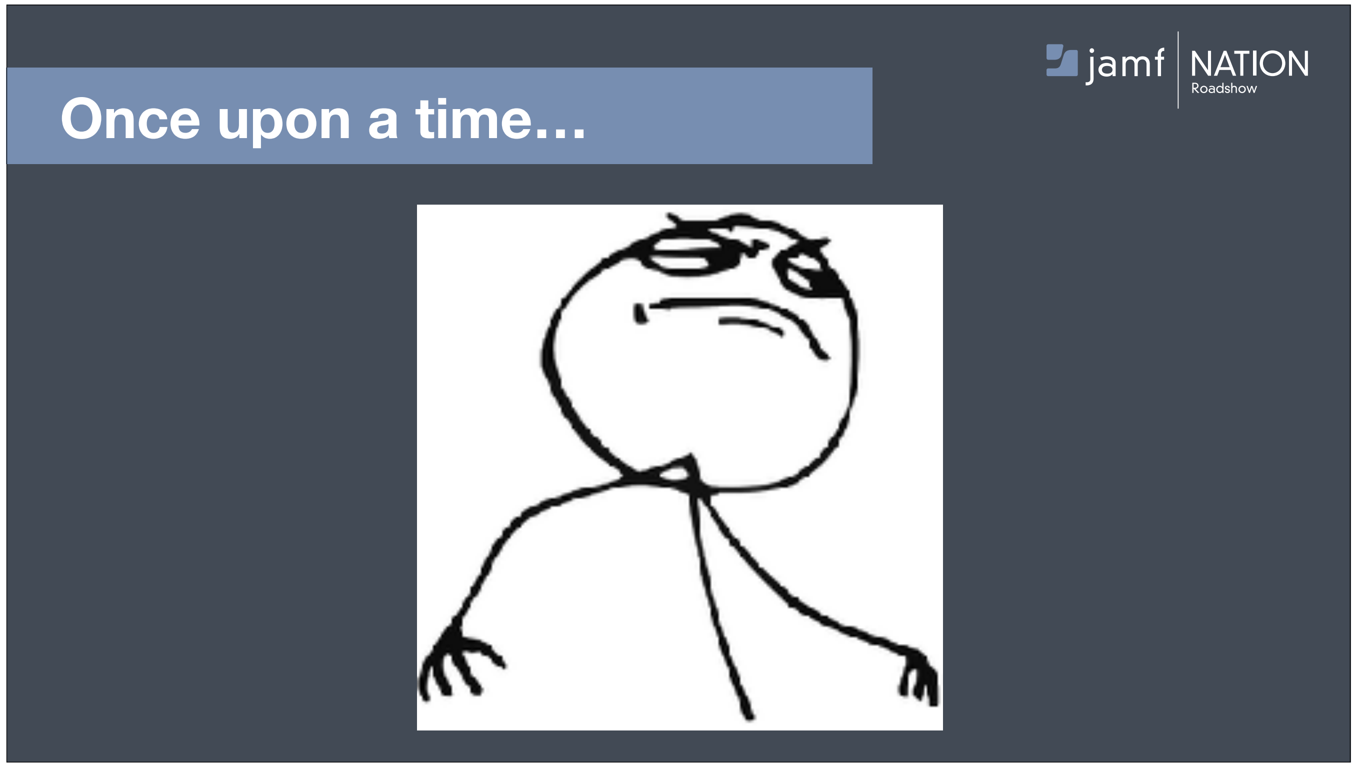


If you were provisioning a new Mac, you'd give it a hostname

<C>, then choose a configuration that told Jamf the version of macOS you wanted and that you wanted it to be a Student facing computer.

Its role (whether the Mac was staff or student facing), was stored in an Extension Attribute.

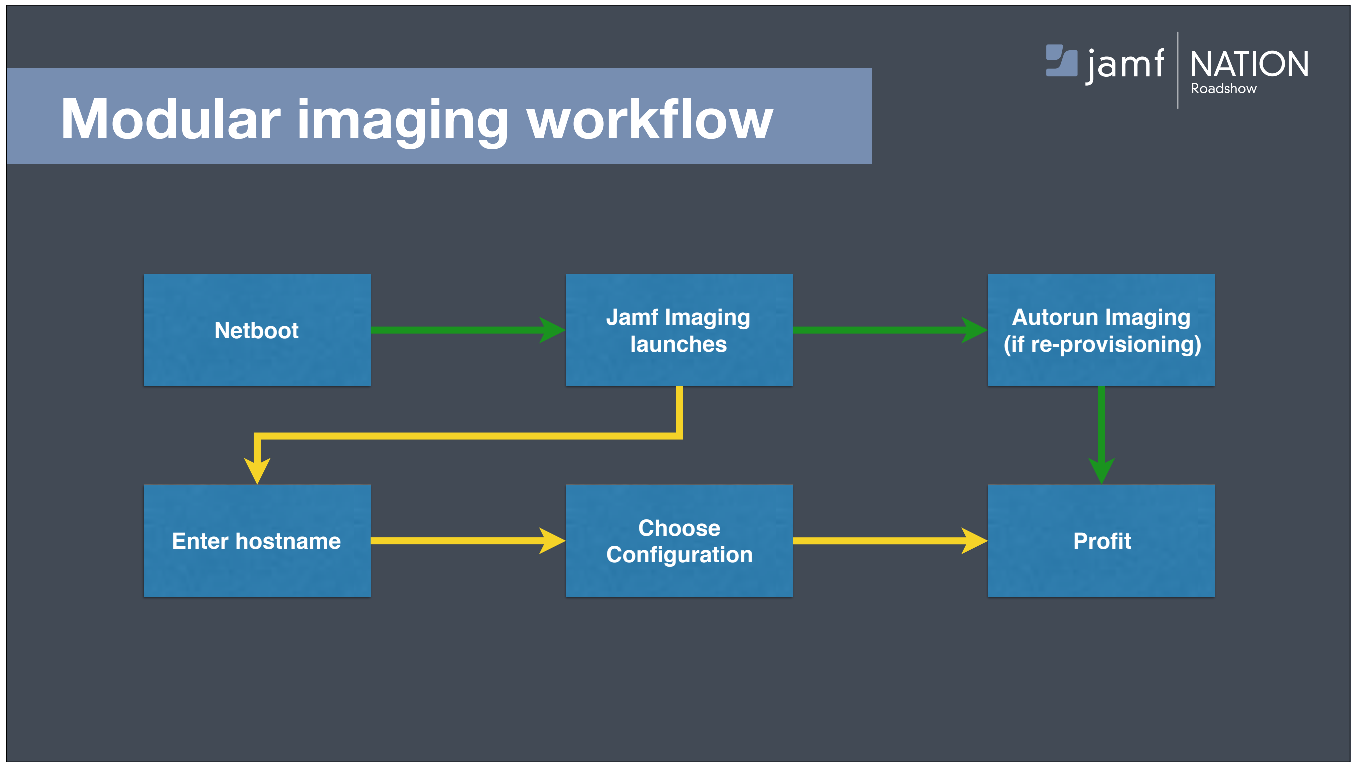
Depending on what you'd select here, staff or student, we'd write a corresponding dummy file to the Library folder and using the API, have a script set the proper value of the Extension Attribute based on that file.



Once upon a time…

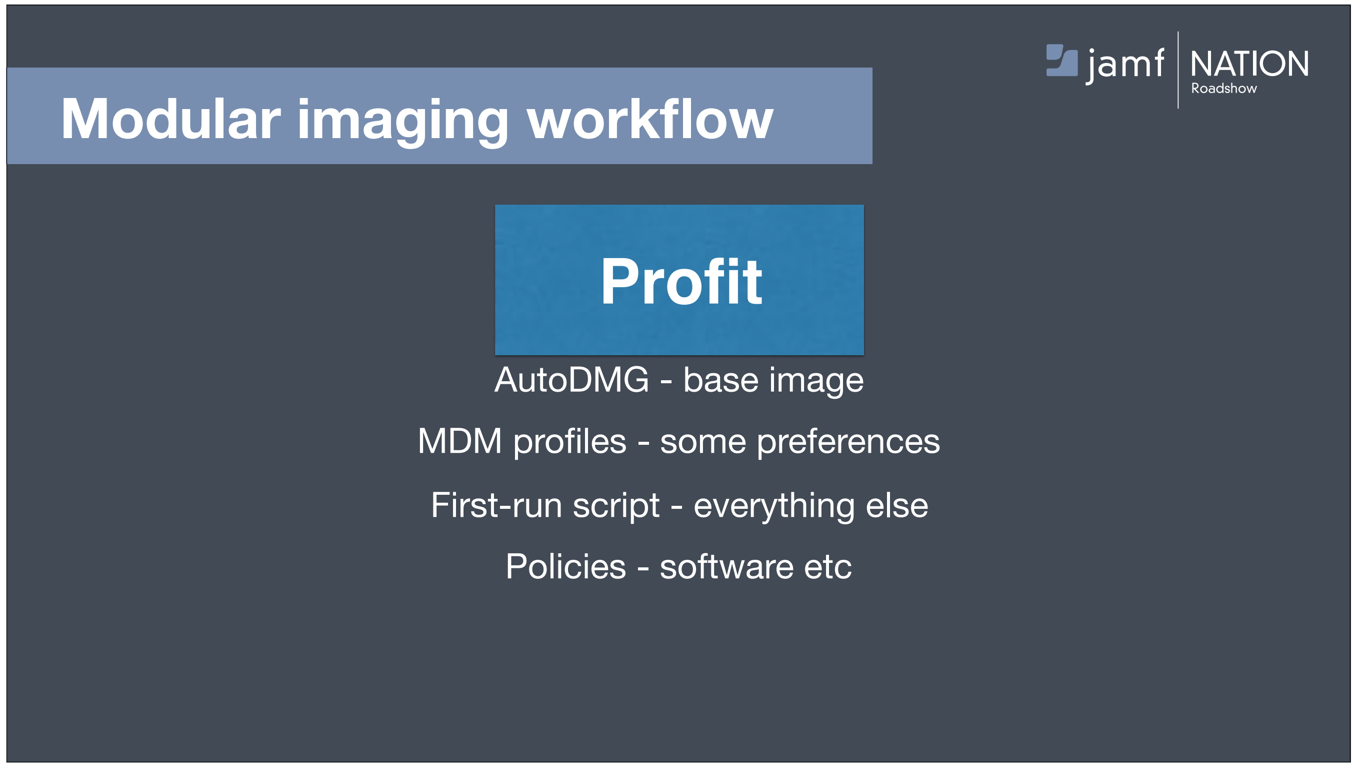
jamf NATION Roadshow

After you clicked the Image button, you'd walk away, feeling pretty good about yourself.



Modular imaging workflow

This modular workflow might look like this. Once you hit that Imaging button…



Modular imaging workflow

Profit

AutoDMG - base image

MDM profiles - some preferences

First-run script - everything else

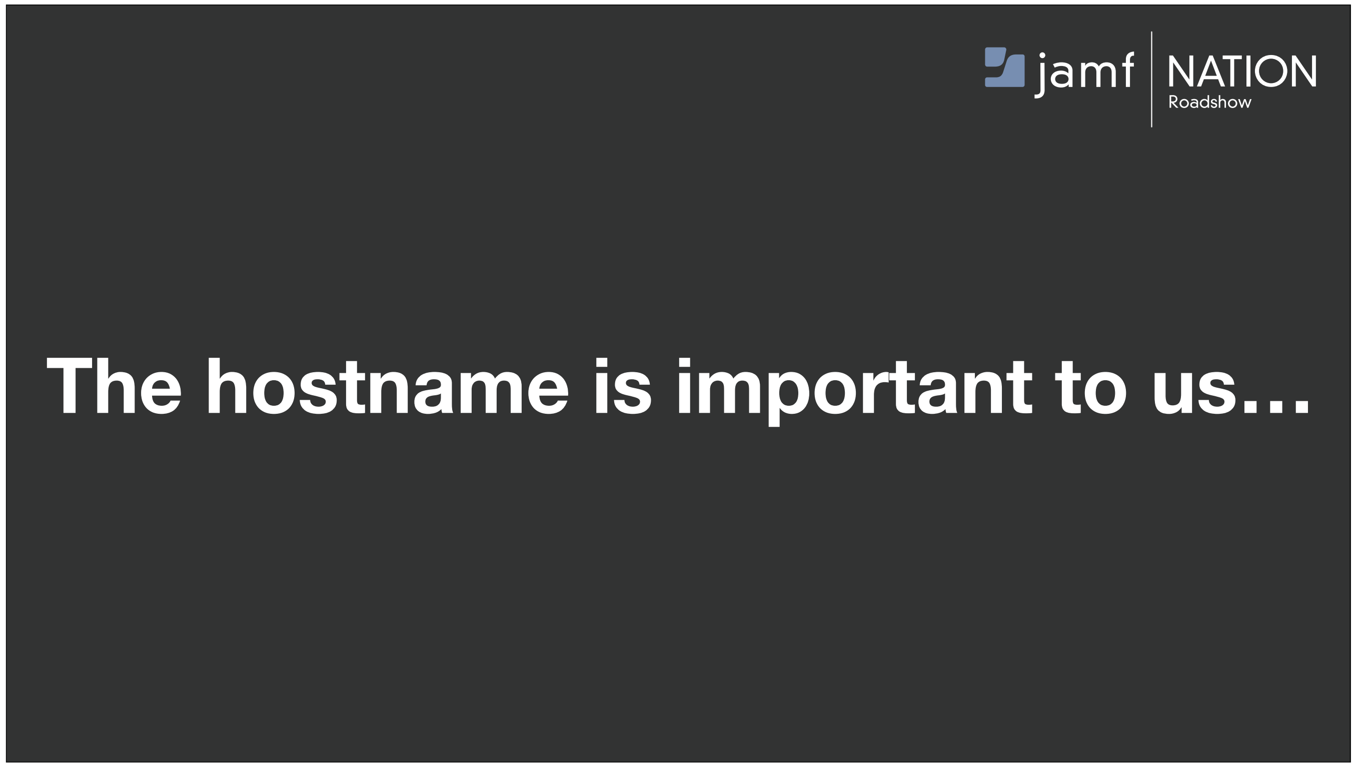
Policies - software etc

We'd lay down a clean, never-booted AutoDMG image…

<C> have MDM profiles set most preferences

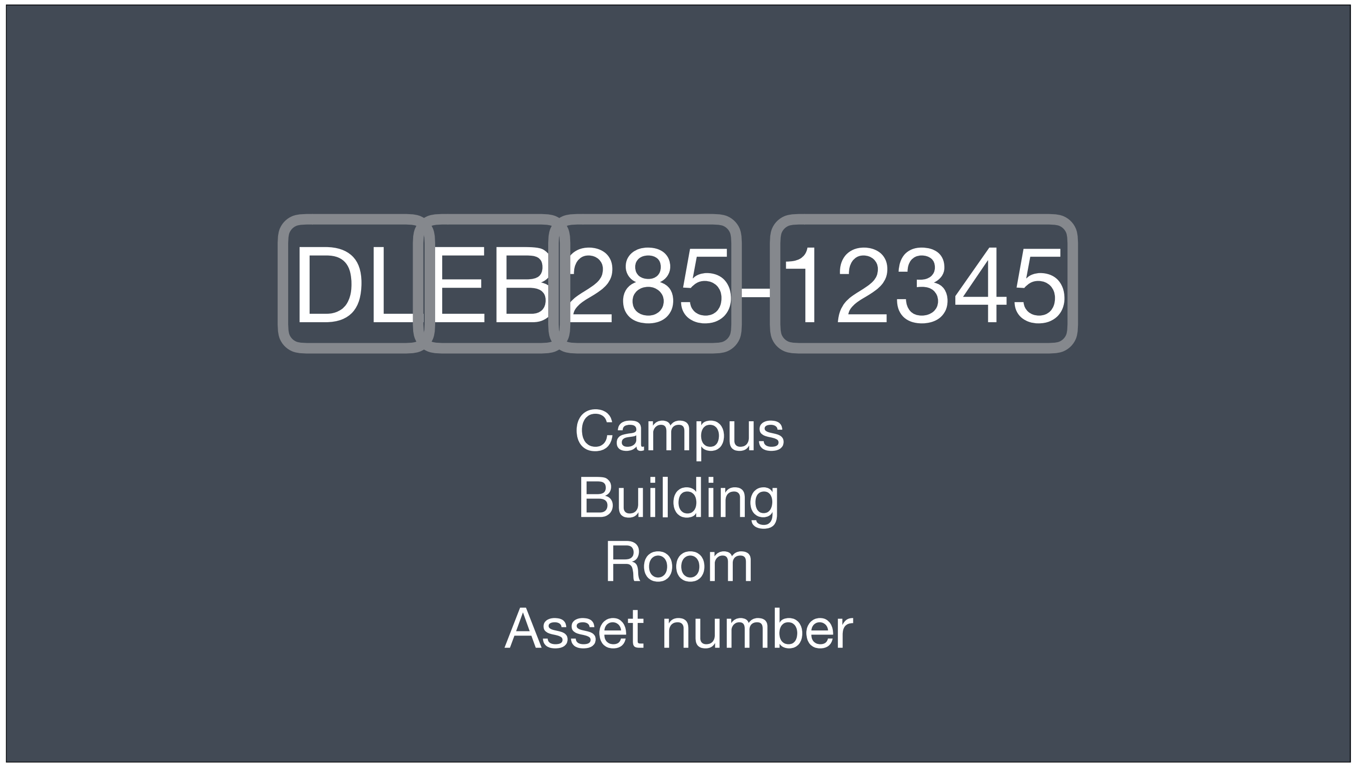
<C> Run a script to do everything else (like enable screen sharing, set the timezone and so on)

<C> and have policies to get all the software installed


jamf | NATION Roadshow

The hostname is important to us…

And for it to work, we really need to set the hostname…

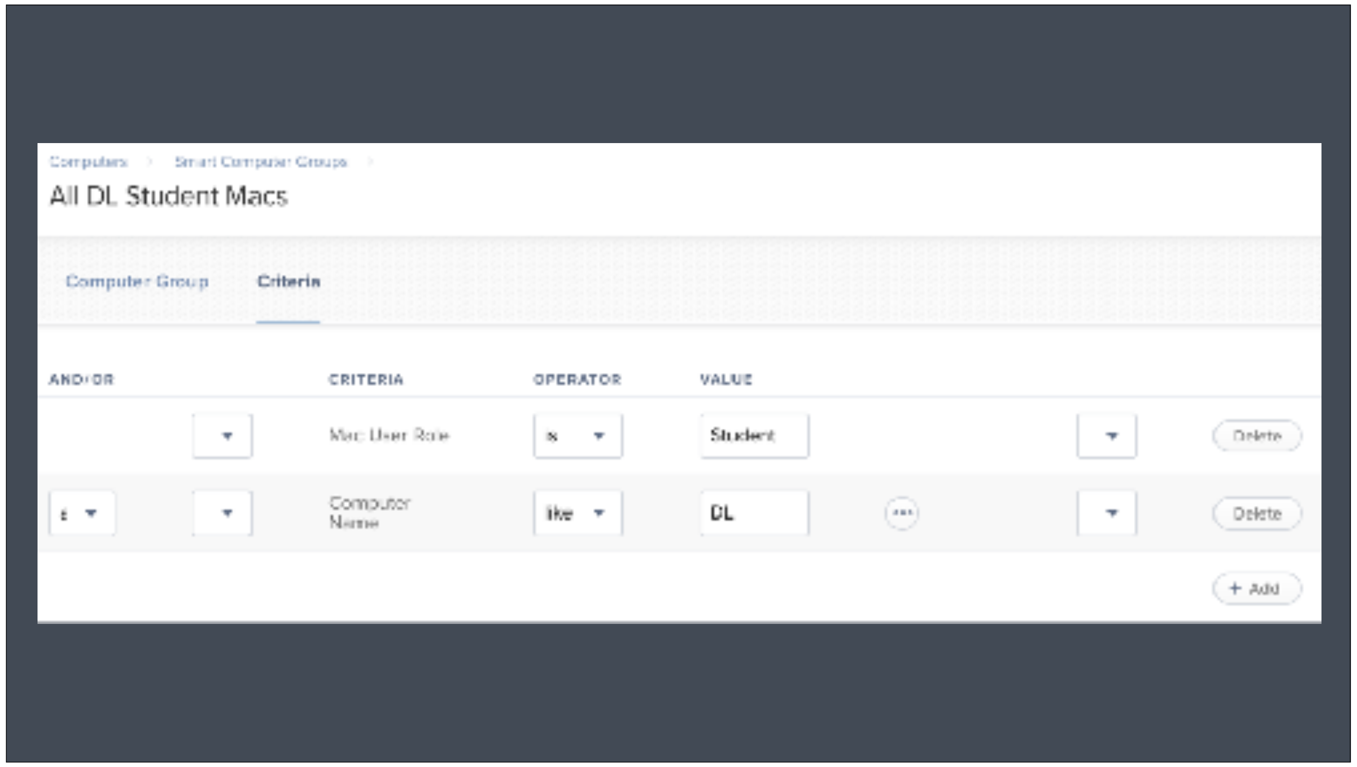


DLEB285-12345

Campus
Building
Room
Asset number

Because our hostnames tell us the <C> Campus (Docklands), <C> Building (East Building), <C>, Room number <C> and our own internal Asset number.

This means we can create smart groups, based on the computer name…

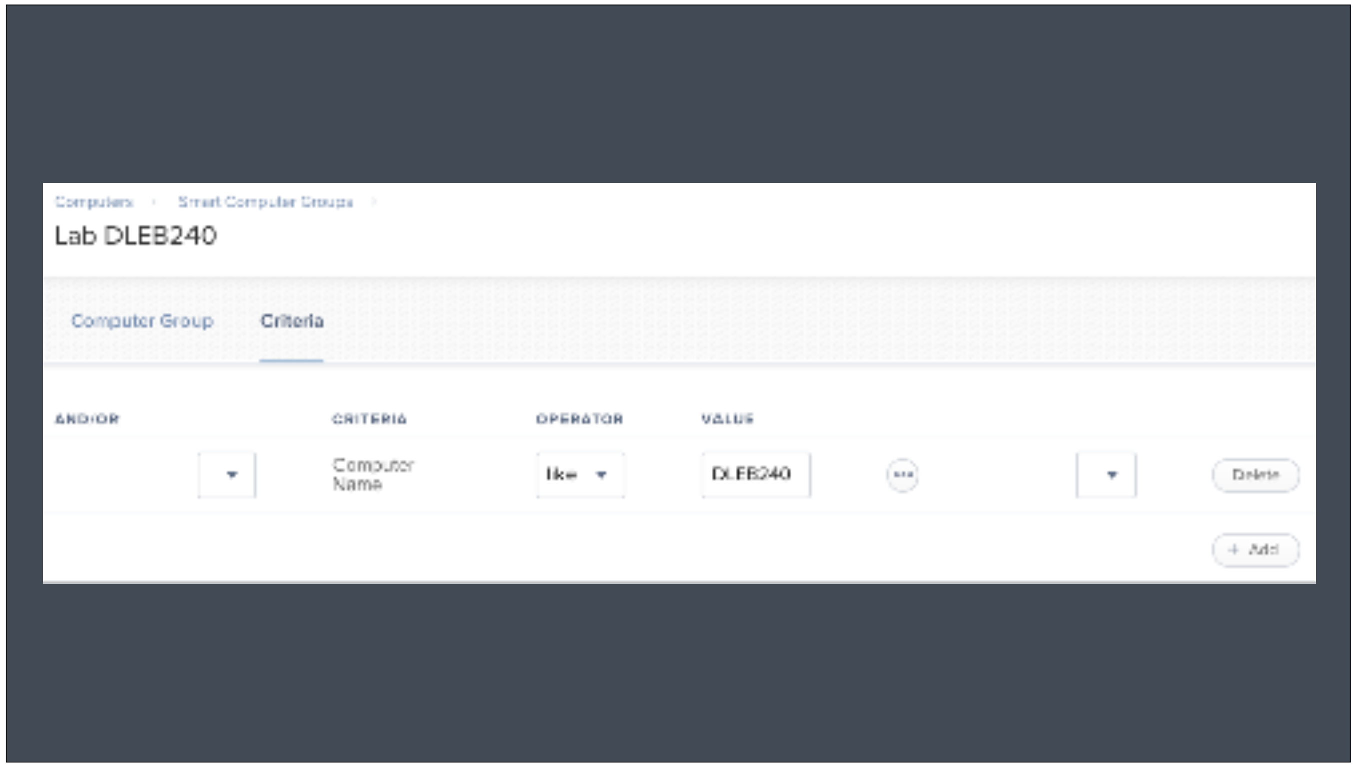

Computers > Smart Computer Groups >

All DL Student Macs

Computer Group | Criteria

AND/OR		CRITERIA	OPERATOR	VALUE
		Mac User Role	is	Student
[illegible]		Computer Name	like	DL

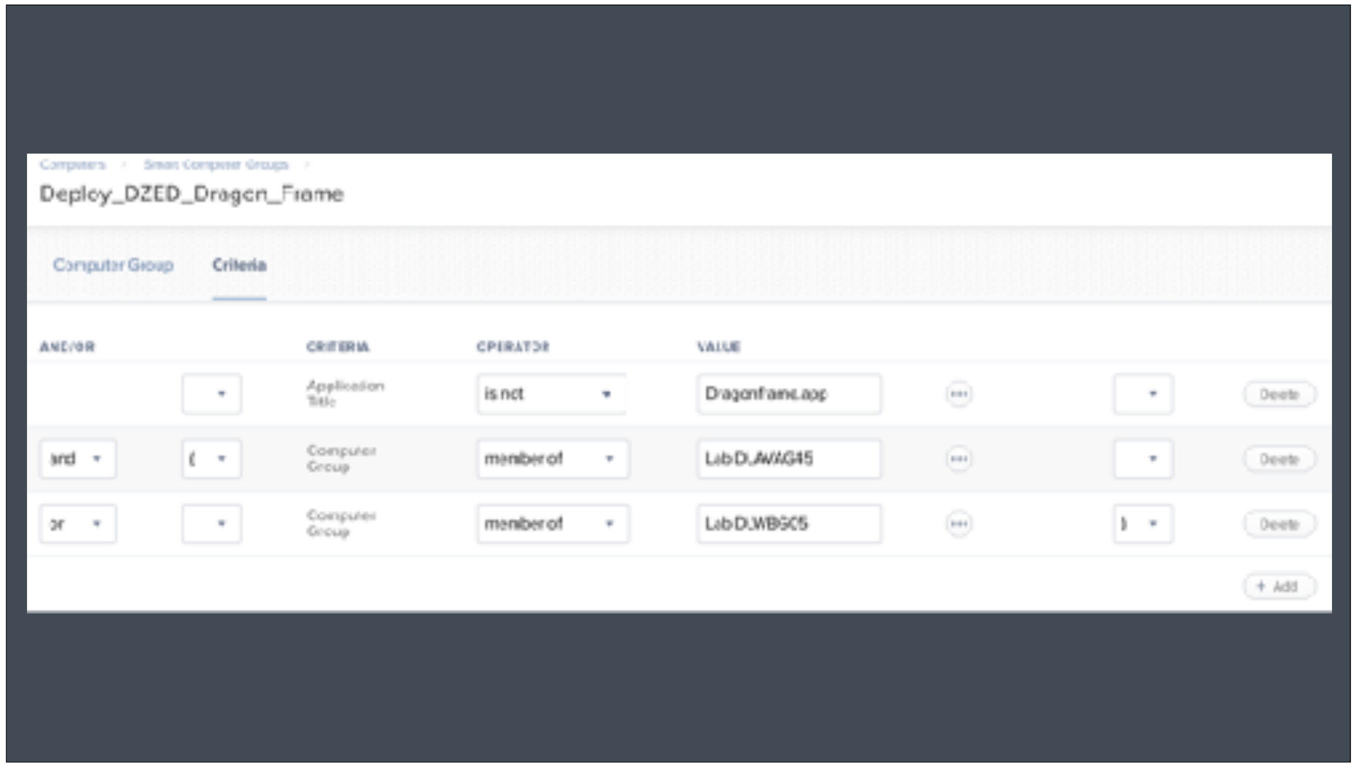
…that have all the student facing Macs in a whole campus or building.



Lab DLEB240

AND/OR	CRITERIA	OPERATOR	VALUE
	Computer Name	like	DLEB240

…or have all the Macs in a specific room or lab…


Computers > Smart Computer Groups >

Deploy_DZED_Dragon_Frame

Computer Group | Criteria

AND/OR		CRITERIA	OPERATOR	VALUE		
		Application Title	is not	Dragonframe.app		
and	(	Computer Group	member of	Lab D.AWAG45		
or		Computer Group	member of	Lab D.WBG05	)	

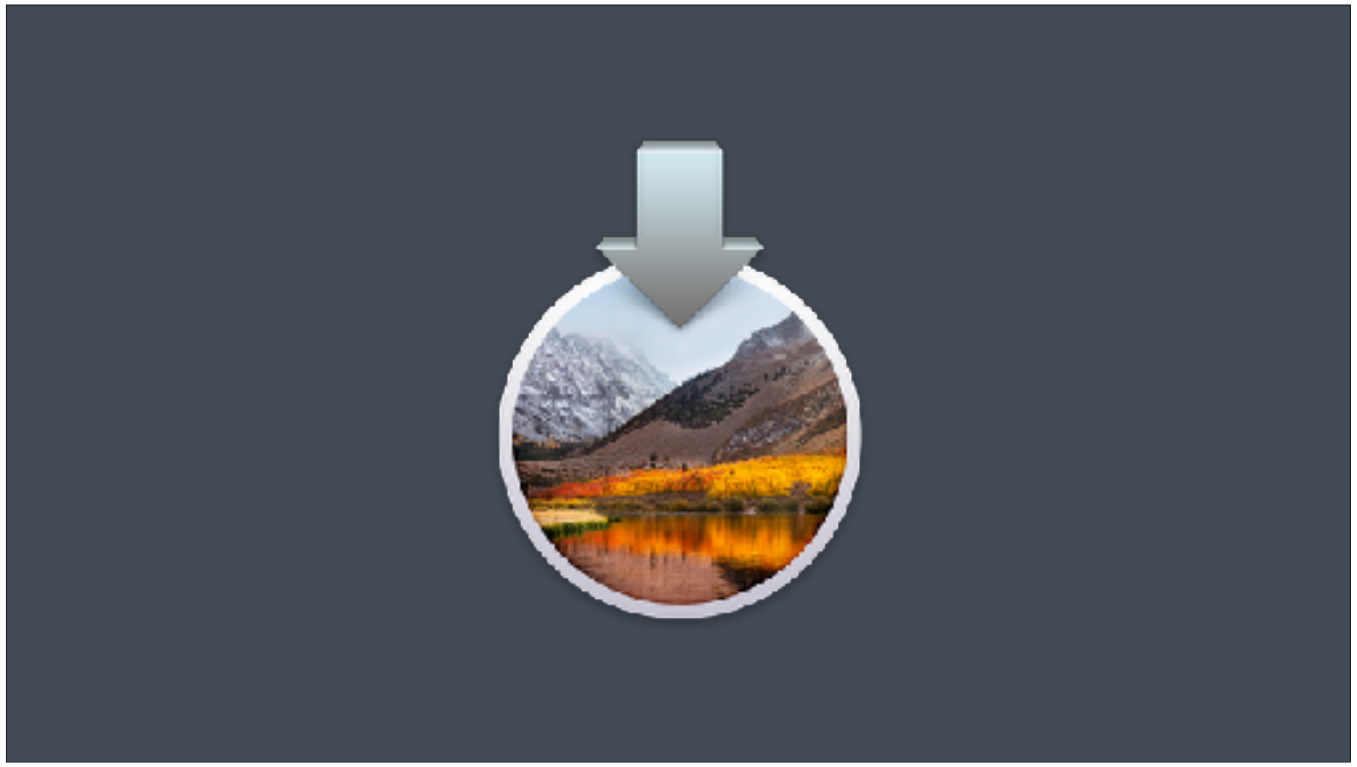
When it comes to deploying an application, say Dragonframe, we could have a Smart Group that looks for Macs that are in the labs that require it but don’t have it installed.

With this, we can scope a policy to this group that installs Dragonframe. If you want to deploy more applications, just create a similar smart group and policy for each one.

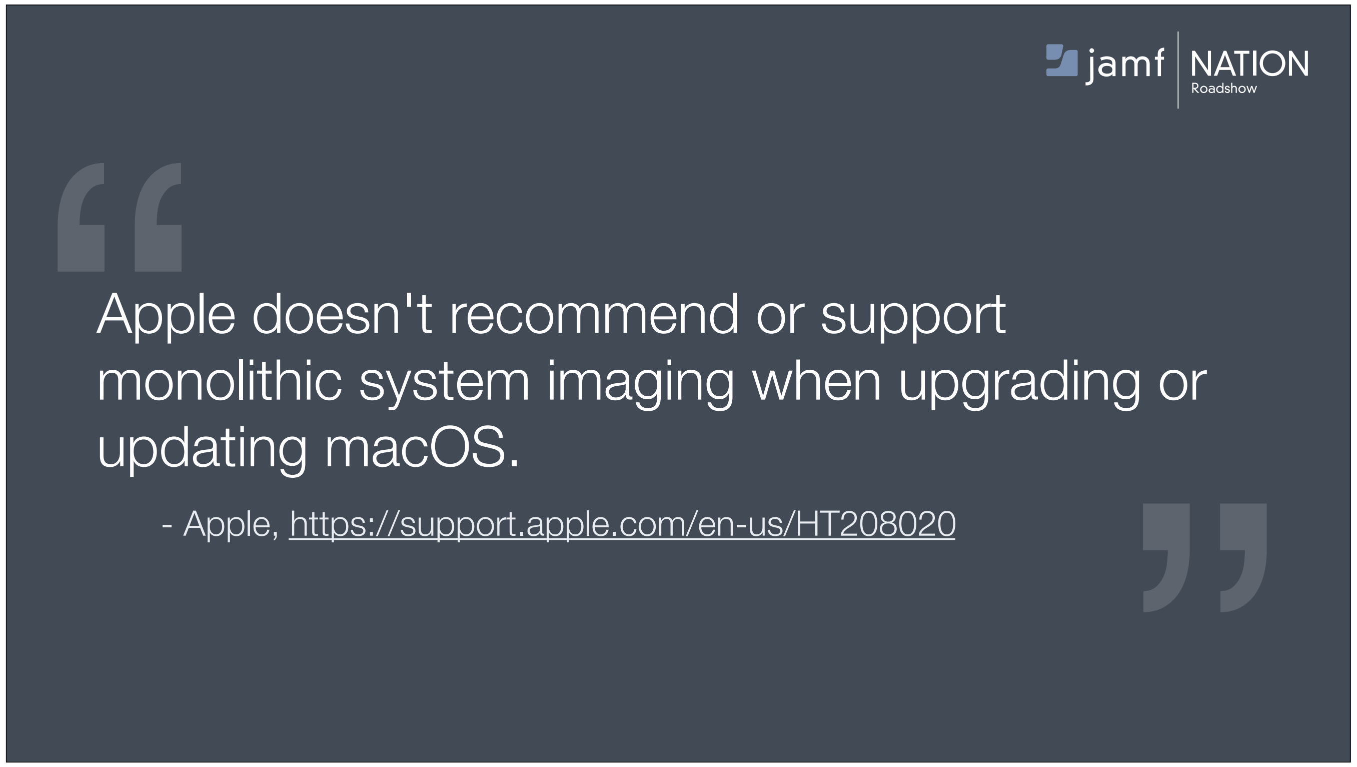
We also deal with version control for each application but that’s a bit beyond the scope of this presentation.

Times were good…

Times were good!



Then High Sierra came along…


jamf | NATION Roadshow

> Apple doesn't recommend or support monolithic system imaging when upgrading or updating macOS.
>
> - Apple, https://support.apple.com/en-us/HT208020

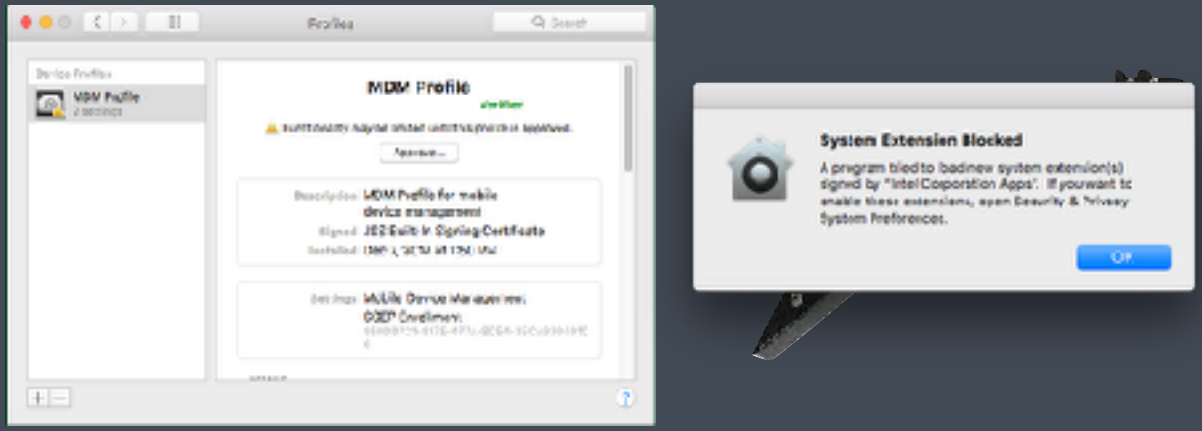
And Apple did this.

APFS and firmware

jamf NATION Roadshow

Because we have a new file system, that needs up to date firmware to boot from. The process of laying down an image doesn’t install firmware.

UAMDM & UAKEL



System Extension Blocked

A program tried to load new system extension(s) signed by "Intel Corporation Apps". If you want to enable these extensions, open Security & Privacy System Preferences.

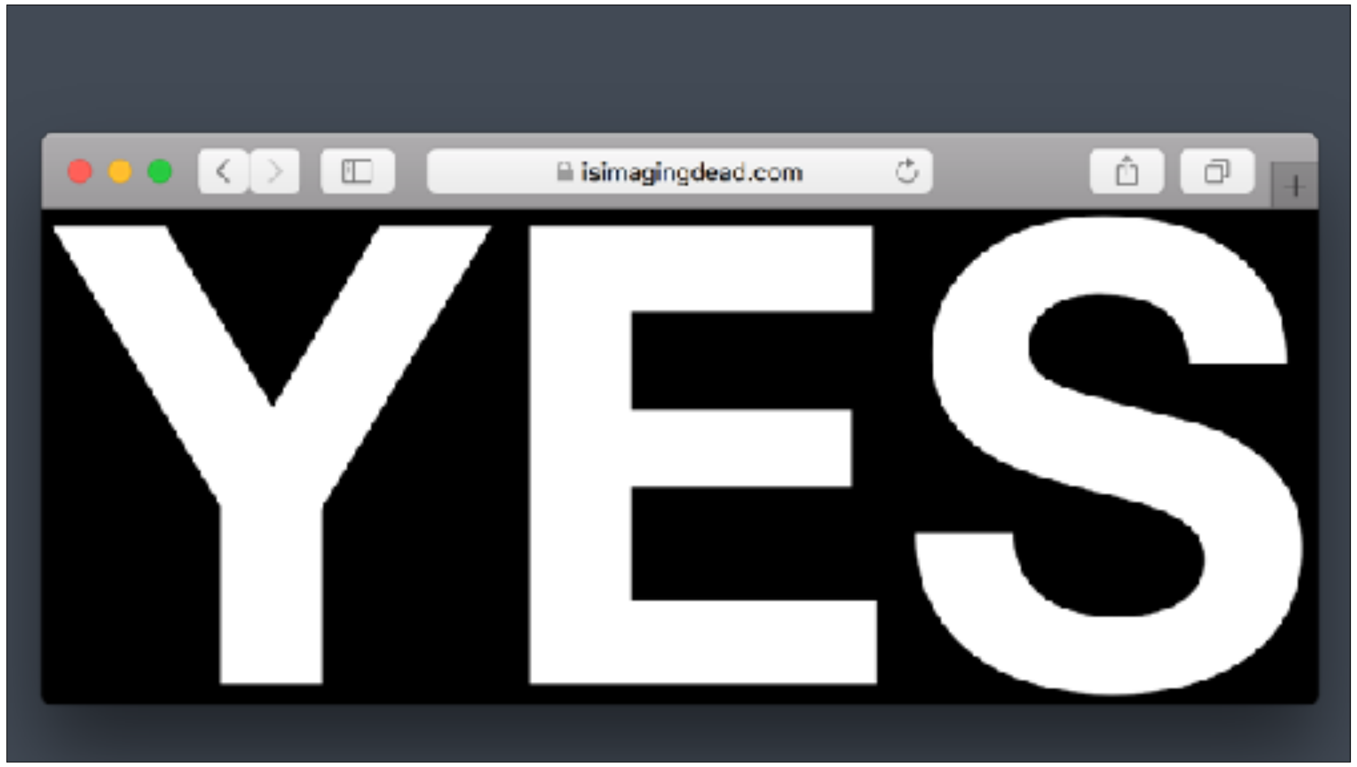
If you image and your Macs are automatically enrolled into your MDM with your imaging tools, you'll have User Approved MDM to contend with. As well as User Approved Kernel Extension Loading.

<C> or Ukulele. You can't even click that Approve button remotely - you have to use a physical mouse on the actual Mac. Once you've approved MDM, you can push a profile to approve specific extensions, or the all the extensions from a specific vendor.

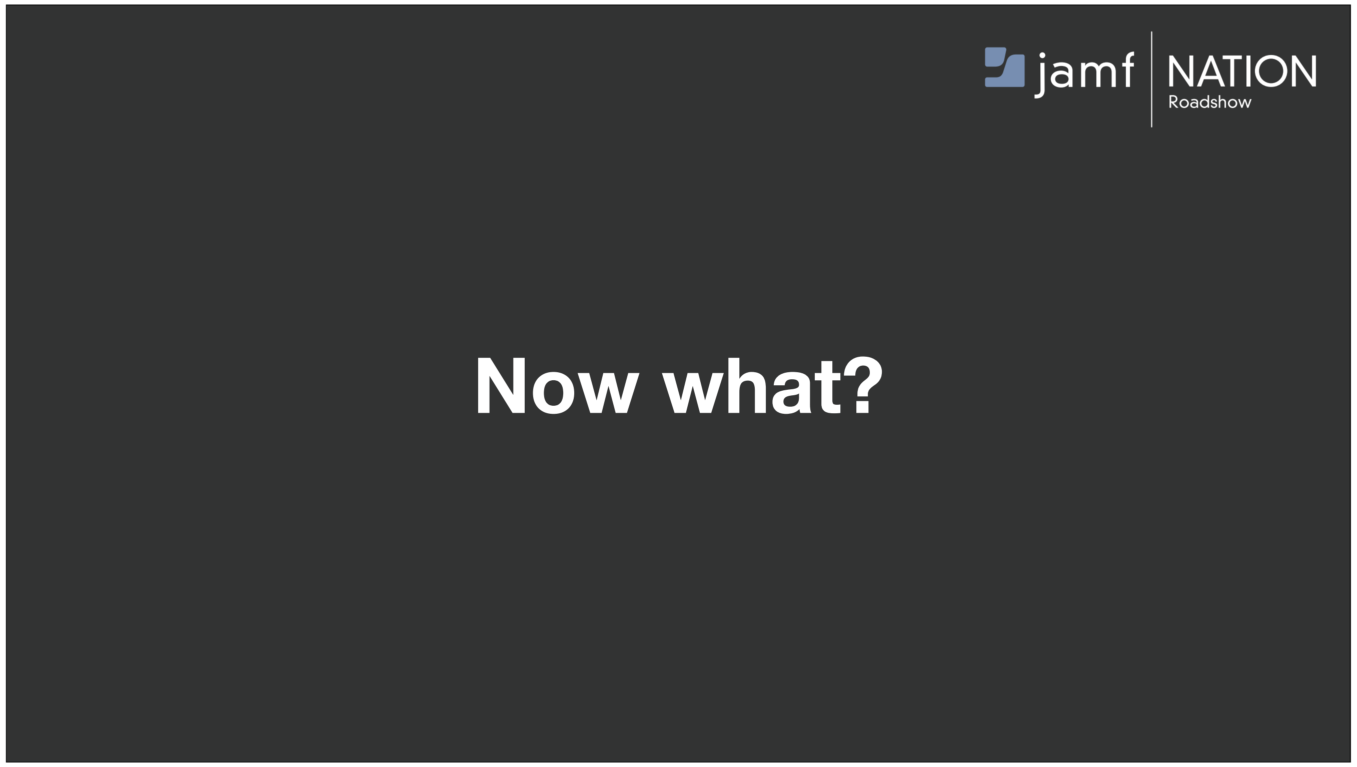
isimagingdead.com

?

So, is imaging dead?

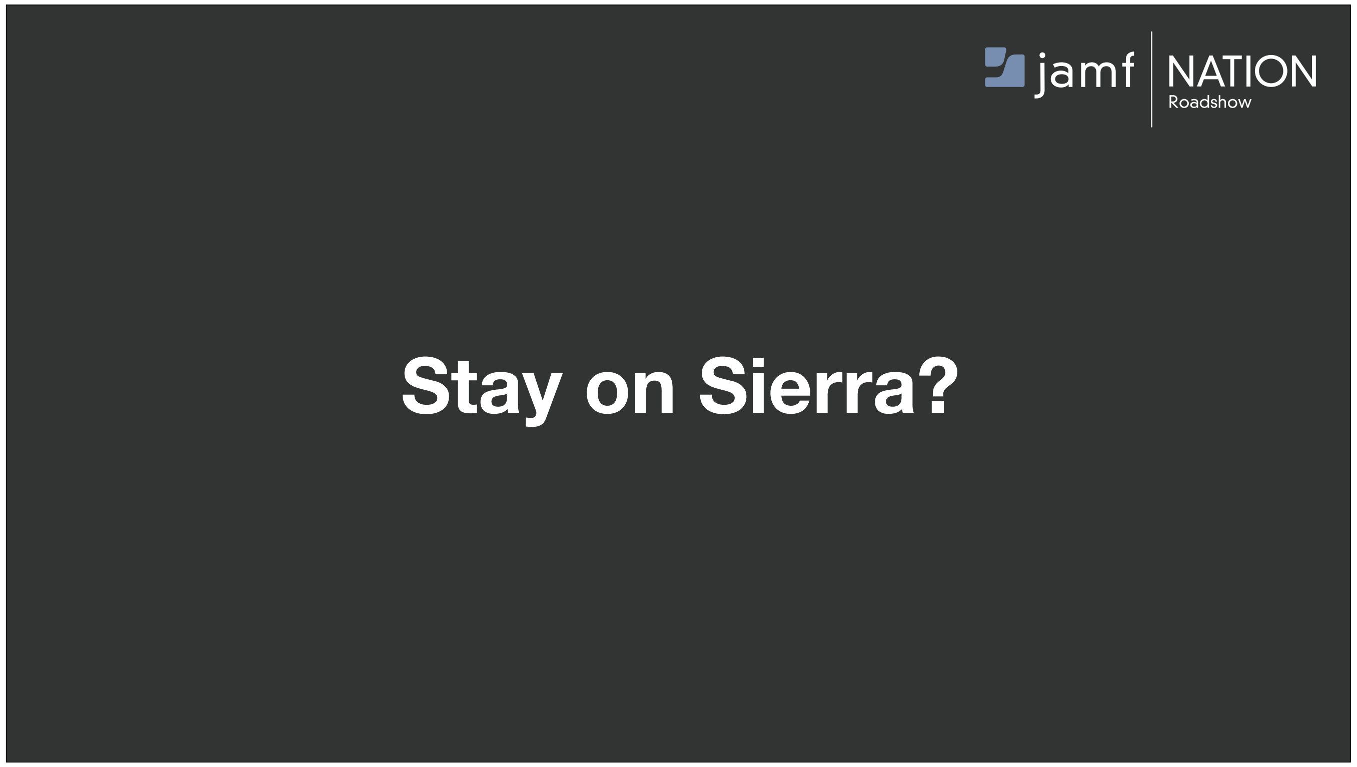


It’s looking that way.


jamf | NATION Roadshow

Now what?

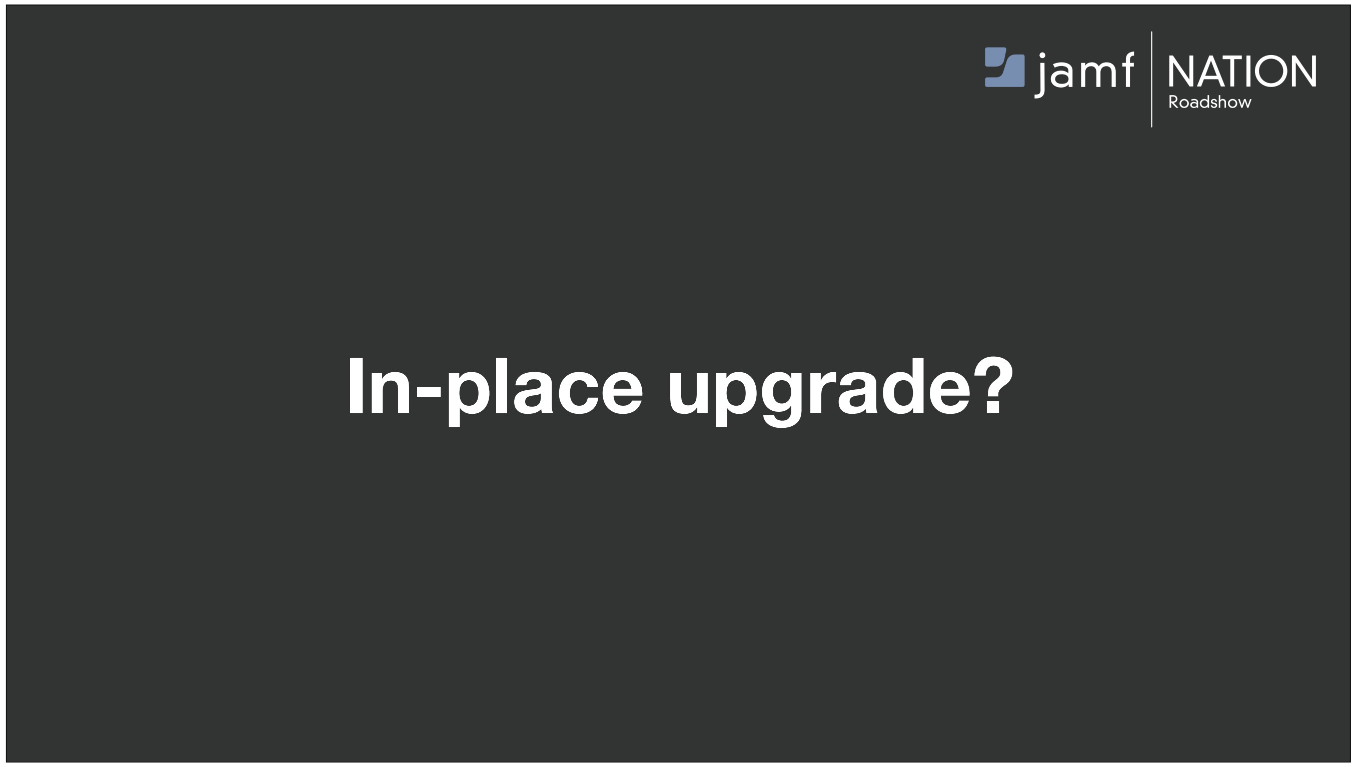
So, what can we do?



Stay on Sierra?

Could we stick with what we have?

Not really. When Apple release new Macs, you probably won't be able to downgrade them. And I don't think this is going to blow over…

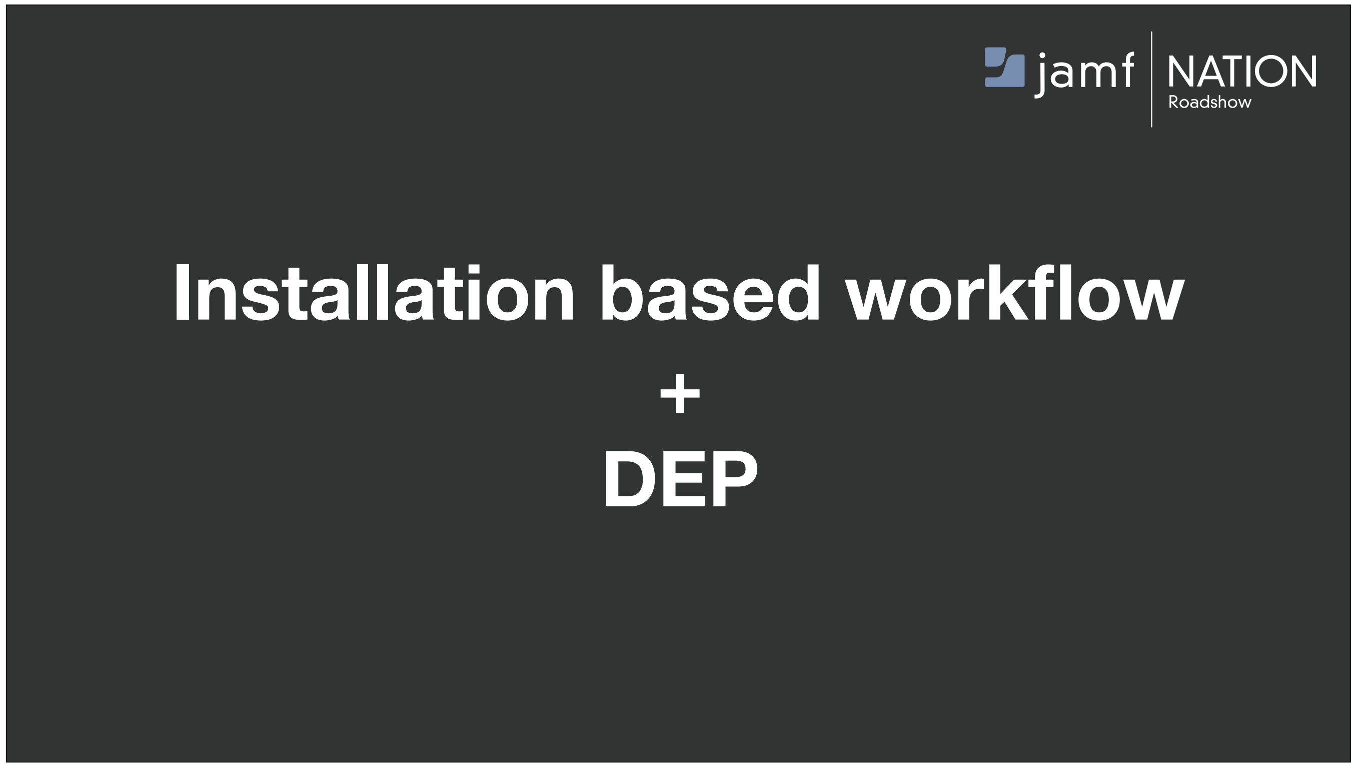

jamf | NATION Roadshow

In-place upgrade?

We could deploy 10.13 as an upgrade. You’d get the right firmware, you wouldn’t have to deal with User Approved MDM and you wouldn’t have to change your imaging workflows.

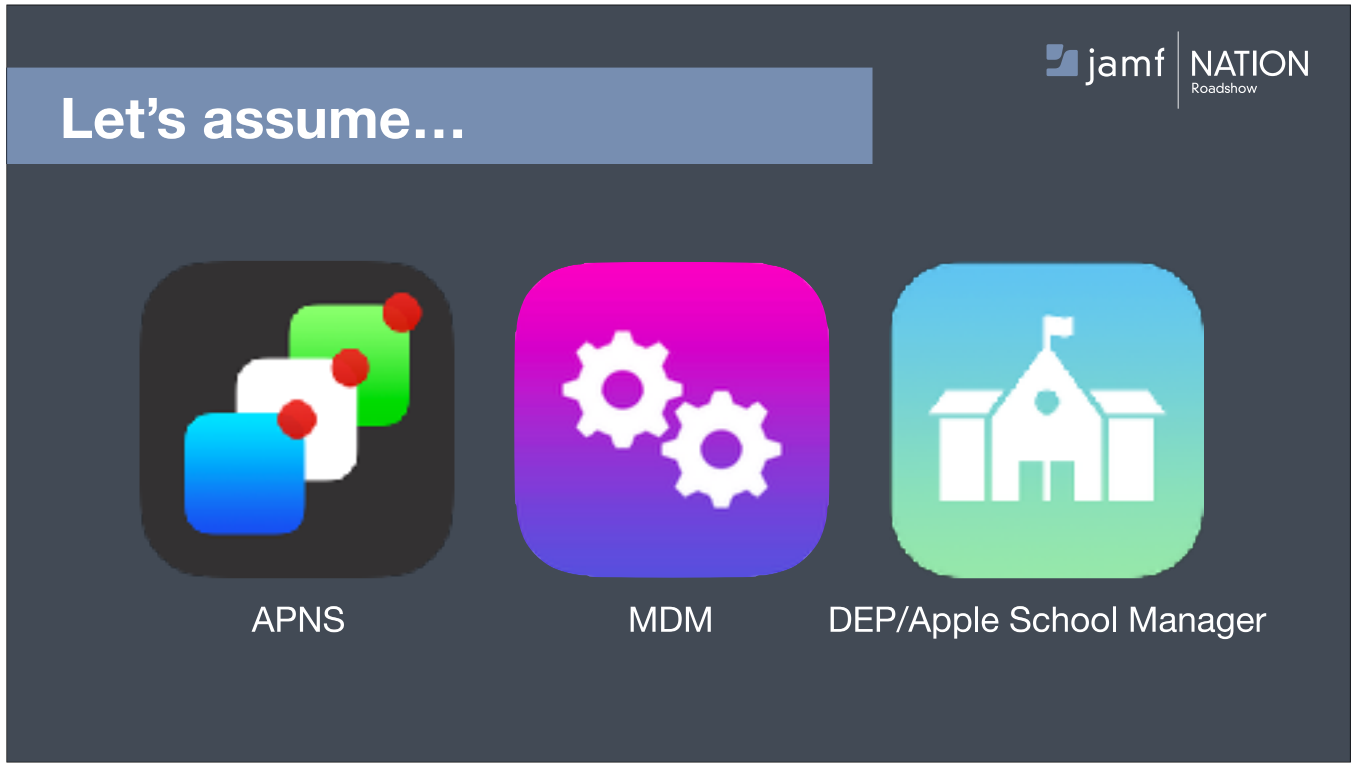
I don’t particularly like this one either.

You end up with bits left over from the previous OS and some Apps that install things all over the place. What about new Macs?



Installation based workflow
+
DEP

So that leads us here. A clean installation of macOS, followed by enrolling it into Jamf through DEP. User Approved MDM is dealt with and we can take care of those kernel extensions with a profile.



Let’s assume…

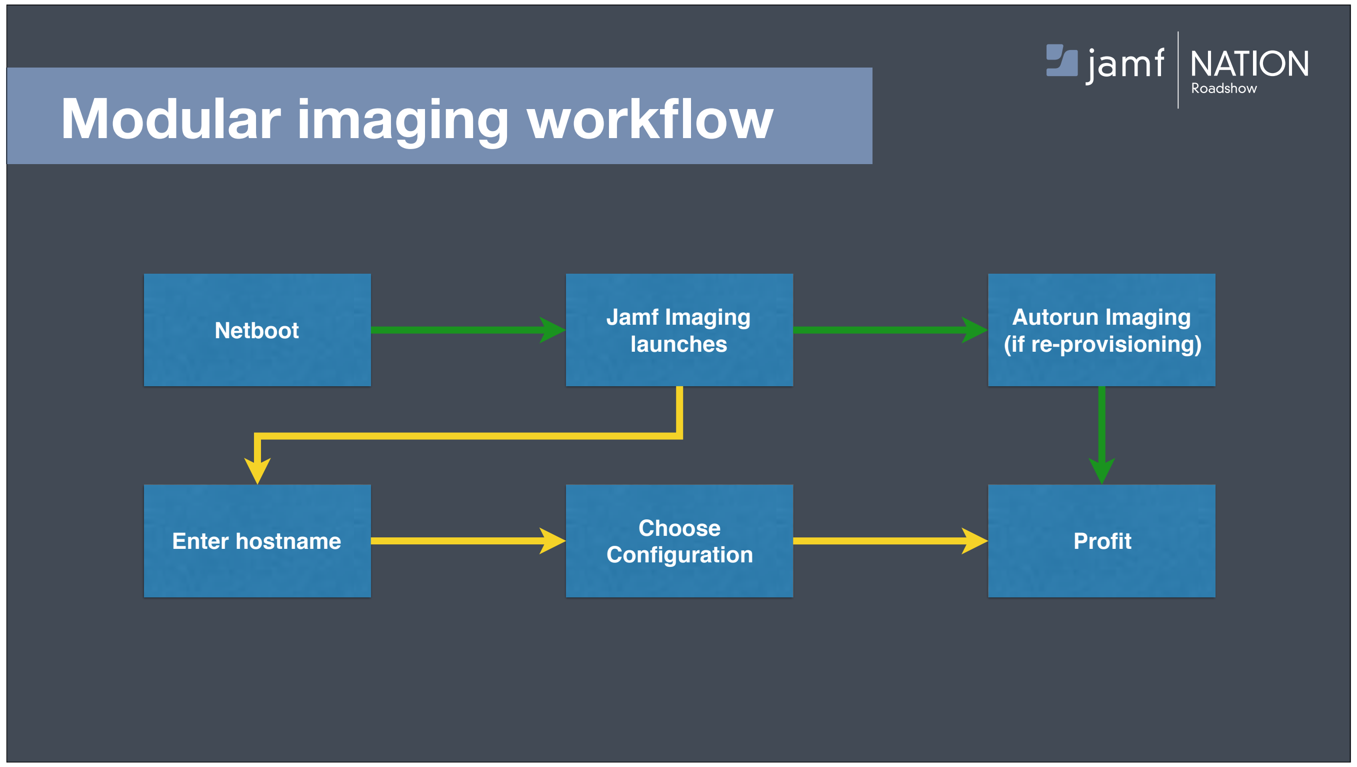
APNS

MDM

DEP/Apple School Manager

At this point I’m going to assume APNS is allowed over the network and that we’re using MDM.

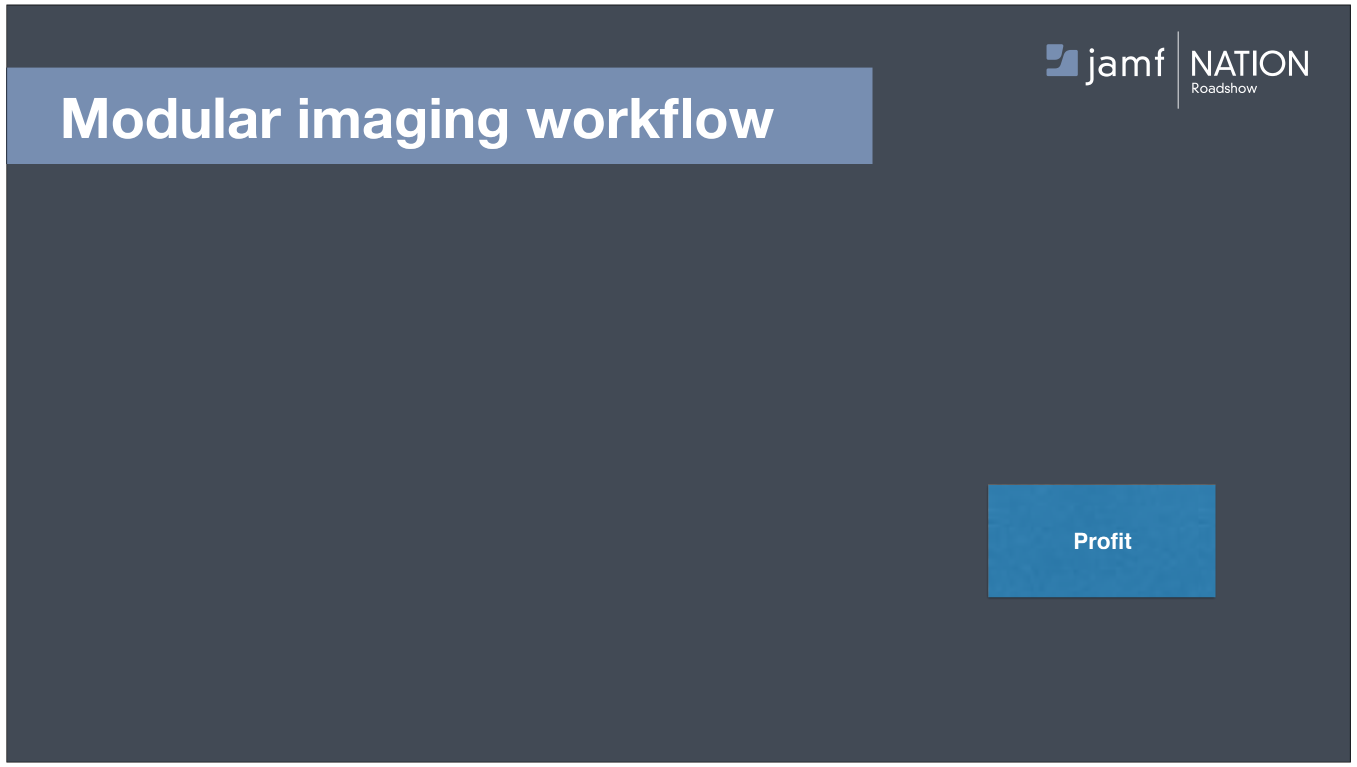
I’m also going to assume that we’ve just set up Apple School Manager (or Business Manager if you’re not in education) and that our Macs are added to it.



Modular imaging workflow

Let’s look at the old workflow again.

We need to complete most of the steps first…



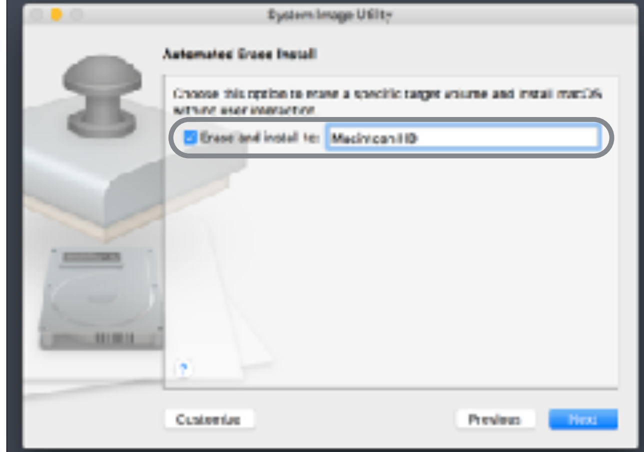
Modular imaging workflow

jamf NATION Roadshow
Profit

…then the OS gets laid down at the end.

But with an install based workflow, we need to get the OS on there at the beginning…

Install a clean macOS


System Image Utility

Automated Erase Install

Choose this option to erase a specific target volume and install macOS without user interaction

☑ Erase and install to: Macintosh HD

```
startosinstall --eraseinstall
```

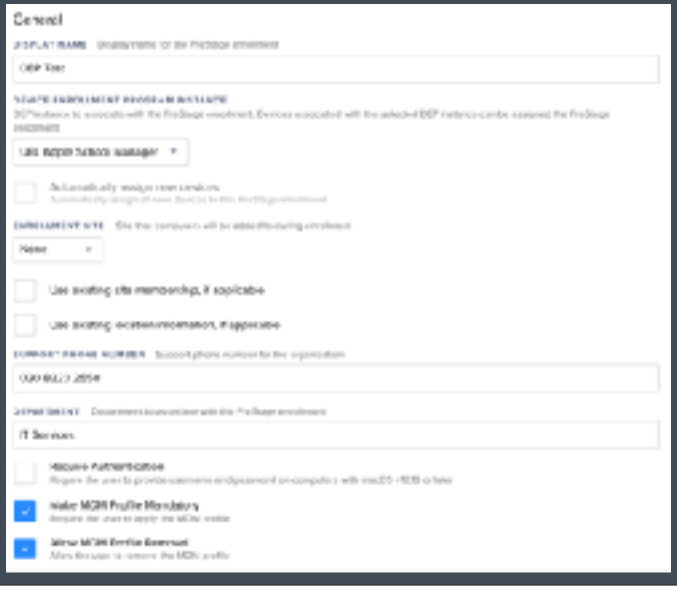
APFS only

Now, with High Sierra 10.13.4, Apple have given us a couple of ways to achieve that.

For the moment, you can still use NetBoot - create a NetInstall set with System Image Utility and set it up to automatically erase and install (this was broken with 10.13 but fixed in the update).

Or with Macs that have APFS volumes, get Apple's installer application onto them and trigger the startosinstall command with a new --eraseinstall flag.

PreStage Enrolment



General

DISPLAY NAME Display name for the PreStage enrollment

DEP Test

[illegible]
DEP instance to associate with the PreStage enrollment. Devices associated with the selected DEP instance can be assigned the PreStage enrollment

[illegible]

- [ ] Automatically assign new devices
  Automatically assign all new devices to this PreStage enrollment

ENROLLMENT SITE Site that computers will be added to during enrollment

None

- [ ] Use existing site membership, if applicable
- [ ] Use existing location information, if applicable

SUPPORT PHONE NUMBER Support phone number for the organization

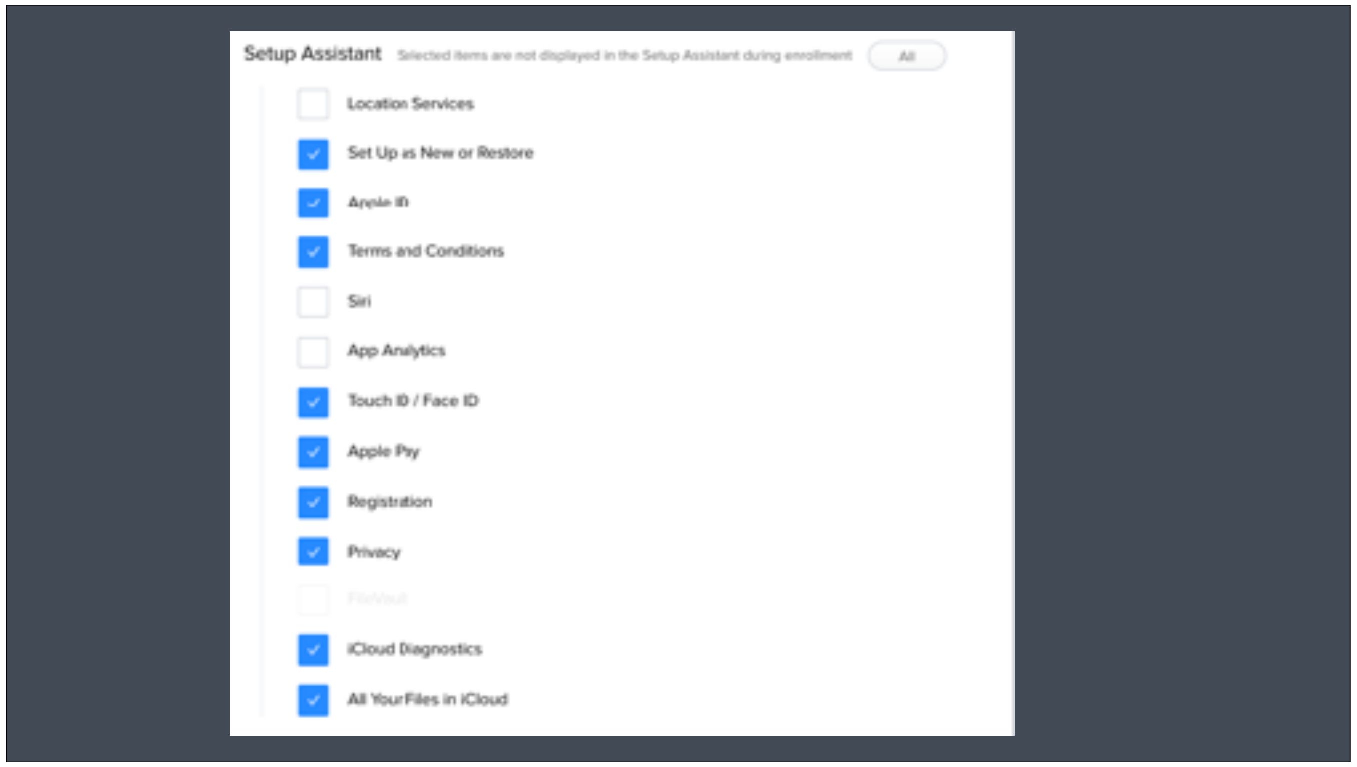
[illegible]

DEPARTMENT Department to associate with the PreStage enrollment

IT Services

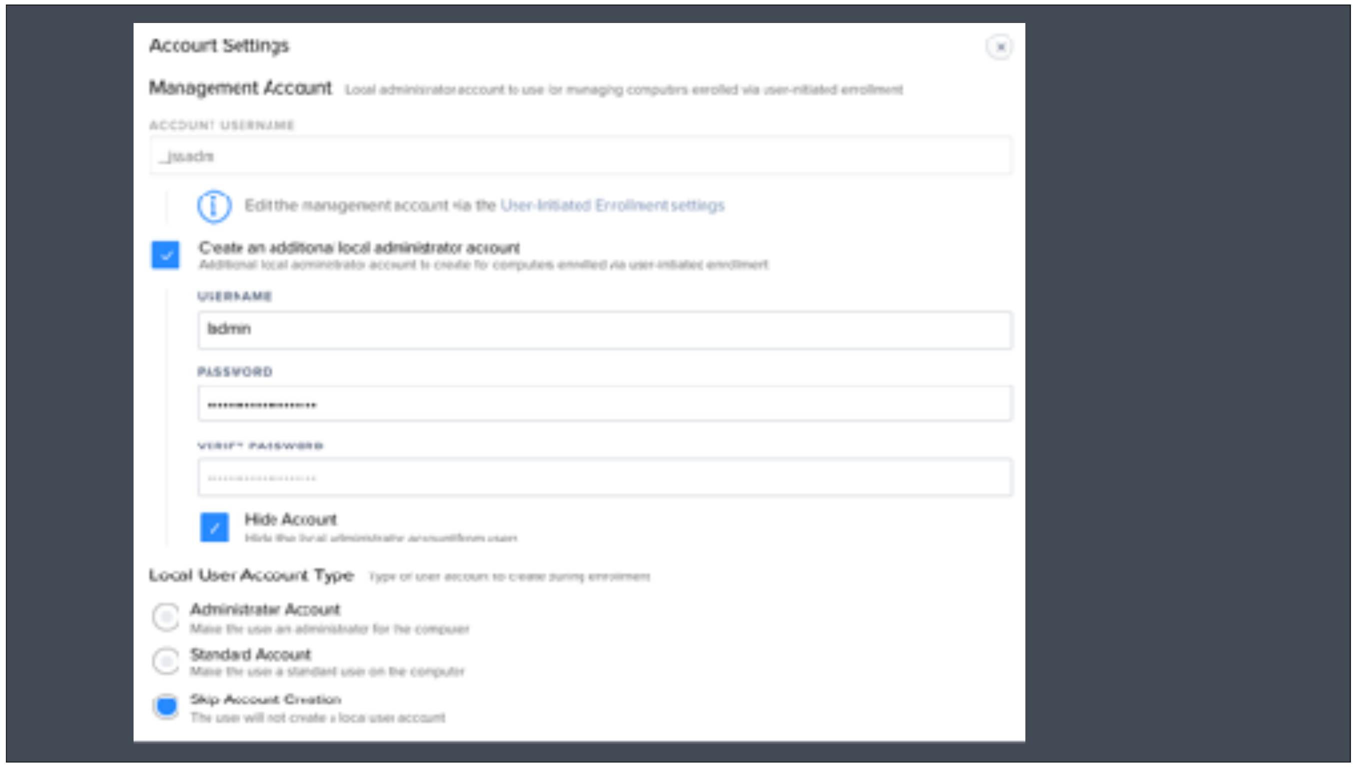
- [ ] Require Authentication
  Require the user to provide username and password on computers with macOS 10.10 or later
- [x] Make MDM Profile Mandatory
  Require the user to apply the MDM profile
- [x] Allow MDM Profile Removal
  Allow the user to remove the MDM profile

Next, You’d need to set up a PreStage Enrolment in Jamf


Setup Assistant Selected items are not displayed in the Setup Assistant during enrollment

- [ ] Location Services
- [✓] Set Up as New or Restore
- [✓] Apple ID
- [✓] Terms and Conditions
- [ ] Siri
- [ ] App Analytics
- [✓] Touch ID / Face ID
- [✓] Apple Pay
- [✓] Registration
- [✓] Privacy
- [ ] FileVault
- [✓] iCloud Diagnostics
- [✓] All YourFiles in iCloud

…this can skip the parts of the Setup Assistant want to



Account Settings

Management Account — Local administrator account to use for managing computers enrolled via user-initiated enrollment

ACCOUNT USERNAME: _jssadm

Edit the management account via the User-Initiated Enrollment settings

[✓] Create an additional local administrator account
Additional local administrator account to create for computers enrolled via user-initiated enrollment

USERNAME: ladmin

PASSWORD: ••••••••••••••••••

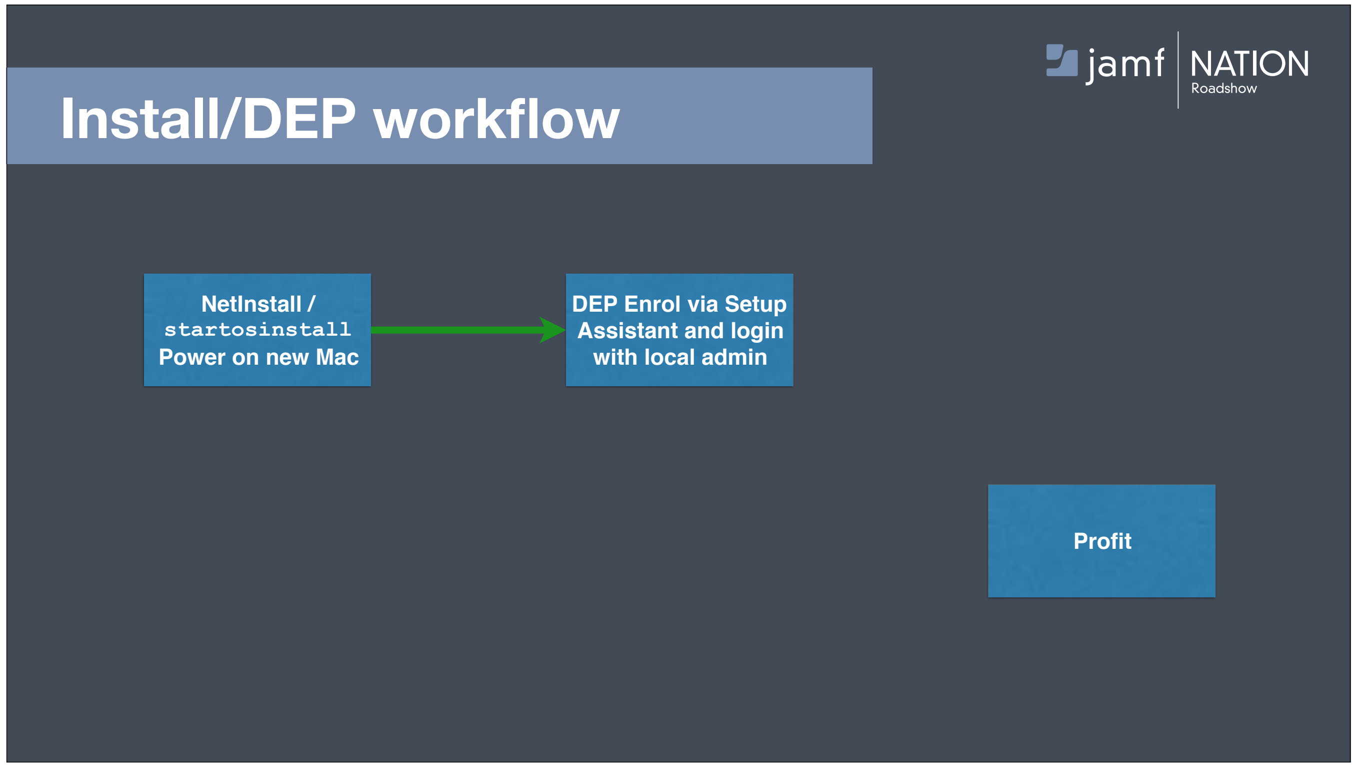
VERIFY PASSWORD: ••••••••••••••••••

[✓] Hide Account
Hide the local administrator account from users

Local User Account Type — Type of user account to create during enrollment

- Administrator Account — Make the user an administrator for the computer
- Standard Account — Make the user a standard user on the computer
- [✓ Skip Account Creation] — The user will not create a local user account

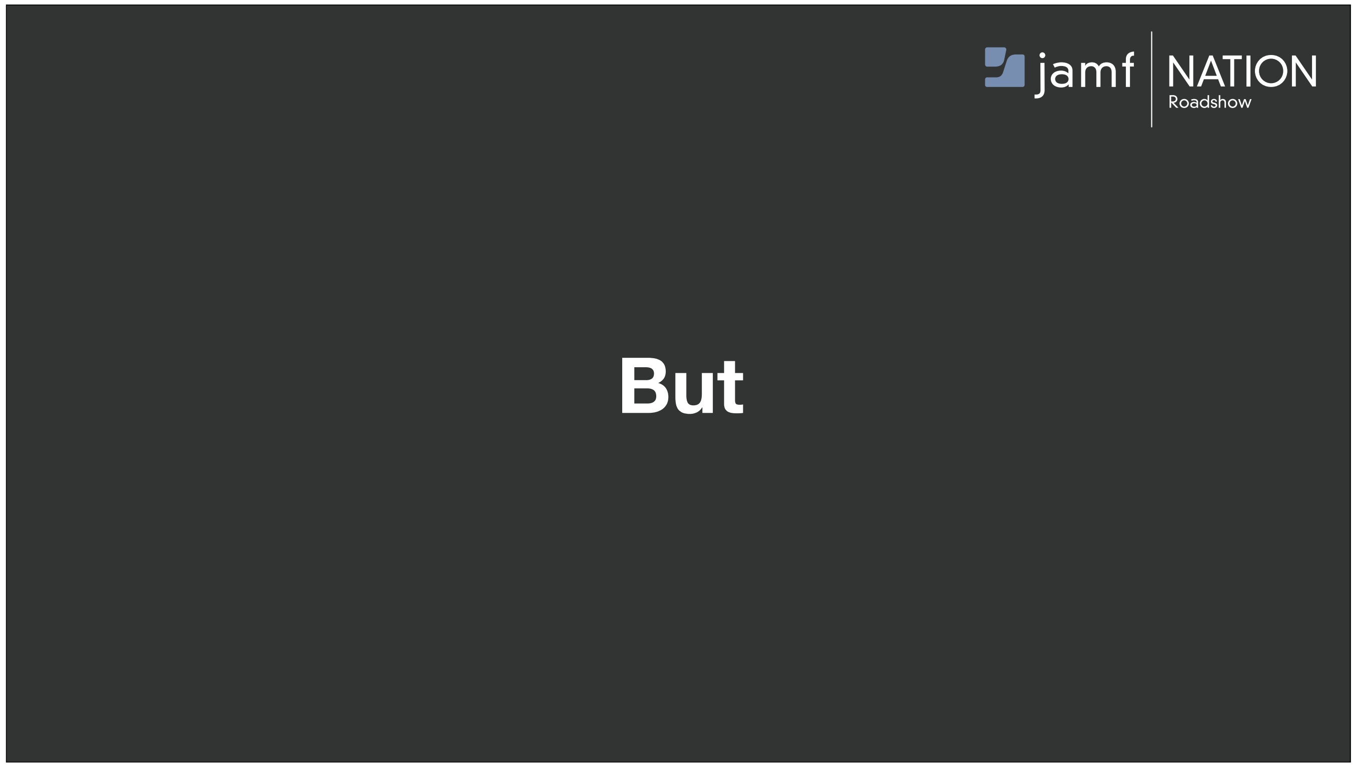
…and create a local administrator account



Install/DEP workflow

jamf NATION Roadshow
NetInstall / startosinstall Power on new Mac
DEP Enrol via Setup Assistant and login with local admin
Profit

In terms of our workflow, we get this far…
macOS gets installed and you can log in with the local admin account.


jamf NATION
Roadshow

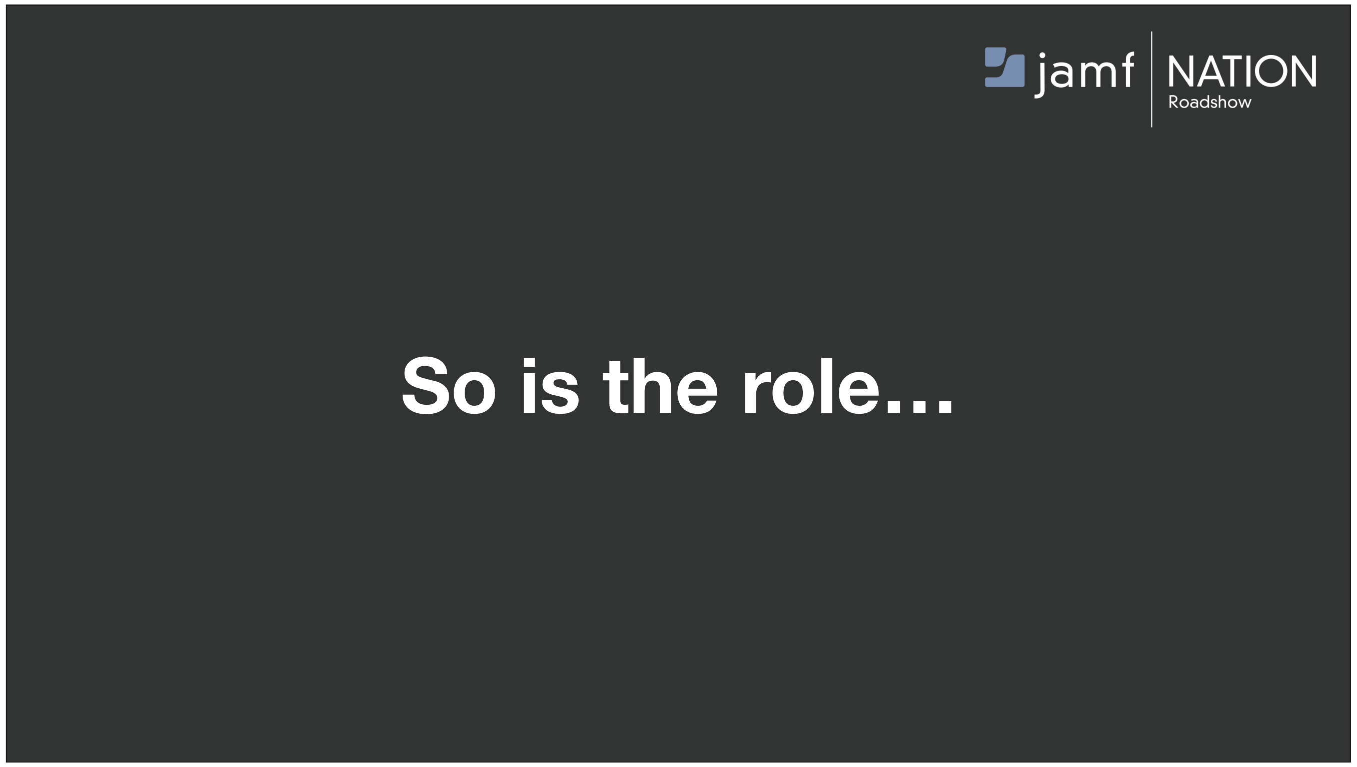
But

However…

jamf | NATION Roadshow

The hostname is important to us…

We still want that hostname.

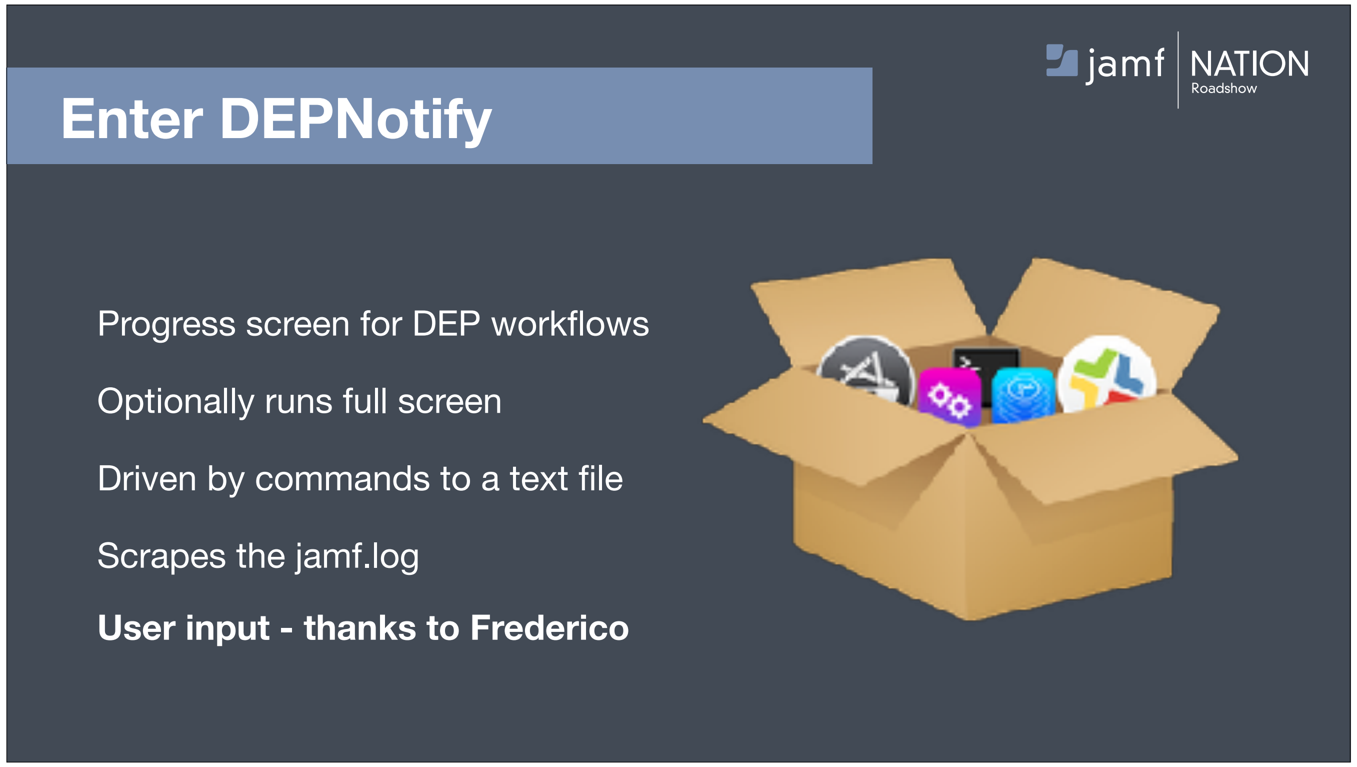


So is the role…

And we still want to set that role.

Because we still want to use our original smart group and policy workflows to determine which software gets installed.

So we need a way to collect that. Apple’s Setup Assistant doesn’t give us a way. But there is a tool that does…



Enter DEPNotify

Progress screen for DEP workflows

Optionally runs full screen

Driven by commands to a text file

Scrapes the jamf.log

User input - thanks to Frederico

DEPNotify, written by Joel Rennich who also wrote NoMAD.

<C> It's a small application that lets you know what's going on during a DEP based enrolment. And it's really easy to use.

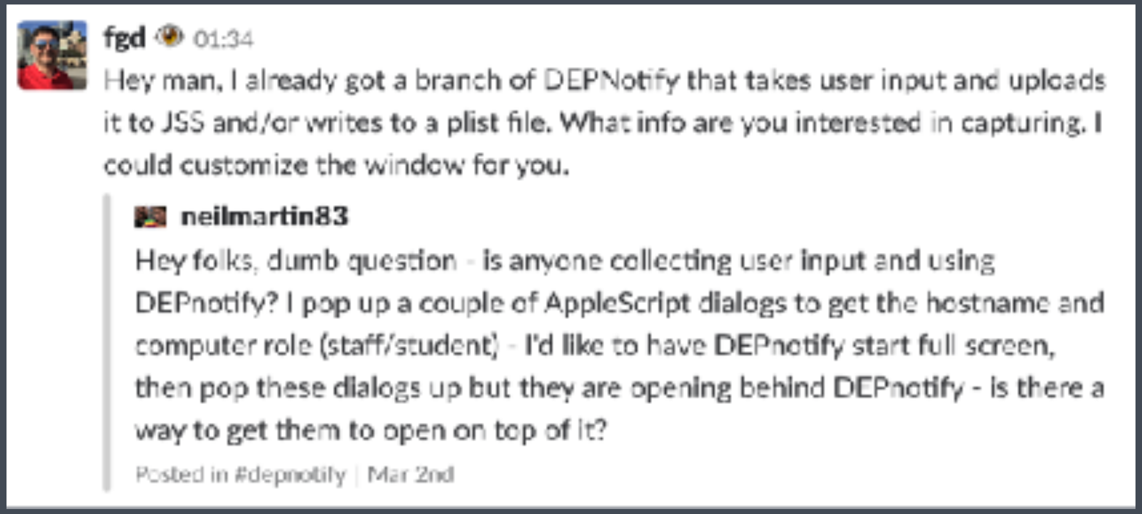
<C> It can cover the whole screen and stop people messing about.

<C> and you can drive it to do different things during provisioning by echoing commands to a text file.

<C> It can also reads the Jamf log and update its status text when it sees policy executions or package installations.

<C> And just now, thanks to Frederico, DEPNotify can take input from the user and write that information to a plist, which we can read with other things.

Meanwhile, in #depnotify



fgd 01:34

Hey man, I already got a branch of DEPNotify that takes user input and uploads it to JSS and/or writes to a plist file. What info are you interested in capturing. I could customize the window for you.

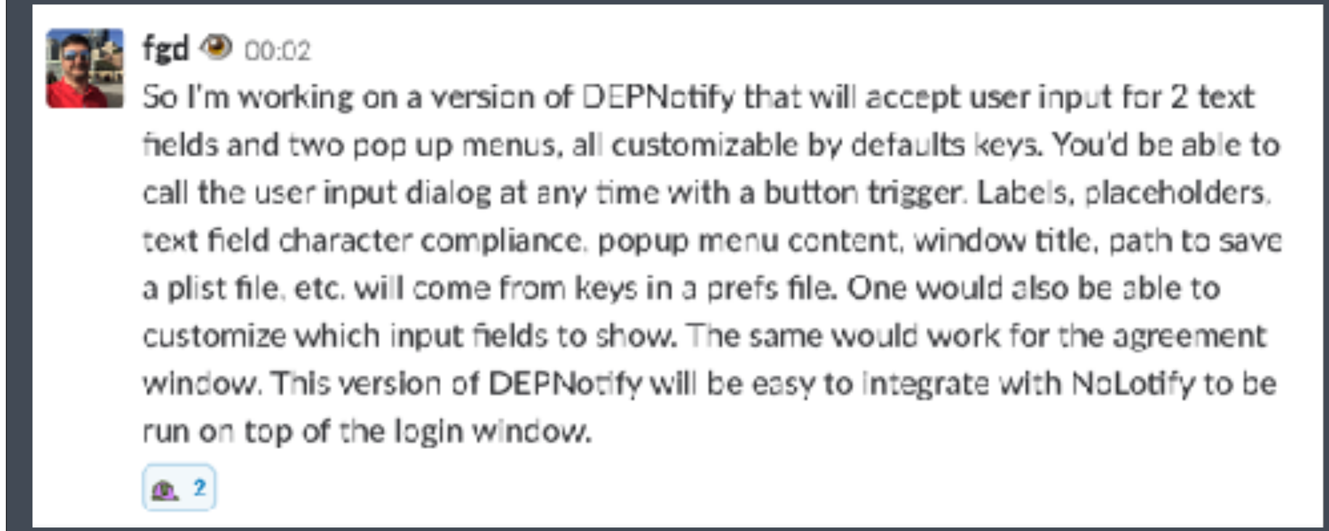
> **neilmartin83**
>
> Hey folks, dumb question - is anyone collecting user input and using DEPnotify? I pop up a couple of AppleScript dialogs to get the hostname and computer role (staff/student) - I'd like to have DEPnotify start full screen, then pop these dialogs up but they are opening behind DEPnotify - is there a way to get them to open on top of it?
>
> Posted in #depnotify | Mar 2nd

Now I think this really highlights how great our community is.

You start with a silly question in the DEPNotify channel on the Mac Admins Slack…

One thing leads to another…

Meanwhile, in #depnotify

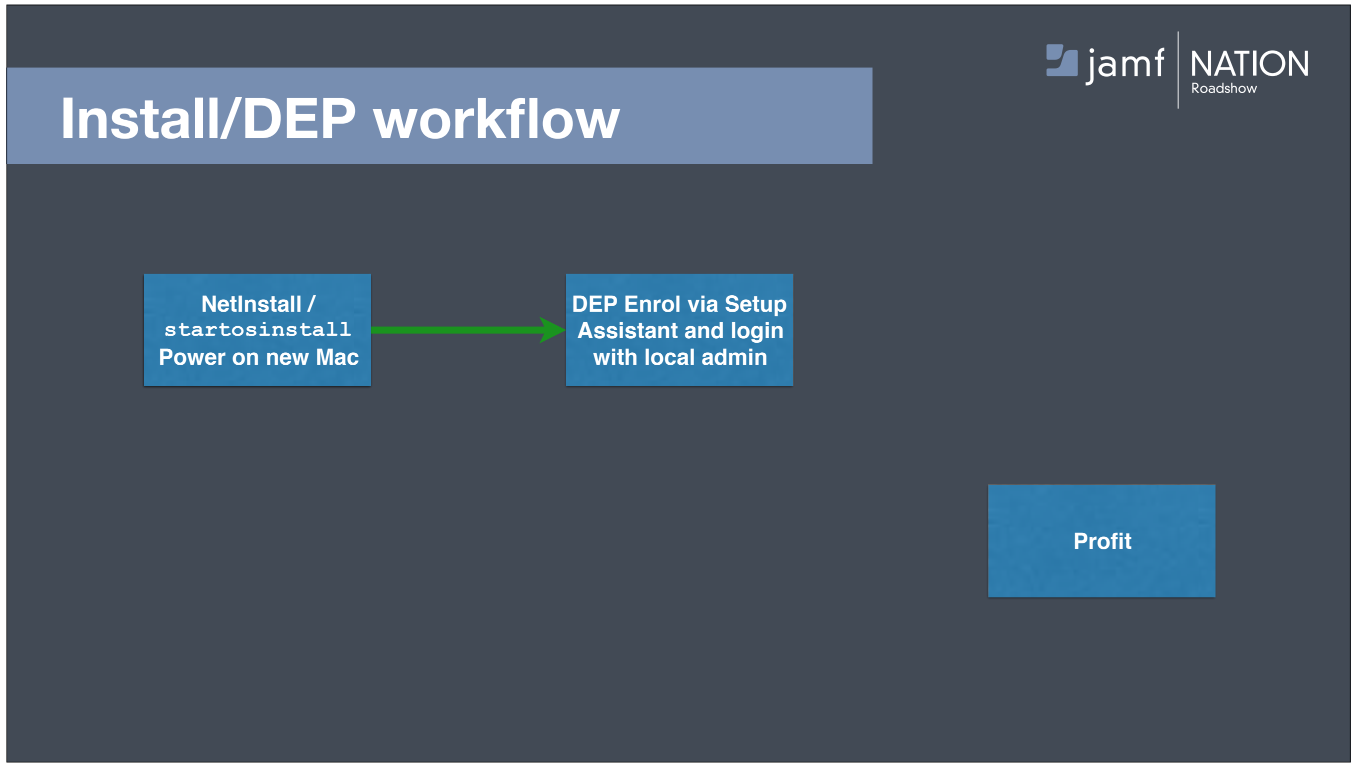


fgd 00:02

So I'm working on a version of DEPNotify that will accept user input for 2 text fields and two pop up menus, all customizable by defaults keys. You'd be able to call the user input dialog at any time with a button trigger. Labels, placeholders, text field character compliance, popup menu content, window title, path to save a plist file, etc. will come from keys in a prefs file. One would also be able to customize which input fields to show. The same would work for the agreement window. This version of DEPNotify will be easy to integrate with NoLoAD to be run on top of the login window.

2

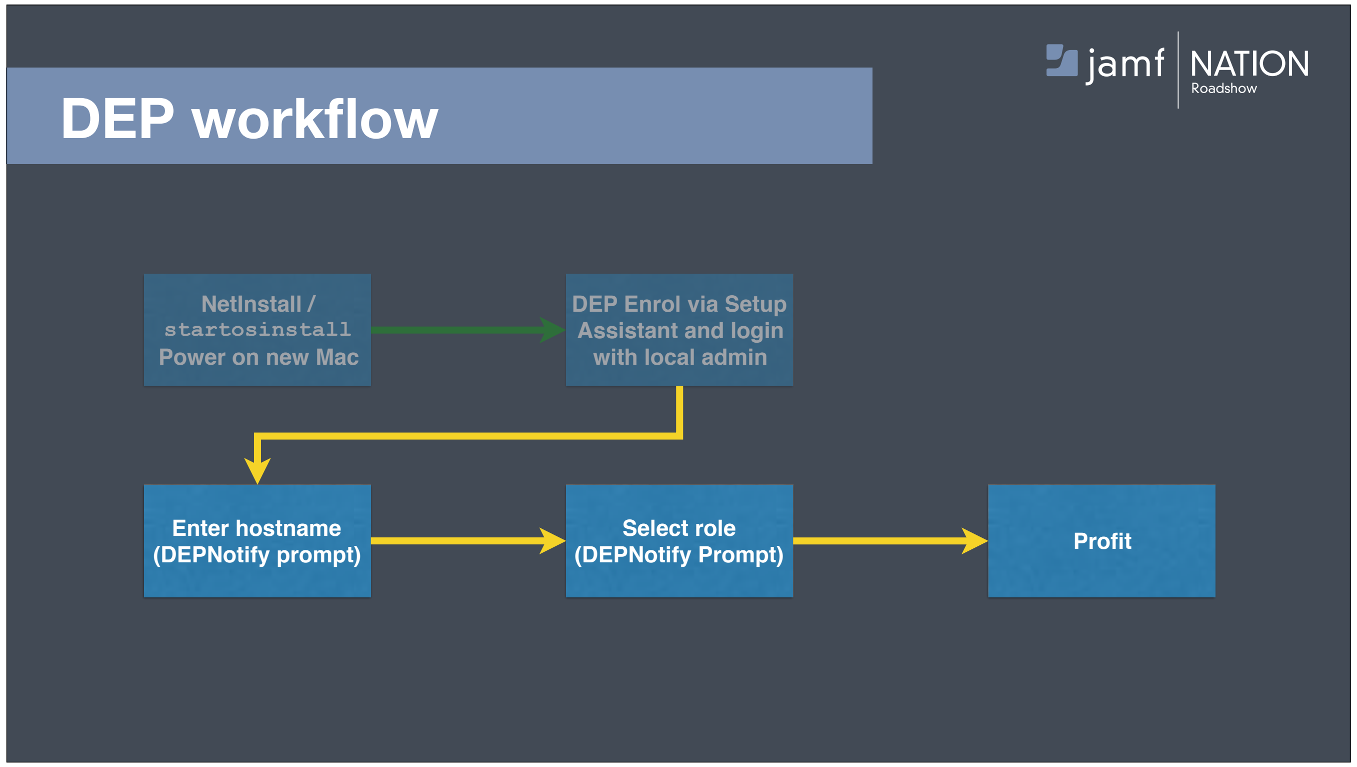
…and after lots of testing, more discussion, and even more testing great things start to happen.



Install/DEP workflow

jamf NATION
Roadshow
NetInstall / startosinstall Power on new Mac
DEP Enrol via Setup Assistant and login with local admin
Profit

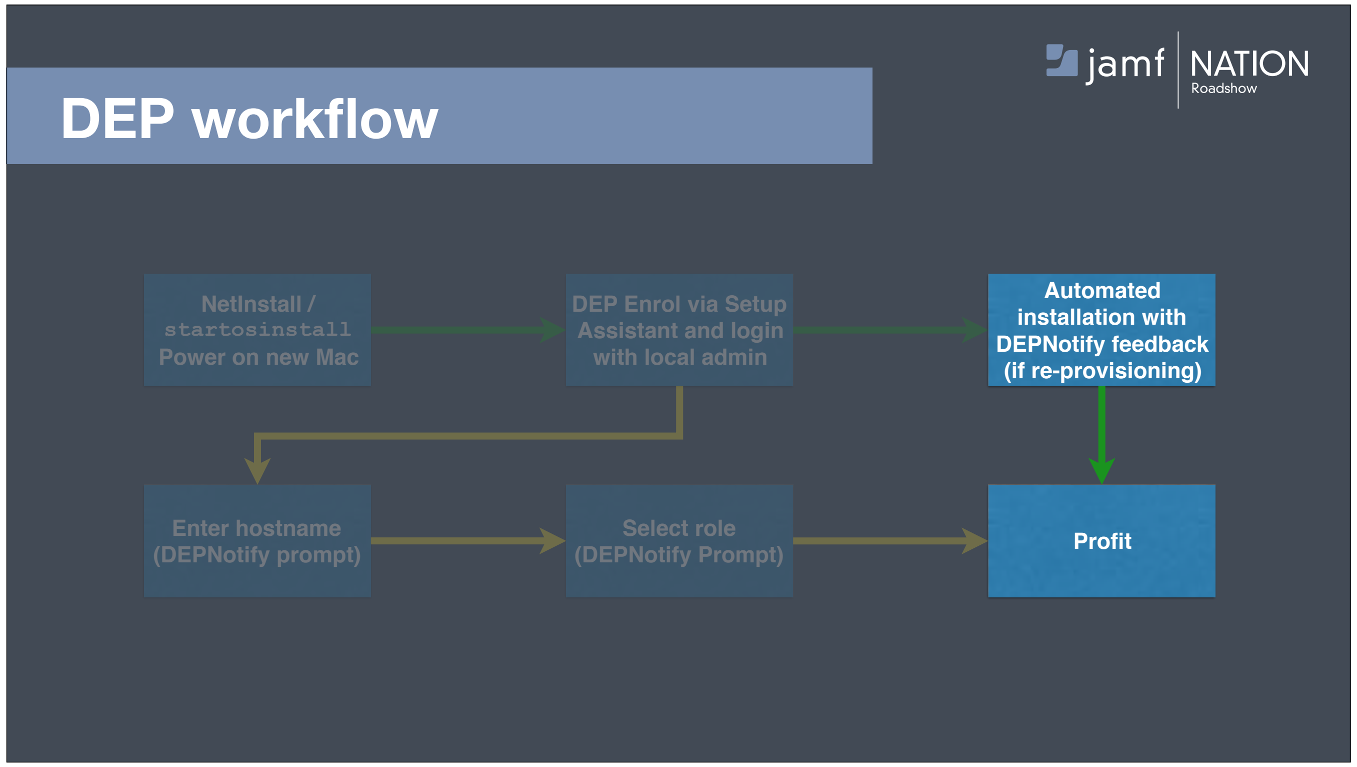
So lets add the missing pieces to our workflow.



DEP workflow

jamf NATION Roadshow
NetInstall / startosinstall Power on new Mac
DEP Enrol via Setup Assistant and login with local admin
Enter hostname (DEPNotify prompt)
Select role (DEPNotify Prompt)
Profit

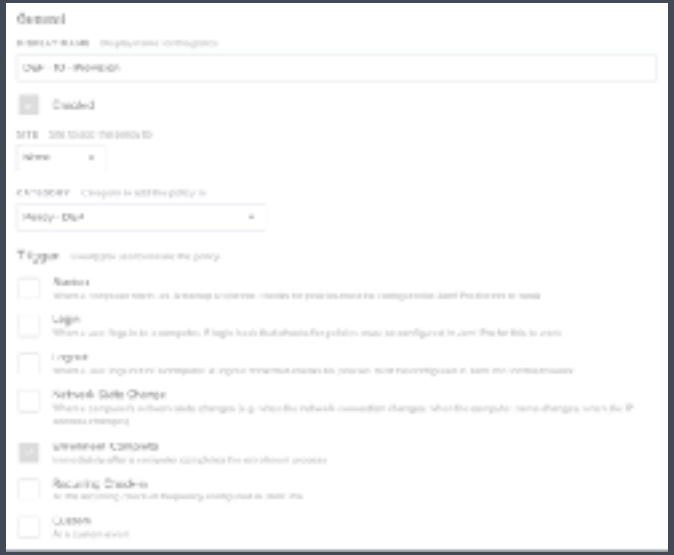
We need to set the hostname and role.


DEP workflow
NetInstall / startosinstall Power on new Mac
DEP Enrol via Setup Assistant and login with local admin
Automated installation with DEPNotify feedback (if re-provisioning)
Enter hostname (DEPNotify prompt)
Select role (DEPNotify Prompt)
Profit

But if we're re-provisioning a Mac that's already in Jamf, we want to skip collecting that data and reuse what we have.

Like we did with autorun imaging.

Policy - Enrolment Complete



General

[illegible]

[illegible]

☑ Enabled

SITE [illegible]

[illegible]

CATEGORY [illegible]

[illegible]

Trigger [illegible]

☐ Startup
[illegible]

☐ Login
[illegible]

☐ Logout
[illegible]

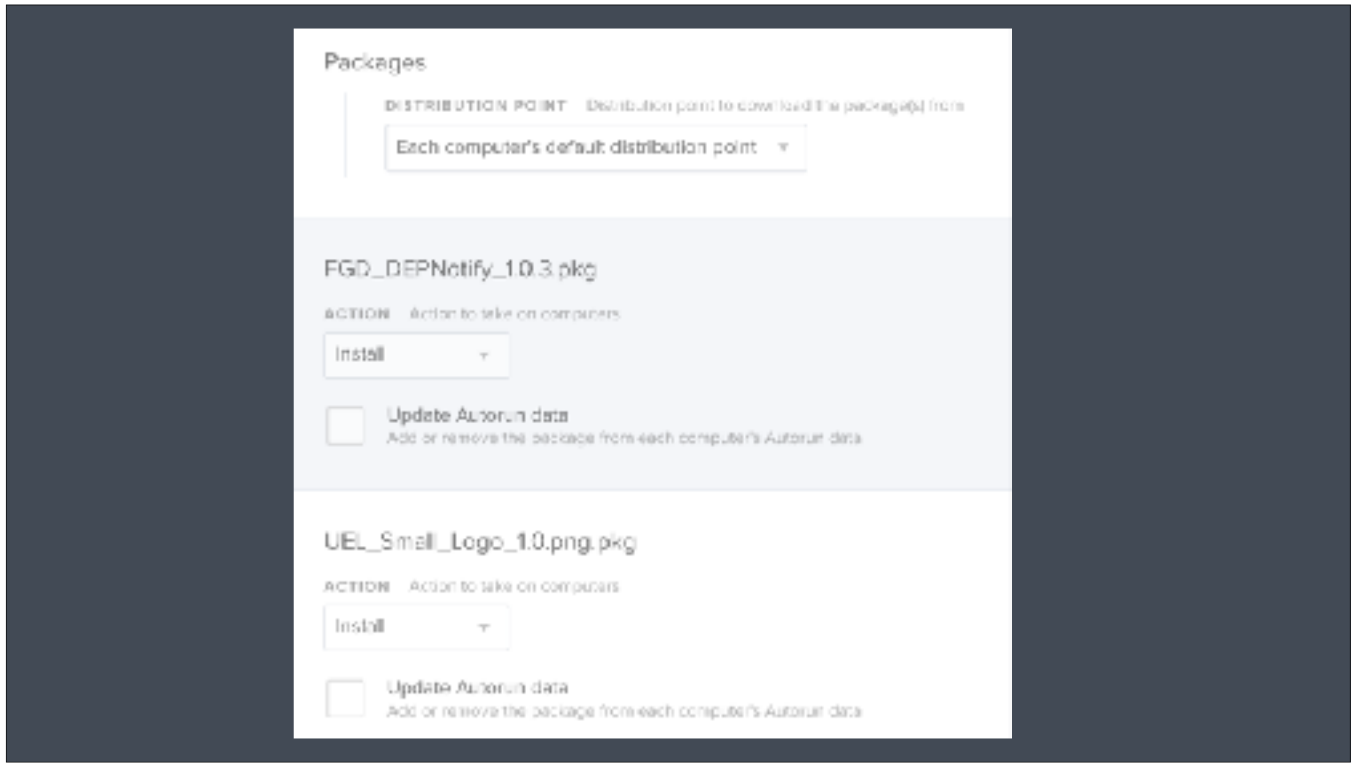
☐ Network State Change
[illegible]

☑ Enrolment Complete
[illegible]

☐ Recurring Check-in
[illegible]

☐ Custom
[illegible]

We need a policy that runs on the Enrolment Complete trigger.



Packages

DISTRIBUTION POINT Distribution point to download the package(s) from

Each computer's default distribution point

FGD_DEPNotify_1.0.3.pkg

ACTION Action to take on computers

Install

☐ Update Autorun data
Add or remove the package from each computer's Autorun data

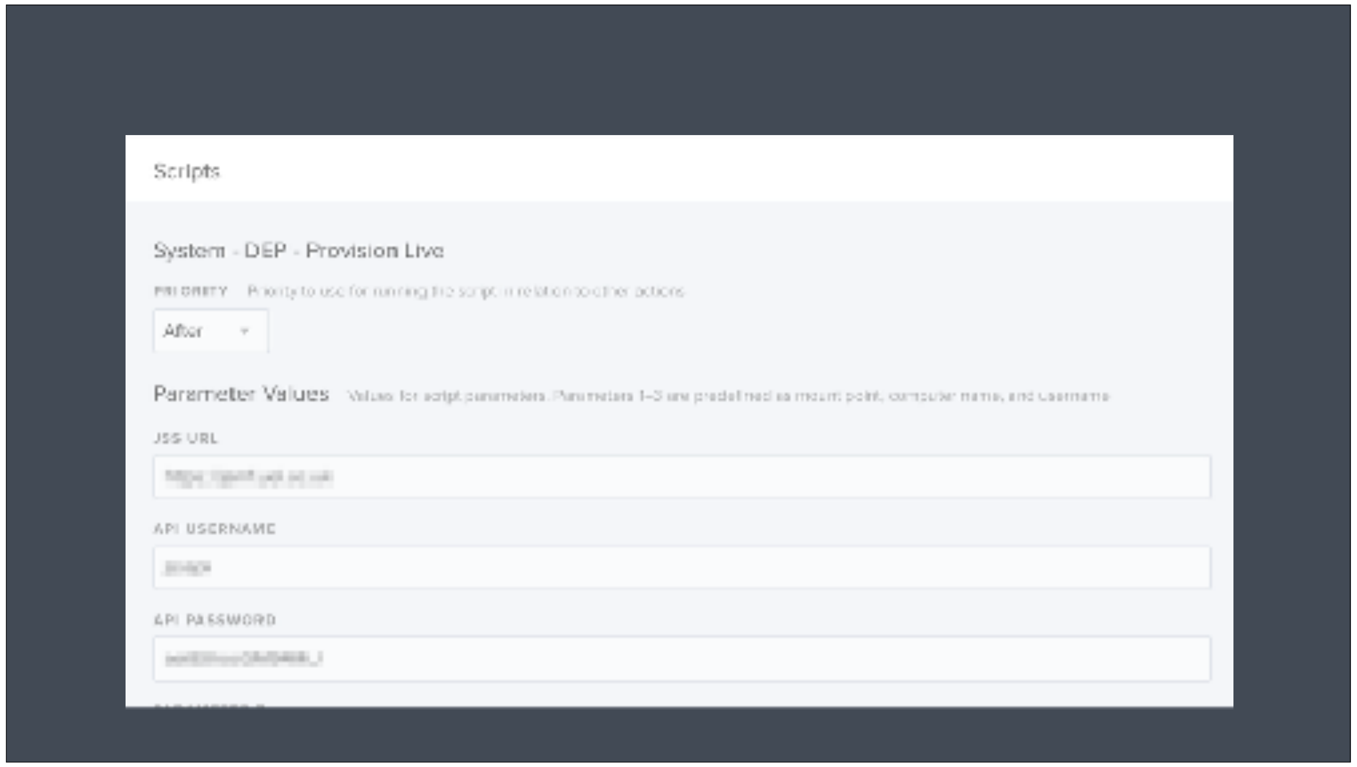
UEL_Small_Logo_1.0.png.pkg

ACTION Action to take on computers

Install

☐ Update Autorun data
Add or remove the package from each computer's Autorun data

This will install DEPNotify and our own branding icon for it.



Scripts

System - DEP - Provision Live

PRIORITY Priority to use for running the script in relation to other actions

After

Parameter Values Values for script parameters. Parameters 1–3 are predefined as mount point, computer name, and username

JSS URL

[redacted]

API USERNAME

[redacted]

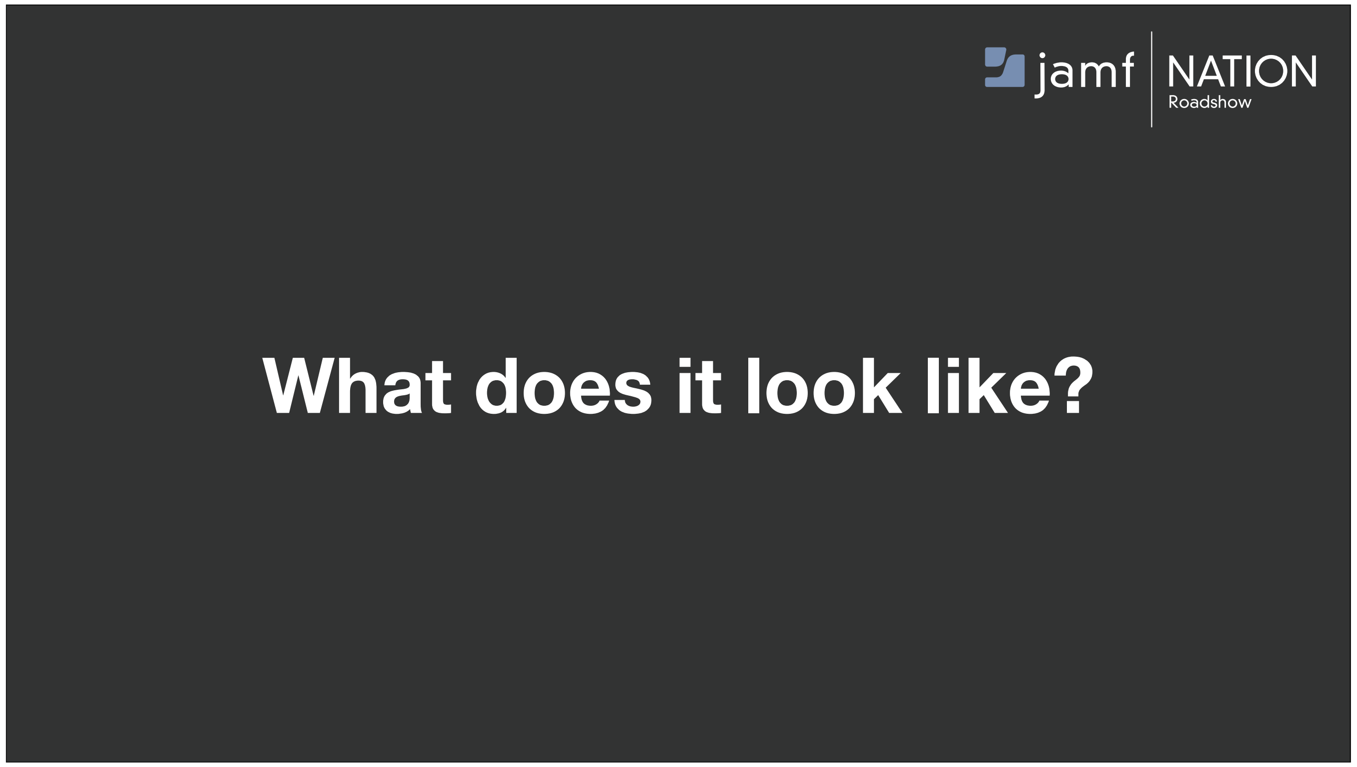
API PASSWORD

[redacted]

And run a script. As you can see here, we're using a few parameters.

In this case, our account credentials for API access.

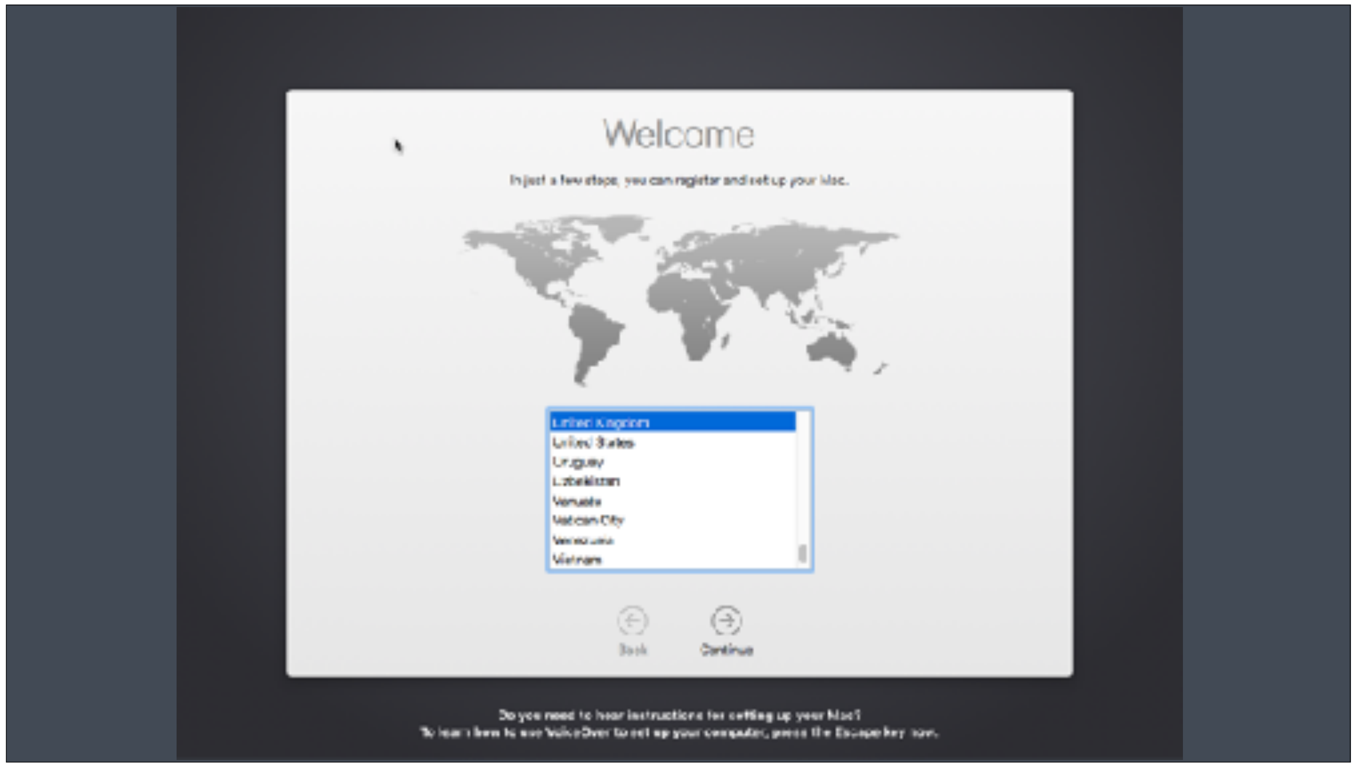
This means you don't have to hard code your API account username and password in the actual script.



What does it look like?

jamf NATION Roadshow

Lets put it all together.



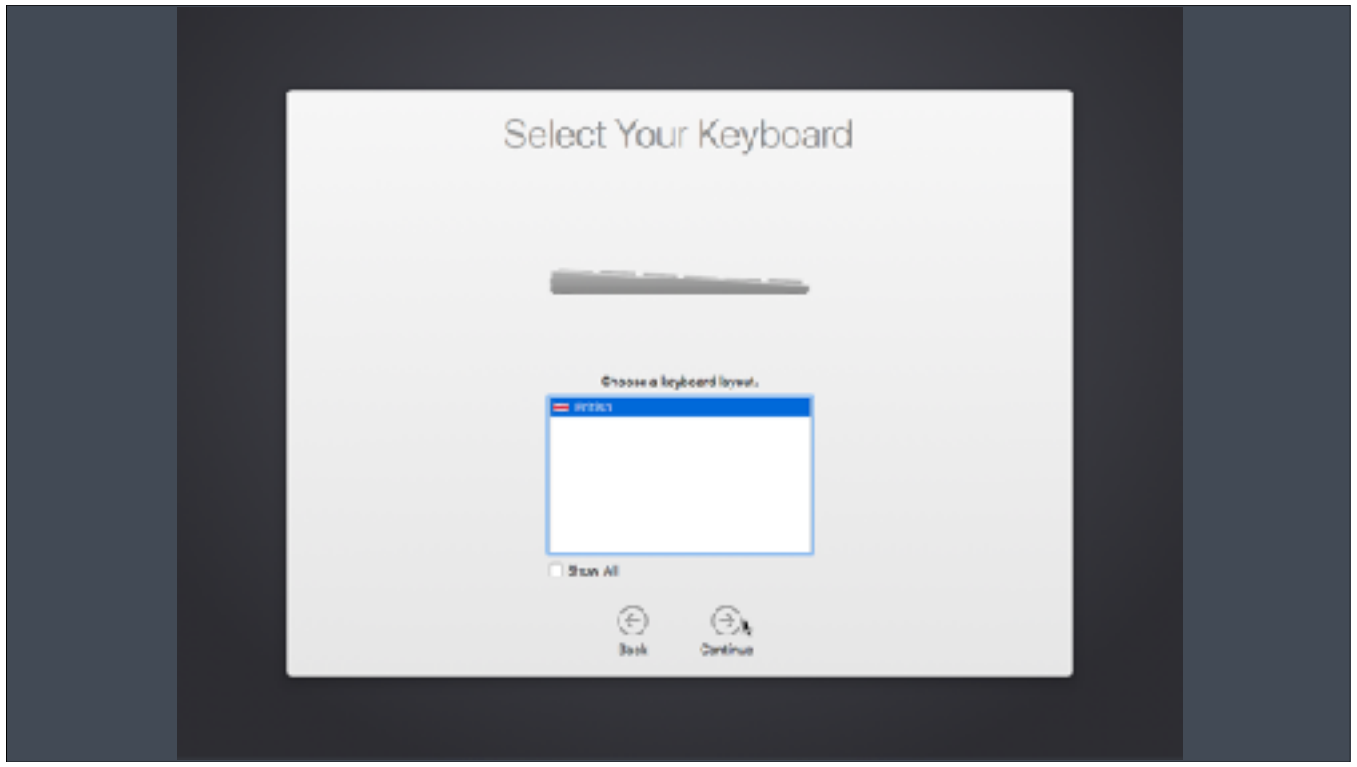
In just a few steps, you can register and set up your Mac.

United Kingdom
United States
Uruguay
Uzbekistan
Vanuatu
Vatican City
Venezuela
Vietnam

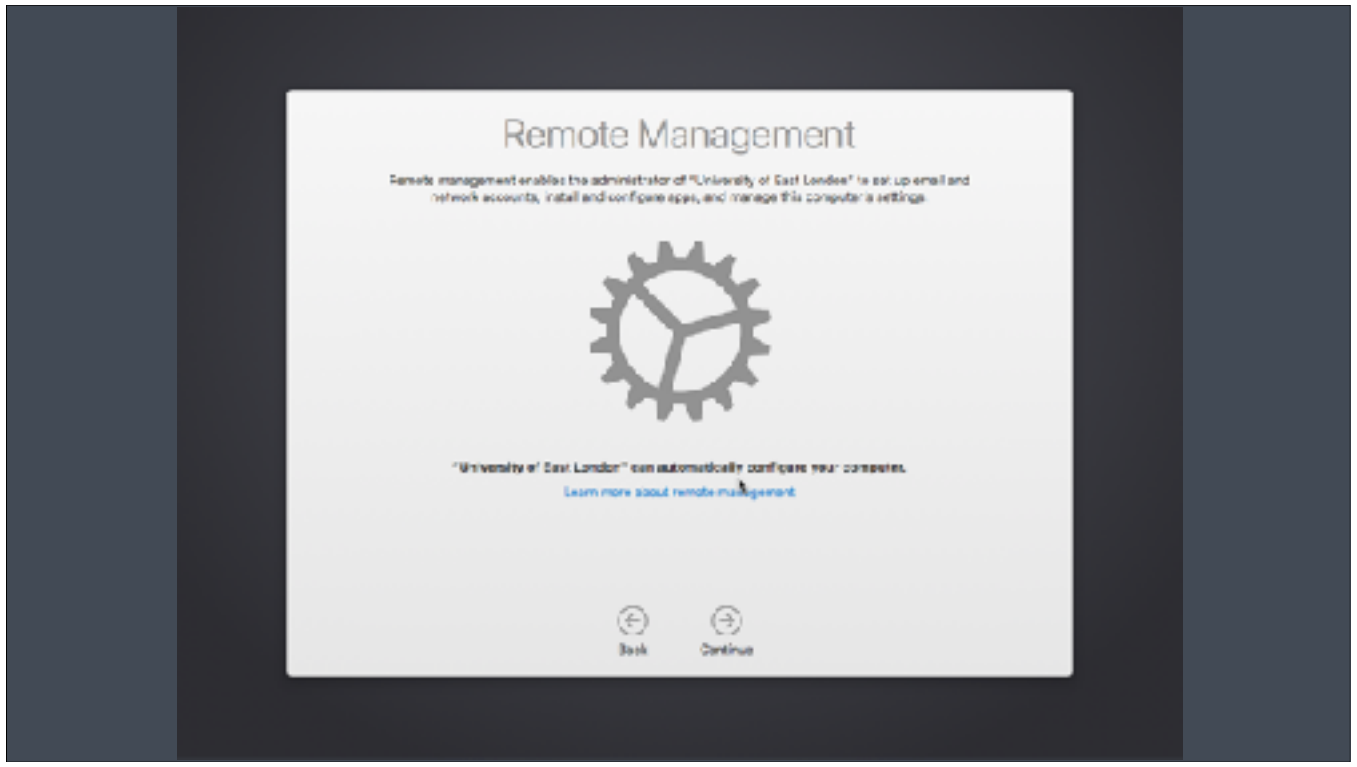
Do you need to hear instructions for setting up your Mac?
To learn how to use VoiceOver to set up your computer, press the Escape key now.

You’ve installed macOS and we’re at the Setup Assistant

Choose your country



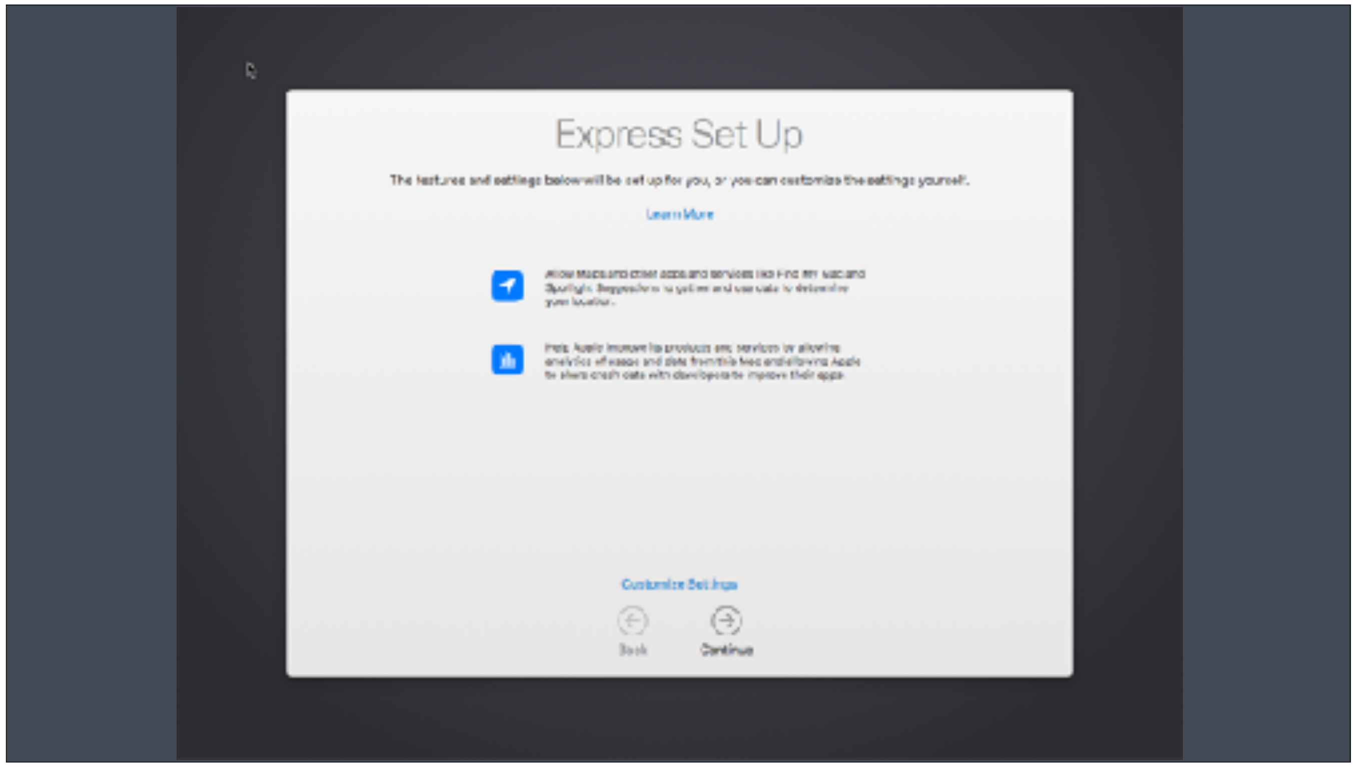
Your keyboard layout


Remote Management

Remote management enables the administrator of "University of East London" to set up email and network accounts, install and configure apps, and manage this computer's settings.

"University of East London" can automatically configure your computer.

Hit the DEP screen



Express Set Up

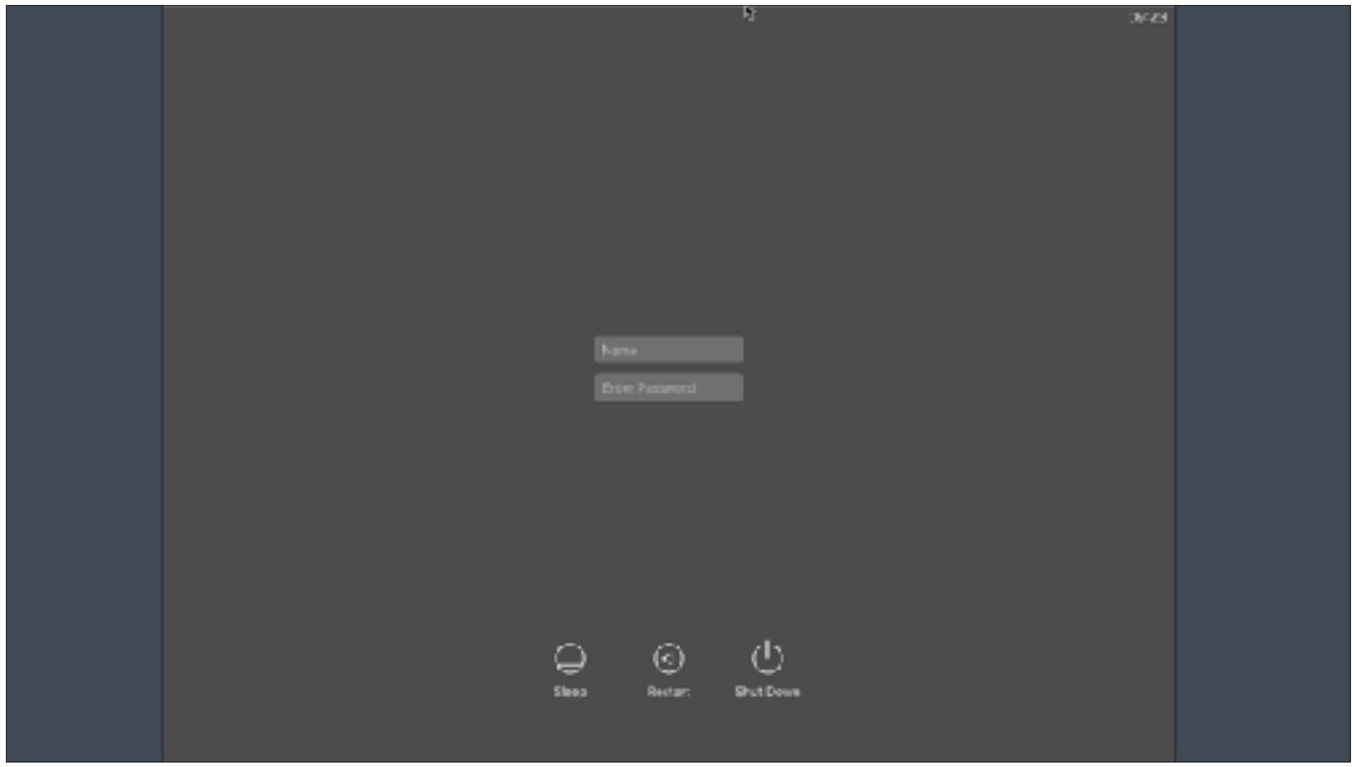
The features and settings below will be set up for you, or you can customize the settings yourself.

Learn More

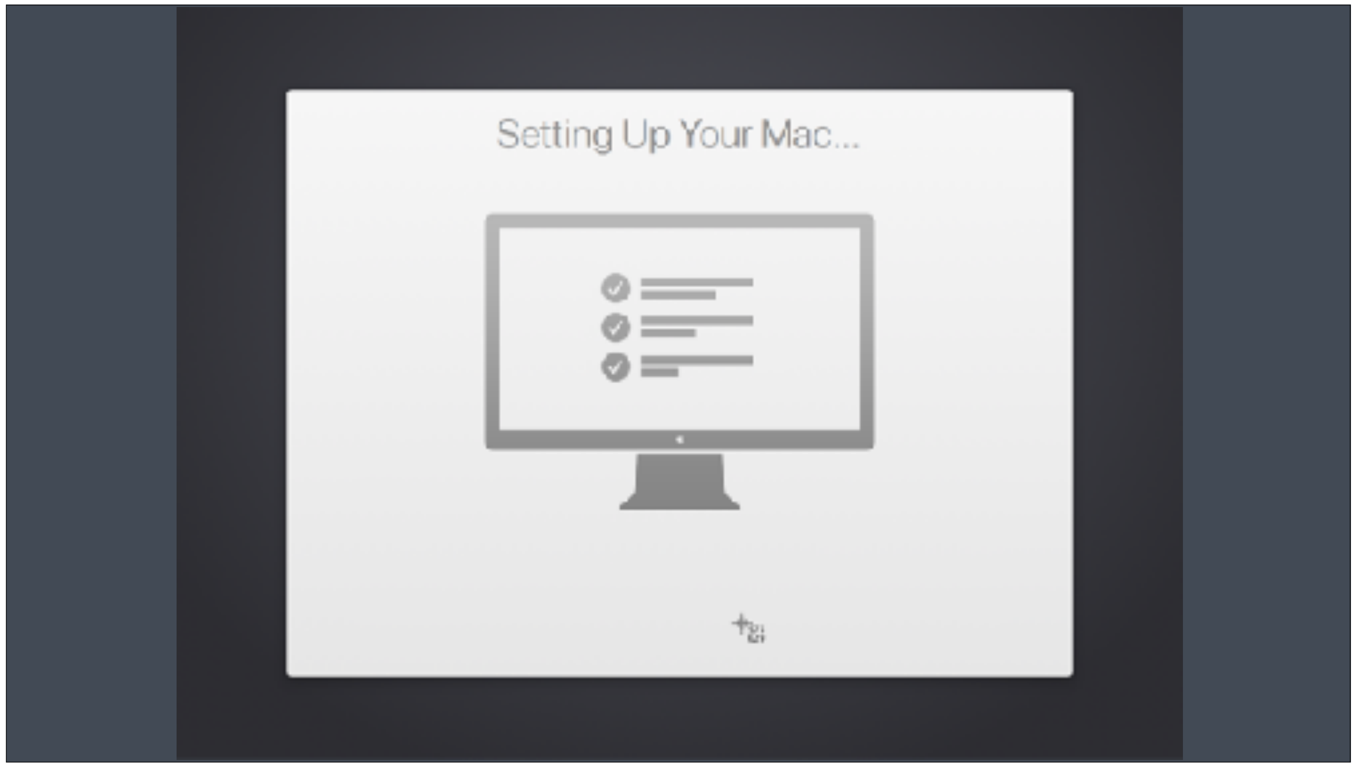
Allow Maps and other apps and services like Find My Mac and Spotlight Suggestions to gather and use data to determine your location.

Help Apple improve its products and services by allowing analytics of usage and data from this Mac and allowing Apple to share crash data with developers to improve their apps.

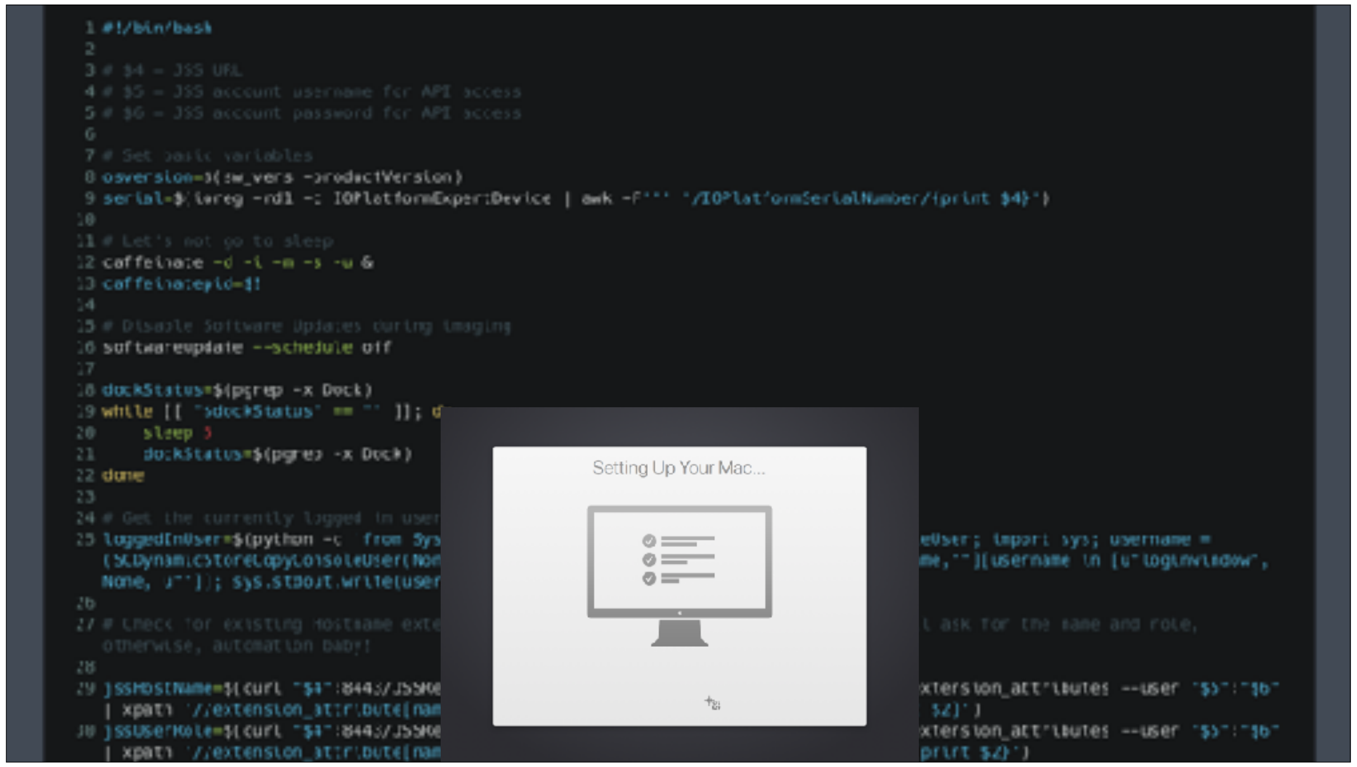
Proceed through the rest of the Setup Assistant



Login as local admin

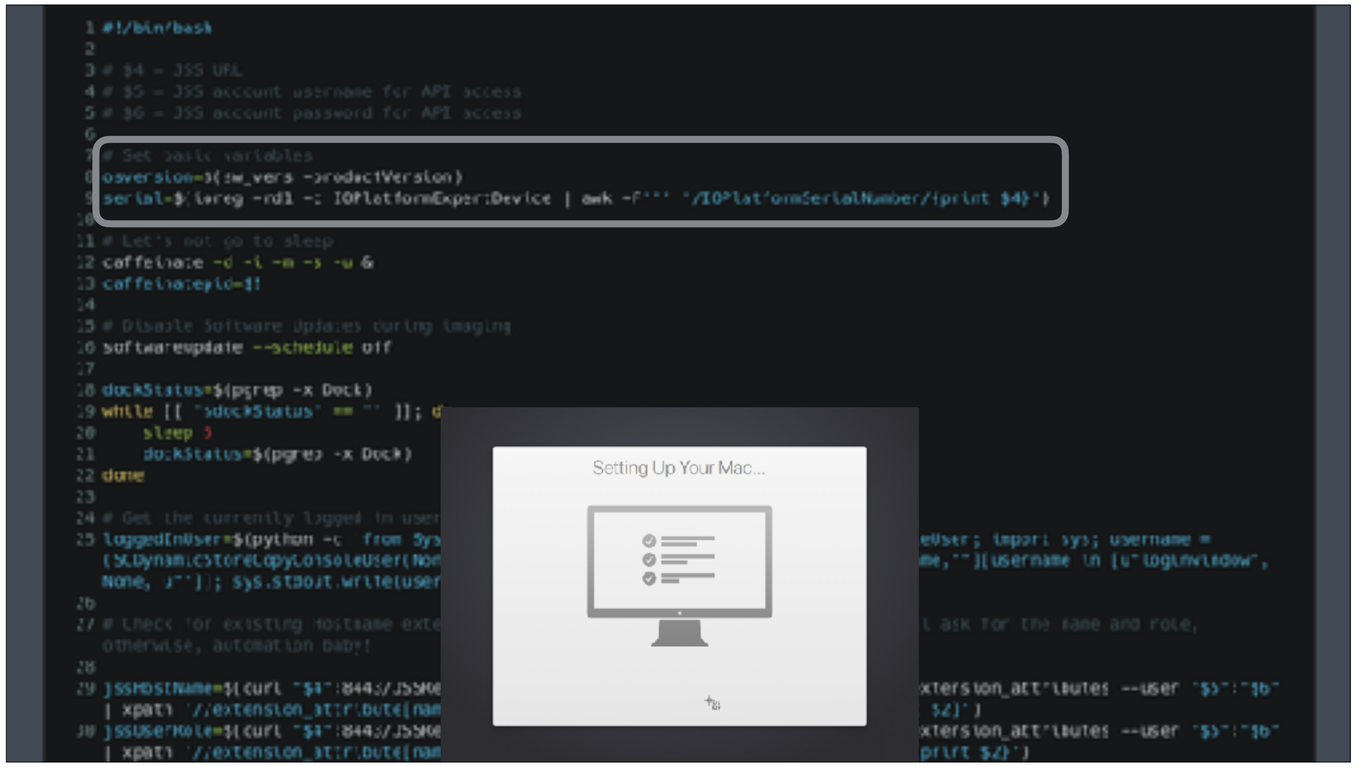


And off we go…



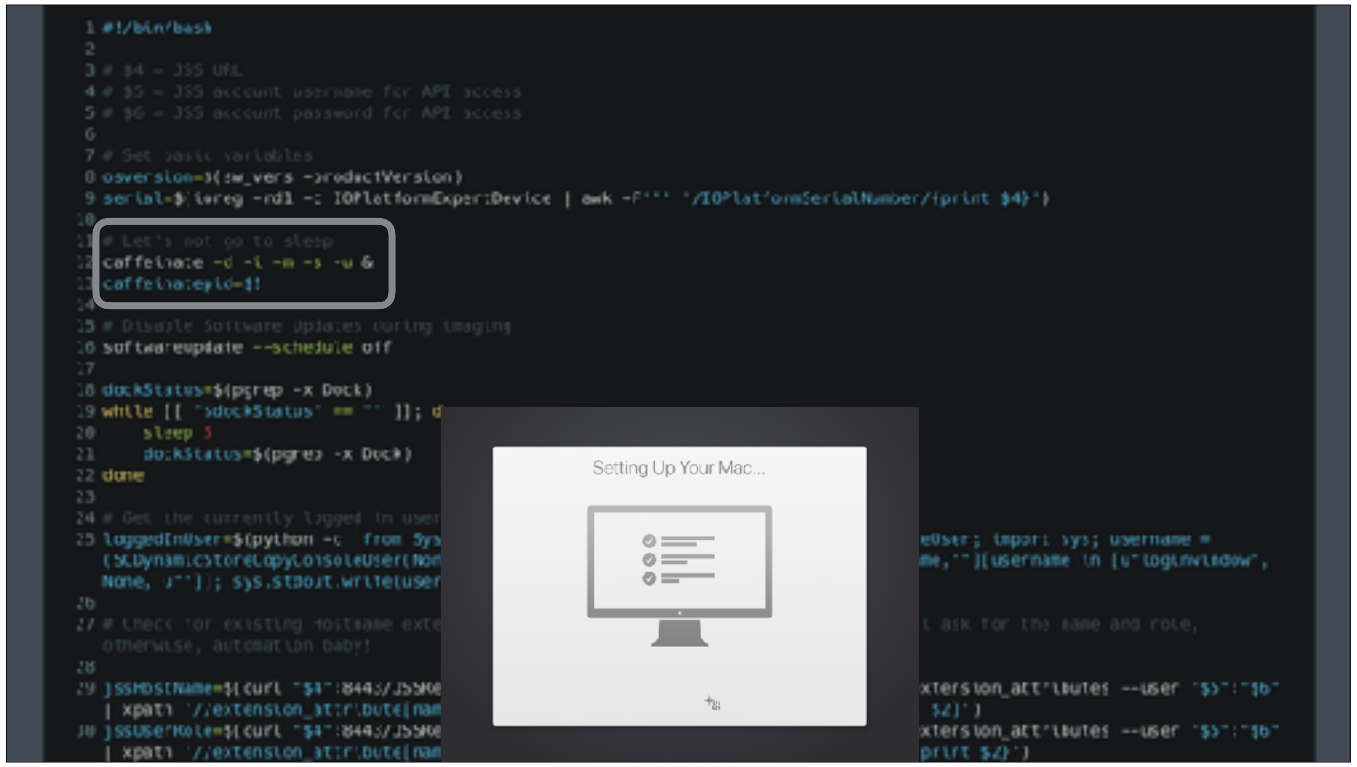
```
#!/bin/bash

# $4 - JSS URL
# $5 - JSS account username for API access
# $6 - JSS account password for API access

# Set basic variables
osversion=$(sw_vers -productVersion)
serial=$(ioreg -rd1 -c IOPlatformExpertDevice | awk -F'"' '/IOPlatformSerialNumber/{print $4}')

# Let's not go to sleep
caffeinate -d -i -m -s -u &
caffeinatepid=$!

# Disable Software Updates during imaging
softwareupdate --schedule off

dockStatus=$(pgrep -x Dock)
while [[ "$dockStatus" == "" ]]; d
    sleep 5
    dockStatus=$(pgrep -x Dock)
done

# Get the currently logged in user
loggedInUser=$(python -c 'from Sys                                  eUser; import sys; username =
(SCDynamicStoreCopyConsoleUser(Non                                  me,""][username in [u"loginwindow",
None, u""]]; sys.stdout.write(user

# Check for existing hostname exte                                  l ask for the name and role,
otherwise, automation baby!

jssHostName=$(curl "$4":8443/JSSRe                                  xtension_attributes --user "$5":"$6"
| xpath '//extension_attribute[nam                                   $2}')
jssUserRole=$(curl "$4":8443/JSSRe                                  xtension_attributes --user "$5":"$6"
| xpath '//extension_attribute[nam                                  print $2}')
```

Our Enrolment Complete policy fires, installing DEPNotify and running this script



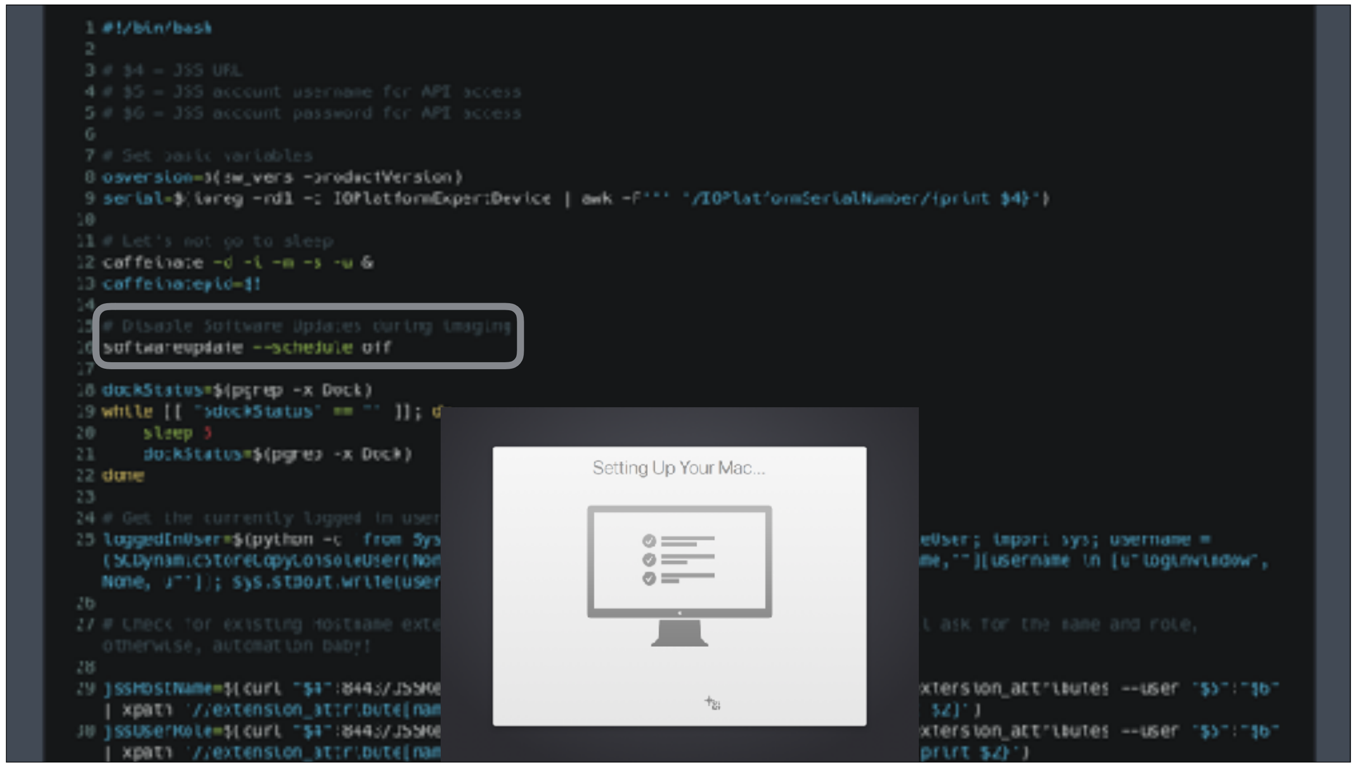
```
#!/bin/bash

# $4 - JSS URL
# $5 - JSS account username for API access
# $6 - JSS account password for API access

# Set basic variables
osversion=$(sw_vers -productVersion)
serial=$(ioreg -rd1 -c IOPlatformExpertDevice | awk -F'"' '/IOPlatformSerialNumber/{print $4}')

# Let's not go to sleep
caffeinate -d -i -m -s -u &
caffeinatepid=$!

# Disable Software Updates during imaging
softwareupdate --schedule off

dockStatus=$(pgrep -x Dock)
while [[ "$dockStatus" == "" ]]; d
    sleep 5
    dockStatus=$(pgrep -x Dock)
done

# Get the currently logged in user
loggedInUser=$(python -c 'from Sys                                   eUser; import sys; username =
(SCDynamicStoreCopyConsoleUser(Non                                    me,""][username in [u"loginwindow",
None, u""]]; sys.stdout.write(user

# Check for existing hostname exte                                    l ask for the name and role,
otherwise, automation baby!

jssHostName=$(curl "$4":8443/JSSRe                                    xtension_attributes --user "$5":"$6"
| xpath '//extension_attribute[nam                                     $2}')
jssUserRole=$(curl "$4":8443/JSSRe                                    xtension_attributes --user "$5":"$6"
| xpath '//extension_attribute[nam                                    print $2}')
```

First we grab the OS version and serial number and store them as a couple of variables



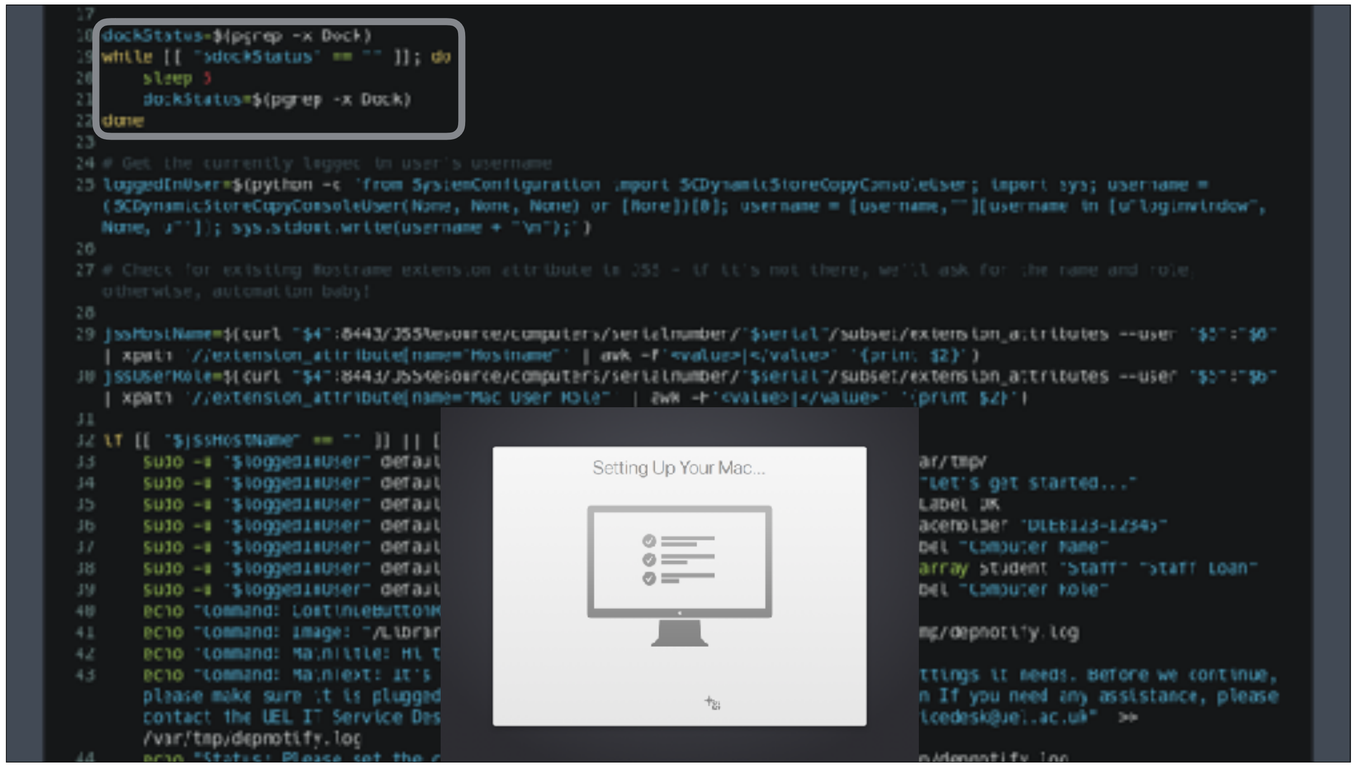
```
#!/bin/bash

# $4 - JSS URL
# $5 - JSS account username for API access
# $6 - JSS account password for API access

# Set basic variables
osversion=$(sw_vers -productVersion)
serial=$(ioreg -rd1 -c IOPlatformExpertDevice | awk -F'"' '/IOPlatformSerialNumber/{print $4}')

# Let's not go to sleep
caffeinate -d -i -m -s -u &
caffeinatepid=$!

# Disable Software Updates during imaging
softwareupdate --schedule off

dockStatus=$(pgrep -x Dock)
while [[ "$dockStatus" == "" ]]; d
    sleep 5
    dockStatus=$(pgrep -x Dock)
done

# Get the currently logged in user
loggedInUser=$(python -c 'from Sys                                   eUser; import sys; username =
(SCDynamicStoreCopyConsoleUser(Non                                   me,"")[username in [u"loginwindow",
None, u""]]; sys.stdout.write(user

# Check for existing hostname exte                                   l ask for the name and role,
otherwise, automation baby!

jssHostName=$(curl "$4":8443/JSSRe                                   xtension_attributes --user "$5":"$6"
| xpath '//extension_attribute[nam                                   $2}')
jssUserRole=$(curl "$4":8443/JSSRe                                   xtension_attributes --user "$5":"$6"
| xpath '//extension_attribute[nam                                   print $2}')
```

We don’t want the Mac to sleep, so we deal with that.



```
#!/bin/bash

# $4 - JSS URL
# $5 - JSS account username for API access
# $6 - JSS account password for API access

# Set basic variables
osversion=$(sw_vers -productVersion)
serial=$(ioreg -rd1 -c IOPlatformExpertDevice | awk -F'"' '/IOPlatformSerialNumber/{print $4}')

# Let's not go to sleep
caffeinate -d -i -m -s -u &
caffeinatepid=$!

# Disable Software Updates during imaging
softwareupdate --schedule off

dockStatus=$(pgrep -x Dock)
while [[ "$dockStatus" == "" ]]; d
    sleep 5
    dockStatus=$(pgrep -x Dock)
done

# Get the currently logged in user
loggedInUser=$(python -c 'from Sys                                    eUser; import sys; username =
(SCDynamicStoreCopyConsoleUser(Non                                    me,"")[username in [u"loginwindow",
None, u""]]; sys.stdout.write(user

# Check for existing hostname exte                                    l ask for the name and role,
otherwise, automation baby!

jssHostName=$(curl "$4":8443/JSSRe                                    xtension_attributes --user "$5":"$6"
| xpath '//extension_attribute[nam                                     $2}')
jssUserRole=$(curl "$4":8443/JSSRe                                    xtension_attributes --user "$5":"$6"
| xpath '//extension_attribute[nam                                    print $2}')
```

We nip any software update checks in the bud.

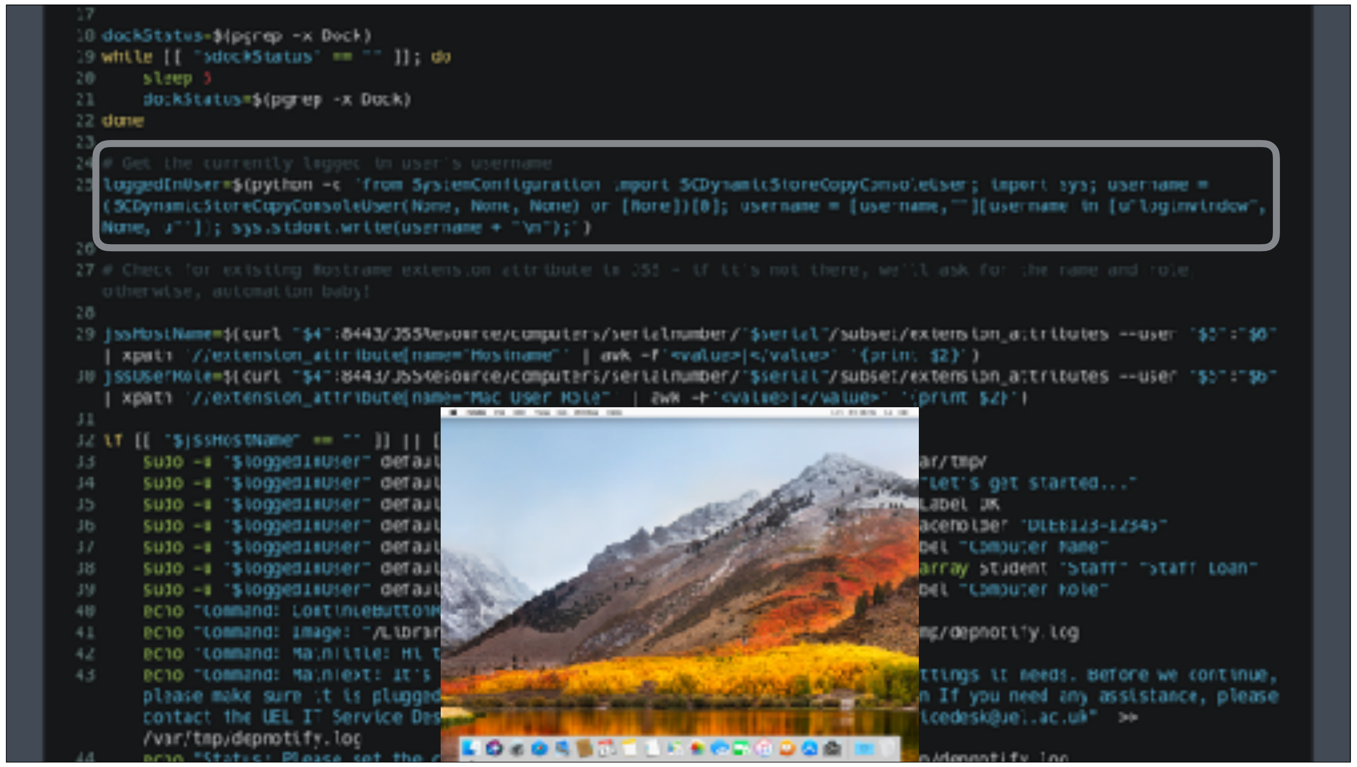


```
dockStatus=$(pgrep -x Dock)
while [[ "$dockStatus" == "" ]]; do
    sleep 5
    dockStatus=$(pgrep -x Dock)
done

# Get the currently logged in user's username
loggedInUser=$(python -c 'from SystemConfiguration import SCDynamicStoreCopyConsoleUser; import sys; username =
(SCDynamicStoreCopyConsoleUser(None, None, None) or [None])[0]; username = [username,""][username in [u"loginwindow",
None, u""]]; sys.stdout.write(username + "\n");')

# Check for existing Hostname extension attribute in JSS - if it's not there, we'll ask for the name and role
otherwise, automation baby!

jssHostName=$(curl "$4":8443/JSSResource/computers/serialnumber/"$serial"/subset/extension_attributes --user "$5":"$6"
 | xpath '//extension_attribute[name="Hostname"]' | awk -F'<value>|</value>' '{print $2}')
jssUserRole=$(curl "$4":8443/JSSResource/computers/serialnumber/"$serial"/subset/extension_attributes --user "$5":"$6"
 | xpath '//extension_attribute[name="Mac User Role"]' | awk -F'<value>|</value>' '{print $2}')

if [[ "$jssHostName" == "" ]] || [
    sudo -u "$loggedInUser" defaul                          ar/tmp/
    sudo -u "$loggedInUser" defaul                          "Let's get started..."
    sudo -u "$loggedInUser" defaul                          Label OK
    sudo -u "$loggedInUser" defaul                          aceholder "DLE8123-12345"
    sudo -u "$loggedInUser" defaul                          bel "Computer Name"
    sudo -u "$loggedInUser" defaul                          array Student "Staff" "Staff Loan"
    sudo -u "$loggedInUser" defaul                          bel "Computer Role"
    echo "Command: ContinueButtonR
    echo "Command: Image: "/Librar                          mp/depnotify.log
    echo "Command: MainTitle: Hi t
    echo "Command: MainText: It's                           ttings it needs. Before we continue,
    please make sure it is plugged                          n If you need any assistance, please
    contact the UEL IT Service Des                          icedesk@uel.ac.uk" >>
    /var/tmp/depnotify.log
    echo "Status: Please set the c                          p/depnotify.log
```

And we wait for login to complete.

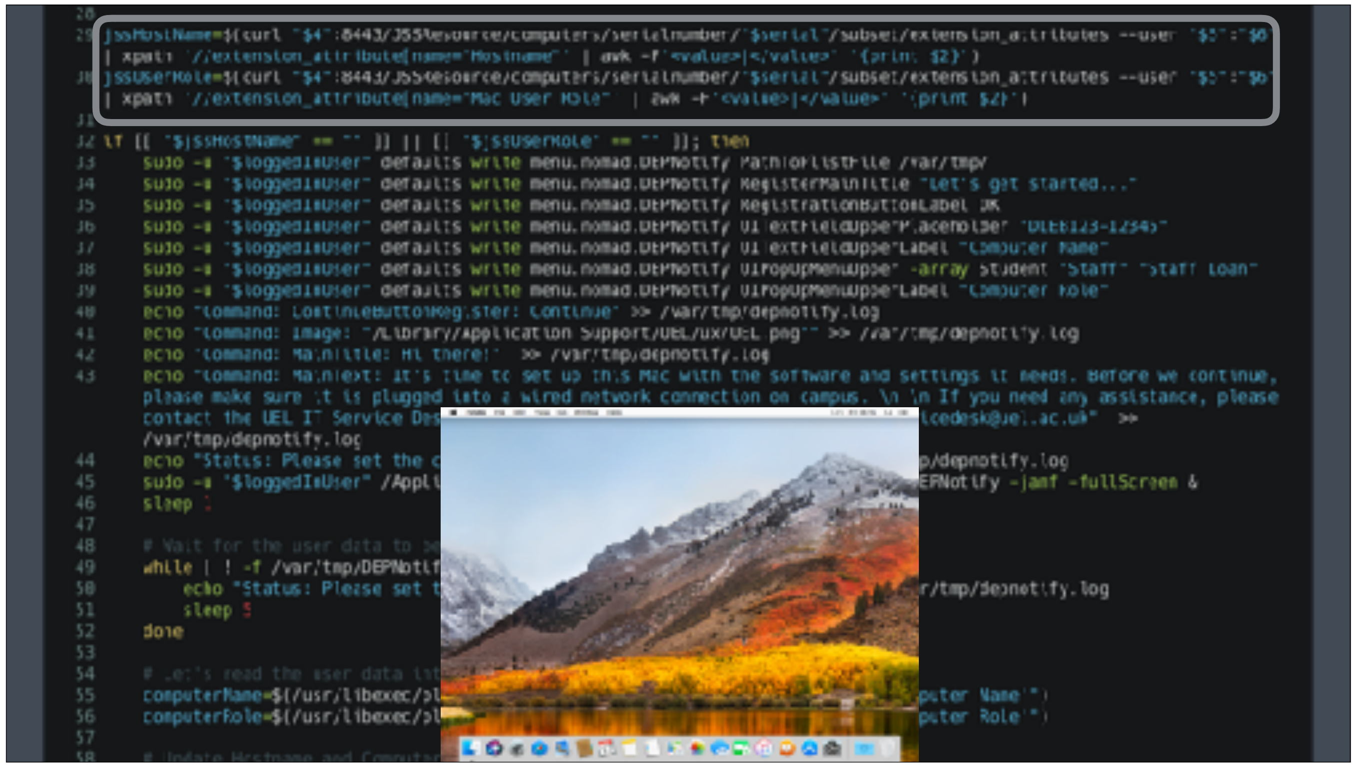
This is important - because of the way Jamf does enrolment with DEP, the script might have started before we logged in. DEPNotify needs to run after login.

One way to detect login is to wait for the Dock process with a while loop.



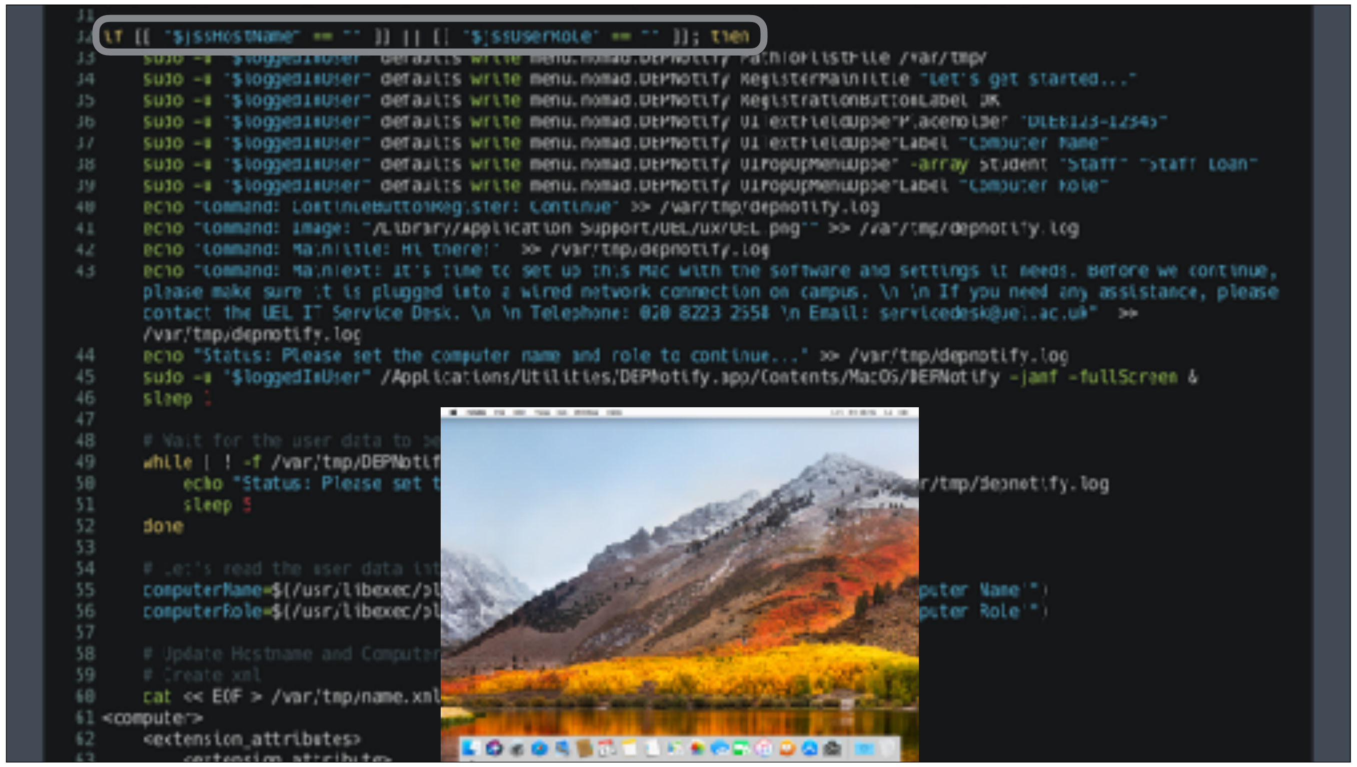
```
dockStatus=$(pgrep -x Dock)
while [[ "$dockStatus" == "" ]]; do
    sleep 5
    dockStatus=$(pgrep -x Dock)
done

# Get the currently logged in user's username
loggedInUser=$(python -c 'from SystemConfiguration import SCDynamicStoreCopyConsoleUser; import sys; username =
(SCDynamicStoreCopyConsoleUser(None, None, None) or [None])[0]; username = [username,""][username in [u"loginwindow",
None, u""]]; sys.stdout.write(username + "\n");')

# Check for existing Hostname extension attribute in JSS - if it's not there, we'll ask for the name and role
otherwise, automation baby!

jssHostName=$(curl "$4":8443/JSSResource/computers/serialnumber/"$serial"/subset/extension_attributes --user "$5":"$6"
| xpath '//extension_attribute[name="Hostname"]' | awk -F'<value>|</value>' '{print $2}')
jssUserRole=$(curl "$4":8443/JSSResource/computers/serialnumber/"$serial"/subset/extension_attributes --user "$5":"$6"
| xpath '//extension_attribute[name="Mac User Role"]' | awk -F'<value>|</value>' '{print $2}')

if [[ "$jssHostName" == "" ]] || [                                                  
    sudo -u "$loggedInUser" defau                                                  ar/tmp/
    sudo -u "$loggedInUser" defau                                                  "Let's get started..."
    sudo -u "$loggedInUser" defau                                                  Label OK
    sudo -u "$loggedInUser" defau                                                  aceholder "DLEE123-12345"
    sudo -u "$loggedInUser" defau                                                  bel "Computer Name"
    sudo -u "$loggedInUser" defau                                                  array Student "Staff" "Staff Loan"
    sudo -u "$loggedInUser" defau                                                  bel "Computer Role"
    echo "Command: ContinueButtonR
    echo "Command: Image: "/Librar                                                 mp/depnotify.log
    echo "Command: MainTitle: Hi t
    echo "Command: MainText: It's                                                  ttings it needs. Before we continue,
    please make sure it is plugged                                                 n If you need any assistance, please
    contact the UEL IT Service Des                                                 icedesk@uel.ac.uk" >>
    /var/tmp/depnotify.log
    echo "Status: Please set the c                                                 p/depnotify.log
```

And we're logged in. Let's get the logged in user's username.



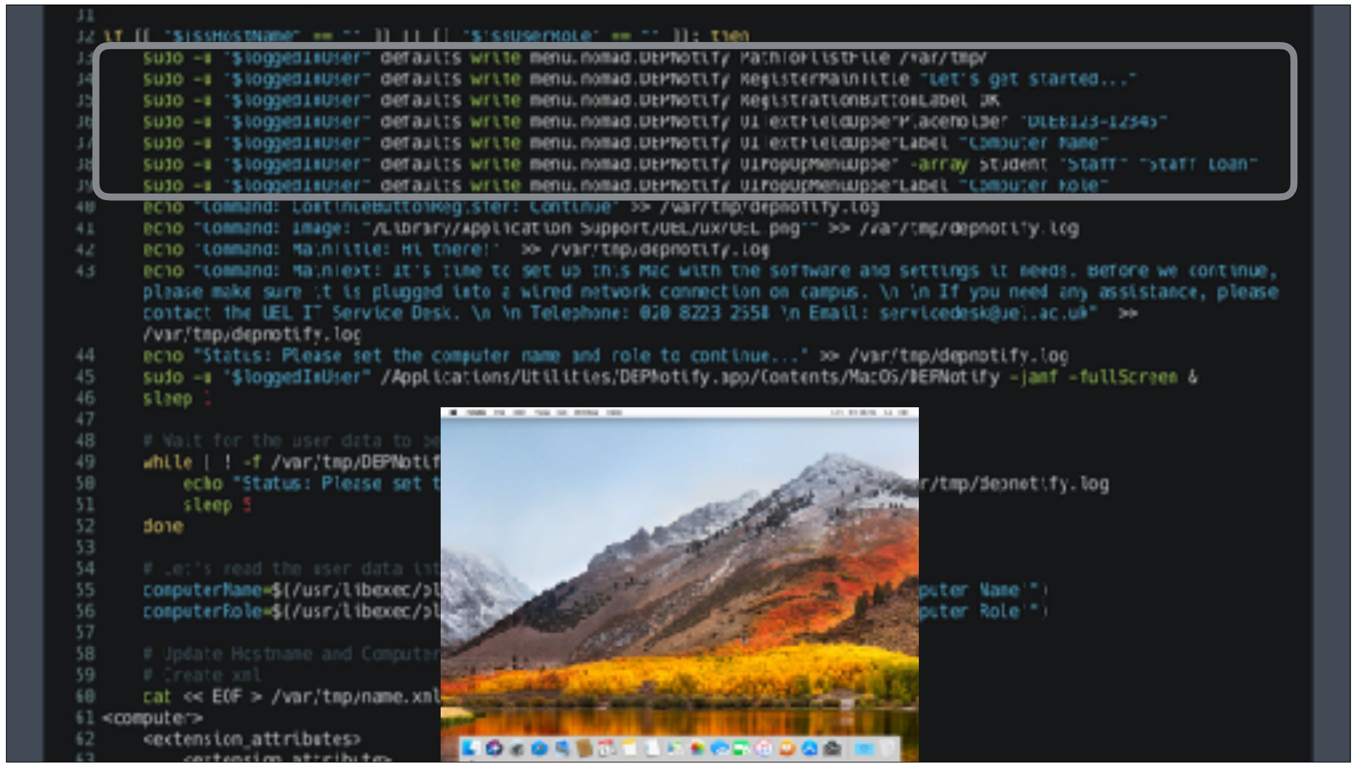
```
jssHostName=$(curl "$4":8443/JSSResource/computers/serialnumber/"$serial"/subset/extension_attributes --user "$5":"$6"
| xpath '//extension_attribute[name="Hostname"]' | awk -F'<value>|</value>' '{print $2}')
jssUserRole=$(curl "$4":8443/JSSResource/computers/serialnumber/"$serial"/subset/extension_attributes --user "$5":"$6"
| xpath '//extension_attribute[name="Mac User Role"]' | awk -F'<value>|</value>' '{print $2}')

if [[ "$jssHostName" == "" ]] || [[ "$jssUserRole" == "" ]]; then
    sudo -u "$loggedInUser" defaults write menu.nomad.DEPNotify PathToPlistFile /var/tmp/
    sudo -u "$loggedInUser" defaults write menu.nomad.DEPNotify RegisterMainTitle "Let's get started..."
    sudo -u "$loggedInUser" defaults write menu.nomad.DEPNotify RegistrationButtonLabel OK
    sudo -u "$loggedInUser" defaults write menu.nomad.DEPNotify UITextFieldUpperPlaceholder "DLEB123-12345"
    sudo -u "$loggedInUser" defaults write menu.nomad.DEPNotify UITextFieldUpperLabel "Computer Name"
    sudo -u "$loggedInUser" defaults write menu.nomad.DEPNotify UIPopUpMenuUpper -array Student "Staff" "Staff Loan"
    sudo -u "$loggedInUser" defaults write menu.nomad.DEPNotify UIPopUpMenuUpperLabel "Computer Role"
    echo "Command: ContinueButtonRegister: Continue" >> /var/tmp/depnotify.log
    echo "Command: Image: "/Library/Application Support/UEL/ux/UEL.png"" >> /var/tmp/depnotify.log
    echo "Command: MainTitle: Hi there!" >> /var/tmp/depnotify.log
    echo "Command: MainText: It's time to set up this Mac with the software and settings it needs. Before we continue,
    please make sure it is plugged into a wired network connection on campus. \n \n If you need any assistance, please
    contact the UEL IT Service Des                                            icedesk@uel.ac.uk" >>
    /var/tmp/depnotify.log
    echo "Status: Please set the c                                            p/depnotify.log
    sudo -u "$loggedInUser" /Appli                                            EPNotify -jamf -fullScreen &
    sleep 

    # Wait for the user data to be
    while [ ! -f /var/tmp/DEPNotif
        echo "Status: Please set t                                            r/tmp/depnotify.log
        sleep 5
    done

    # Let's read the user data int
    computerName=$(/usr/libexec/pl                                            puter Name")
    computerRole=$(/usr/libexec/pl                                            puter Role")

    # Update Hostname and Computer
```

And let's do a 2 API calls to read back our hostname and role Extension Attributes, which will only be there if the Mac already has a computer record in Jamf and we're re-provisioning it.



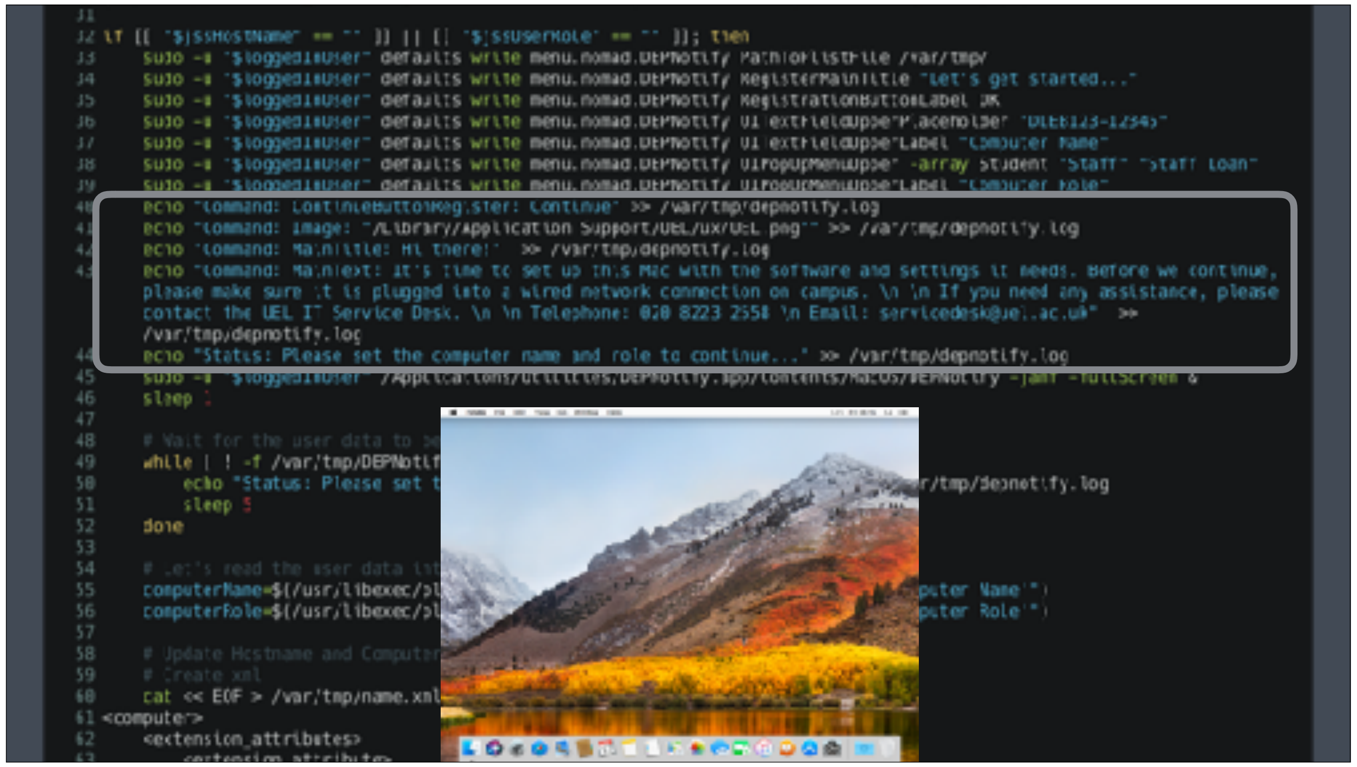
```
if [[ "$jssHostName" == "" ]] || [[ "$jssUserRole" == "" ]]; then
    sudo -u "$loggedInUser" defaults write menu.nomad.DEPNotify PathToPlistFile /var/tmp/
    sudo -u "$loggedInUser" defaults write menu.nomad.DEPNotify RegisterMainTitle "Let's get started..."
    sudo -u "$loggedInUser" defaults write menu.nomad.DEPNotify RegistrationButtonLabel OK
    sudo -u "$loggedInUser" defaults write menu.nomad.DEPNotify UITextFieldUpperPlaceholder "UEL0123-12345"
    sudo -u "$loggedInUser" defaults write menu.nomad.DEPNotify UITextFieldUpperLabel "Computer Name"
    sudo -u "$loggedInUser" defaults write menu.nomad.DEPNotify UIPopUpMenuUpper -array Student "Staff" "Staff Loan"
    sudo -u "$loggedInUser" defaults write menu.nomad.DEPNotify UIPopUpMenuUpperLabel "Computer Role"
    echo "Command: ContinueButtonRegister: Continue" >> /var/tmp/depnotify.log
    echo "Command: Image: "/Library/Application Support/UEL/ux/UEL.png"" >> /var/tmp/depnotify.log
    echo "Command: MainTitle: Hi there!" >> /var/tmp/depnotify.log
    echo "Command: MainText: It's time to set up this Mac with the software and settings it needs. Before we continue,
    please make sure it is plugged into a wired network connection on campus. \n \n If you need any assistance, please
    contact the UEL IT Service Desk. \n \n Telephone: 020 8223 2558 \n Email: servicedesk@uel.ac.uk" >>
    /var/tmp/depnotify.log
    echo "Status: Please set the computer name and role to continue..." >> /var/tmp/depnotify.log
    sudo -u "$loggedInUser" /Applications/Utilities/DEPNotify.app/Contents/MacOS/DEPNotify -jamf -fullScreen &
    sleep 1

    # Wait for the user data to be
    while [ ! -f /var/tmp/DEPNotif
        echo "Status: Please set t                                            r/tmp/depnotify.log
        sleep 5
    done

    # Let's read the user data int
    computerName=$(/usr/libexec/pl                                            puter Name")
    computerRole=$(/usr/libexec/pl                                            puter Role")

    # Update Hostname and Computer
    # Create xml
    cat << EOF > /var/tmp/name.xml
<computer>
    <extension_attributes>
```

If either of those Extension Attributes are empty, we're going to set up DEPNotify to prompt for user input.



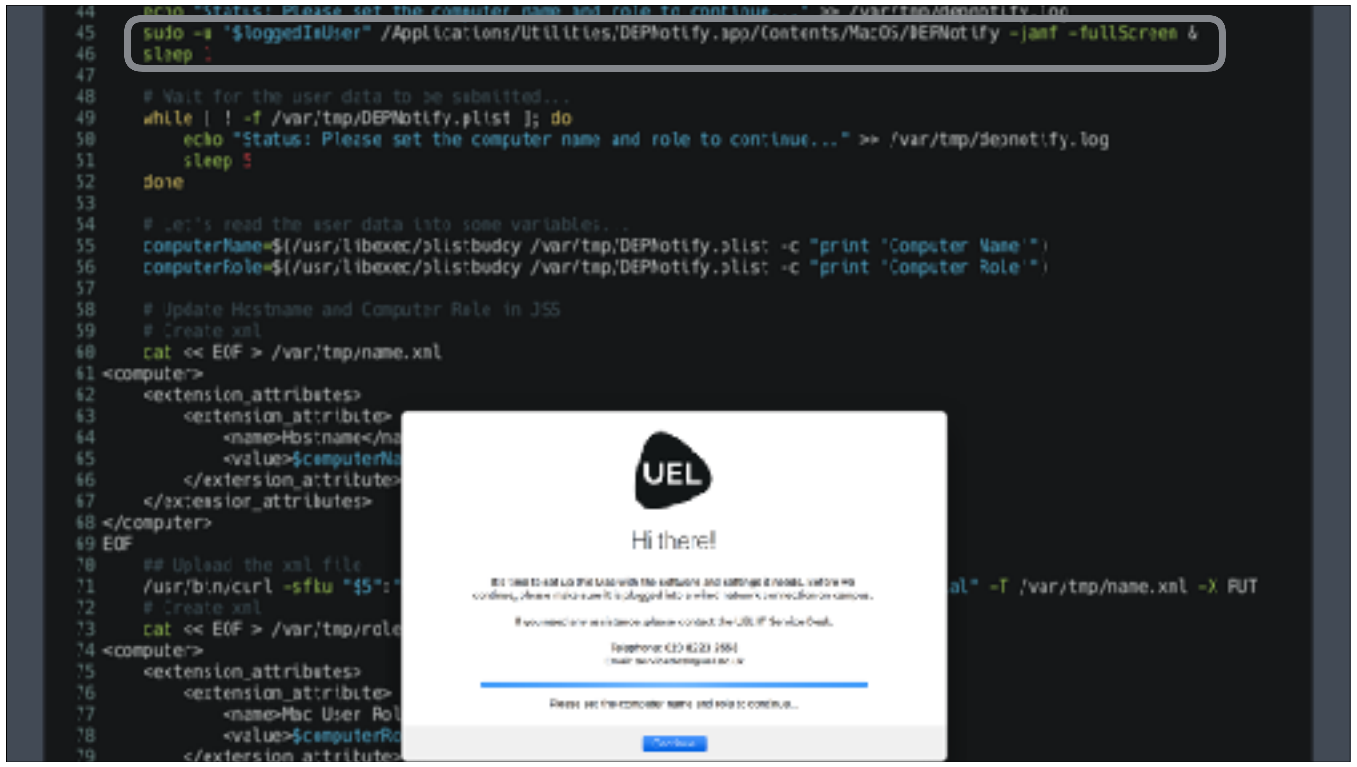
```
if [[ "$isHostName" == "" ]] || [[ "$isUserRole" == "" ]]; then
    sudo -u "$loggedInUser" defaults write menu.nomad.DEPNotify PathToPlistFile /var/tmp/
    sudo -u "$loggedInUser" defaults write menu.nomad.DEPNotify RegisterMainTitle "Let's get started..."
    sudo -u "$loggedInUser" defaults write menu.nomad.DEPNotify RegistrationButtonLabel OK
    sudo -u "$loggedInUser" defaults write menu.nomad.DEPNotify UITextFieldUpperPlaceholder "UELE123-12345"
    sudo -u "$loggedInUser" defaults write menu.nomad.DEPNotify UITextFieldUpperLabel "Computer Name"
    sudo -u "$loggedInUser" defaults write menu.nomad.DEPNotify UIPopUpMenuUpper -array Student "Staff" "Staff Loan"
    sudo -u "$loggedInUser" defaults write menu.nomad.DEPNotify UIPopUpMenuUpperLabel "Computer Role"
    echo "Command: ContinueButtonRegister: Continue" >> /var/tmp/depnotify.log
    echo "Command: Image: "/Library/Application Support/UEL/ux/UEL.png"" >> /var/tmp/depnotify.log
    echo "Command: MainTitle: Hi there!"  >> /var/tmp/depnotify.log
    echo "Command: MainText: It's time to set up this Mac with the software and settings it needs. Before we continue,
    please make sure it is plugged into a wired network connection on campus. \n \n If you need any assistance, please
    contact the UEL IT Service Desk. \n \n Telephone: 020 8223 2558 \n Email: servicedesk@uel.ac.uk"  >>
    /var/tmp/depnotify.log
    echo "Status: Please set the computer name and role to continue..." >> /var/tmp/depnotify.log
    sudo -u "$loggedInUser" /Applications/Utilities/DEPNotify.app/Contents/MacOS/DEPNotify -jamf -fullScreen &
    sleep 1

    # Wait for the user data to be
    while [ ! -f /var/tmp/DEPNotif                                                  r/tmp/depnotify.log
        echo "Status: Please set t
        sleep 5
    done

    # Let's read the user data int
    computerName=$(/usr/libexec/pl                                                  puter Name'")
    computerRole=$(/usr/libexec/pl                                                  puter Role'")

    # Update Hostname and Computer
    # Create xml
    cat << EOF > /var/tmp/name.xml
<computer>
    <extension_attributes>
        <extension_attribute>
```

We write some preferences to it - we're defining where it'll put the resultant plist file, the title of the dialog along with a text field for the hostname and pop up menu for the role.



```
if [[ "$jssHostName" == "" ]] || [[ "$jssUserRole" == "" ]]; then
    sudo -u "$loggedInUser" defaults write menu.nomad.DEPNotify PathToPlistFile /var/tmp/
    sudo -u "$loggedInUser" defaults write menu.nomad.DEPNotify RegisterMainTitle "Let's get started..."
    sudo -u "$loggedInUser" defaults write menu.nomad.DEPNotify RegistrationButtonLabel OK
    sudo -u "$loggedInUser" defaults write menu.nomad.DEPNotify UITextFieldUpperPlaceholder "UEL0123-12345"
    sudo -u "$loggedInUser" defaults write menu.nomad.DEPNotify UITextFieldUpperLabel "Computer Name"
    sudo -u "$loggedInUser" defaults write menu.nomad.DEPNotify UIPopupMenuUpper -array Student "Staff" "Staff Loan"
    sudo -u "$loggedInUser" defaults write menu.nomad.DEPNotify UIPopupMenuUpperLabel "Computer Role"
    echo "Command: ContinueButtonRegister: Continue" >> /var/tmp/depnotify.log
    echo "Command: Image: "/Library/Application Support/UEL/ux/UEL.png"" >> /var/tmp/depnotify.log
    echo "Command: MainTitle: Hi there!" >> /var/tmp/depnotify.log
    echo "Command: MainText: It's time to set up this Mac with the software and settings it needs. Before we continue,
    please make sure it is plugged into a wired network connection on campus. \n \n If you need any assistance, please
    contact the UEL IT Service Desk. \n \n Telephone: 020 8223 2558 \n Email: servicedesk@uel.ac.uk" >>
    /var/tmp/depnotify.log
    echo "Status: Please set the computer name and role to continue..." >> /var/tmp/depnotify.log
    sudo -u "$loggedInUser" /Applications/Utilities/DEPNotify.app/Contents/MacOS/DEPNotify -jamf -fullScreen &
    sleep 1

    # Wait for the user data to be
    while [ ! -f /var/tmp/DEPNotif
        echo "Status: Please set t                          r/tmp/depnotify.log
        sleep 5
    done

    # Let's read the user data int
    computerName=$(/usr/libexec/pl                          puter Name'")
    computerRole=$(/usr/libexec/pl                          puter Role'")

    # Update Hostname and Computer
    # Create xml
    cat << EOF > /var/tmp/name.xml
<computer>
    <extension_attributes>
        <extension_attribute>
```

Next, we're echoing commands so when DEPNotify launches, it will give us a button, use our icon and have the title, main text and status text we want.



```
    echo "Status: Please set the computer name and role to continue..." >> /var/tmp/depnotify.log
    sudo -u "$loggedInUser" /Applications/Utilities/DEPNotify.app/Contents/MacOS/DEPNotify -jamf -fullScreen &
    sleep [illegible]

    # Wait for the user data to be submitted...
    while [ ! -f /var/tmp/DEPNotify.plist ]; do
        echo "Status: Please set the computer name and role to continue..." >> /var/tmp/depnotify.log
        sleep 5
    done

    # Let's read the user data into some variables...
    computerName=$(/usr/libexec/plistbuddy /var/tmp/DEPNotify.plist -c "print 'Computer Name'")
    computerRole=$(/usr/libexec/plistbuddy /var/tmp/DEPNotify.plist -c "print 'Computer Role'")

    # Update Hostname and Computer Role in JSS
    # Create xml
    cat << EOF > /var/tmp/name.xml
<computer>
    <extension_attributes>
        <extension_attribute>
            <name>Hostname</na
            <value>$computerNa
        </extension_attribute>
    </extension_attributes>
</computer>
EOF
    ## Upload the xml file
    /usr/bin/curl -sfku "$5":"                                  al" -T /var/tmp/name.xml -X PUT
    # Create xml
    cat << EOF > /var/tmp/role
<computer>
    <extension_attributes>
        <extension_attribute>
            <name>Mac User Rol
            <value>$computerRo
        </extension_attribute>
```

Hi there!

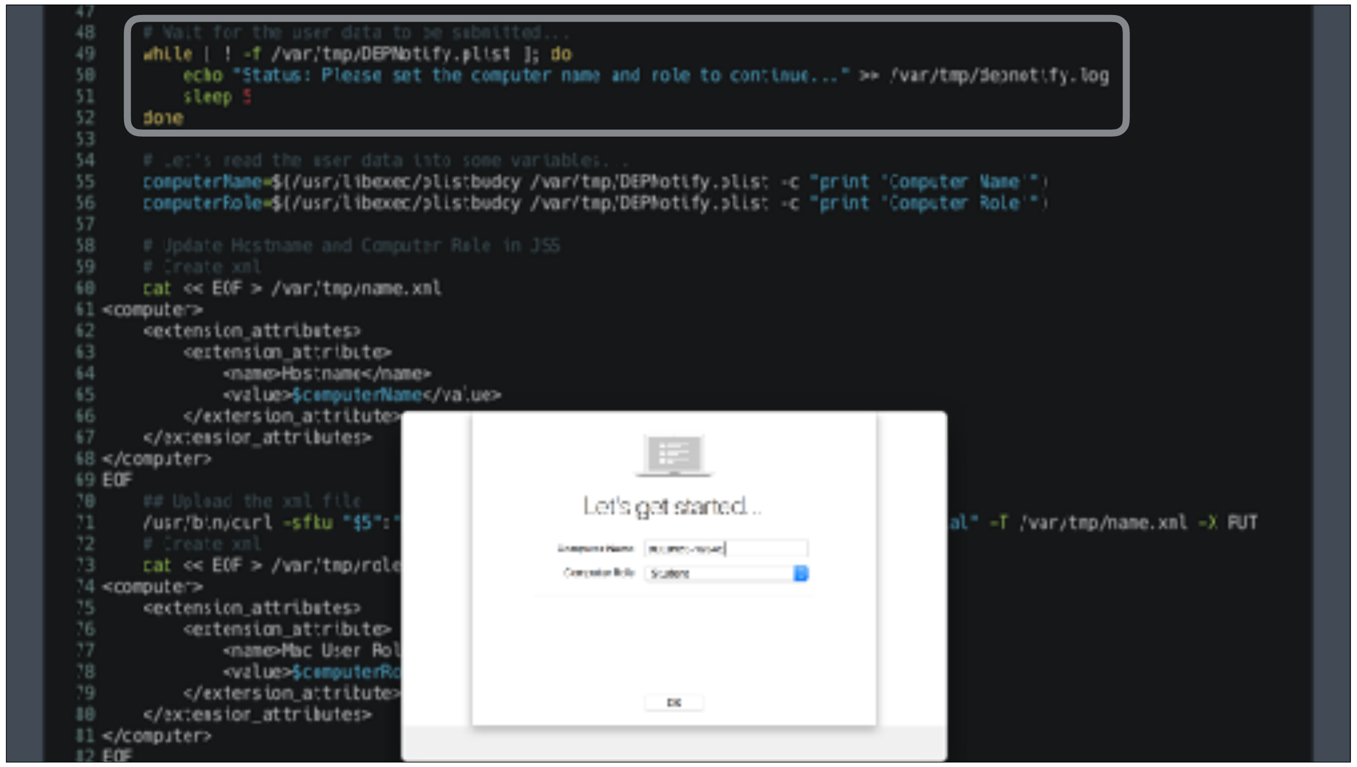
[illegible]

Now we launch DEPNotify as the logged in user, along with the Jamf and fullscreen switches.

Its window appears and the surrounding screen area is blurred.

It will read the Jamf log and update the status text when policies run and packages are installed.

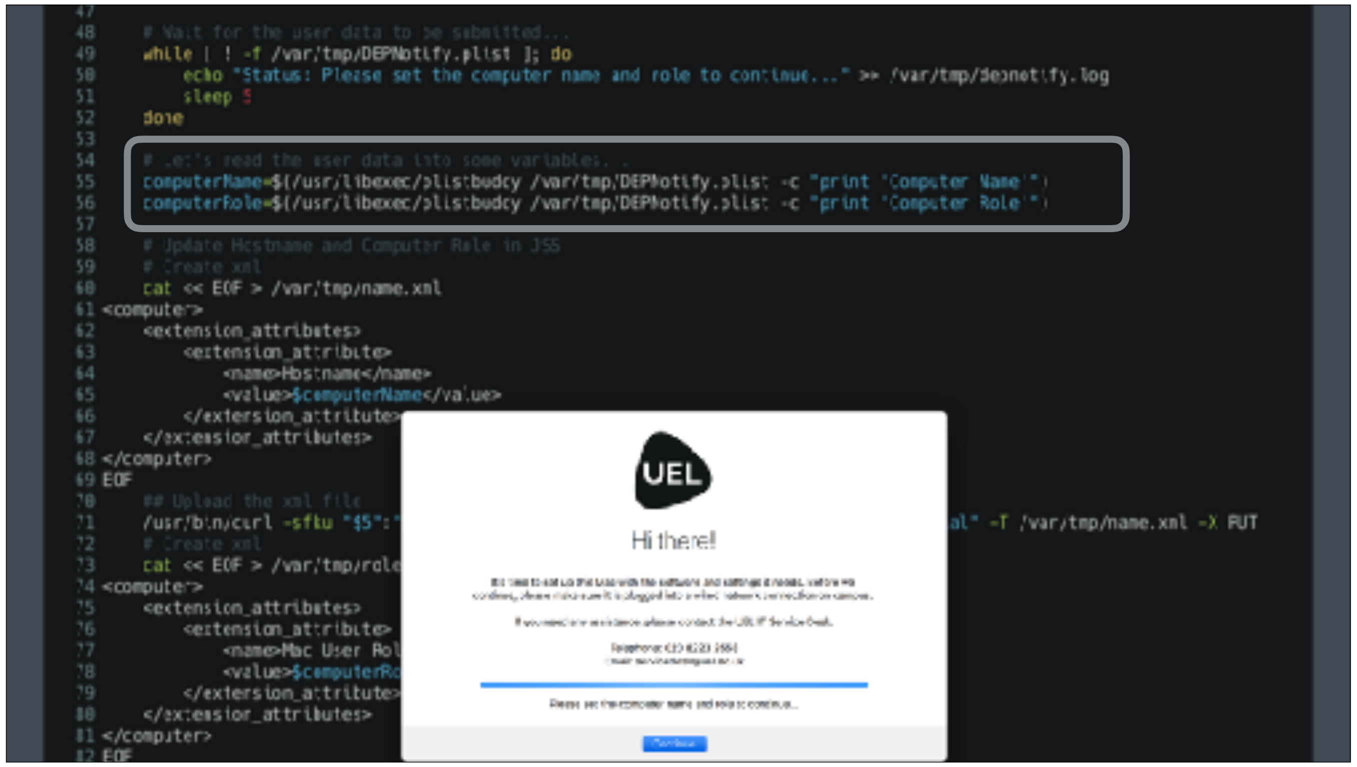
We click the Continue button



```
    # Wait for the user data to be submitted...
    while [ ! -f /var/tmp/DEPNotify.plist ]; do
        echo "Status: Please set the computer name and role to continue..." >> /var/tmp/depnotify.log
        sleep 5
    done

    # Let's read the user data into some variables...
    computerName=$(/usr/libexec/plistbuddy /var/tmp/DEPNotify.plist -c "print 'Computer Name'")
    computerRole=$(/usr/libexec/plistbuddy /var/tmp/DEPNotify.plist -c "print 'Computer Role'")

    # Update Hostname and Computer Role in JSS
    # Create xml
    cat << EOF > /var/tmp/name.xml
<computer>
    <extension_attributes>
        <extension_attribute>
            <name>Hostname</name>
            <value>$computerName</value>
        </extension_attribute>
    </extension_attributes>
</computer>
EOF
    ## Upload the xml file
    /usr/bin/curl -sfku "$5":"                                        al" -T /var/tmp/name.xml -X PUT
    # Create xml
    cat << EOF > /var/tmp/role
<computer>
    <extension_attributes>
        <extension_attribute>
            <name>Mac User Rol
            <value>$computerRo
        </extension_attribute>
    </extension_attributes>
</computer>
EOF
```

We won't continue until the plist containing our user input data is written.

Our technician enters the hostname and chooses the role, then click OK.



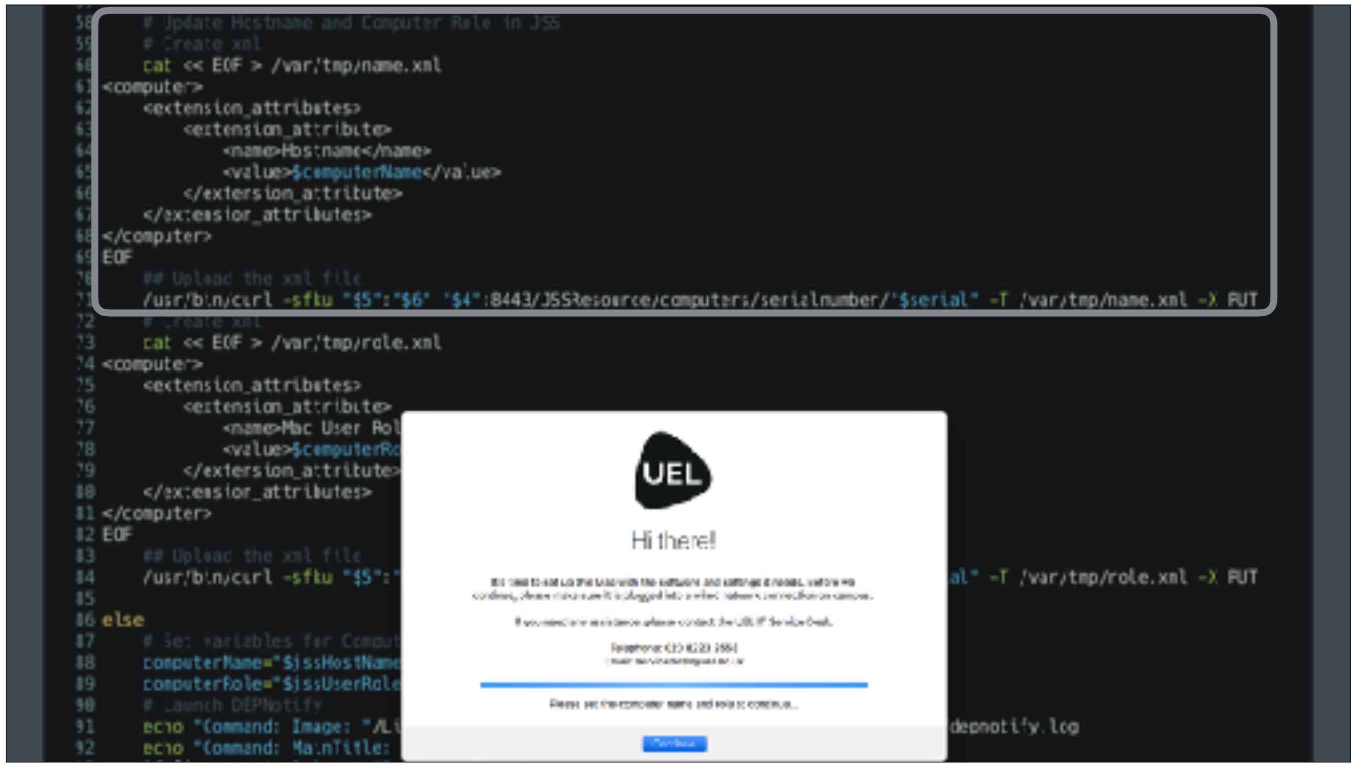
```
    # Wait for the user data to be submitted...
    while [ ! -f /var/tmp/DEPNotify.plist ]; do
        echo "Status: Please set the computer name and role to continue..." >> /var/tmp/depnotify.log
        sleep 5
    done

    # Let's read the user data into some variables...
    computerName=$(/usr/libexec/plistbuddy /var/tmp/DEPNotify.plist -c "print 'Computer Name'")
    computerRole=$(/usr/libexec/plistbuddy /var/tmp/DEPNotify.plist -c "print 'Computer Role'")

    # Update Hostname and Computer Role in JSS
    # Create xml
    cat << EOF > /var/tmp/name.xml
<computer>
    <extension_attributes>
        <extension_attribute>
            <name>Hostname</name>
            <value>$computerName</value>
        </extension_attribute>
    </extension_attributes>
</computer>
EOF
    ## Upload the xml file
    /usr/bin/curl -sfku "$5":"                                          al" -T /var/tmp/name.xml -X PUT
    # Create xml
    cat << EOF > /var/tmp/role
<computer>
    <extension_attributes>
        <extension_attribute>
            <name>Mac User Rol
            <value>$computerRo
        </extension_attribute>
    </extension_attributes>
</computer>
EOF
```

Hi there!

[illegible]

Please set the computer name and role to continue...

DEPNotify writes the plist and we read back the hostname and role into a couple of variables.

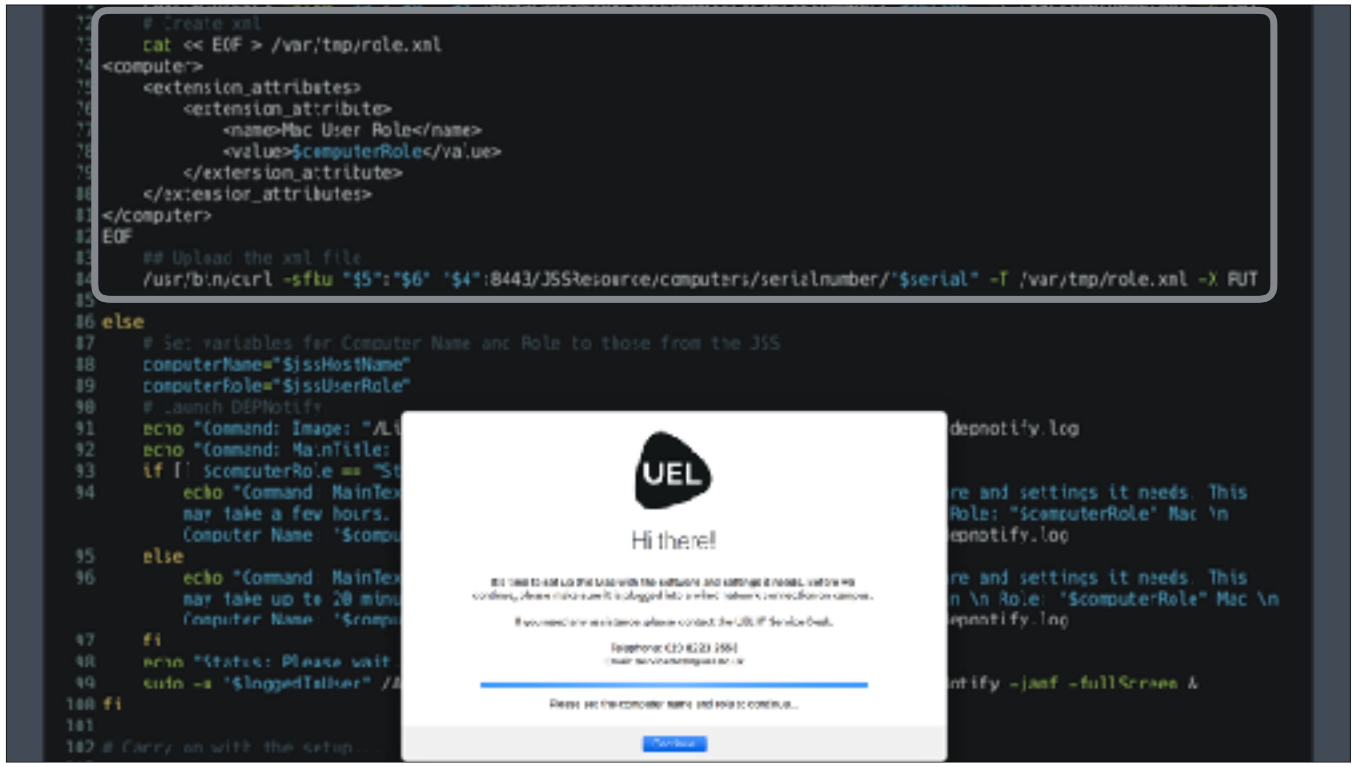


```
    # Update Hostname and Computer Role in JSS
    # Create xml
    cat << EOF > /var/tmp/name.xml
<computer>
    <extension_attributes>
        <extension_attribute>
            <name>Hostname</name>
            <value>$computerName</value>
        </extension_attribute>
    </extension_attributes>
</computer>
EOF
    ## Upload the xml file
    /usr/bin/curl -sfku "$5":"$6" "$4":8443/JSSResource/computers/serialnumber/"$serial" -T /var/tmp/name.xml -X PUT
    # Create xml
    cat << EOF > /var/tmp/role.xml
<computer>
    <extension_attributes>
        <extension_attribute>
            <name>Mac User Rol
            <value>$computerRo
        </extension_attribute>
    </extension_attributes>
</computer>
EOF
    ## Upload the xml file
    /usr/bin/curl -sfku "$5":"                                    al" -T /var/tmp/role.xml -X PUT

else
    # Set variables for Comput
    computerName="$jssHostName
    computerRole="$jssUserRole
    # Launch DEPNotify
    echo "Command: Image: "/Li                                    depnotify.log
    echo "Command: MainTitle:
```

Hi there!

[illegible]

Next, we create some XML containing the name and use the API to update the Computer Name Extension Attribute with it.

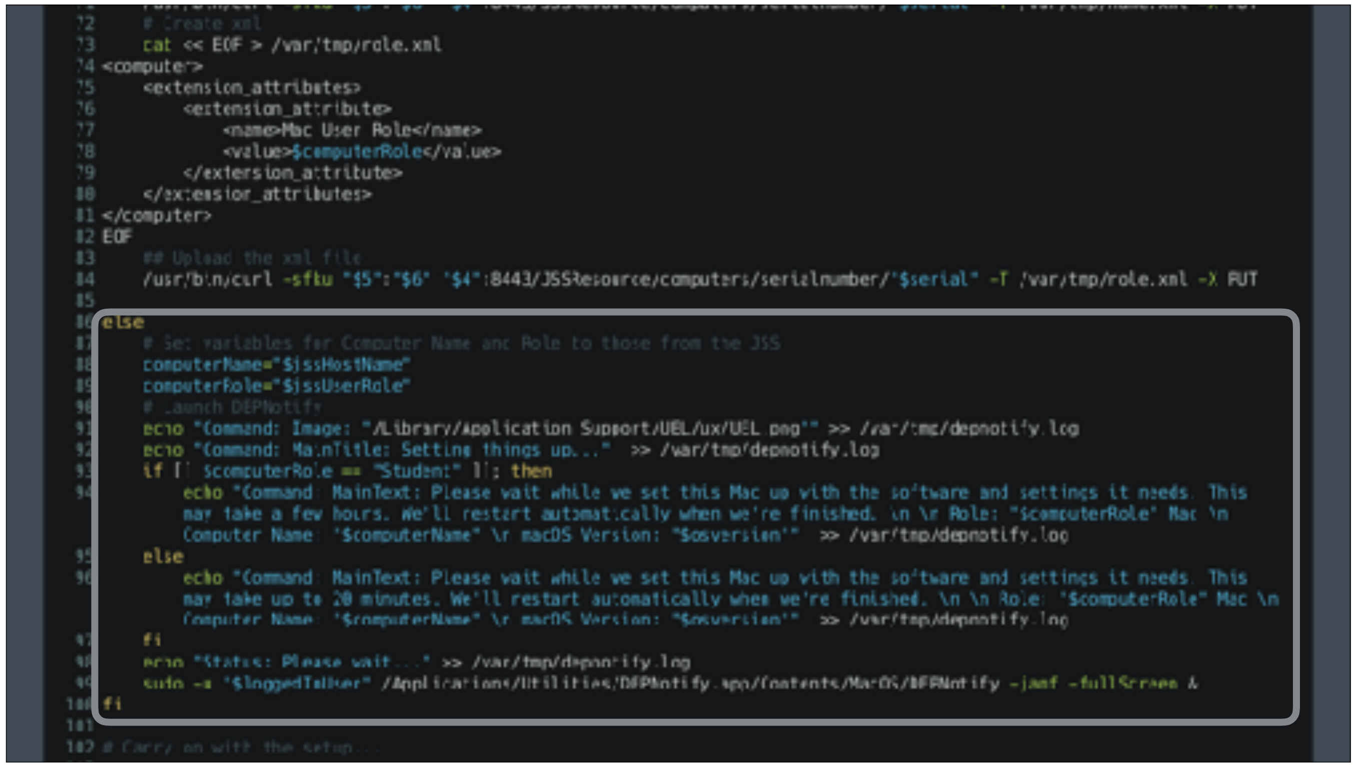


```
    # Create xml
    cat << EOF > /var/tmp/role.xml
<computer>
    <extension_attributes>
        <extension_attribute>
            <name>Mac User Role</name>
            <value>$computerRole</value>
        </extension_attribute>
    </extension_attributes>
</computer>
EOF
    ## Upload the xml file
    /usr/bin/curl -sfku "$5":"$6" "$4":8443/JSSResource/computers/serialnumber/"$serial" -T /var/tmp/role.xml -X PUT

else
    # Set variables for Computer Name and Role to those from the JSS
    computerName="$jssHostName"
    computerRole="$jssUserRole"
    # Launch DEPNotify
    echo "Command: Image: "/Li                                          depnotify.log
    echo "Command: MainTitle:
    if [[ $computerRole == "St
        echo "Command: MainTex                                          re and settings it needs. This
        may take a few hours.                                           Role: "$computerRole" Mac \n
        Computer Name: '$compu                                          epnotify.log
    else
        echo "Command: MainTex                                          re and settings it needs. This
        may take up to 20 minu                                          n \n Role: "$computerRole" Mac \n
        Computer Name: '$compu                                          epnotify.log
    fi
    echo "Status: Please wait.
    sudo -u "$loggedInUser" /A                                          otify -jamf -fullScreen &
fi

# Carry on with the setup...
```

Hi there!

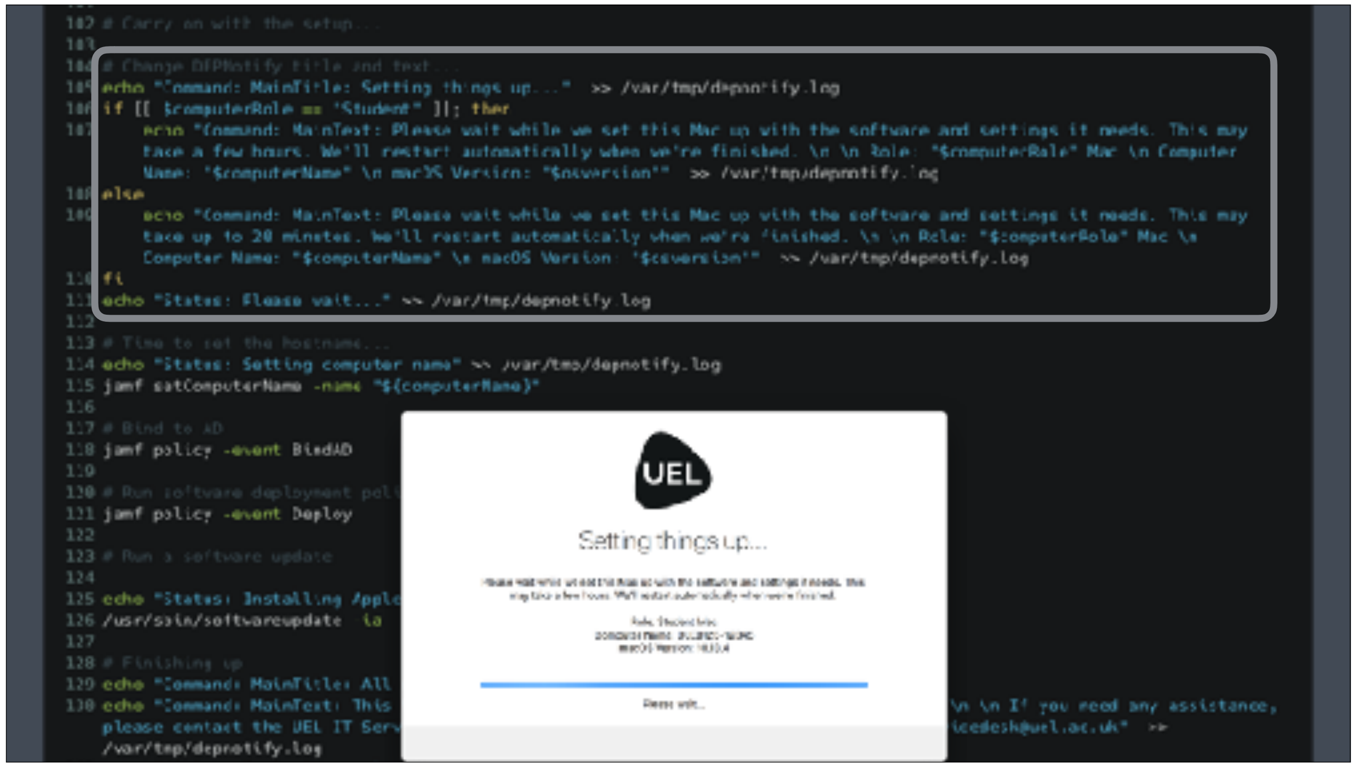
[illegible]

We do the same for the role.



```
    # Create xml
    cat << EOF > /var/tmp/role.xml
<computer>
    <extension_attributes>
        <extension_attribute>
            <name>Mac User Role</name>
            <value>$computerRole</value>
        </extension_attribute>
    </extension_attributes>
</computer>
EOF
    ## Upload the xml file
    /usr/bin/curl -sfku "$5":"$6" "$4":8443/JSSResource/computers/serialnumber/"$serial" -T /var/tmp/role.xml -X PUT

else
    # Set variables for Computer Name and Role to those from the JSS
    computerName="$jssHostName"
    computerRole="$jssUserRole"
    # Launch DEPNotify
    echo "Command: Image: "/Library/Application Support/UEL/ux/UEL.png"" >> /var/tmp/depnotify.log
    echo "Command: MainTitle: Setting things up..."  >> /var/tmp/depnotify.log
    if [[ $computerRole == "Student" ]]; then
        echo "Command: MainText: Please wait while we set this Mac up with the software and settings it needs. This
        may take a few hours. We'll restart automatically when we're finished. \n \r Role: "$computerRole" Mac \n
        Computer Name: "$computerName" \r macOS Version: "$osversion""  >> /var/tmp/depnotify.log
    else
        echo "Command: MainText: Please wait while we set this Mac up with the software and settings it needs. This
        may take up to 20 minutes. We'll restart automatically when we're finished. \n \n Role: "$computerRole" Mac \n
        Computer Name: "$computerName" \r macOS Version: "$osversion""  >> /var/tmp/depnotify.log
    fi
    echo "Status: Please wait..." >> /var/tmp/depnotify.log
    sudo -u "$loggedInUser" /Applications/Utilities/DEPNotify.app/Contents/MacOS/DEPNotify -jamf -fullScreen &
fi

# Carry on with the setup...
```

Now, if the name and role are already set, we'd skip all of that - so we're not prompted if we're re-provisioning a Mac that's already in Jamf.



```
# Carry on with the setup...

# Change DEPNotify title and text...
echo "Command: MainTitle: Setting things up..."  >> /var/tmp/depnotify.log
if [[ $computerRole == 'Student' ]]; then
    echo "Command: MainText: Please wait while we set this Mac up with the software and settings it needs. This may
    take a few hours. We'll restart automatically when we're finished. \n \n Role: "$computerRole" Mac \n Computer
    Name: "$computerName" \n macOS Version: "$osversion""  >> /var/tmp/depnotify.log
else
    echo "Command: MainText: Please wait while we set this Mac up with the software and settings it needs. This may
    take up to 20 minutes. We'll restart automatically when we're finished. \n \n Role: "$computerRole" Mac \n
    Computer Name: "$computerName" \n macOS Version: "$osversion""  >> /var/tmp/depnotify.log
fi
echo "Status: Please wait..." >> /var/tmp/depnotify.log

# Time to set the hostname...
echo "Status: Setting computer name" >> /var/tmp/depnotify.log
jamf setComputerName -name "${computerName}"

# Bind to AD
jamf policy -event BindAD

# Run software deployment pol
jamf policy -event Deploy

# Run a software update

echo "Status: Installing Apple
/usr/sbin/softwareupdate -ia

# Finishing up
echo "Command: MainTitle: All
echo "Command: MainText: This                                                 \n \n If you need any assistance,
please contact the UEL IT Serv                                                icedesk@uel.ac.uk" >>
/var/tmp/depnotify.log
```

Setting things up...

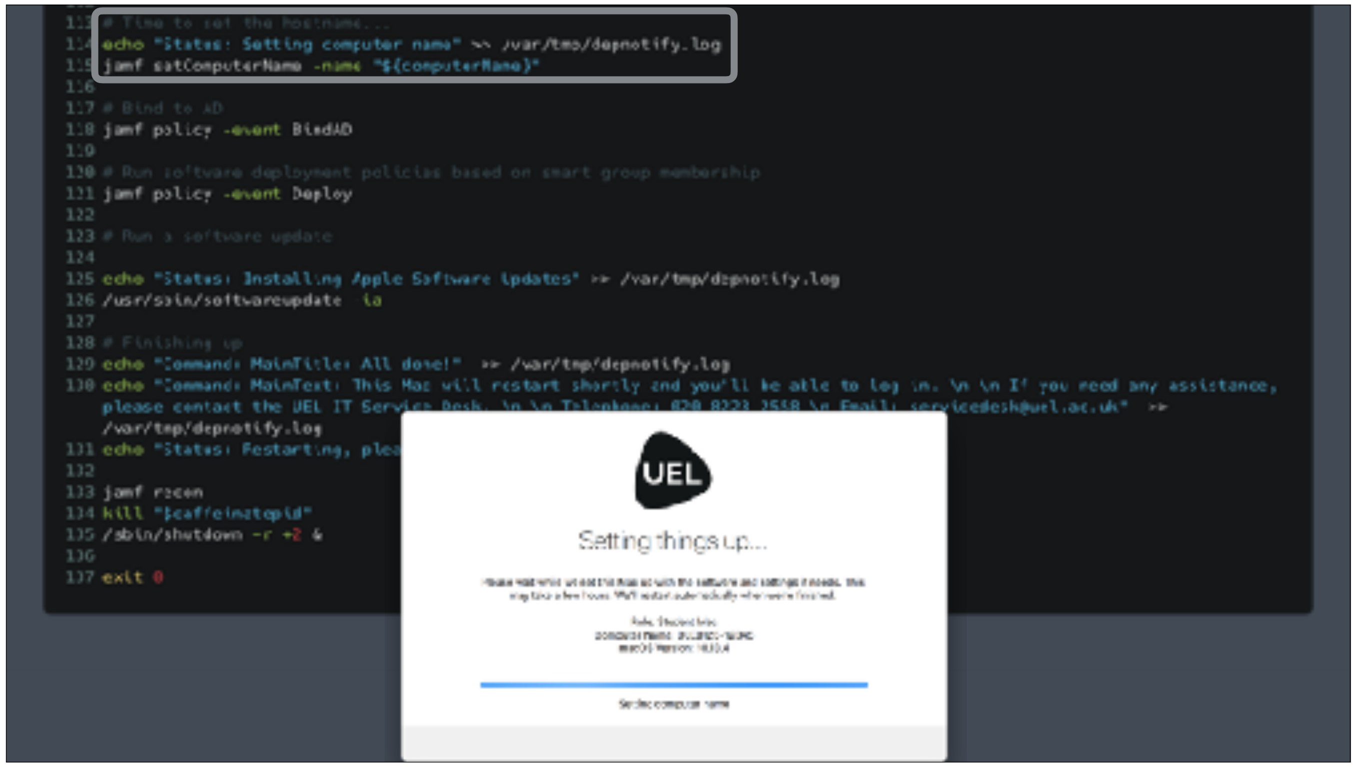
[illegible]

Please wait...

We carry on - changing the title, main text and status to let us know what’s happening.

We can also include the hostname and version of macOS here.

If it’s a student facing Mac, our main text lets us know it’ll take longer to set up than a staff facing Mac.



```
# Time to set the hostname...
echo "Status: Setting computer name" >> /var/tmp/depnotify.log
jamf setComputerName -name "${computerName}"

# Bind to AD
jamf policy -event BindAD

# Run software deployment policies based on smart group membership
jamf policy -event Deploy

# Run a software update

echo "Status: Installing Apple Software Updates" >> /var/tmp/depnotify.log
/usr/sbin/softwareupdate -ia

# Finishing up
echo "Command: MainTitle: All done!"  >> /var/tmp/depnotify.log
echo "Command: MainText: This Mac will restart shortly and you'll be able to log in. \n \n If you need any assistance,
please contact the UEL IT Service Desk. \n \n Telephone: 020 8223 2558 \n Email: servicedesk@uel.ac.uk"  >>
/var/tmp/depnotify.log
echo "Status: Restarting, plea

jamf recon
kill "$caffeinatepid"
/sbin/shutdown -r +2 &

exit 0
```

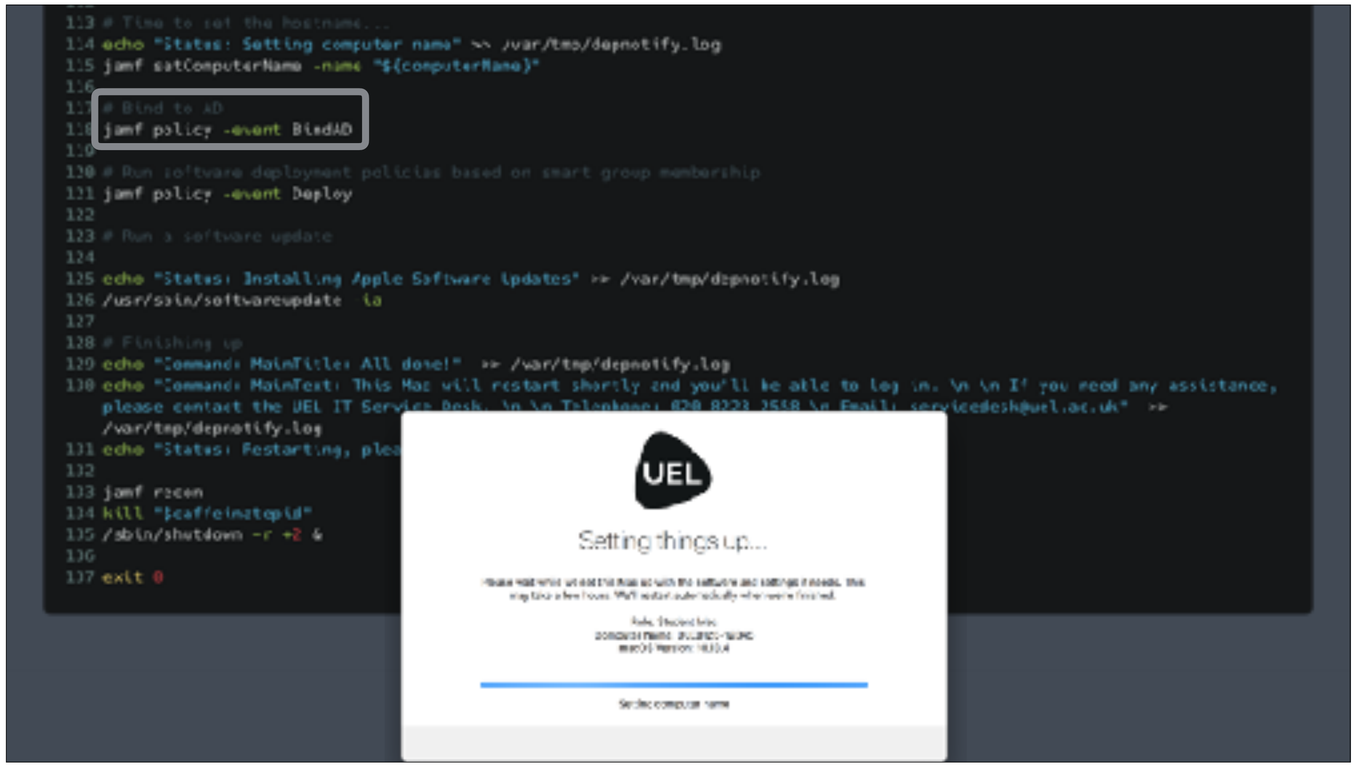
Setting things up...

[illegible]

[illegible]

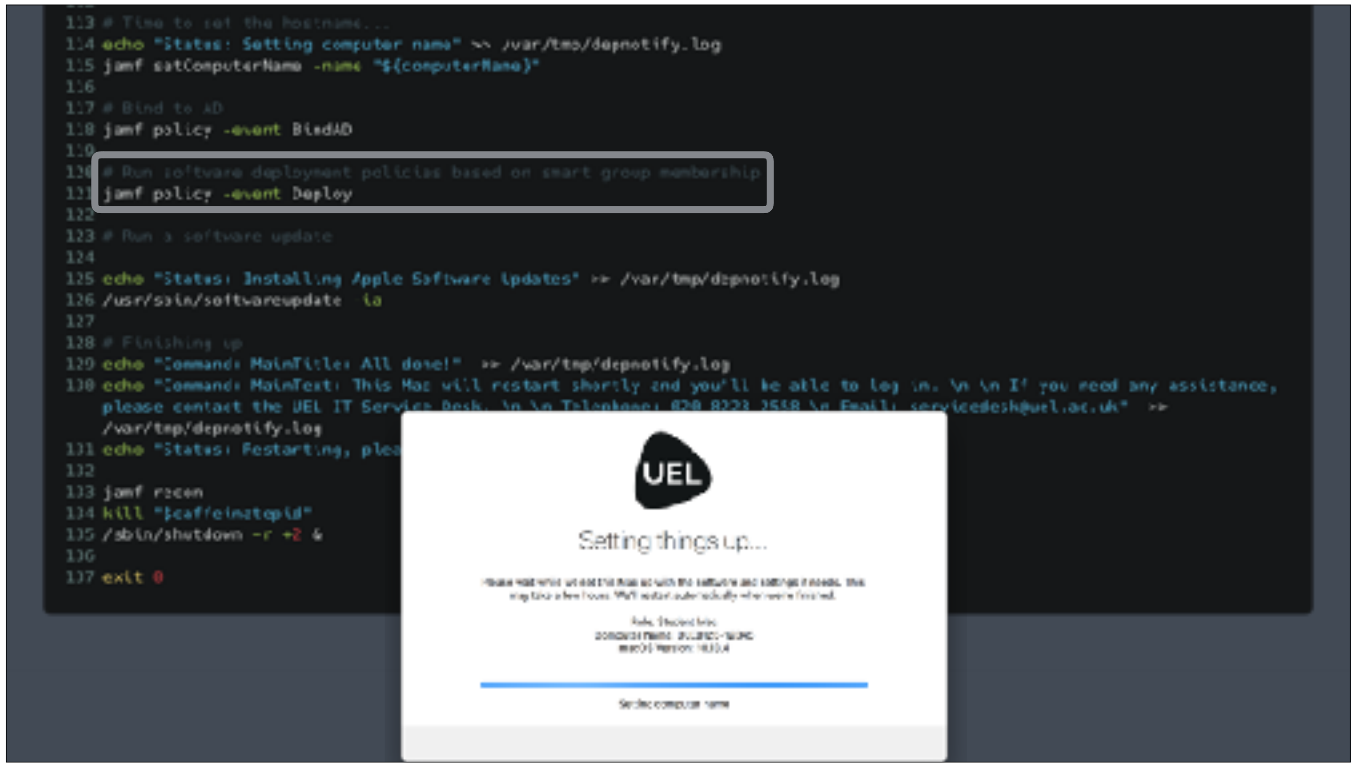
Setting computer name

We set the hostname



```
# Time to set the hostname...
echo "Status: Setting computer name" >> /var/tmp/depnotify.log
jamf setComputerName -name "${computerName}"

# Bind to AD
jamf policy -event BindAD

# Run software deployment policies based on smart group membership
jamf policy -event Deploy

# Run a software update

echo "Status: Installing Apple Software Updates" >> /var/tmp/depnotify.log
/usr/sbin/softwareupdate -ia

# Finishing up
echo "Command: MainTitle: All done!" >> /var/tmp/depnotify.log
echo "Command: MainText: This Mac will restart shortly and you'll be able to log in. \n \n If you need any assistance,
please contact the UEL IT Service Desk. \n \n Telephone: 020 8223 2558 \n Email: servicedesk@uel.ac.uk" >>
/var/tmp/depnotify.log
echo "Status: Restarting, plea

jamf recon
kill "$caffeinatepid"
/sbin/shutdown -r +2 &

exit 0
```

Bind to Active Directory

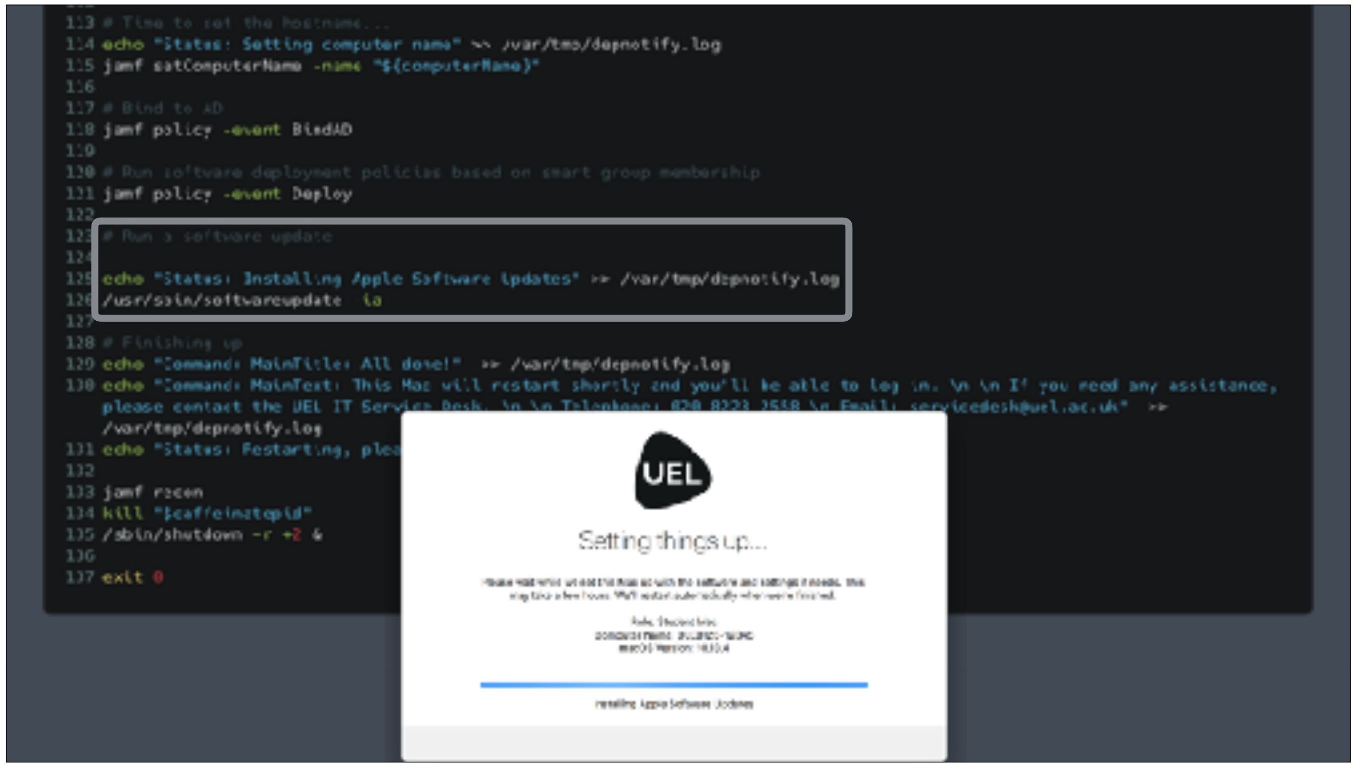


```
# Time to set the hostname...
echo "Status: Setting computer name" >> /var/tmp/depnotify.log
jamf setComputerName -name "${computerName}"

# Bind to AD
jamf policy -event BindAD

# Run software deployment policies based on smart group membership
jamf policy -event Deploy

# Run a software update

echo "Status: Installing Apple Software Updates" >> /var/tmp/depnotify.log
/usr/sbin/softwareupdate -ia

# Finishing up
echo "Command: MainTitle: All done!" >> /var/tmp/depnotify.log
echo "Command: MainText: This Mac will restart shortly and you'll be able to log in. \n \n If you need any assistance,
please contact the UEL IT Service Desk. \n \n Telephone: 020 8223 2558 \n Email: servicedesk@uel.ac.uk" >>
/var/tmp/depnotify.log
echo "Status: Restarting, plea

jamf recon
kill "$caffeinatepid"
/sbin/shutdown -r +2 &

exit 0
```

Setting things up...

[illegible]

And run any custom policy triggers we like. Our Mac will be in the right lab smart group and the relevant policies to install that lab's specific software will be scoped to that smart group. Because of that, those policies can all have the same custom trigger.



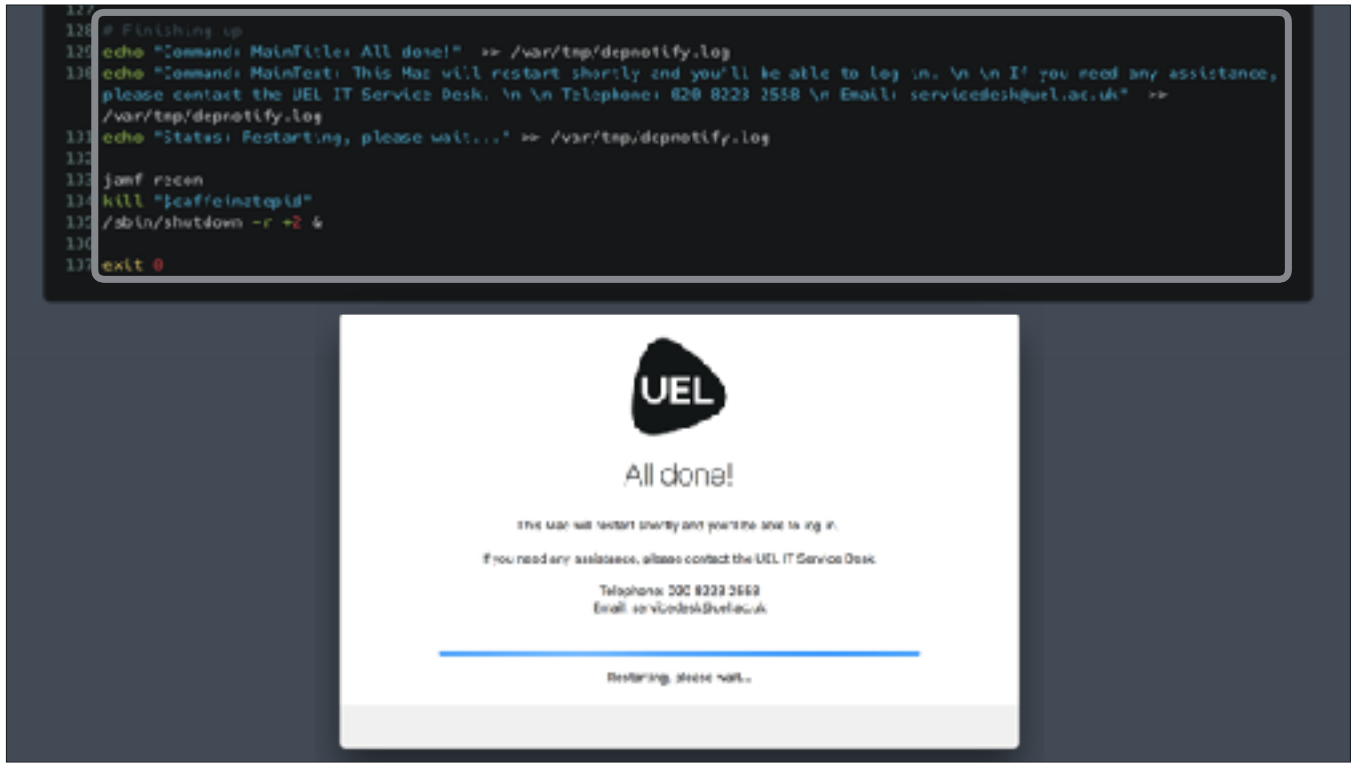
```
# Time to set the hostname...
echo "Status: Setting computer name" >> /var/tmp/depnotify.log
jamf setComputerName -name "${computerName}"

# Bind to AD
jamf policy -event BindAD

# Run software deployment policies based on smart group membership
jamf policy -event Deploy

# Run a software update

echo "Status: Installing Apple Software Updates" >> /var/tmp/depnotify.log
/usr/sbin/softwareupdate -ia

# Finishing up
echo "Command: MainTitle: All done!" >> /var/tmp/depnotify.log
echo "Command: MainText: This Mac will restart shortly and you'll be able to log in. \n \n If you need any assistance,
    please contact the UEL IT Service Desk. \n \n Telephone: 020 8223 2558 \n Email: servicedesk@uel.ac.uk" >>
    /var/tmp/depnotify.log
echo "Status: Restarting, plea

jamf recon
kill "$caffeinatepid"
/sbin/shutdown -r +2 &

exit 0
```

Setting things up...

[illegible]

Installing Apple Software Updates

We run a software update - that’s important because when you install a clean version of macOS, it doesn’t include security updates or other updates like iTunes and Safari, like an image can.



```
# Finishing up
echo "Command: MainTitle: All done!"  >> /var/tmp/depnotify.log
echo "Command: MainText: This Mac will restart shortly and you'll be able to log in. \n \n If you need any assistance, please contact the UEL IT Service Desk. \n \n Telephone: 020 8223 2558 \n Email: servicedesk@uel.ac.uk"  >> /var/tmp/depnotify.log
echo "Status: Restarting, please wait..." >> /var/tmp/depnotify.log

jamf recon
kill "$caffeinatepid"
/sbin/shutdown -r +2 &

exit 0
```

All done!

This Mac will restart shortly and you'll be able to log in.

If you need any assistance, please contact the UEL IT Service Desk.

Telephone: 020 8223 2558
Email: servicedesk@uel.ac.uk

Restarting, please wait...

And we’re finished, so we tell DEPNotify to let us know, update inventory and restart with a 2 minute delay - so the policy has time to complete and write its log back to Jamf.

The future

- Skip the Setup Assistant entirely
- Do it without logging in
- Asset management DB integration
- Apple don’t sit still

So, I said at the beginning that this was a journey I was TAKING. There are lots of places we could continue to take this to.

<C> This is not zero touch - it’s as close as I can get it, but maybe Apple could include an option to skip the entire Setup Assistant

<C> Do we really need to log in? At the moment, yes. DEPNotify won’t run on top of the Login Window. But that might change and there are some exciting conversations happening…

<C> Instead of storing and reading the hostname and role from within Jamf, we could point to an asset management tool and store that data outside. That means we could delete the computer record in Jamf before we start, but still skip asking for the data. I do sometimes notice issues like profiles not pushing properly when we re-provision a Mac with an existing computer record. - this might improve things.

<C> Apple change stuff. Our workflows might break tomorrow. We can only do what we can. And because I called this talk “Dawn of the DEP”…

jamf NATION Roadshow

It might be best to go to the Winchester, have a nice cold pint, and wait for all this to blow over.

jamf NATION Roadshow

Thank you