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.

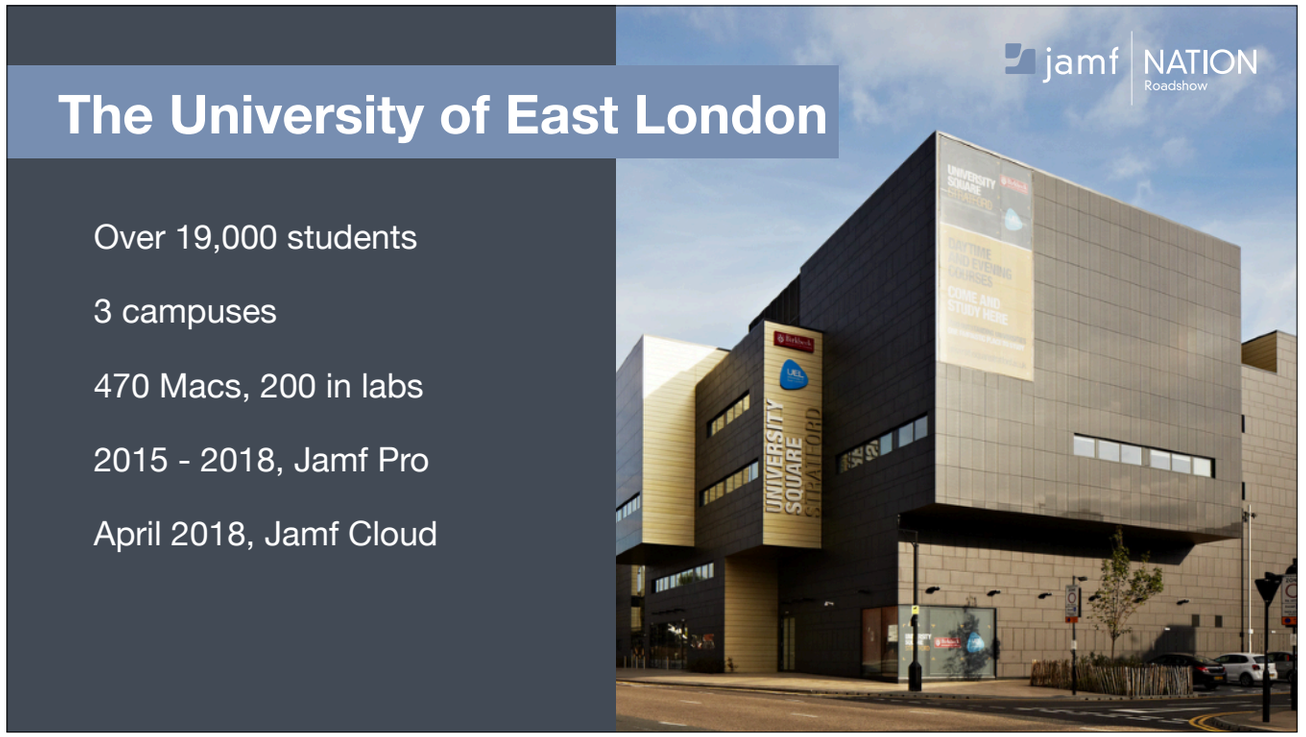
Are you concerned about the direction Apple is taking as it seems to be dropping support for this? Me too. Who’s moving, or has moved their workflows to DEP? Me too - ish.

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.

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.



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.

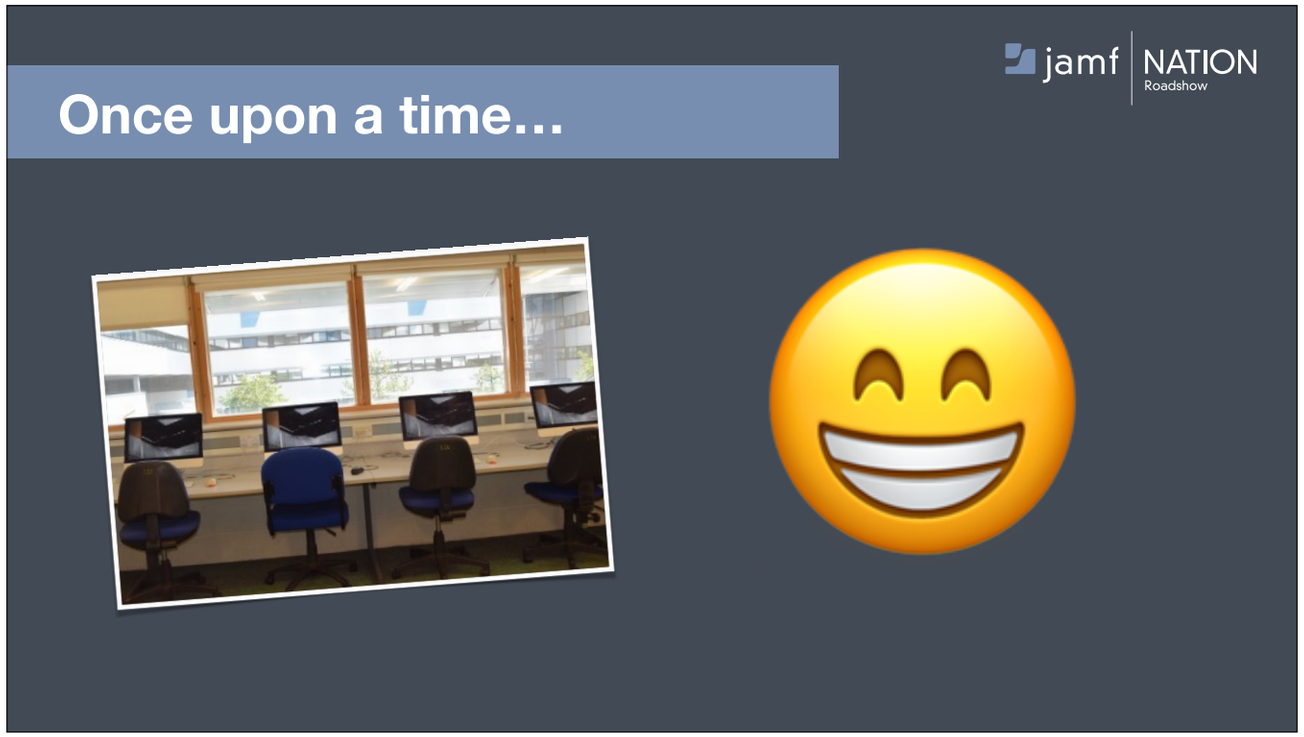
We’ve been managing our Macs with Jamf Pro since 2015 and we’ve just migrated to Jamf Cloud. The migration was really smooth - we worked with Moof-IT and Jamf, who were both great - and it meant we could get rid of 4 on-premise servers, saving my team and I time as well as resources.

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/2018/05/17/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.



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.

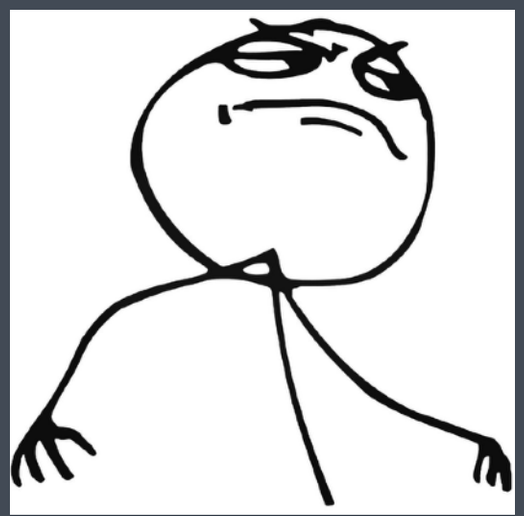
So, you’d NetBoot the lab, remotely.

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

Once upon a time…

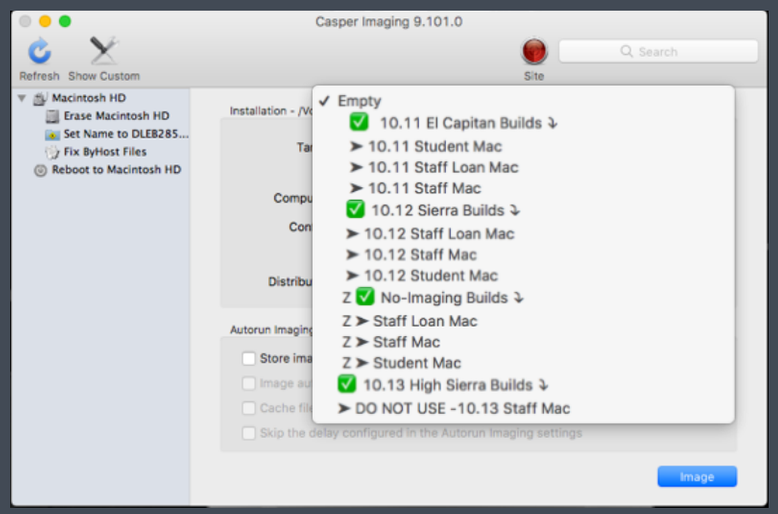
Magic!

Once upon a time…



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

Once upon a time…



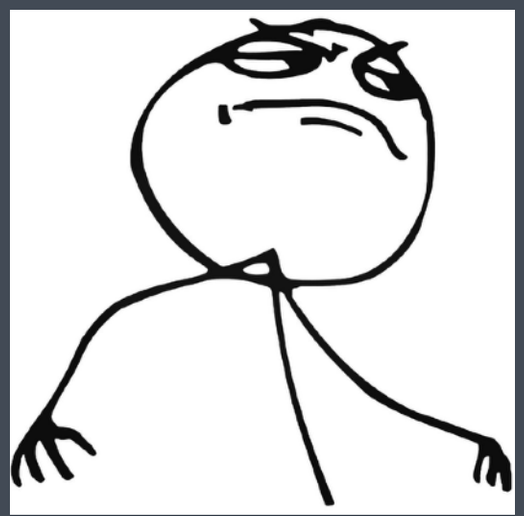
If you were provisioning a new Mac, you'd give it a hostname

<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.

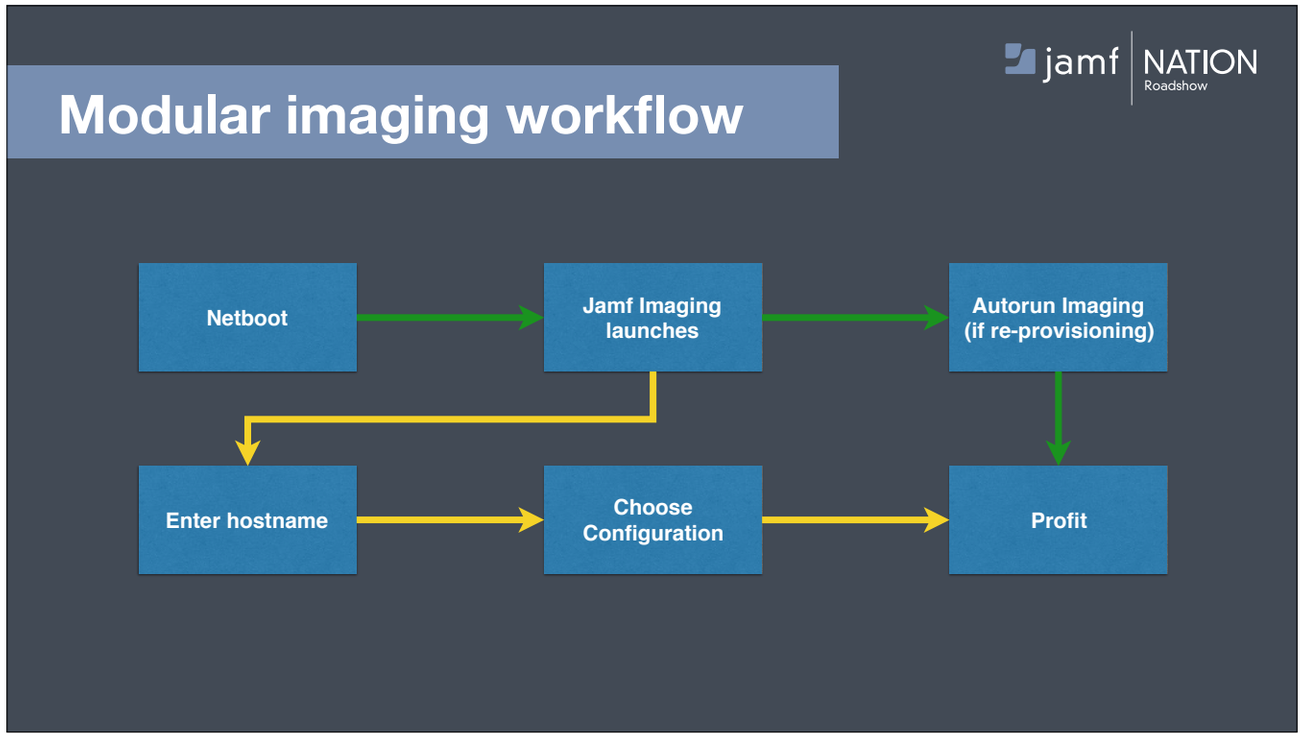
Its role (whether the Mac was staff or student facing), was stored in an Extension Attribute.

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.

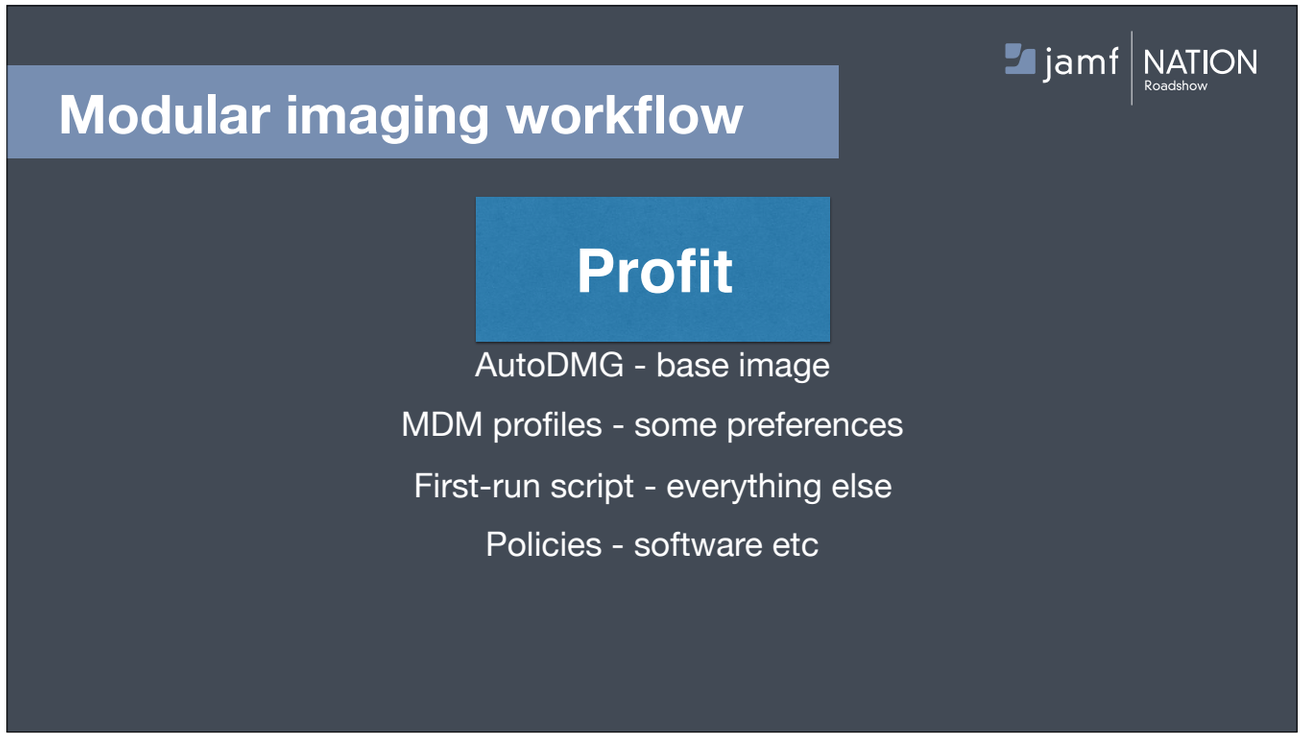
Once upon a time…



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



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



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

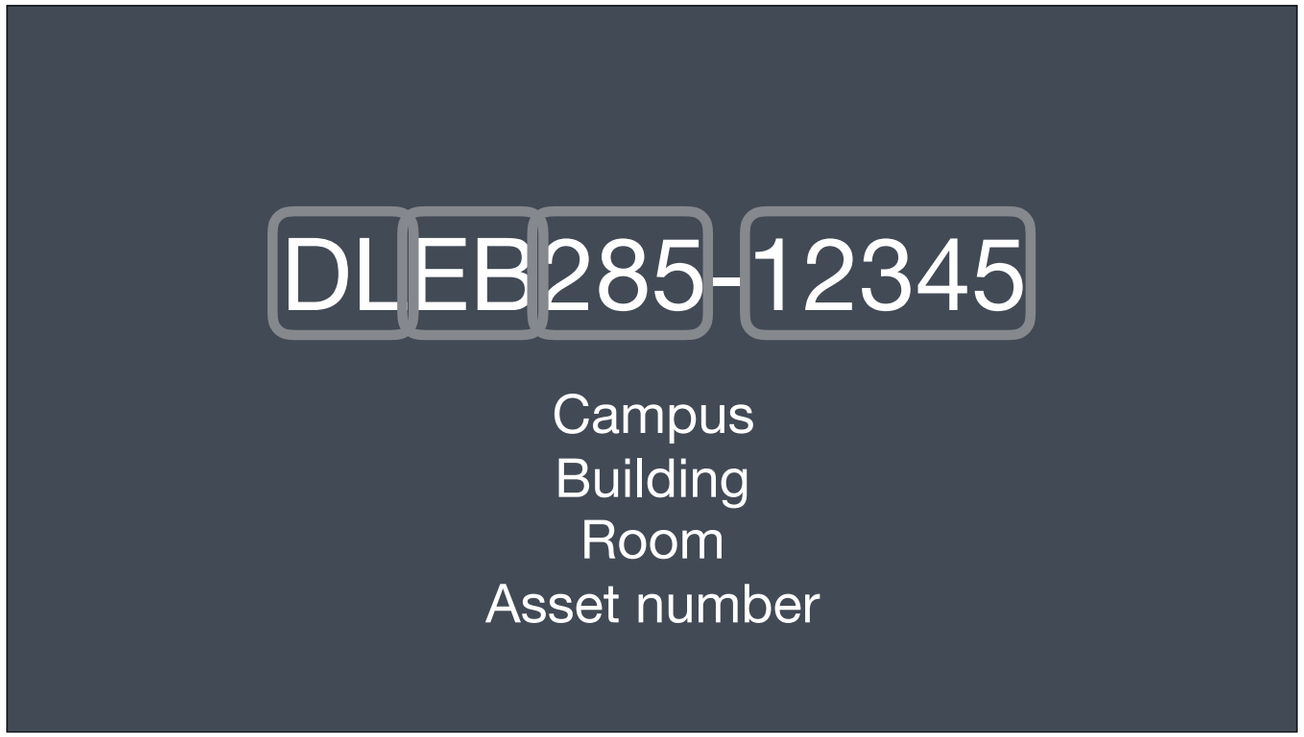
<C> have MDM profiles set most preferences

<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

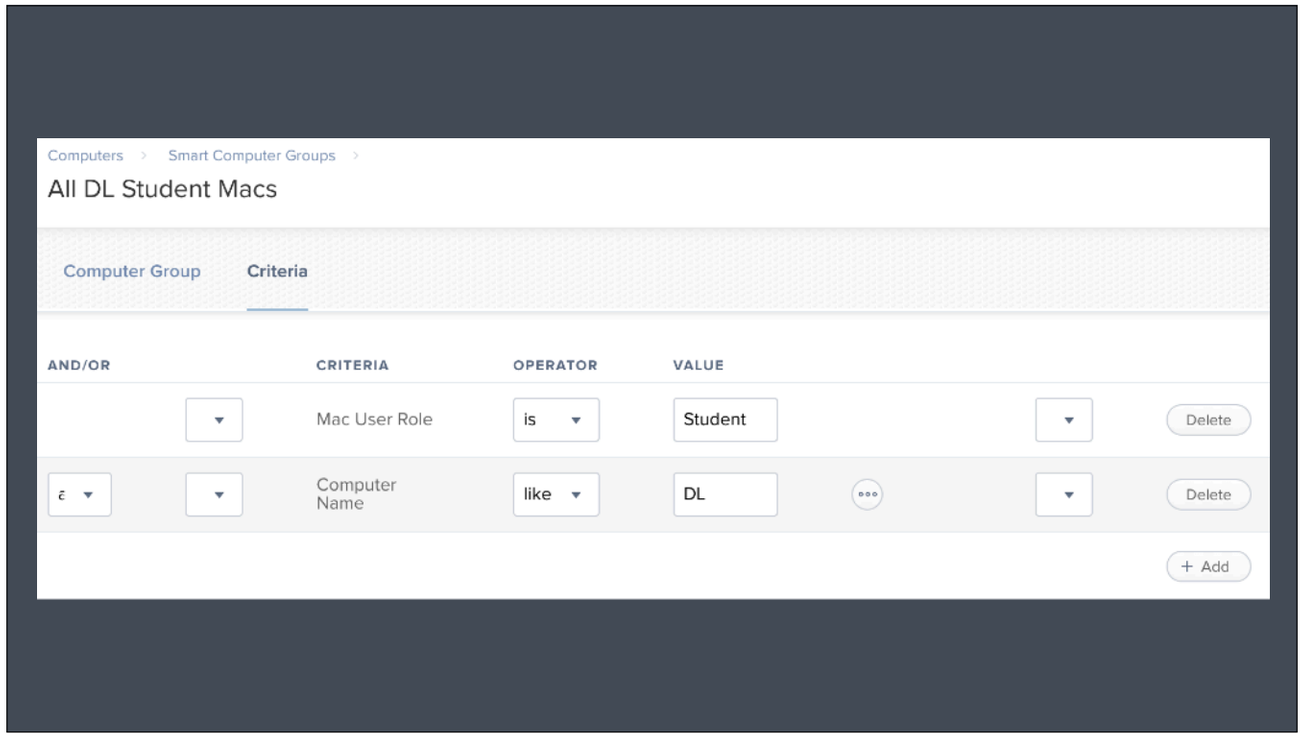
The hostname is important to us…

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

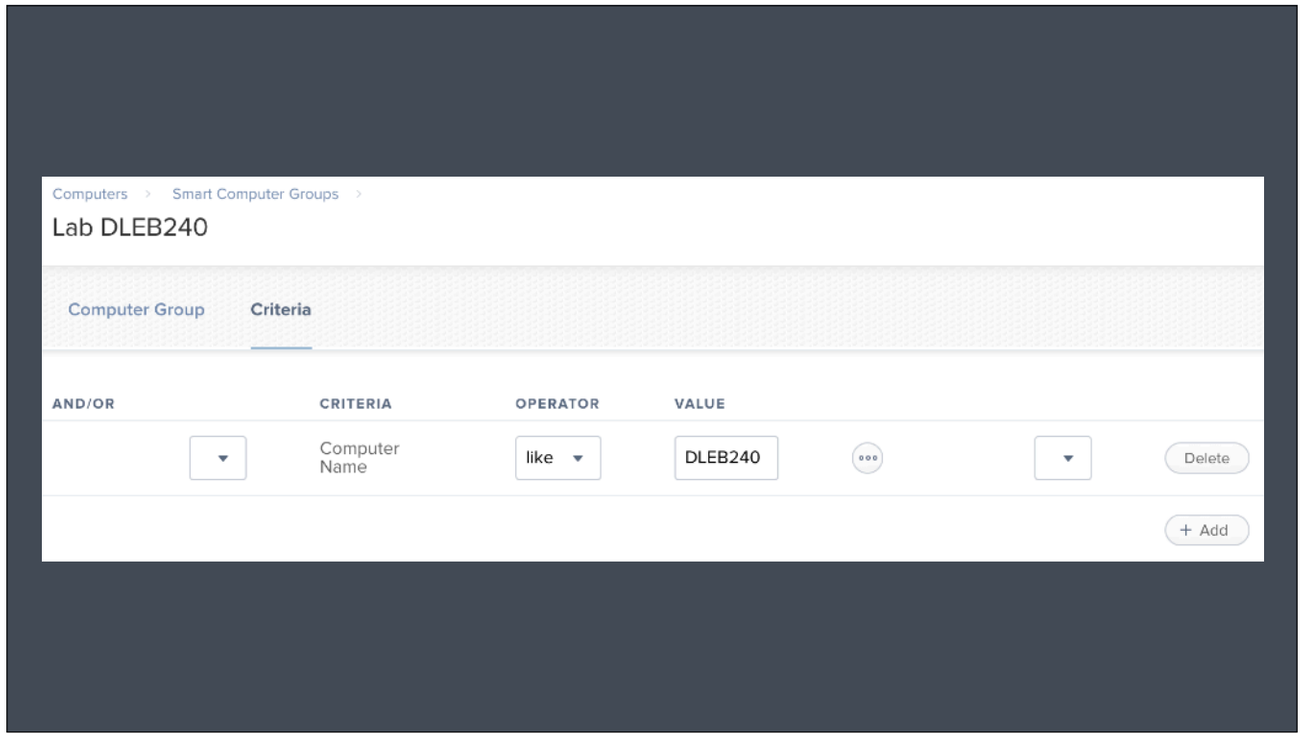


Because our hostnames tell us the <C> Campus (Docklands), <C> Building (East Building), <C>, Room number <C> and our own internal Asset number.

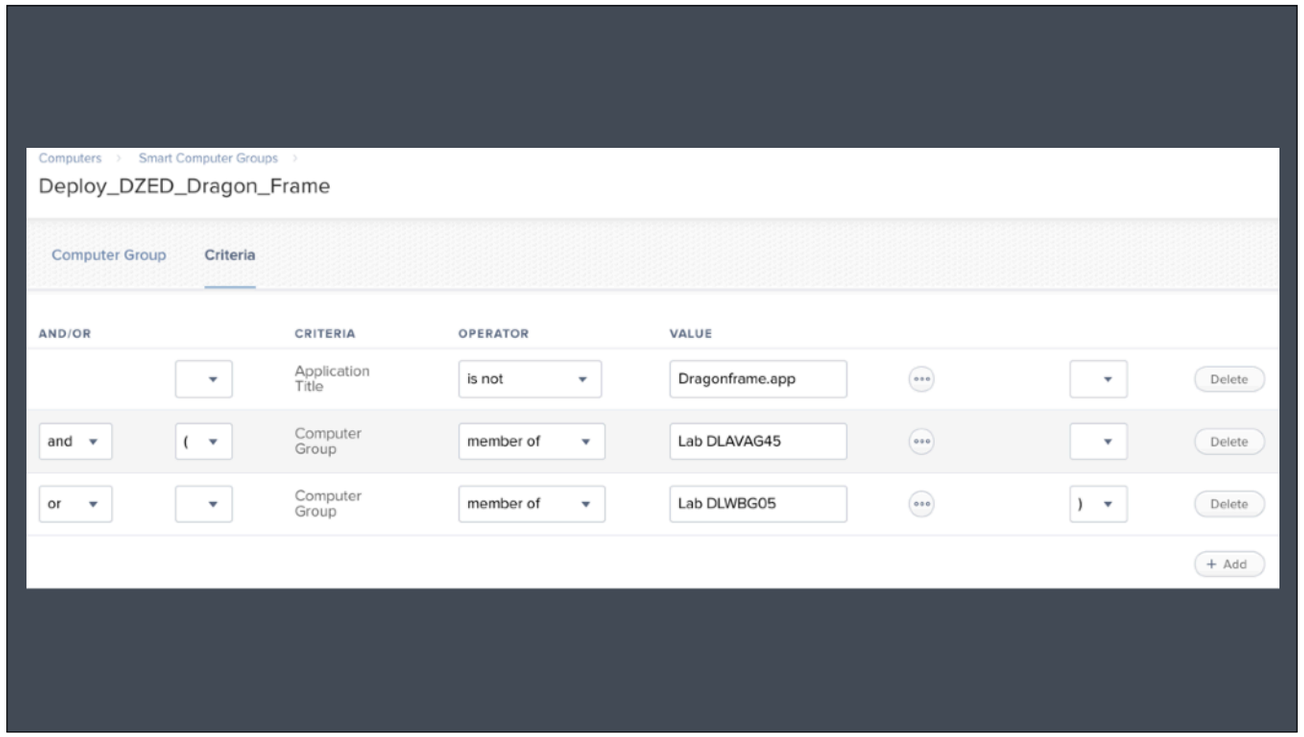
This means we can create smart groups, based on the computer name…



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



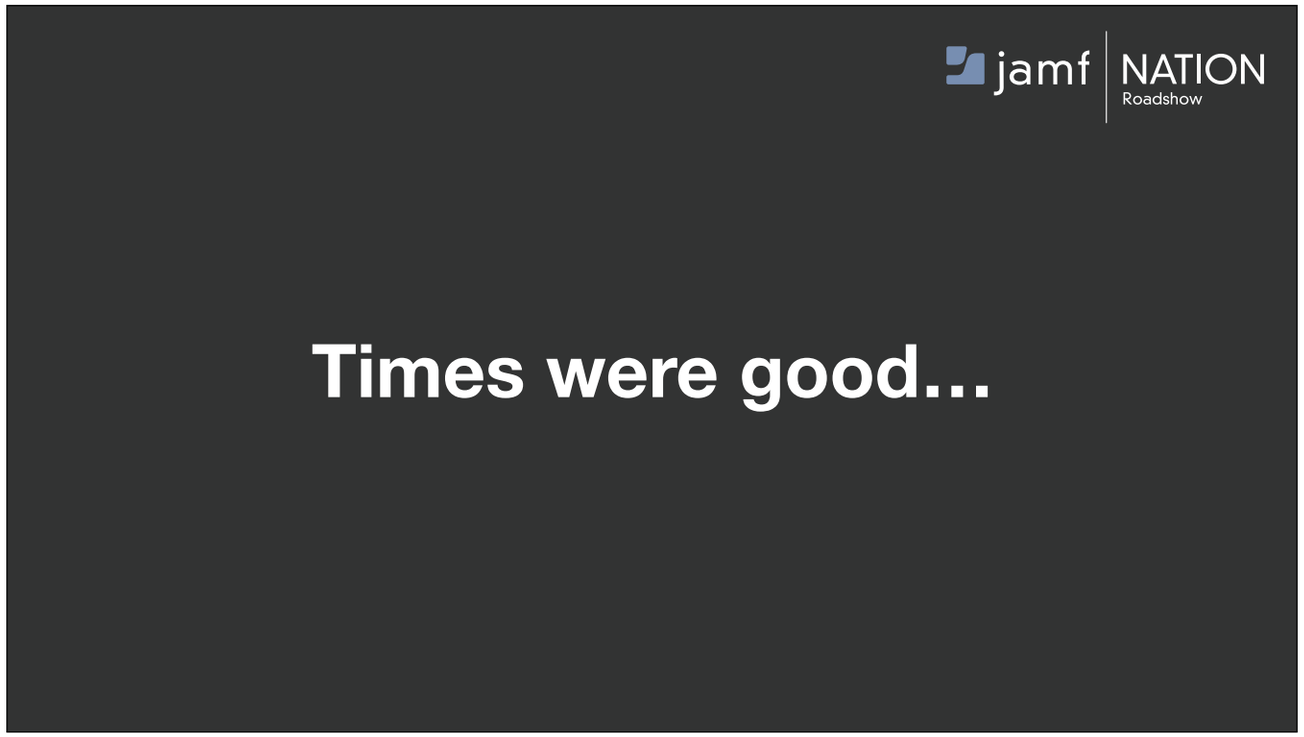
…or have all the Macs in a specific room or lab…



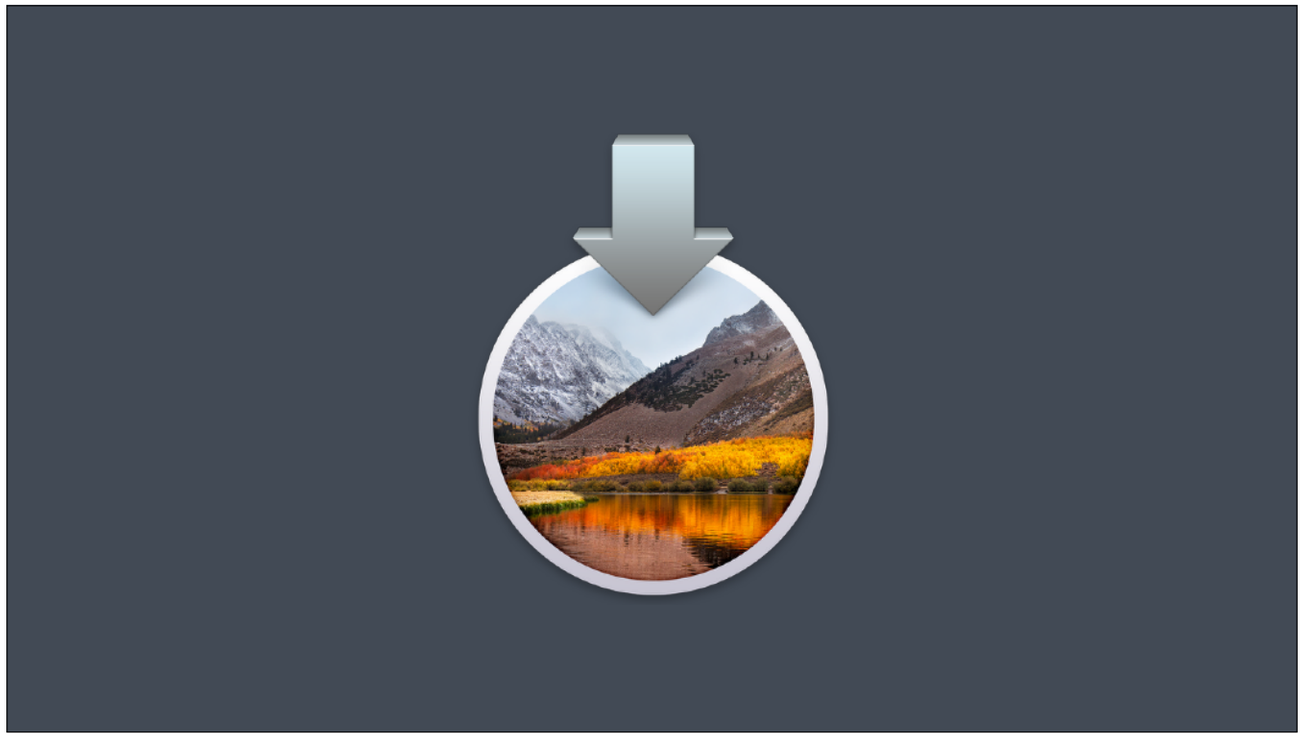
When it comes to deploying an application, say Dragonframe, we could have a Smart Group that looks for Macs that are in the labs that require it but don’t have it installed.

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!



Then High Sierra came along…

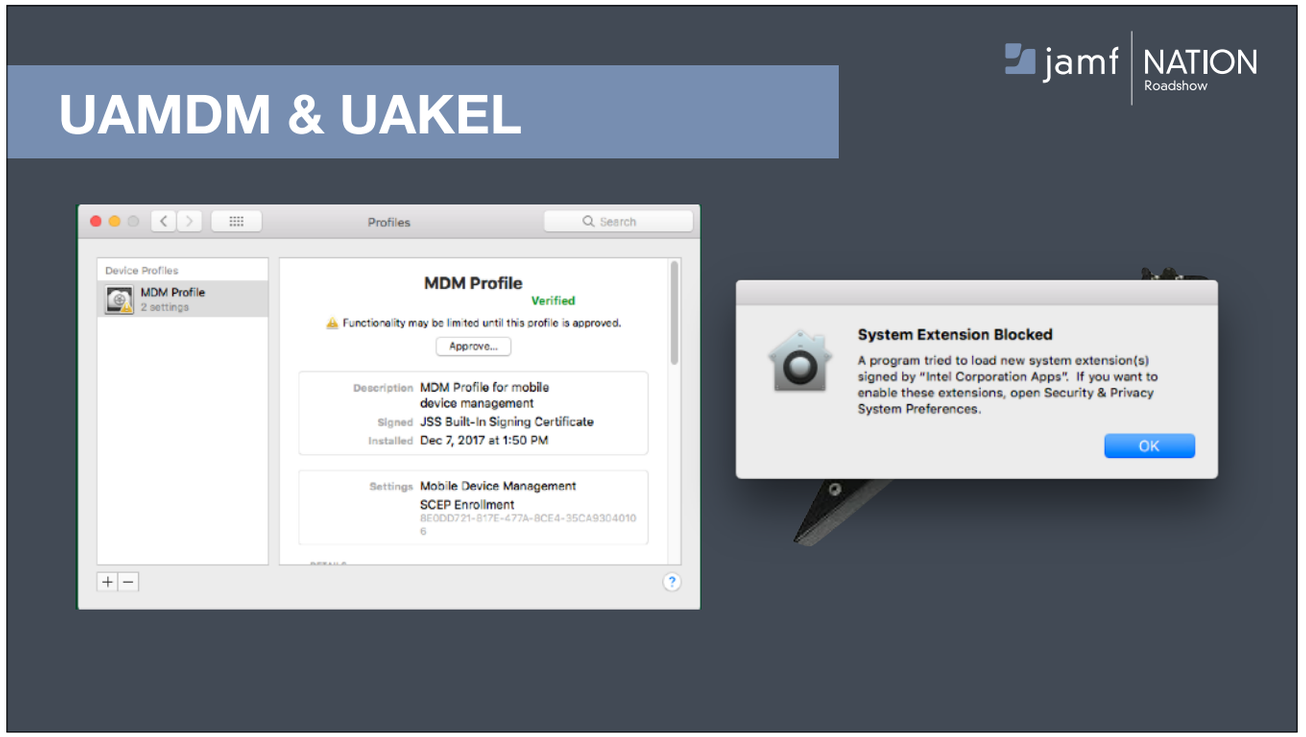
jamf NATION Roadshow

> 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

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.



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.

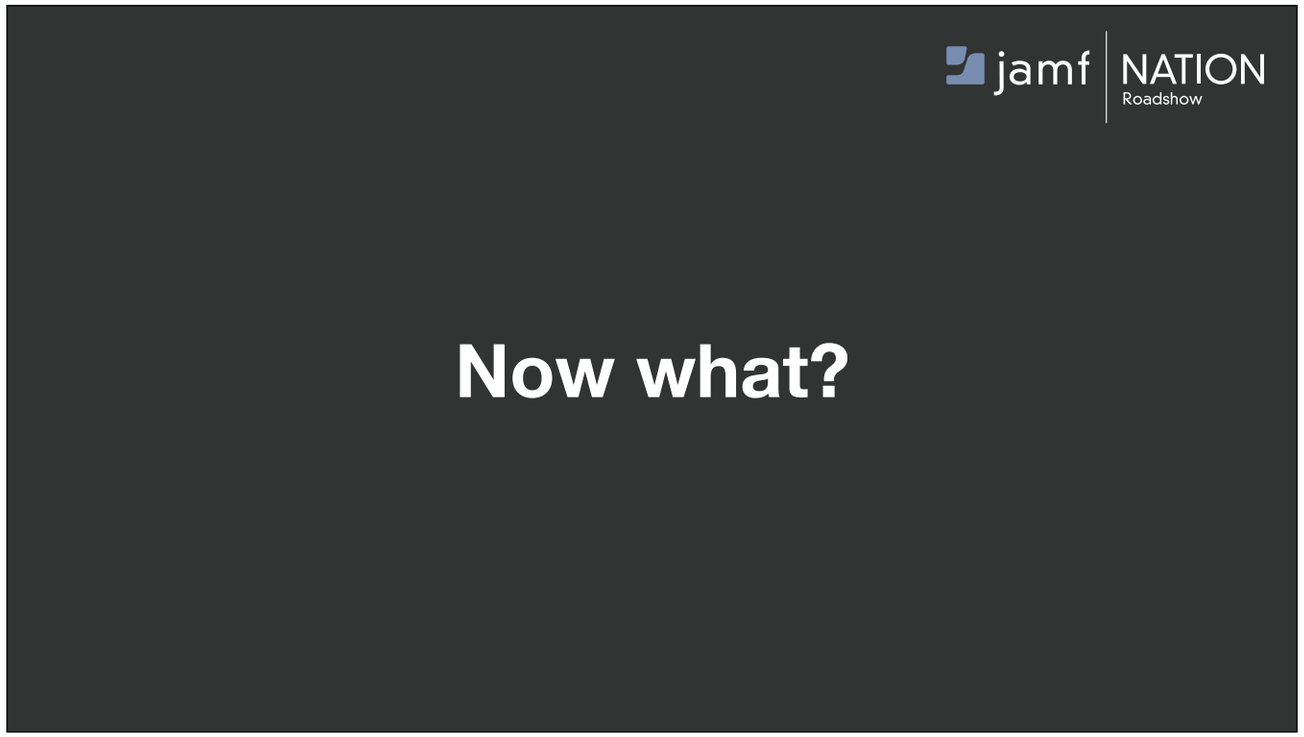
<C> or Ukulele. You can't even click that Approve button remotely - you have to use a physical mouse on the actual Mac. Once you've approved MDM, you can push a profile to approve specific extensions, or the all the extensions from a specific vendor.

isimagingdead.com

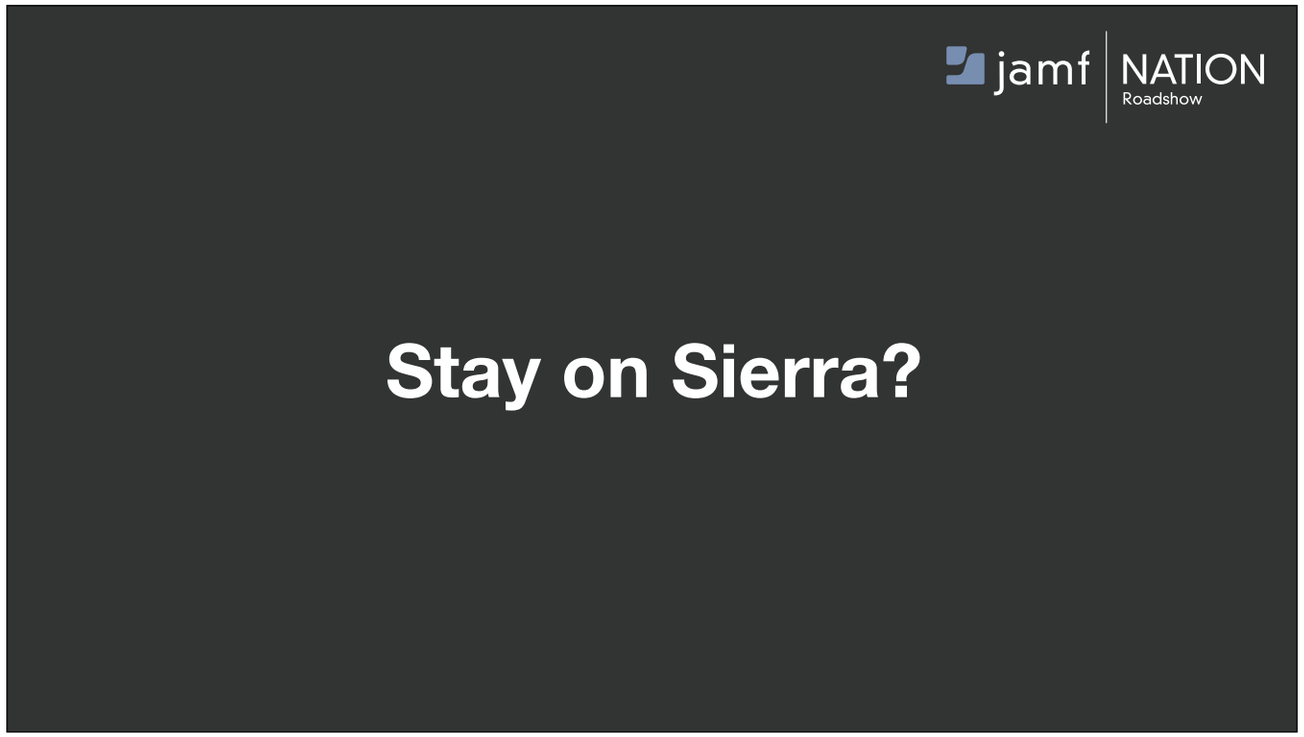
?

So, is imaging dead?

It’s looking that way.

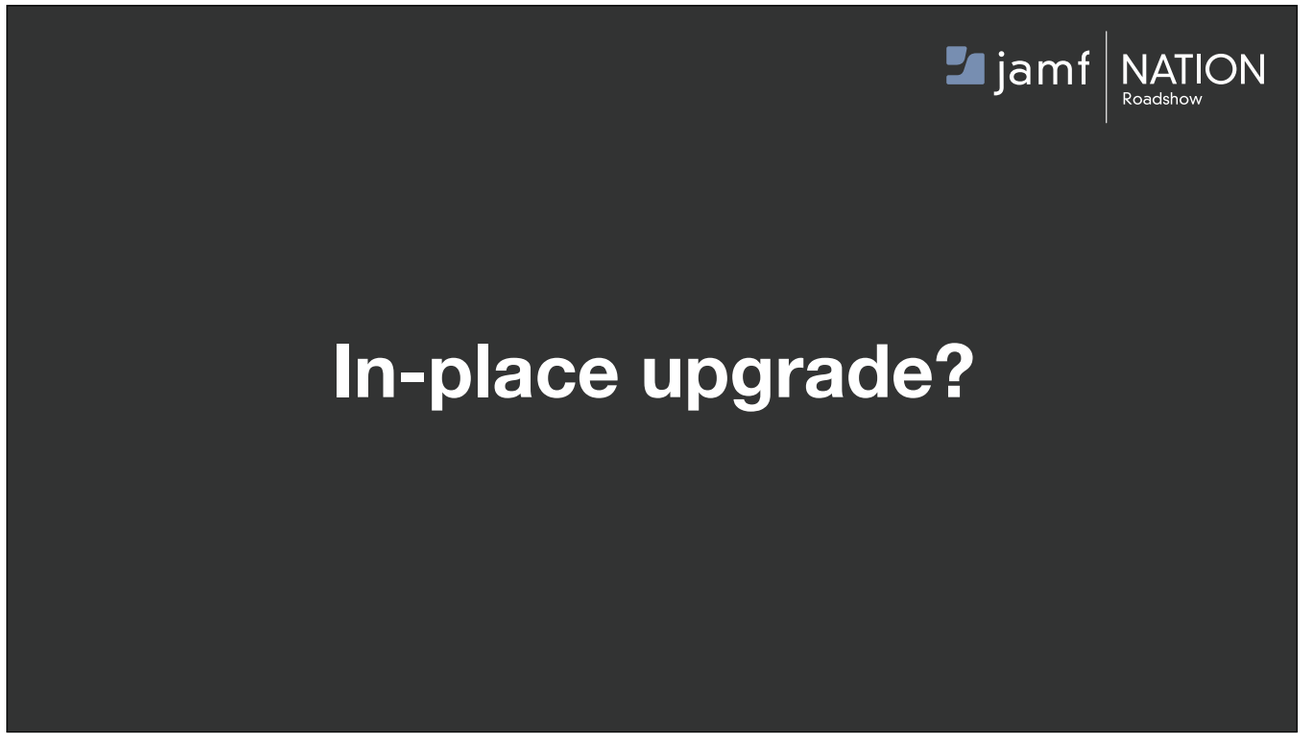


So, what can we do?



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…

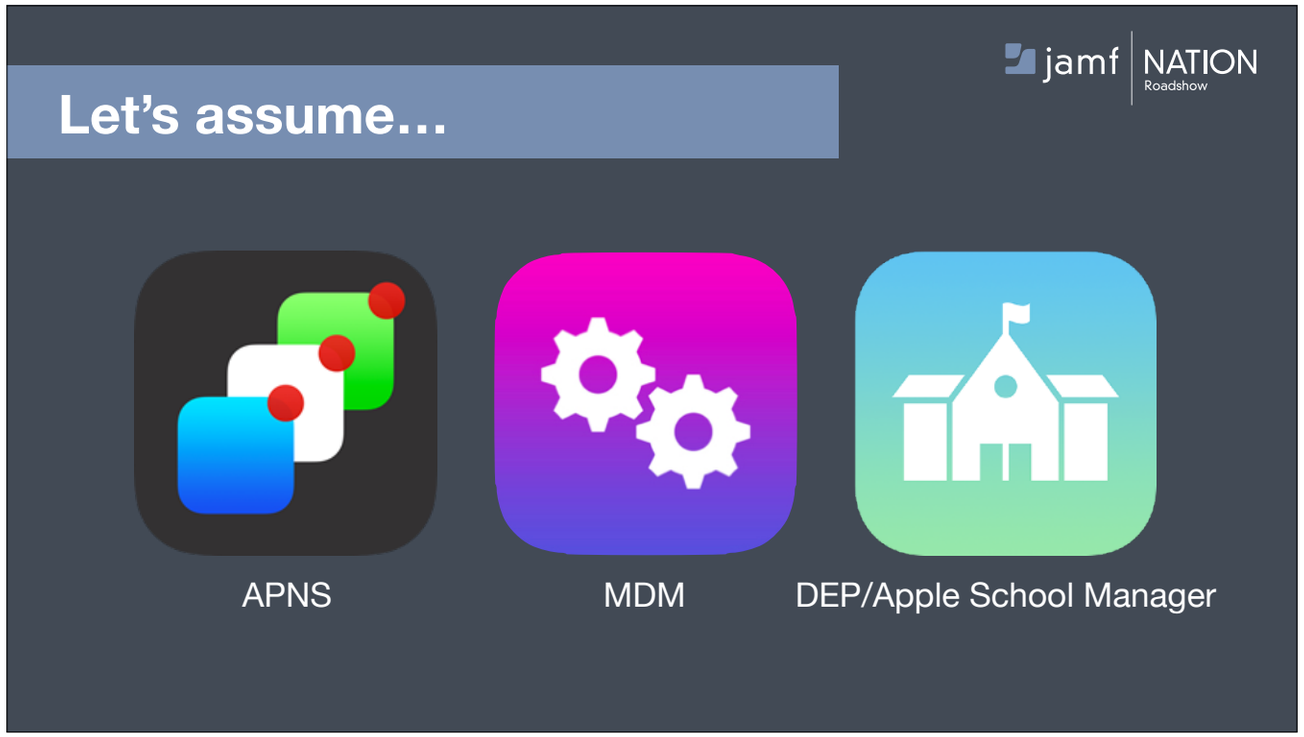


We could deploy 10.13 as an upgrade. You'd get the right firmware, you wouldn't have to deal with User Approved MDM and you wouldn't have to change your imaging workflows.

I don't particularly like this one either.

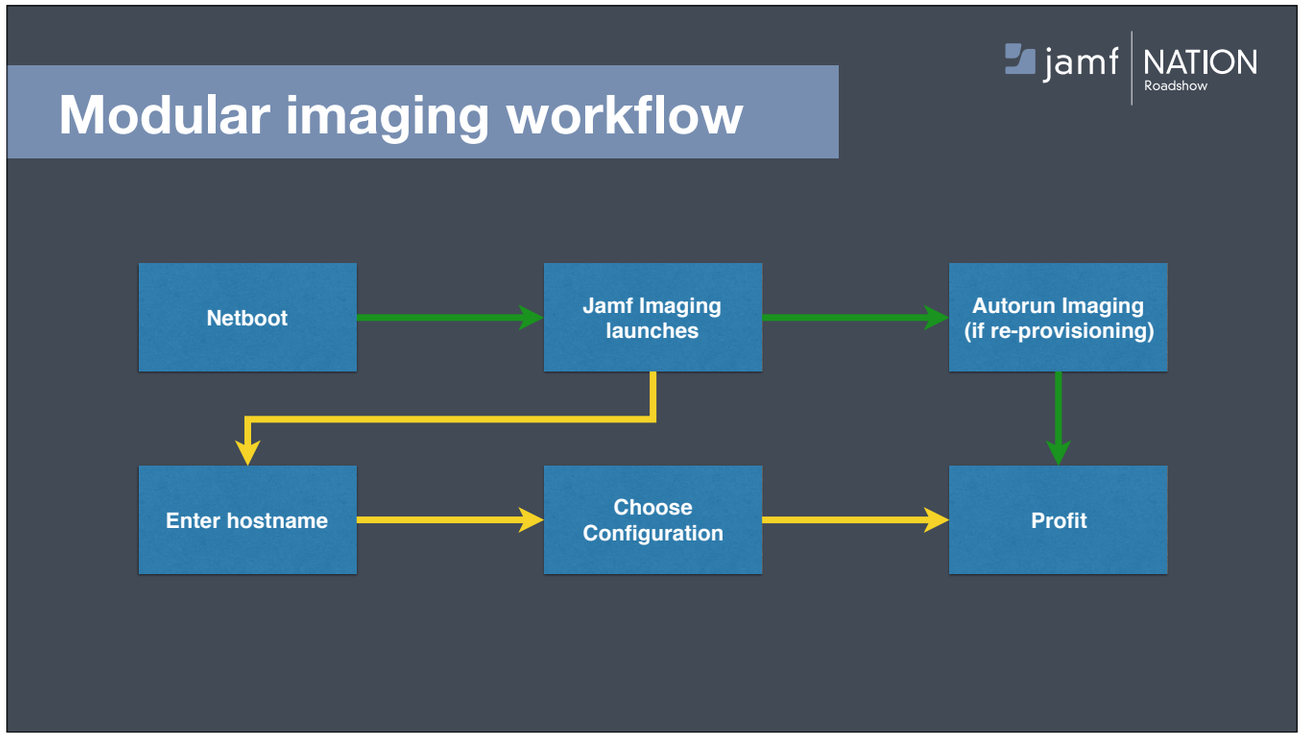
You end up with bits left over from the previous OS and some Apps that install things all over the place. What about new Macs?

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.



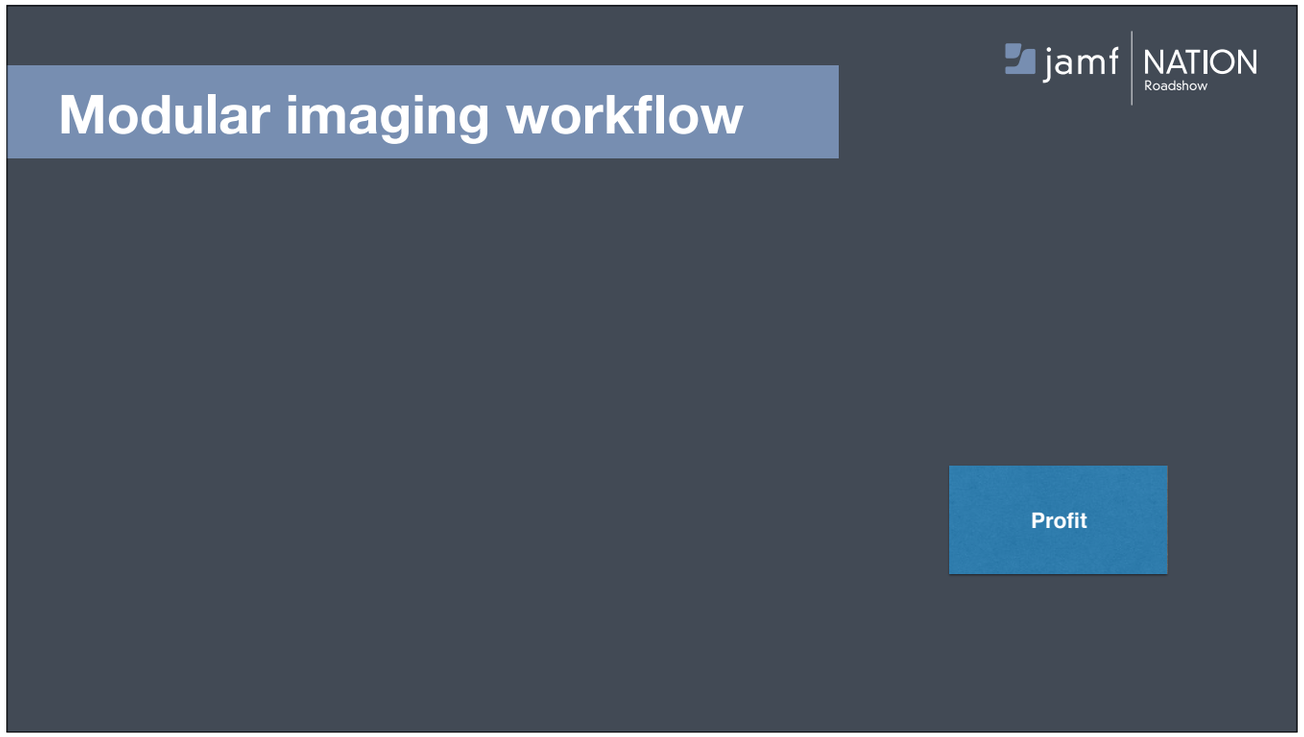
At this point I’m going to assume APNS is allowed over the network and that we’re using MDM.

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.



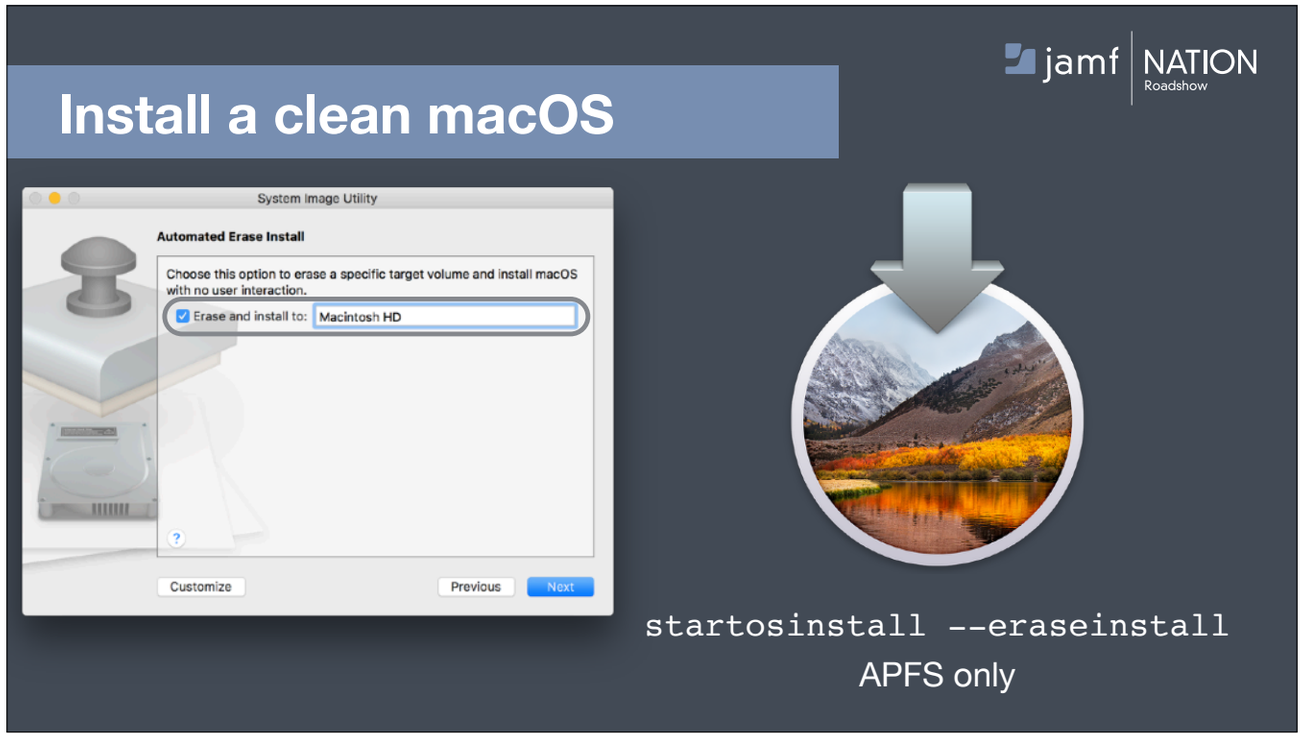
Let’s look at the old workflow again.

We need to complete most of the steps first…



…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…

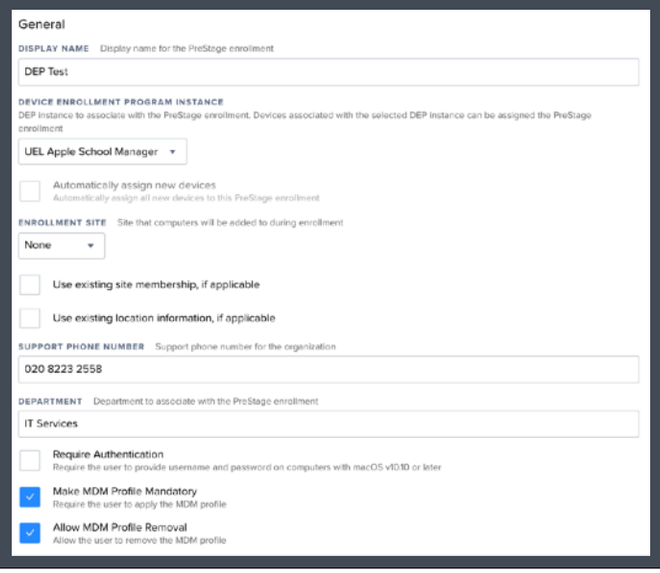


Now, with High Sierra 10.13.4, Apple have given us a couple of ways to achieve that.

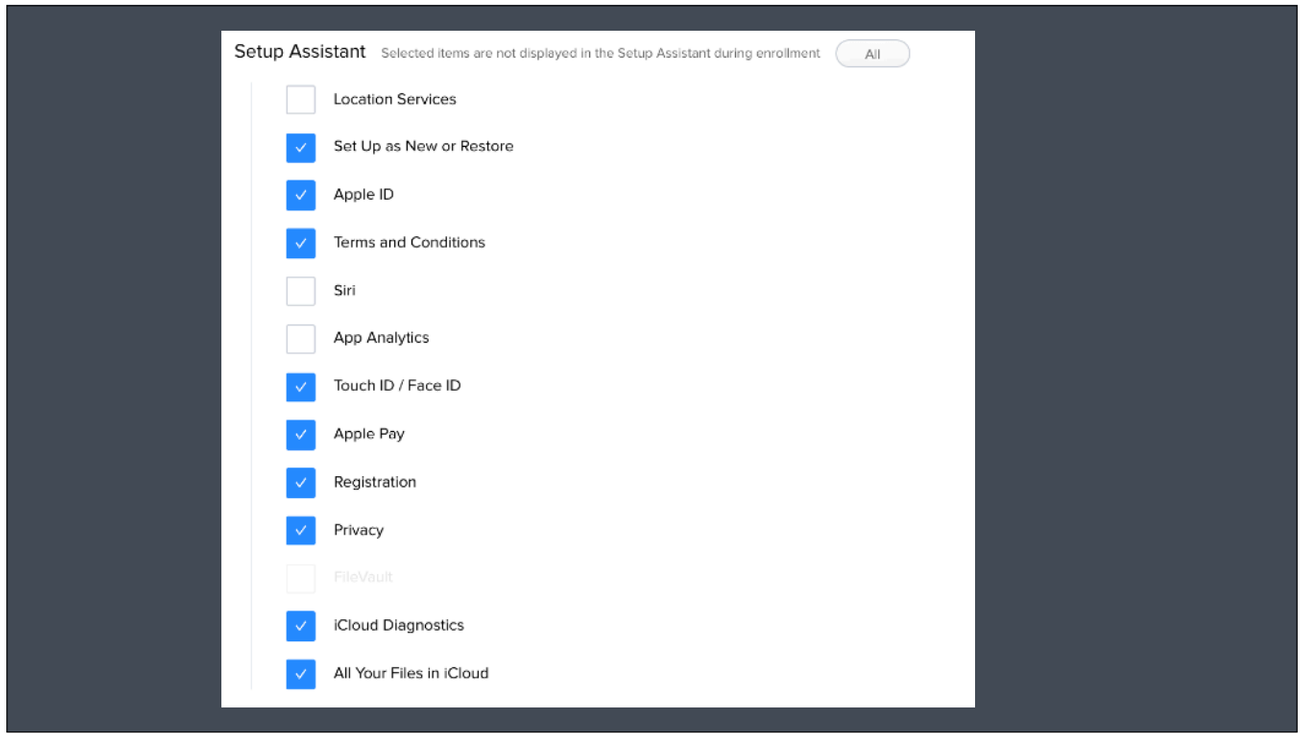
For the moment, you can still use NetBoot - create a NetInstall set with System Image Utility and set it up to automatically erase and install (this was broken with 10.13 but fixed in the update).

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

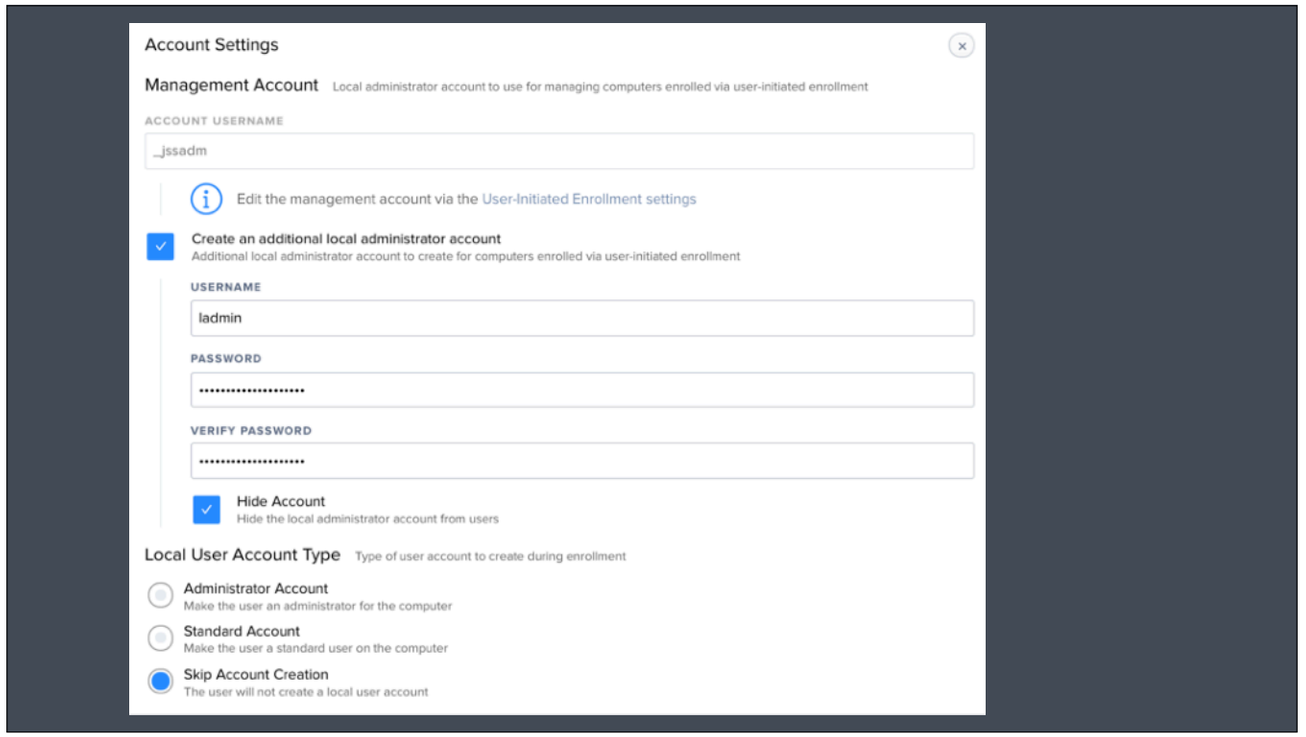
PreStage Enrolment



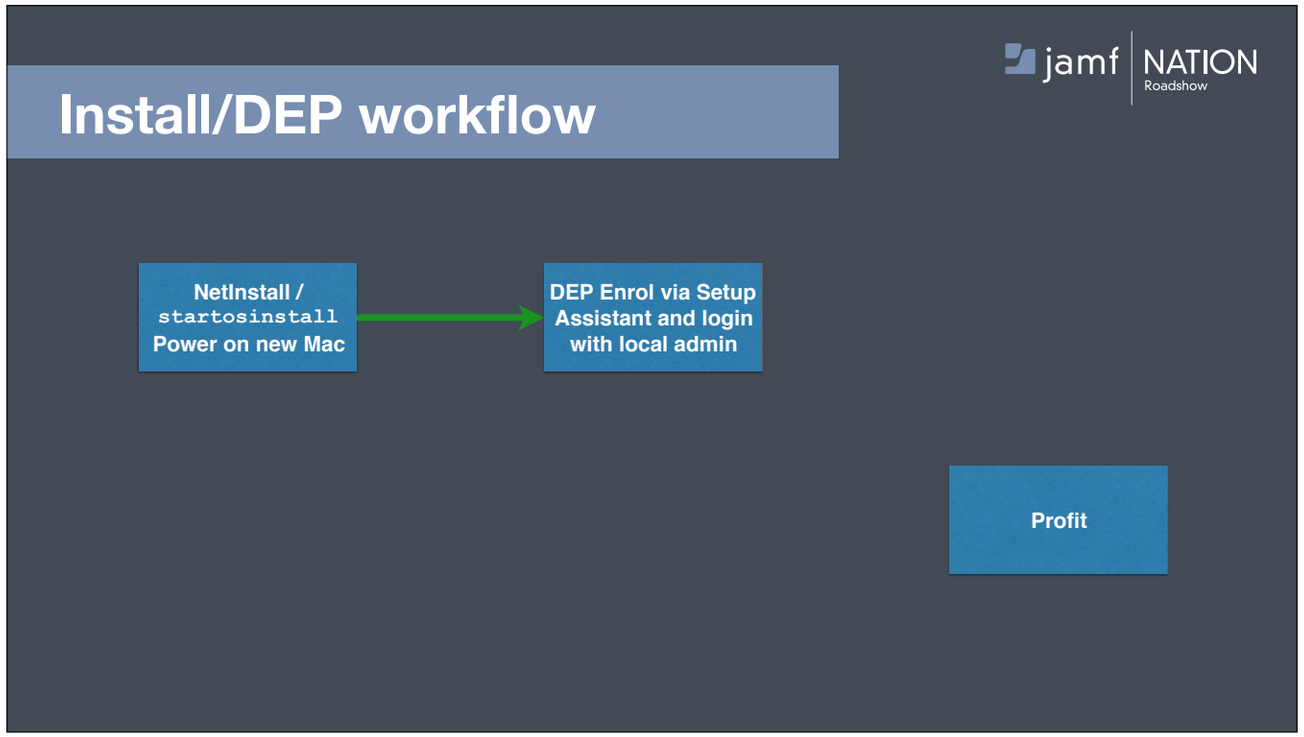
Next, You’d need to set up a PreStage Enrolment in Jamf



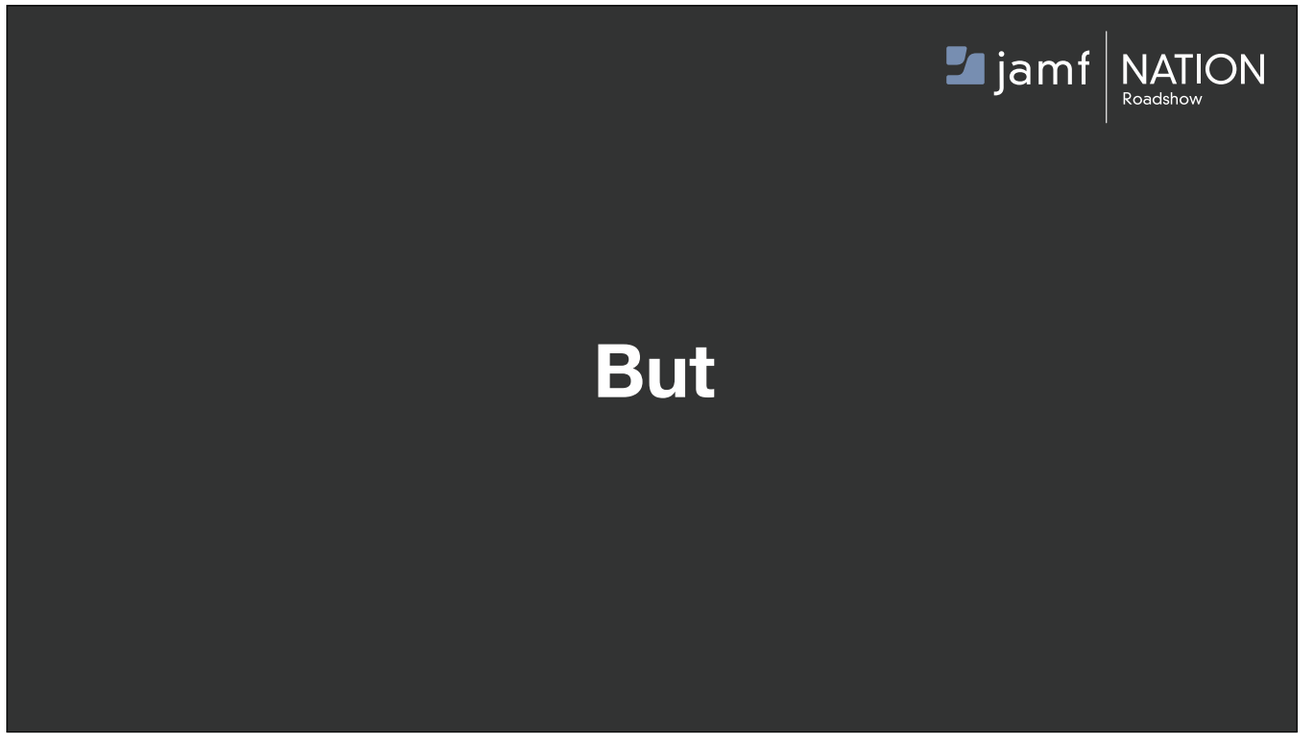
…this can skip the parts of the Setup Assistant you want to…



…and create a local administrator account.



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



However…

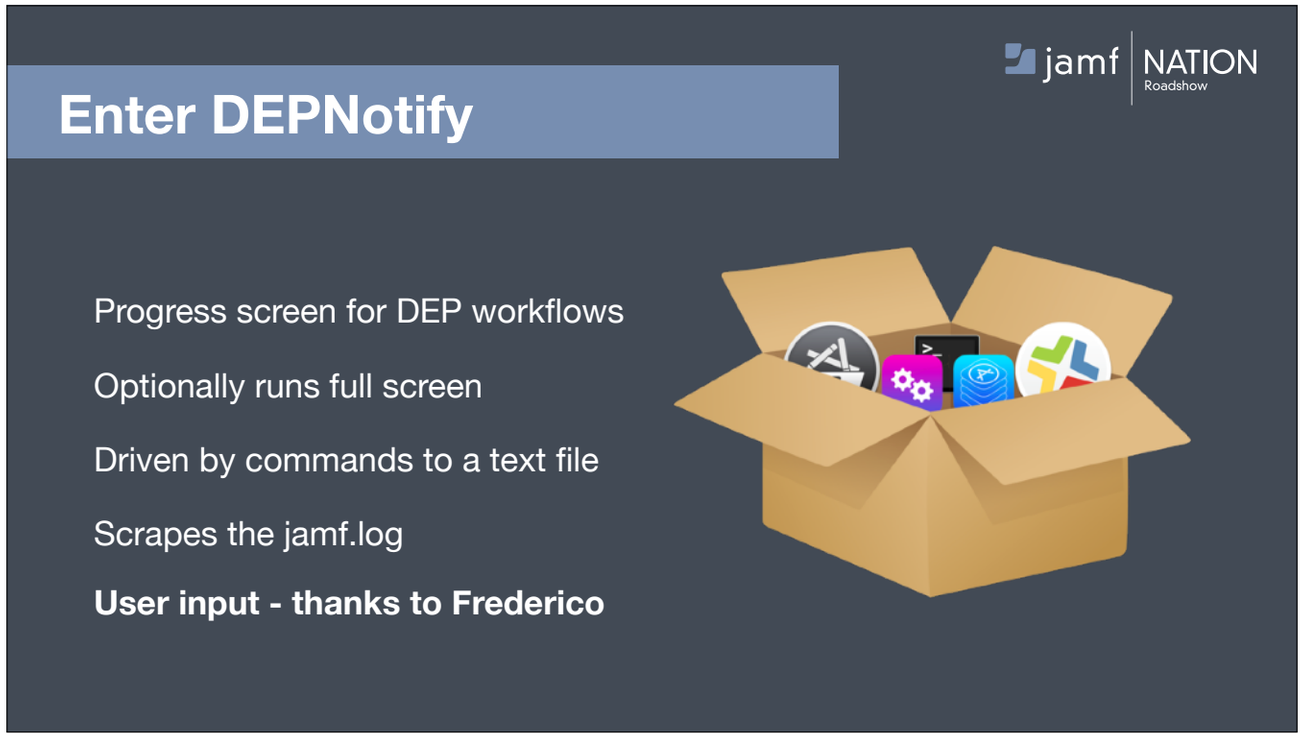
The hostname is important to us…

We still want that hostname.

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…



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…

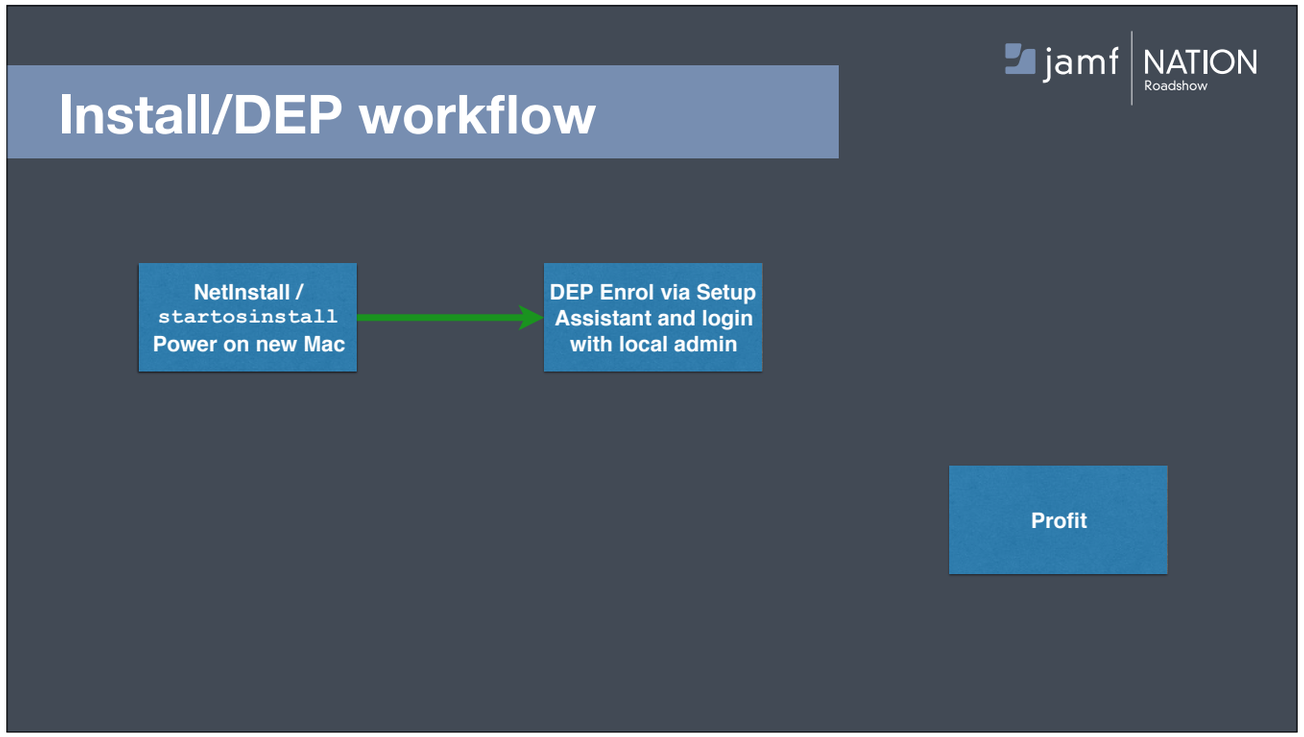
Meanwhile, in #depnotify

fgd 00:02

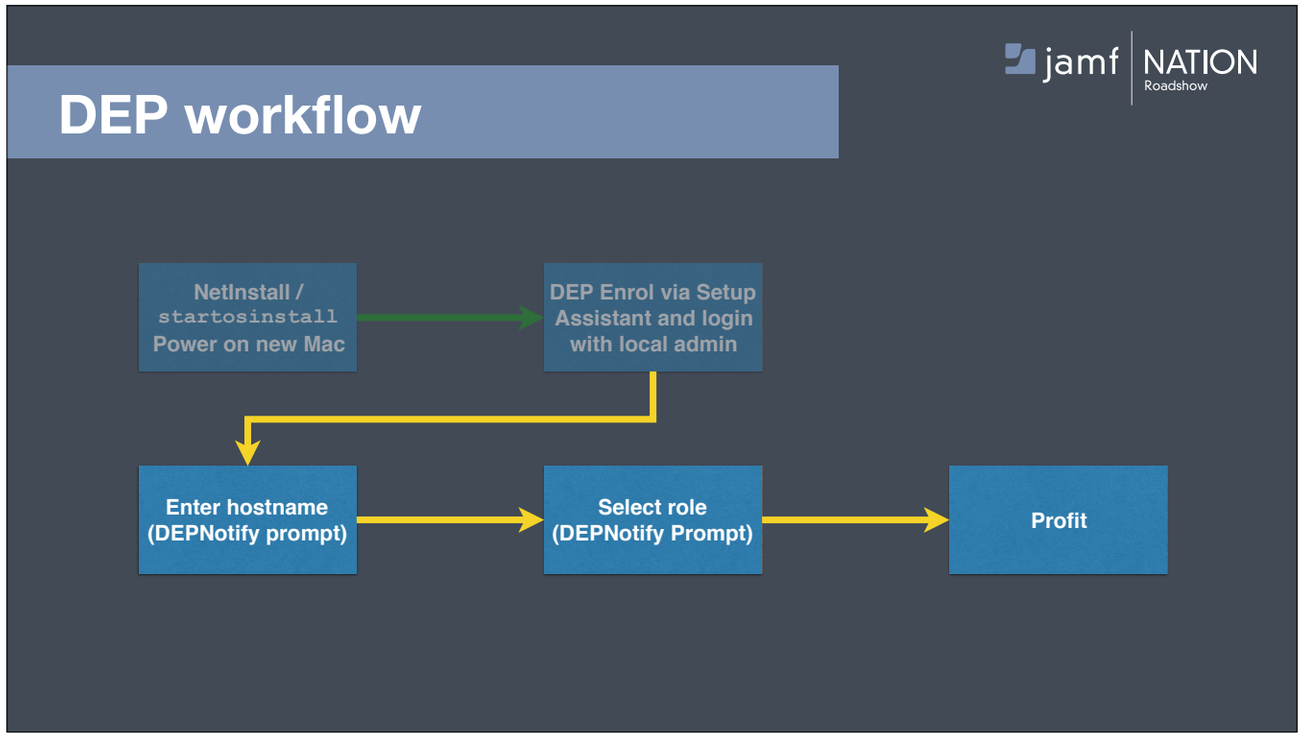
So I'm working on a version of DEPNotify that will accept user input for 2 text fields and two pop up menus, all customizable by defaults keys. You'd be able to call the user input dialog at any time with a button trigger. Labels, placeholders, text field character compliance, popup menu content, window 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 NoLotify to be run on top of the login window.

2

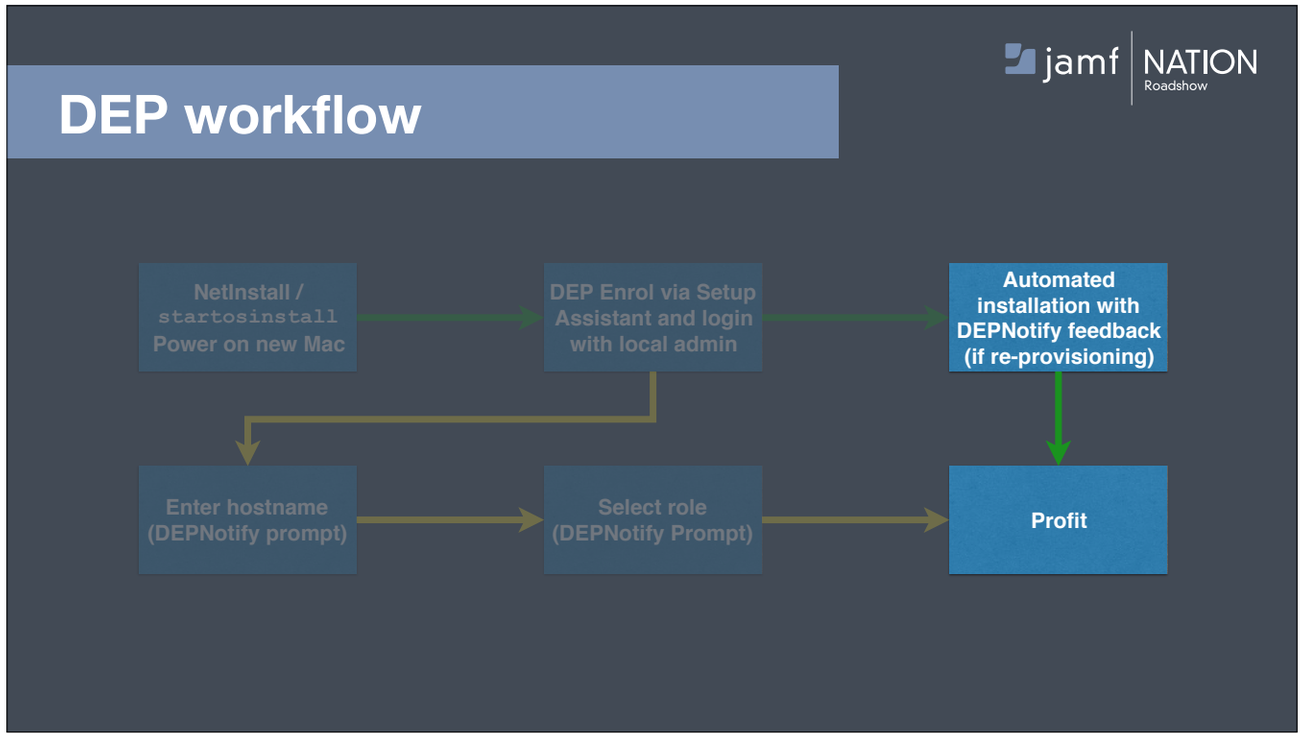
…and after lots of testing, more discussion, and even more testing great things start to happen.



So let’s add the missing pieces to our workflow.



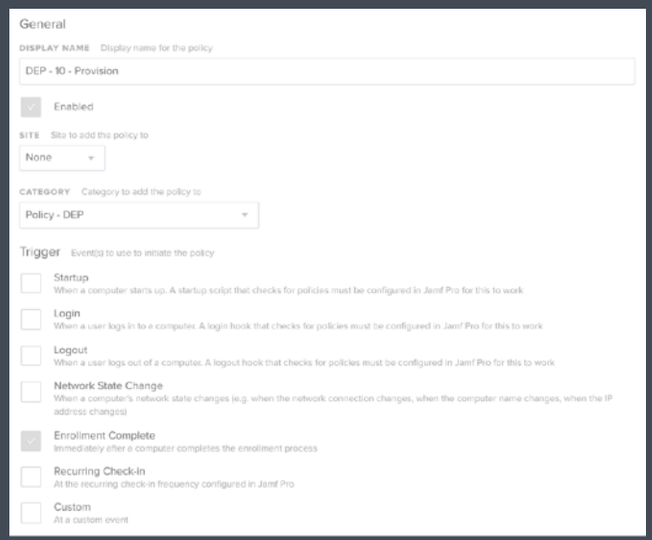
We need to set the hostname and role.



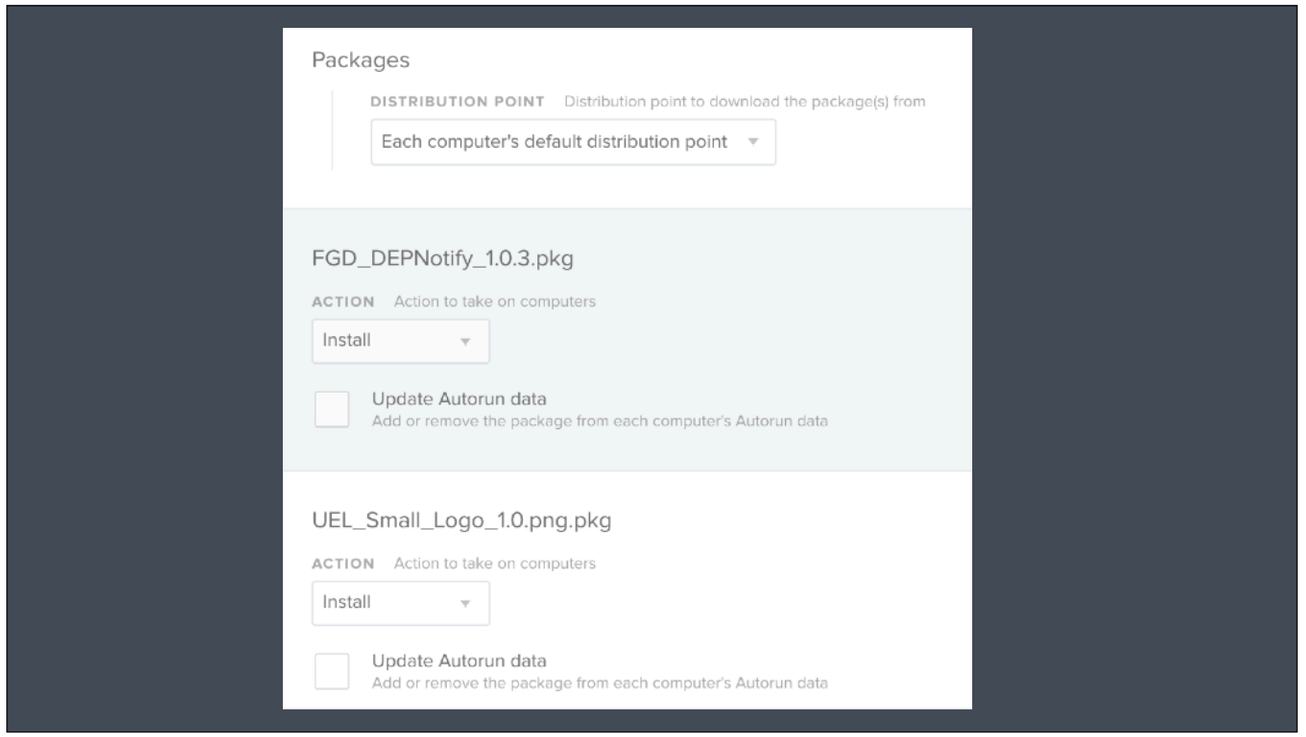
But if we're re-provisioning a Mac that's already in Jamf, we want to skip collecting that data and reuse what we have.

Like we did with autorun imaging.

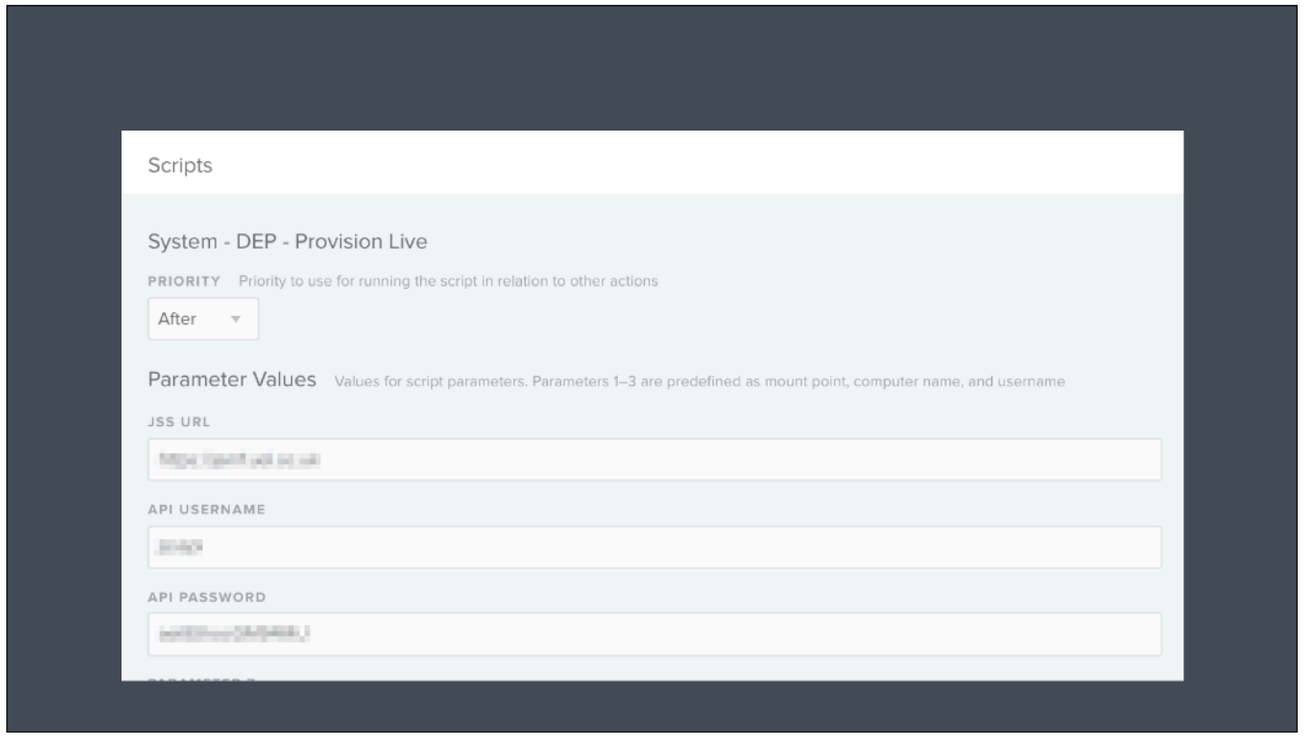
Policy - Enrolment Complete



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



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



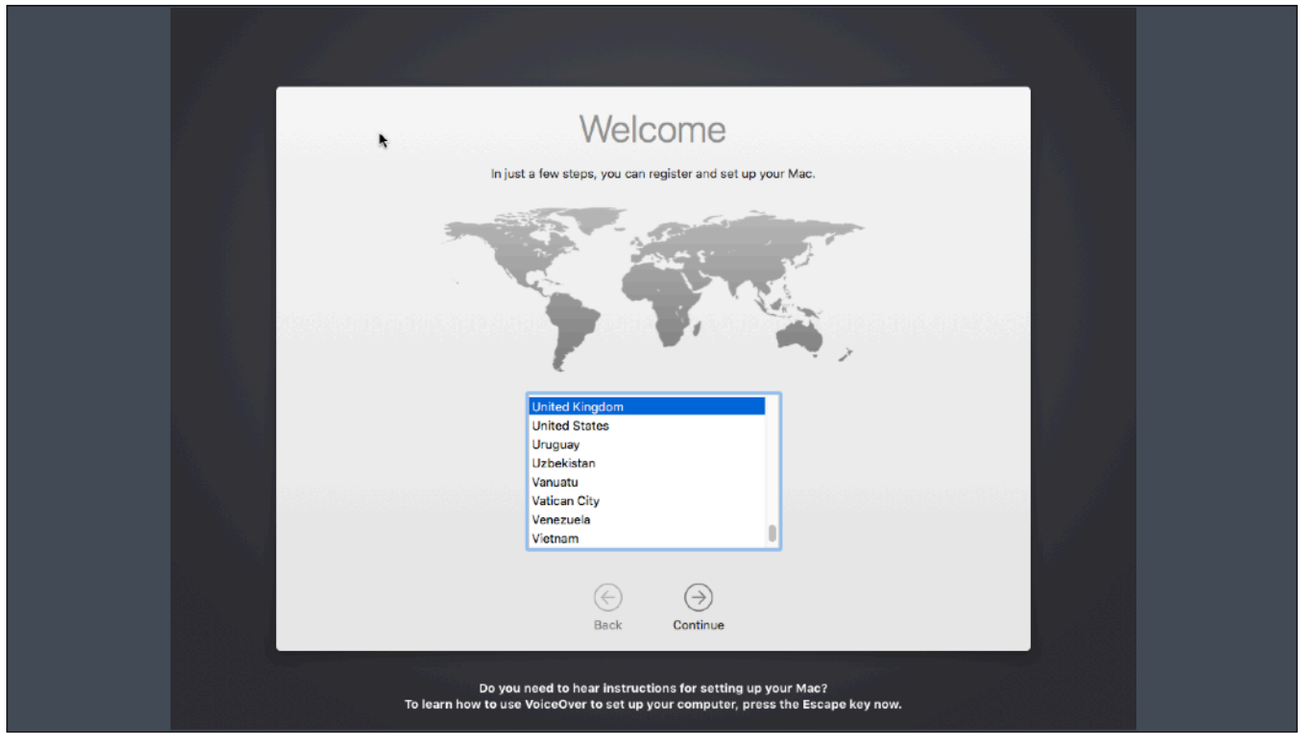
And run a script. As you can see here, we're using a few parameters.

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?

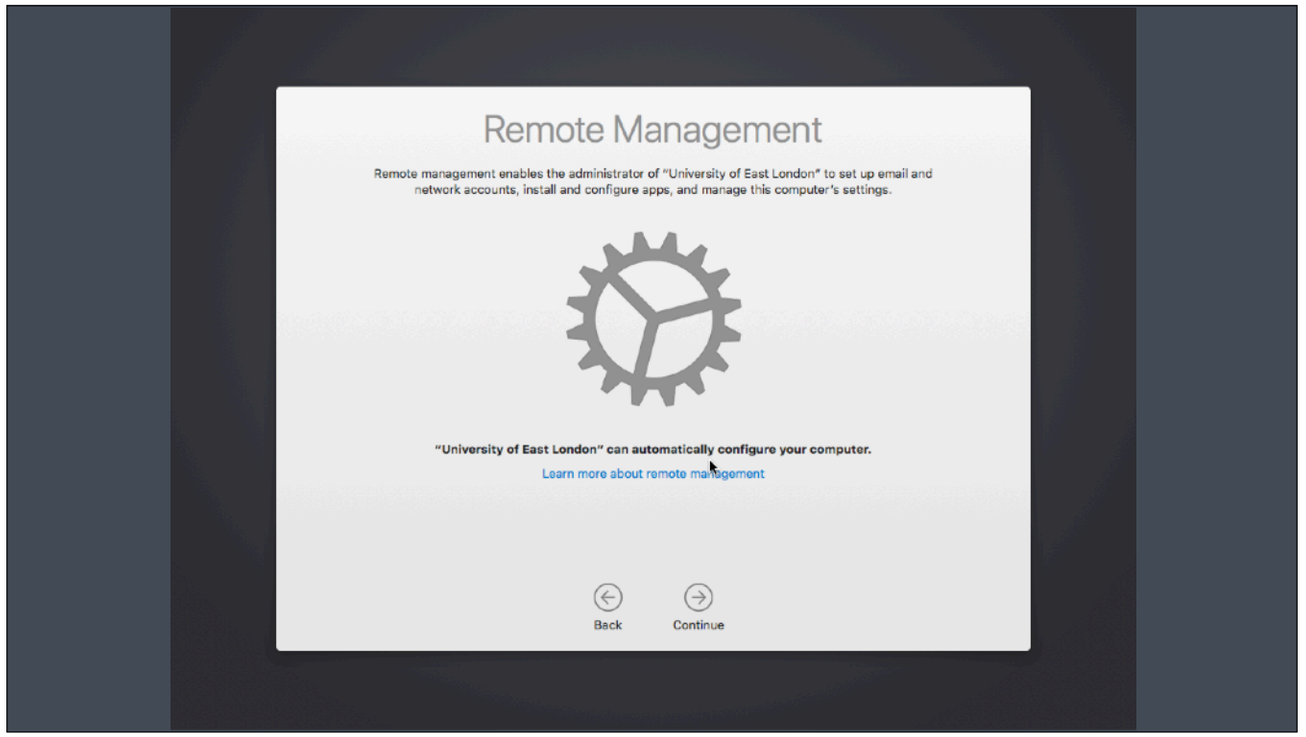
Lets put it all together.



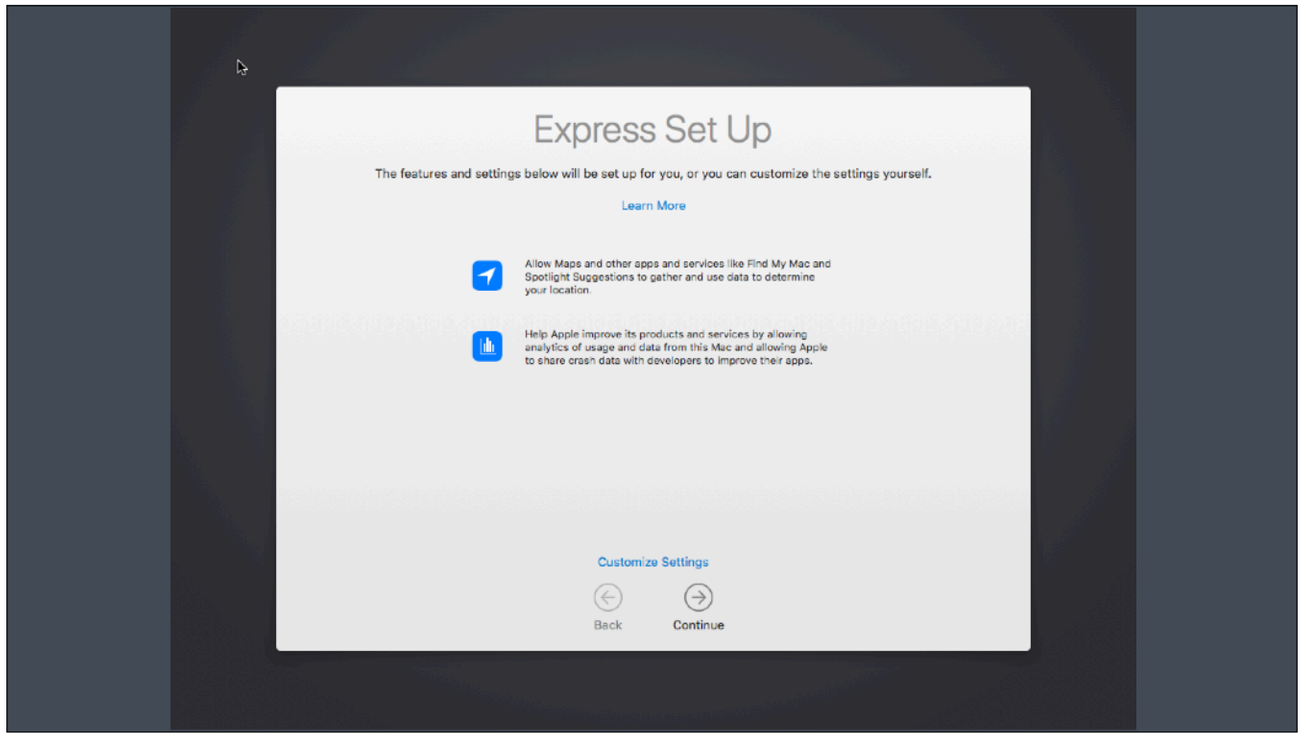
You’ve installed macOS and we’re at the Setup Assistant

Choose your country

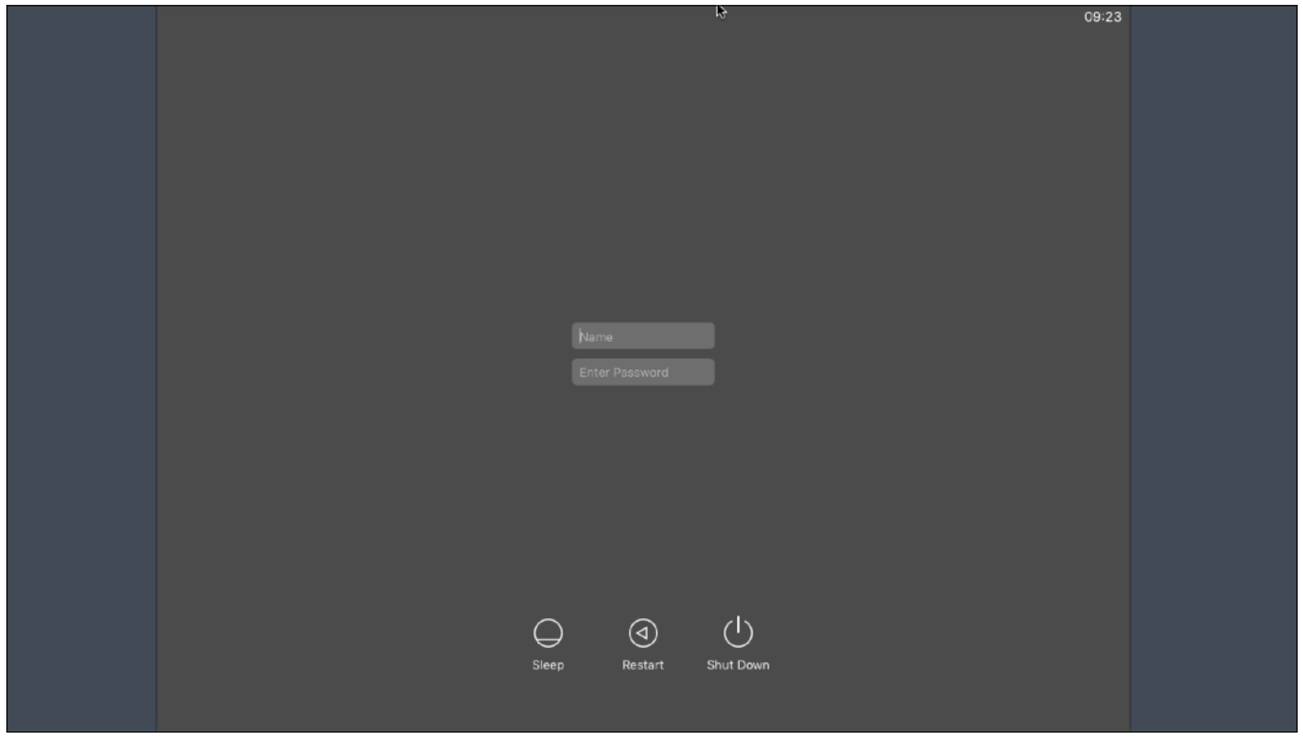
Your keyboard layout



Hit the DEP screen

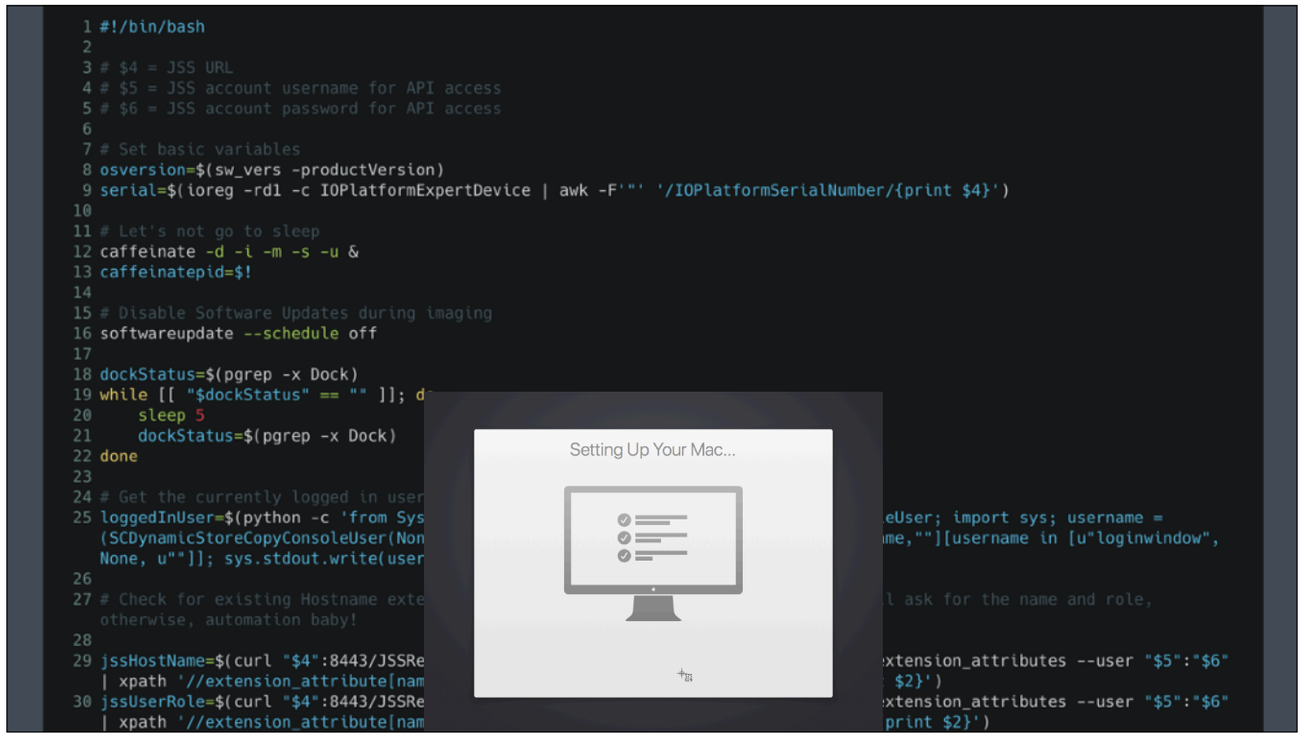


Proceed through the rest of the Setup Assistant

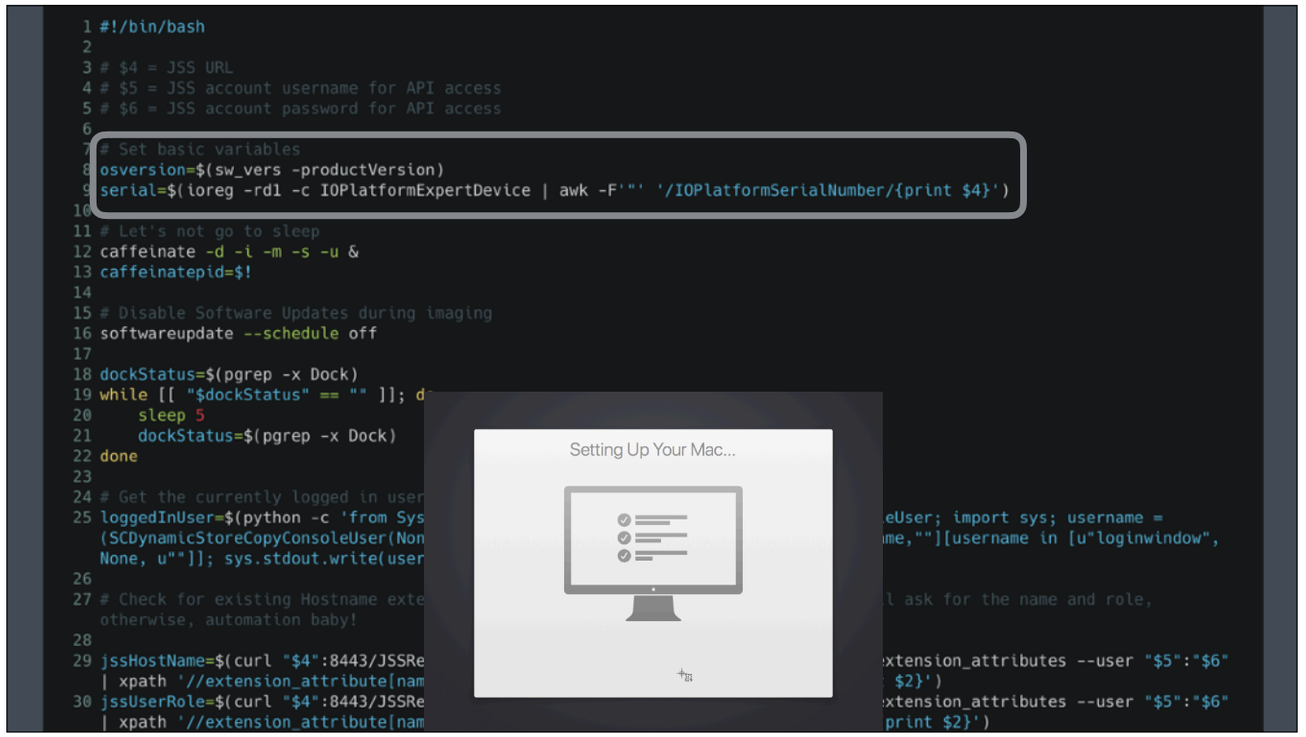


Login as local admin

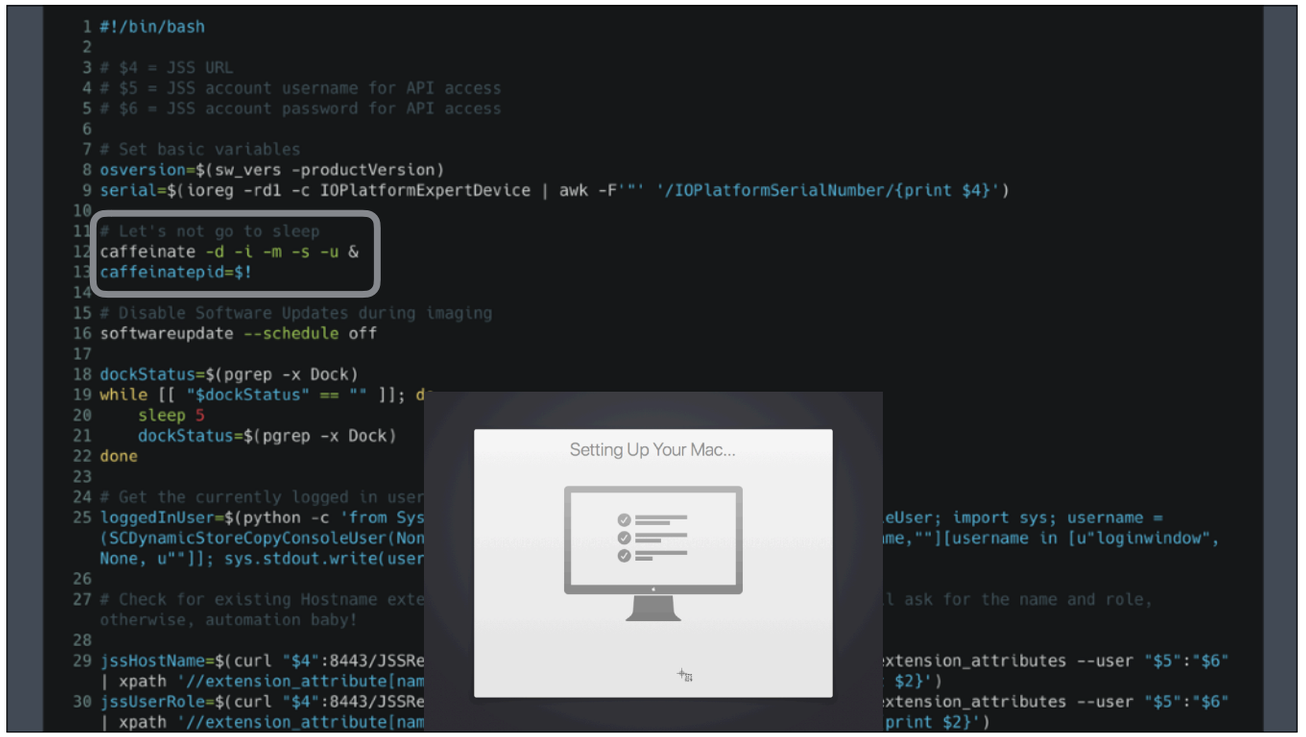
And off we go…



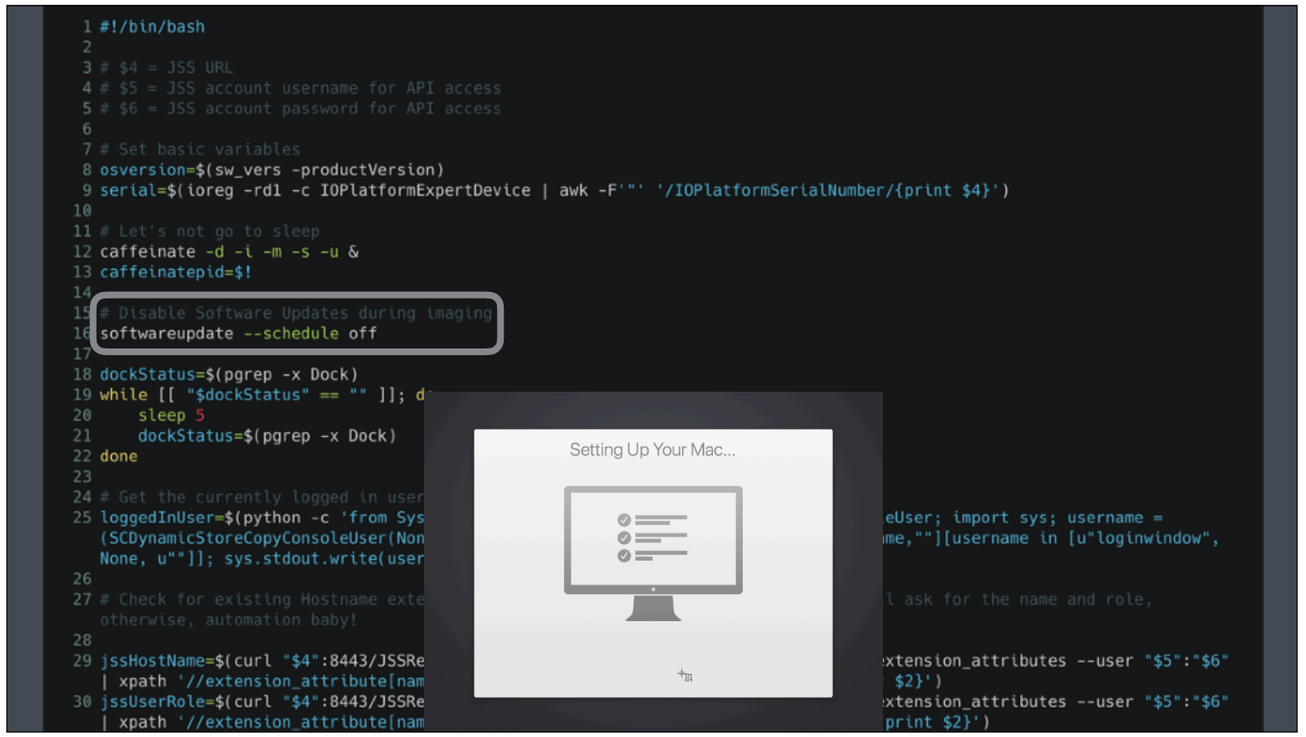
Our Enrolment Complete policy fires, installing DEPNotify and running this script



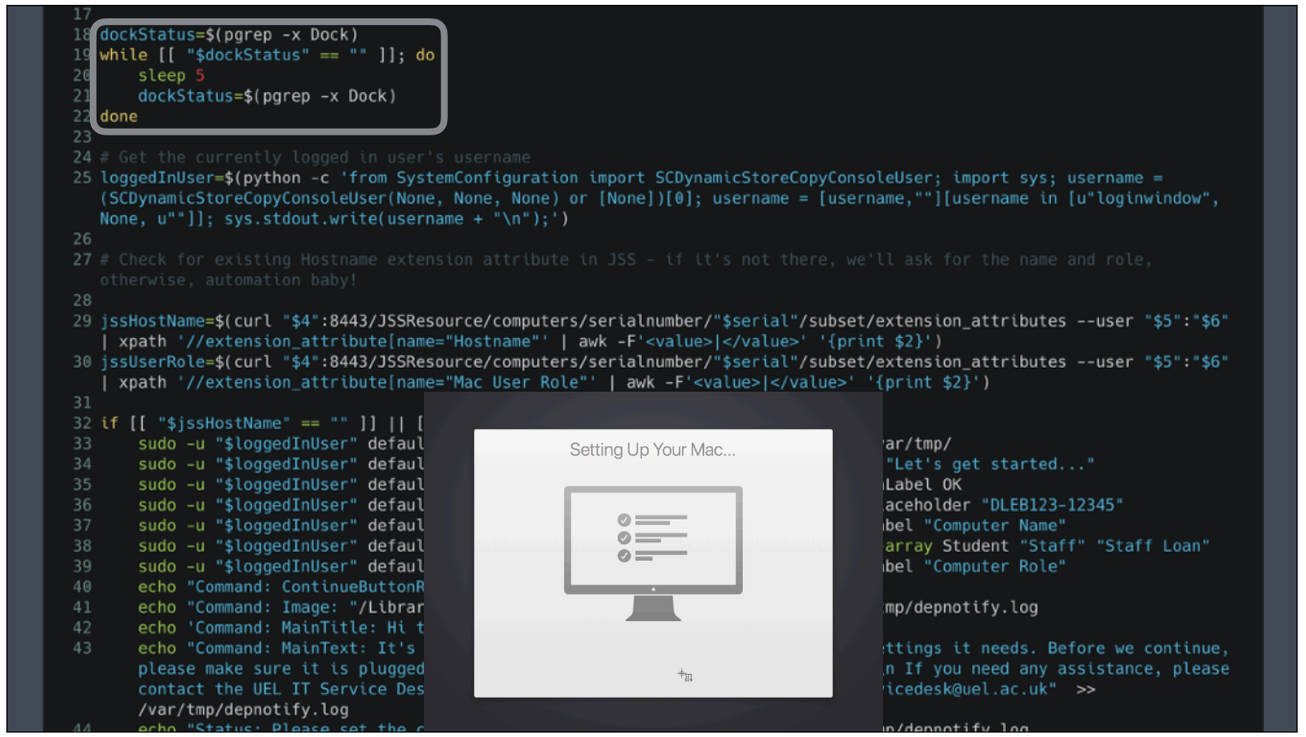
First we grab the OS version and serial number and store them as a couple of variables



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



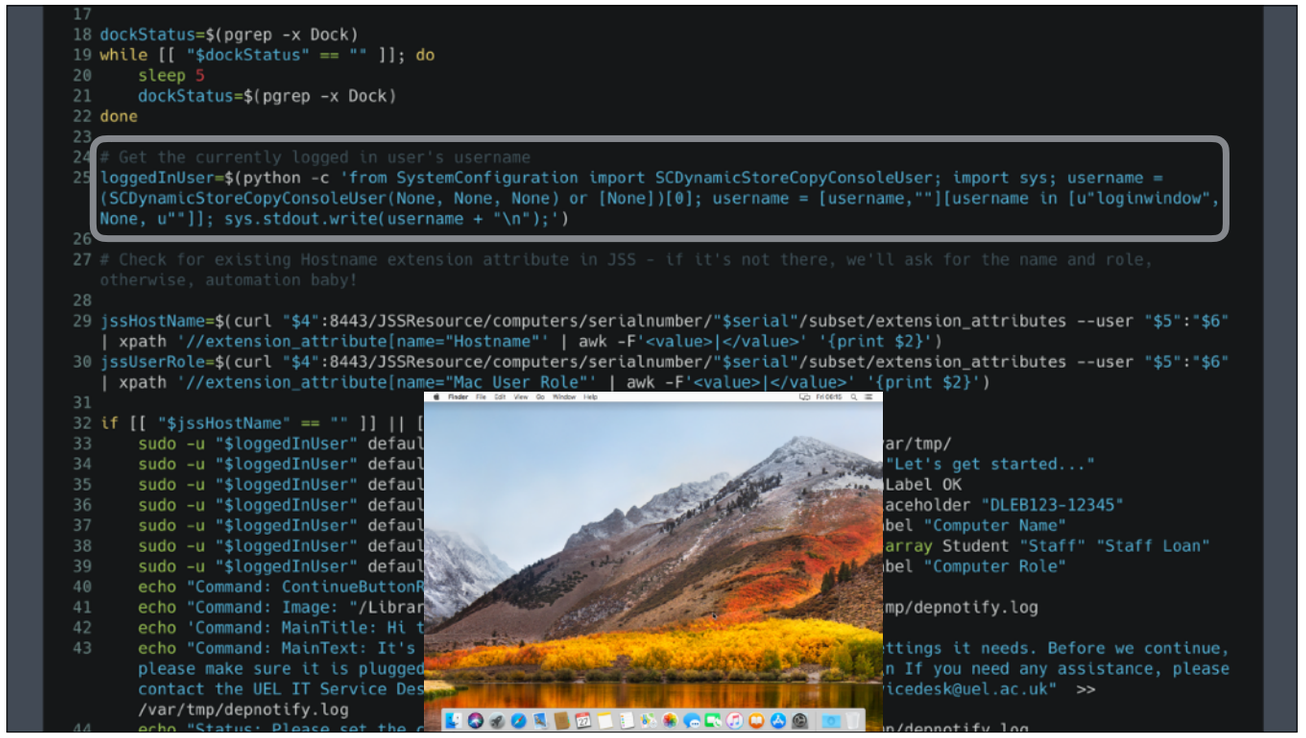
We nip any software update checks in the bud.



And we wait for login to complete.

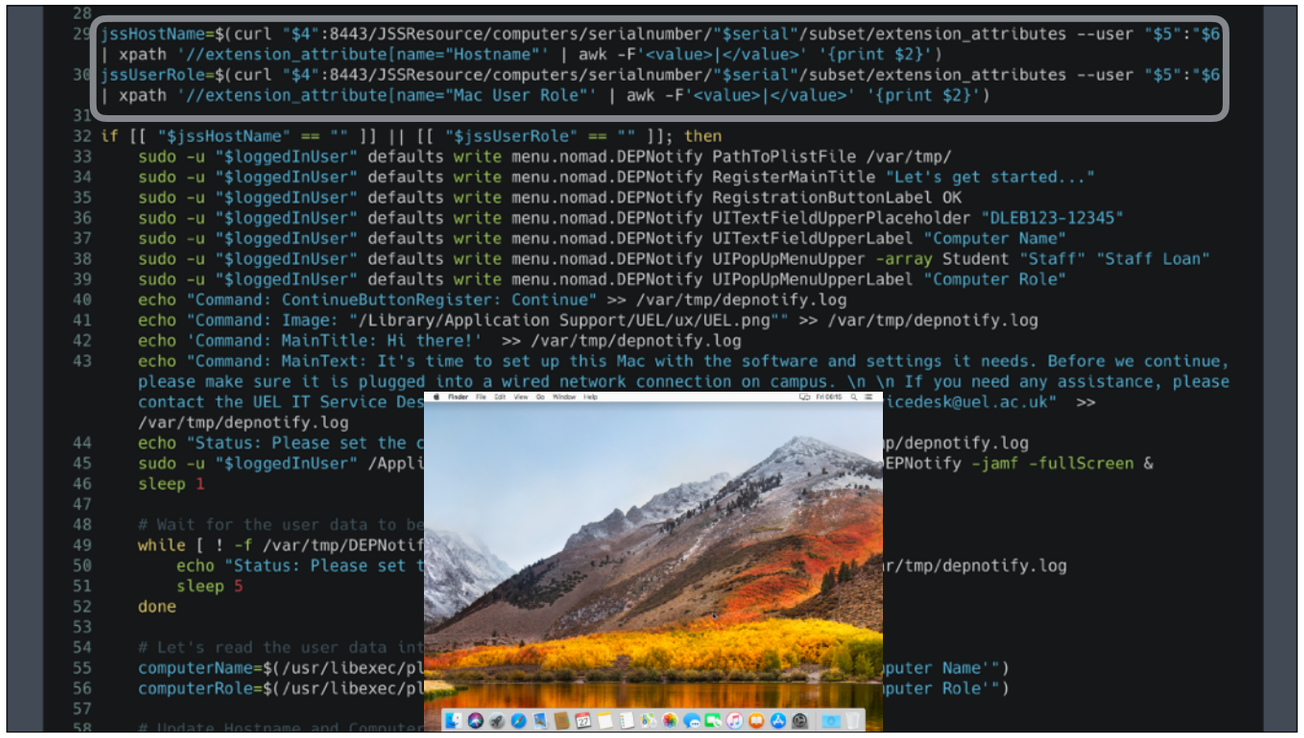
This is important - because of the way Jamf does enrolment with DEP, the script might have started before we logged in. DEPNotify needs to run after login.

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" defaul                                    ar/tmp/
    sudo -u "$loggedInUser" defaul                                    "Let's get started..."
    sudo -u "$loggedInUser" defaul                                    Label OK
    sudo -u "$loggedInUser" defaul                                    aceholder "DLEB123-12345"
    sudo -u "$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                                    n/depnotify.log
```

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



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.

```
31
32 if [[ "$jssHostName" == "" ]] || [[ "$jssUserRole" == "" ]]; then
33     sudo -u "$loggedInUser" defaults write menu.nomad.DEPNotify PathToPlistFile /var/tmp/
34     sudo -u "$loggedInUser" defaults write menu.nomad.DEPNotify RegisterMainTitle "Let's get started..."
35     sudo -u "$loggedInUser" defaults write menu.nomad.DEPNotify RegistrationButtonLabel OK
36     sudo -u "$loggedInUser" defaults write menu.nomad.DEPNotify UITextFieldUpperPlaceholder "DLEB123-12345"
37     sudo -u "$loggedInUser" defaults write menu.nomad.DEPNotify UITextFieldUpperLabel "Computer Name"
38     sudo -u "$loggedInUser" defaults write menu.nomad.DEPNotify UIPopUpMenuUpper -array Student "Staff" "Staff Loan"
39     sudo -u "$loggedInUser" defaults write menu.nomad.DEPNotify UIPopUpMenuUpperLabel "Computer Role"
40     echo "Command: ContinueButtonRegister: Continue" >> /var/tmp/depnotify.log
41     echo "Command: Image: "/Library/Application Support/UEL/ux/UEL.png"" >> /var/tmp/depnotify.log
42     echo 'Command: MainTitle: Hi there!'  >> /var/tmp/depnotify.log
43     echo "Command: MainText: It's time to set up this Mac with the software and settings it needs. Before we continue,
       please make sure it is plugged into a wired network connection on campus. \n \n If you need any assistance, please
       contact the UEL IT Service Desk. \n \n Telephone: 020 8223 2558 \n Email: servicedesk@uel.ac.uk"  >>
       /var/tmp/depnotify.log
44     echo "Status: Please set the computer name and role to continue..." >> /var/tmp/depnotify.log
45     sudo -u "$loggedInUser" /Applications/Utilities/DEPNotify.app/Contents/MacOS/DEPNotify -jamf -fullScreen &
46     sleep 1
47
48     # Wait for the user data to be
49     while [ ! -f /var/tmp/DEPNotif
50         echo "Status: Please set t                                    ar/tmp/depnotify.log
51         sleep 5
52     done
53
54     # Let's read the user data int
55     computerName=$(/usr/libexec/pl                                  puter Name'")
56     computerRole=$(/usr/libexec/pl                                  puter Role'")
57
58     # Update Hostname and Computer
59     # Create xml
60     cat << EOF > /var/tmp/name.xml
61 <computer>
62     <extension_attributes>
63         <extension_attributes>
```

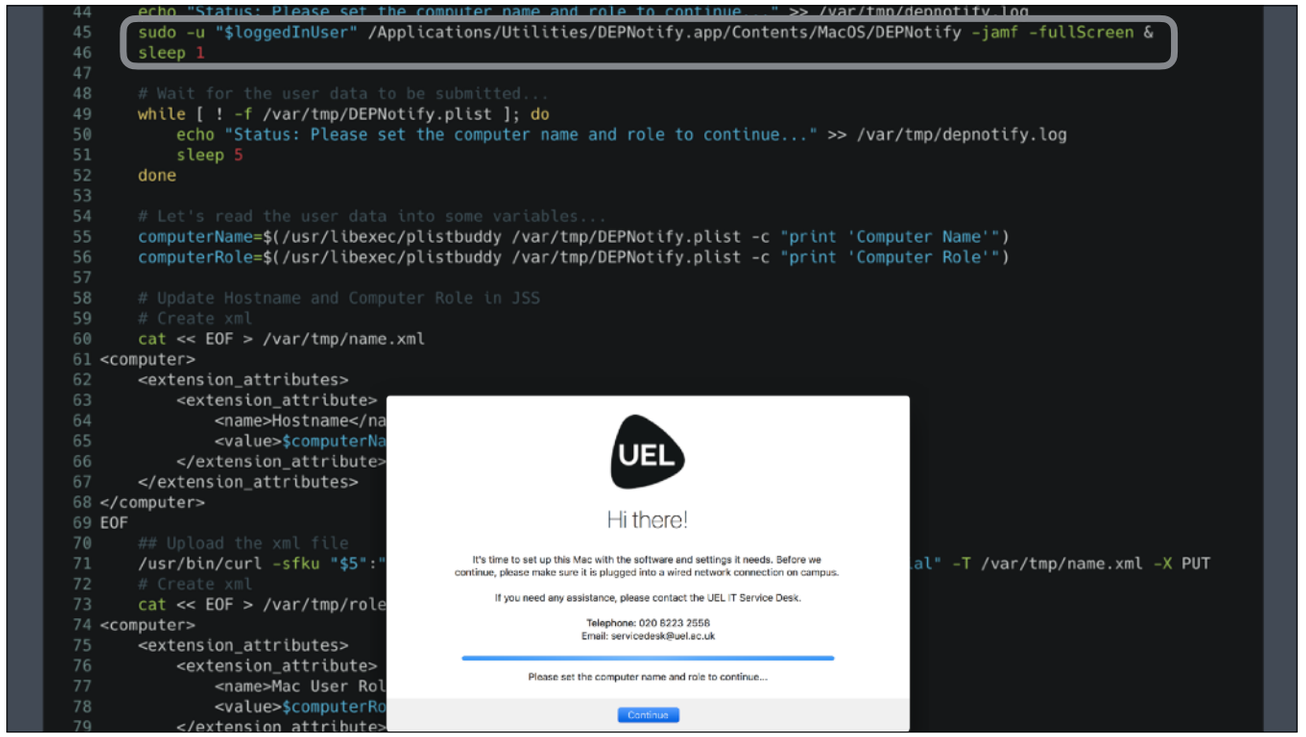
If either of those Extension Attributes are empty, we're going to set up DEPNotify to prompt for user input.

```
31
32 if [[ "$issHostName" == "" ]] || [[ "$issUserRole" == "" ]]; then
33     sudo -u "$loggedInUser" defaults write menu.nomad.DEPNotify PathToPlistFile /var/tmp/
34     sudo -u "$loggedInUser" defaults write menu.nomad.DEPNotify RegisterMainTitle "Let's get started..."
35     sudo -u "$loggedInUser" defaults write menu.nomad.DEPNotify RegistrationButtonLabel OK
36     sudo -u "$loggedInUser" defaults write menu.nomad.DEPNotify UITextFieldUpperPlaceholder "DLEB123-12345"
37     sudo -u "$loggedInUser" defaults write menu.nomad.DEPNotify UITextFieldUpperLabel "Computer Name"
38     sudo -u "$loggedInUser" defaults write menu.nomad.DEPNotify UIPopUpMenuUpper -array Student "Staff" "Staff Loan"
39     sudo -u "$loggedInUser" defaults write menu.nomad.DEPNotify UIPopUpMenuUpperLabel "Computer Role"
40     echo "Command: ContinueButtonRegister: Continue" >> /var/tmp/depnotify.log
41     echo "Command: Image: "/Library/Application Support/UEL/ux/UEL.png"" >> /var/tmp/depnotify.log
42     echo 'Command: MainTitle: Hi there!'  >> /var/tmp/depnotify.log
43     echo "Command: MainText: It's time to set up this Mac with the software and settings it needs. Before we continue,
       please make sure it is plugged into a wired network connection on campus. \n \n If you need any assistance, please
       contact the UEL IT Service Desk. \n \n Telephone: 020 8223 2558 \n Email: servicedesk@uel.ac.uk"  >>
       /var/tmp/depnotify.log
44     echo "Status: Please set the computer name and role to continue..." >> /var/tmp/depnotify.log
45     sudo -u "$loggedInUser" /Applications/Utilities/DEPNotify.app/Contents/MacOS/DEPNotify -jamf -fullScreen &
46     sleep 1
47
48     # Wait for the user data to be
49     while [ ! -f /var/tmp/DEPNotif
50         echo "Status: Please set t                                    ar/tmp/depnotify.log
51         sleep 5
52     done
53
54     # Let's read the user data int
55     computerName=$(/usr/libexec/pl                                  puter Name'")
56     computerRole=$(/usr/libexec/pl                                  puter Role'")
57
58     # Update Hostname and Computer
59     # Create xml
60     cat << EOF > /var/tmp/name.xml
61 <computer>
62     <extension_attributes>
63         <extension_attributes>
```

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.

```
31
32 if [[ "$jssHostName" == "" ]] || [[ "$jssUserRole" == "" ]]; then
33     sudo -u "$loggedInUser" defaults write menu.nomad.DEPNotify PathToPlistFile /var/tmp/
34     sudo -u "$loggedInUser" defaults write menu.nomad.DEPNotify RegisterMainTitle "Let's get started..."
35     sudo -u "$loggedInUser" defaults write menu.nomad.DEPNotify RegistrationButtonLabel OK
36     sudo -u "$loggedInUser" defaults write menu.nomad.DEPNotify UITextFieldUpperPlaceholder "DLEB123-12345"
37     sudo -u "$loggedInUser" defaults write menu.nomad.DEPNotify UITextFieldUpperLabel "Computer Name"
38     sudo -u "$loggedInUser" defaults write menu.nomad.DEPNotify UIPopUpMenuUpper -array Student "Staff" "Staff Loan"
39     sudo -u "$loggedInUser" defaults write menu.nomad.DEPNotify UIPopUpMenuUpperLabel "Computer Role"
40     echo "Command: ContinueButtonRegister: Continue" >> /var/tmp/depnotify.log
41     echo "Command: Image: "/Library/Application Support/UEL/ux/UEL.png"" >> /var/tmp/depnotify.log
42     echo 'Command: MainTitle: Hi there!'  >> /var/tmp/depnotify.log
43     echo "Command: MainText: It's time to set up this Mac with the software and settings it needs. Before we continue,
       please make sure it is plugged into a wired network connection on campus. \n \n If you need any assistance, please
       contact the UEL IT Service Desk. \n \n Telephone: 020 8223 2558 \n Email: servicedesk@uel.ac.uk"  >>
       /var/tmp/depnotify.log
44     echo "Status: Please set the computer name and role to continue..." >> /var/tmp/depnotify.log
45     sudo -u "$loggedInUser" /Applications/Utilities/DEPNotify.app/Contents/MacOS/DEPNotify -jamf -fullScreen &
46     sleep 1
47
48     # Wait for the user data to be
49     while [ ! -f /var/tmp/DEPNotif
50         echo "Status: Please set t                                    ar/tmp/depnotify.log
51         sleep 5
52     done
53
54     # Let's read the user data int
55     computerName=$(/usr/libexec/pl                                  puter Name'")
56     computerRole=$(/usr/libexec/pl                                  puter Role'")
57
58     # Update Hostname and Computer
59     # Create xml
60     cat << EOF > /var/tmp/name.xml
61 <computer>
62     <extension_attributes>
63         <extension_attributes>
```

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.

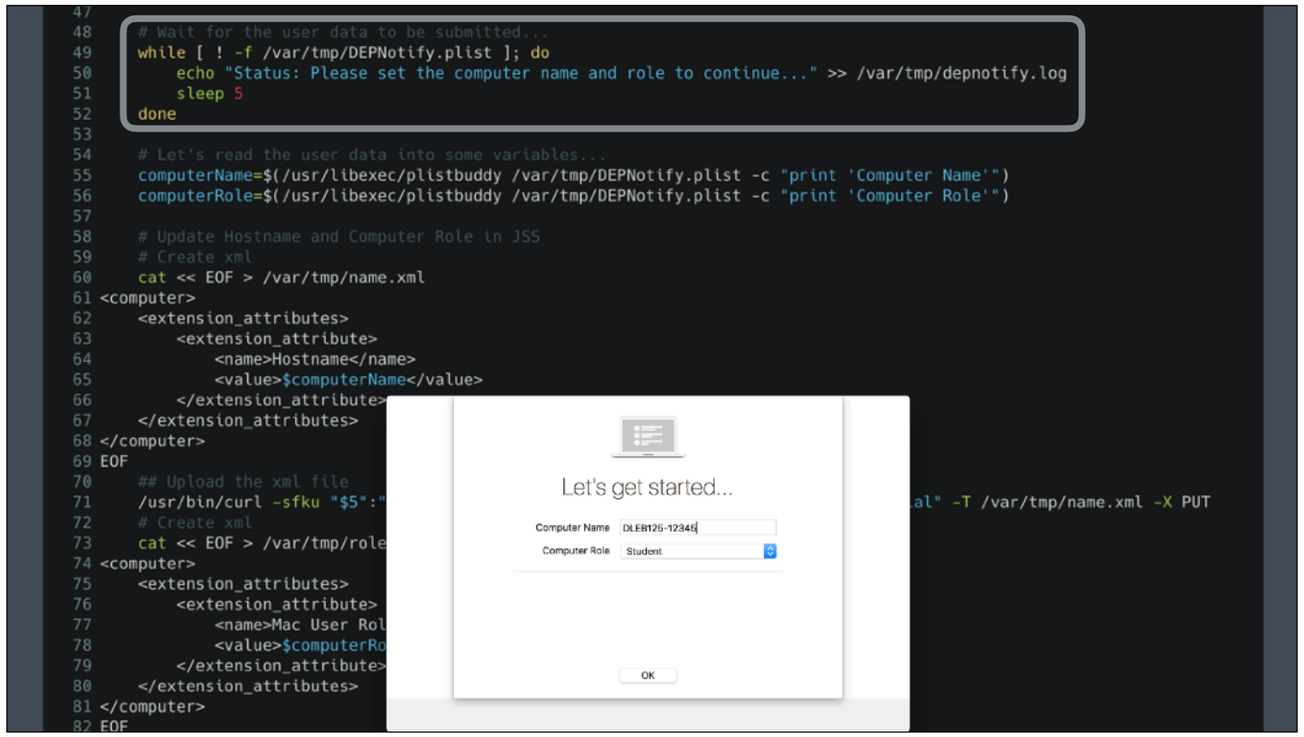


Now we launch DEPNotify as the logged in user, along with the Jamf and fullscreen switches.

Its window appears and the surrounding screen area is blurred.

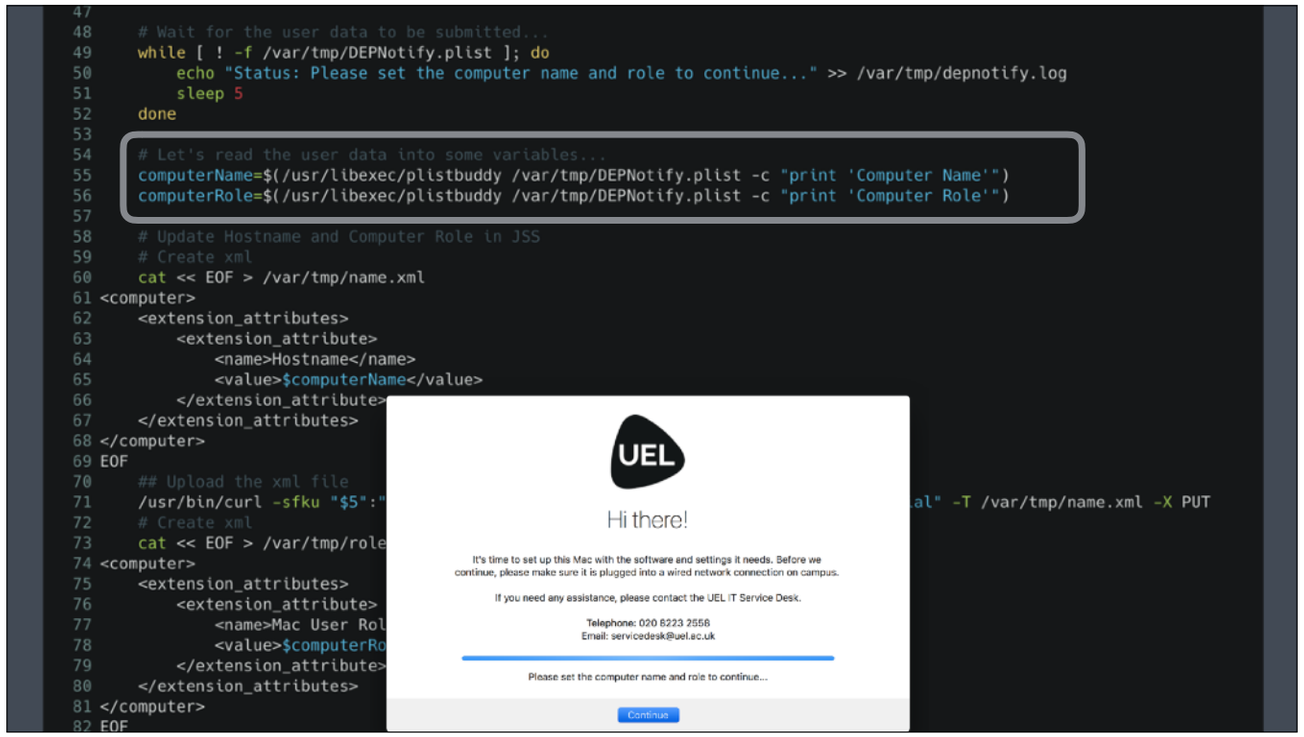
It will read the Jamf log and update the status text when policies run and packages are installed.

We click the Continue button



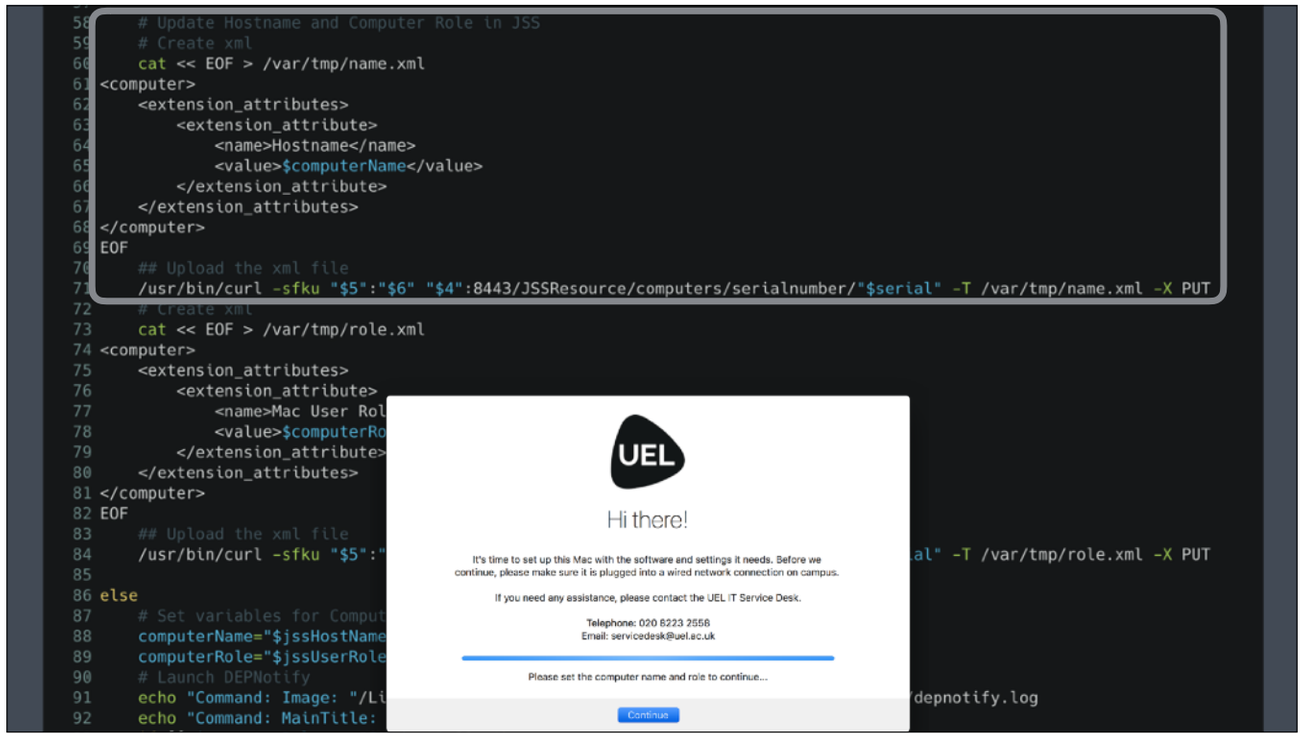
We won’t continue until the plist containing our user input data is written.

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

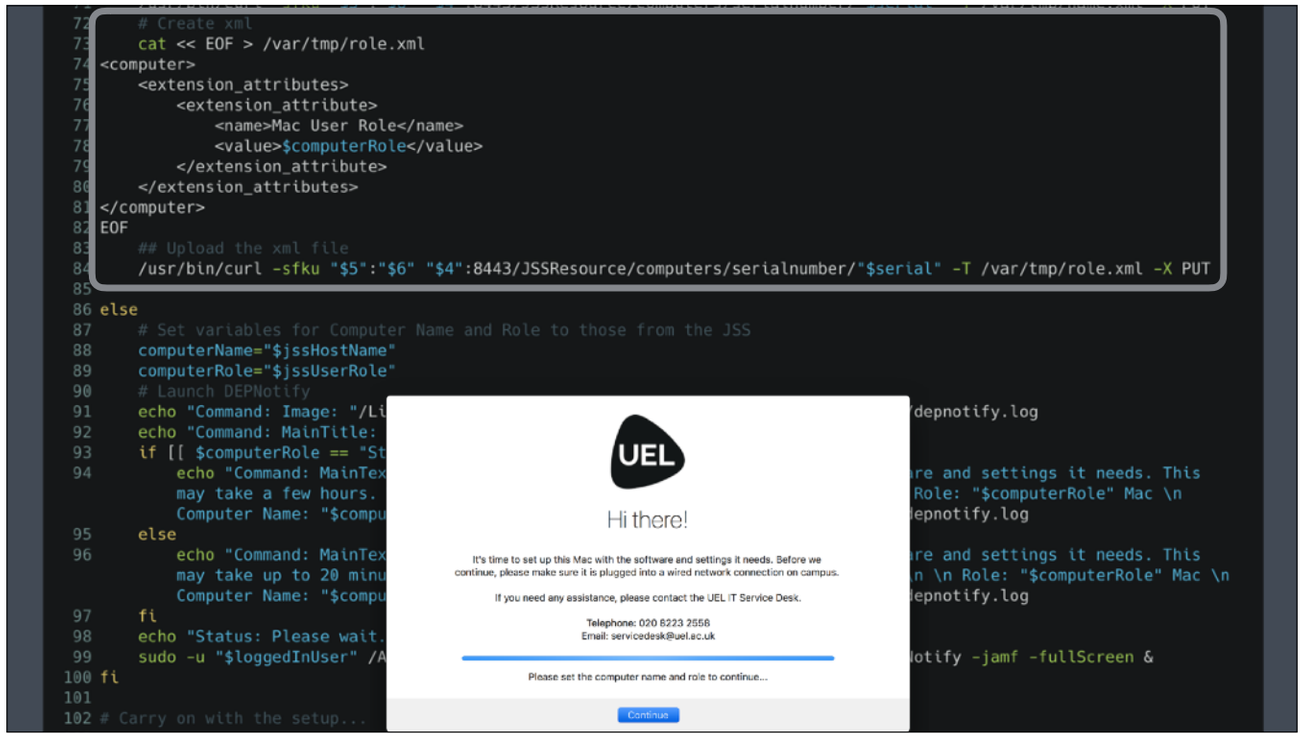



```
47
48      # Wait for the user data to be submitted...
49      while [ ! -f /var/tmp/DEPNotify.plist ]; do
50          echo "Status: Please set the computer name and role to continue..." >> /var/tmp/depnotify.log
51          sleep 5
52      done
53
54      # Let's read the user data into some variables...
55      computerName=$(/usr/libexec/plistbuddy /var/tmp/DEPNotify.plist -c "print 'Computer Name'")
56      computerRole=$(/usr/libexec/plistbuddy /var/tmp/DEPNotify.plist -c "print 'Computer Role'")
57
58      # Update Hostname and Computer Role in JSS
59      # Create xml
60      cat << EOF > /var/tmp/name.xml
61  <computer>
62      <extension_attributes>
63          <extension_attribute>
64              <name>Hostname</name>
65              <value>$computerName</value>
66          </extension_attribute>
67      </extension_attributes>
68  </computer>
69  EOF
70      ## Upload the xml file
71      /usr/bin/curl -sfku "$5":"                                    al" -T /var/tmp/name.xml -X PUT
72      # Create xml
73      cat << EOF > /var/tmp/role
74  <computer>
75      <extension_attributes>
76          <extension_attribute>
77              <name>Mac User Rol
78              <value>$computerRo
79          </extension_attribute>
80      </extension_attributes>
81  </computer>
82  EOF
```

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



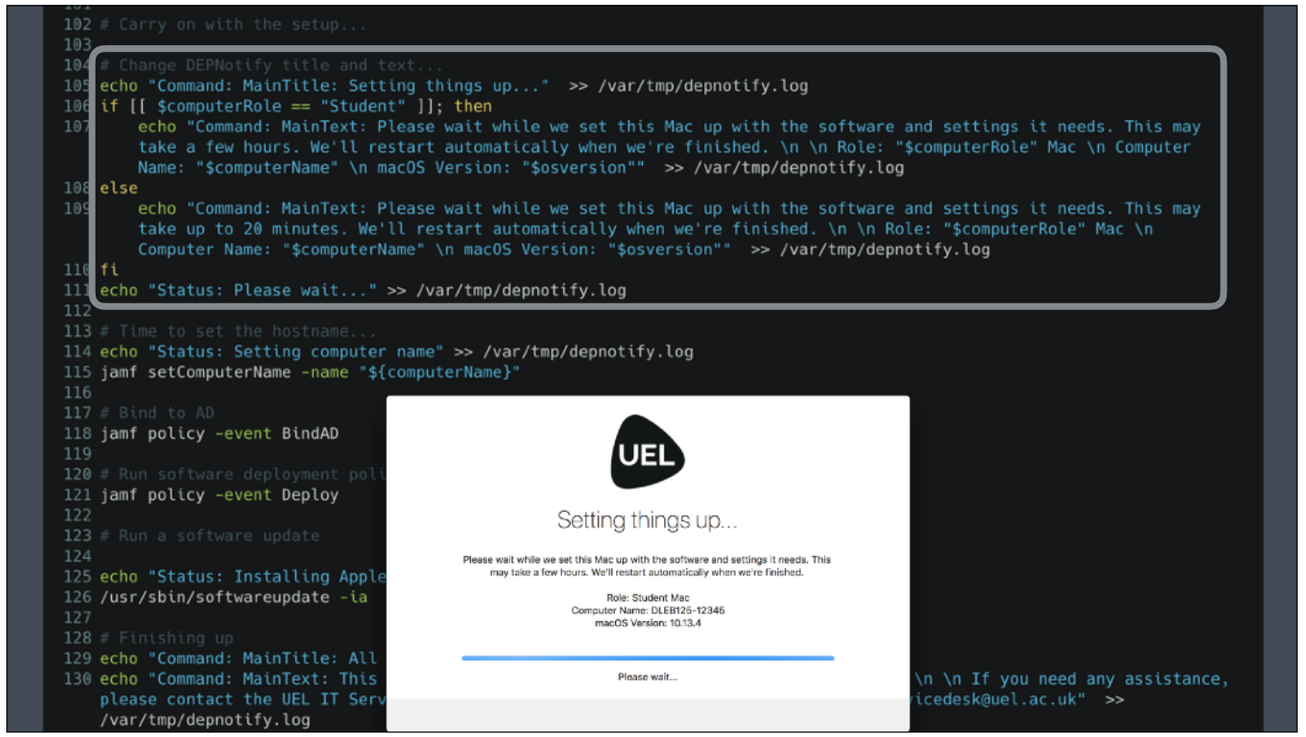
Next, we create some XML containing the name and use the API to update the Computer Name Extension Attribute with it.



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 [[ $computerRole == "Student" ]]; then
        echo "Command: MainText: Please wait while we set this Mac up with the software and settings it needs. This
        may take a few hours. We'll restart automatically when we're finished. \n \n Role: "$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: "$osversion""  >> /var/tmp/depnotify.log
    fi
    echo "Status: Please wait..." >> /var/tmp/depnotify.log
    sudo -u "$loggedInUser" /Applications/Utilities/DEPNotify.app/Contents/MacOS/DEPNotify -jamf -fullScreen &
fi

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

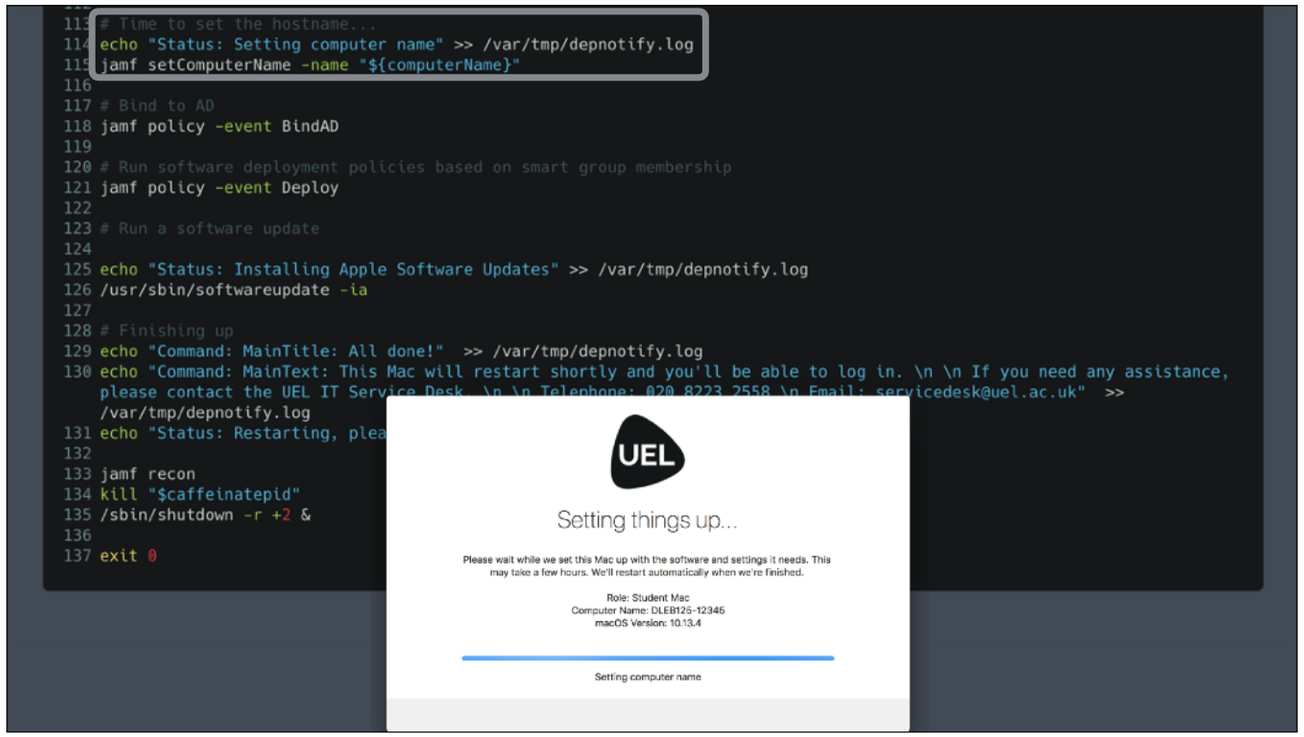
Now, if the name and role are already set, we'd skip all of that - so we're not prompted if we're re-provisioning a Mac that's already in Jamf.



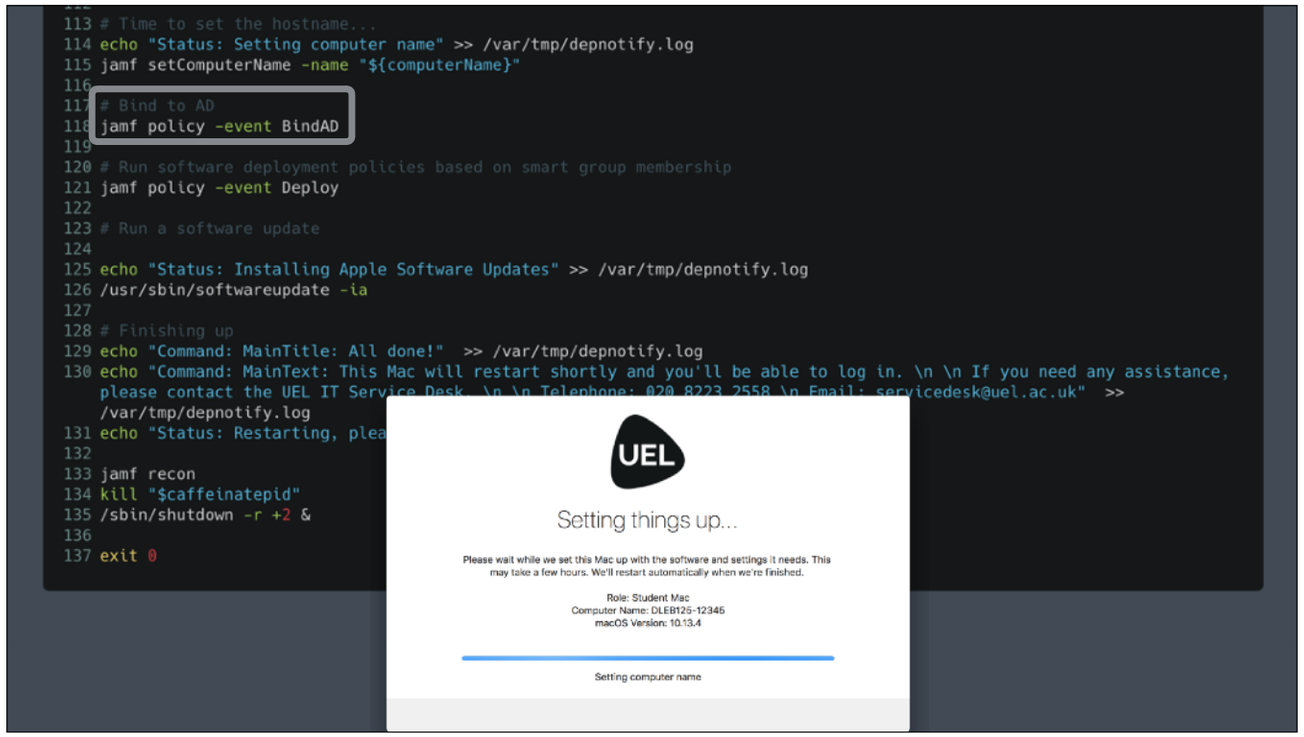
We carry on - changing the title, main text and status to let us know what's happening.

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

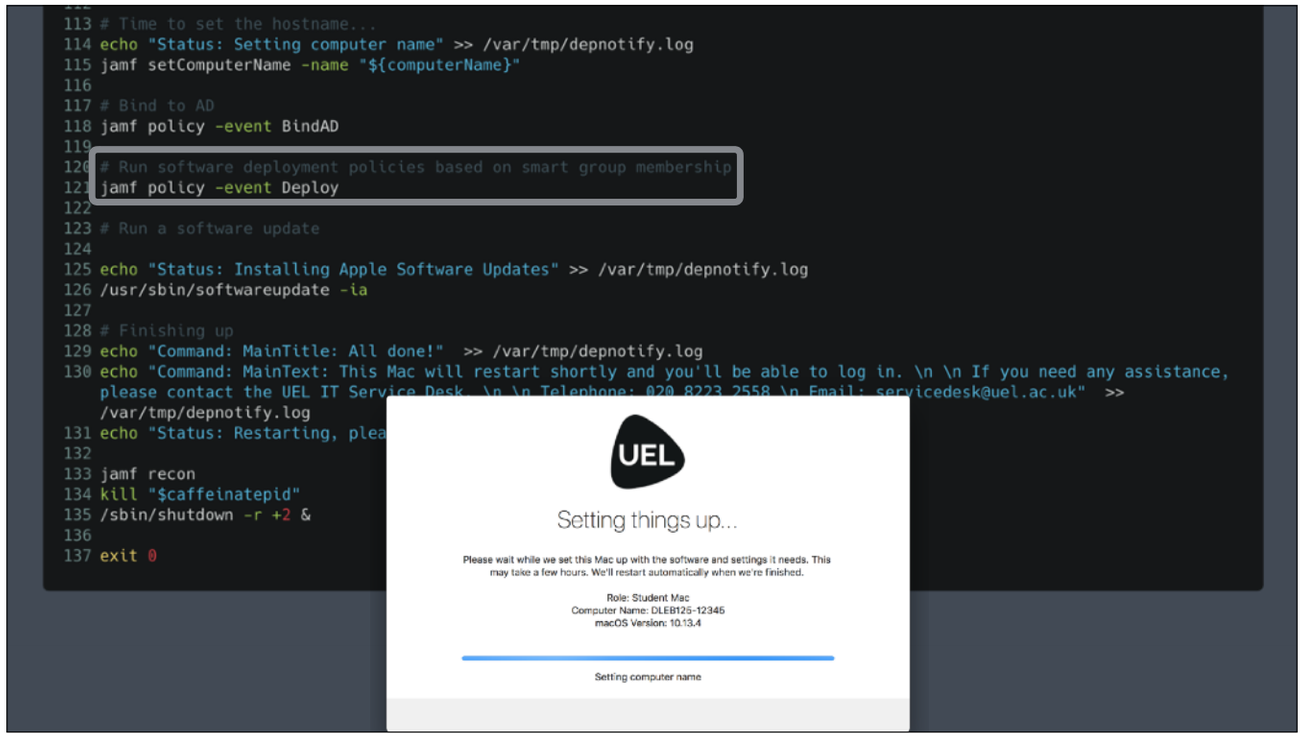
If it's a student facing Mac, our main text lets us know it'll take longer to set up than a staff facing Mac.



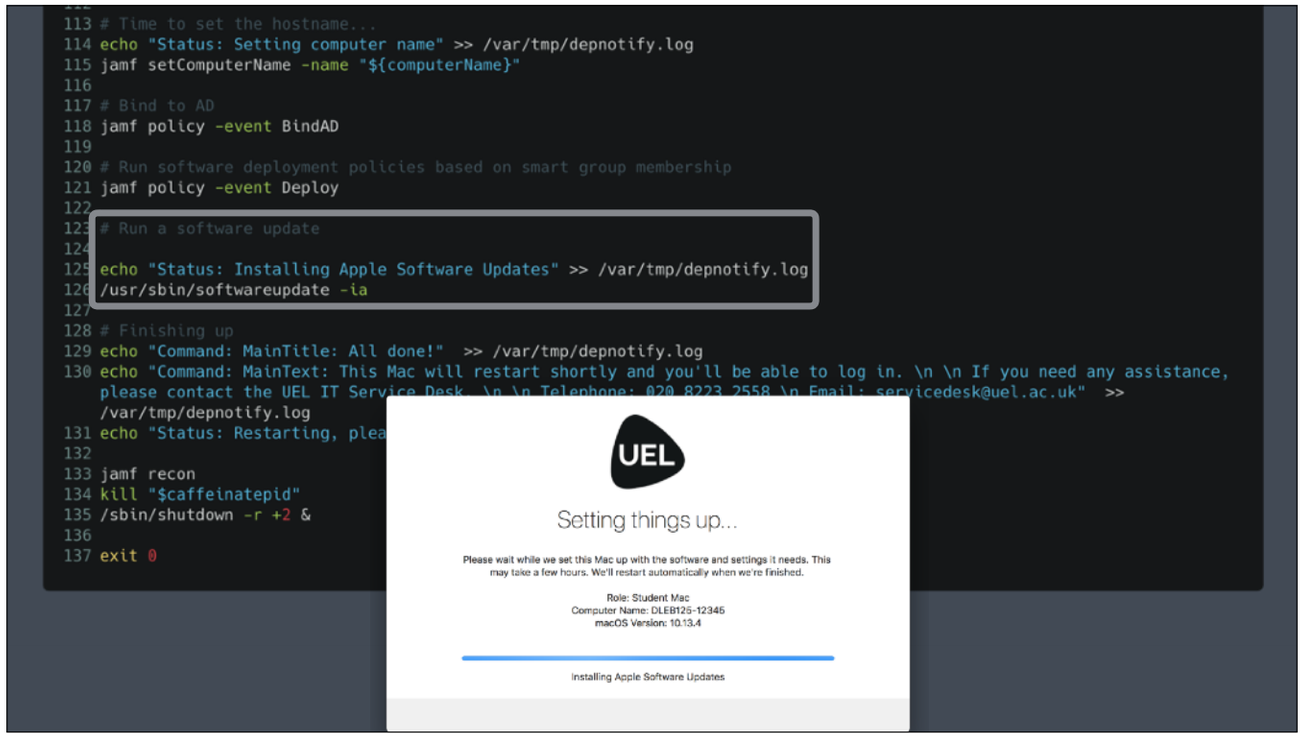
We set the hostname



Bind to Active Directory

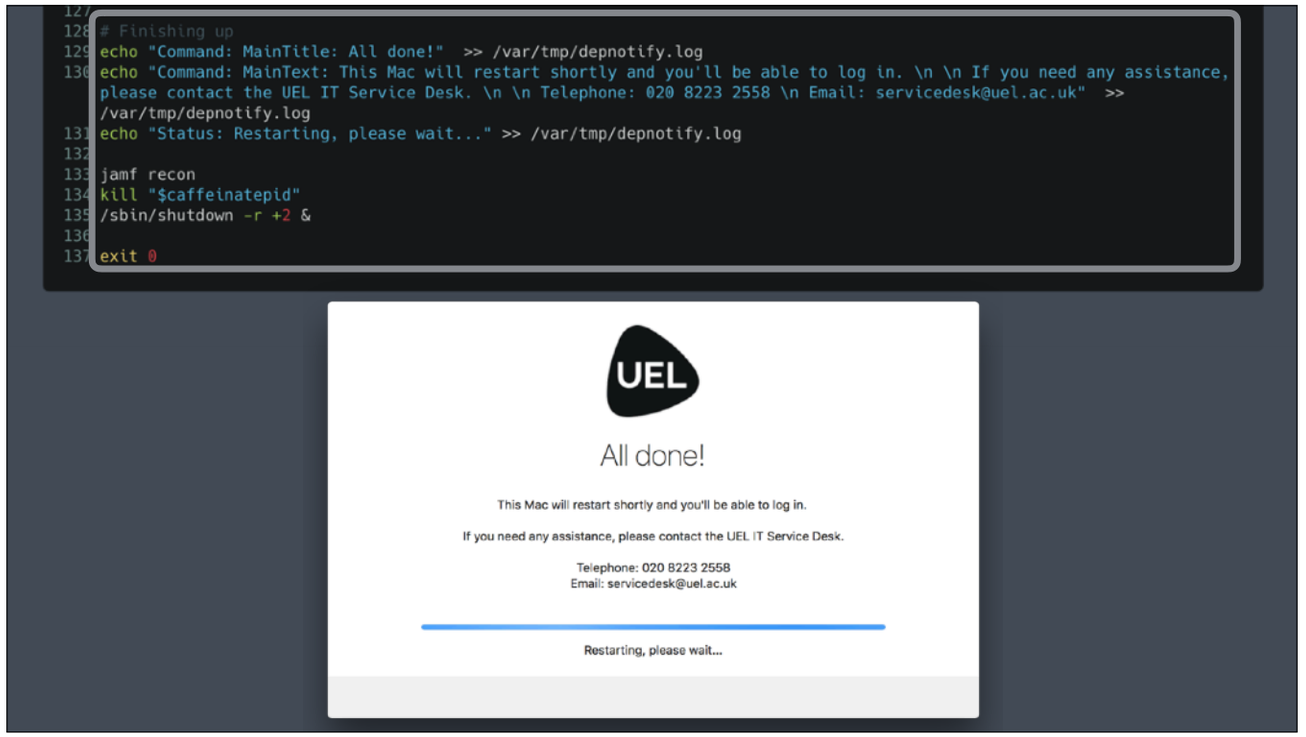


And run any custom policy triggers we like. Our Mac will be in the right lab smart group and the relevant policies to install that lab's specific software will be scoped to that smart group. Because of that, those policies can all have the same custom trigger.



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
```



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

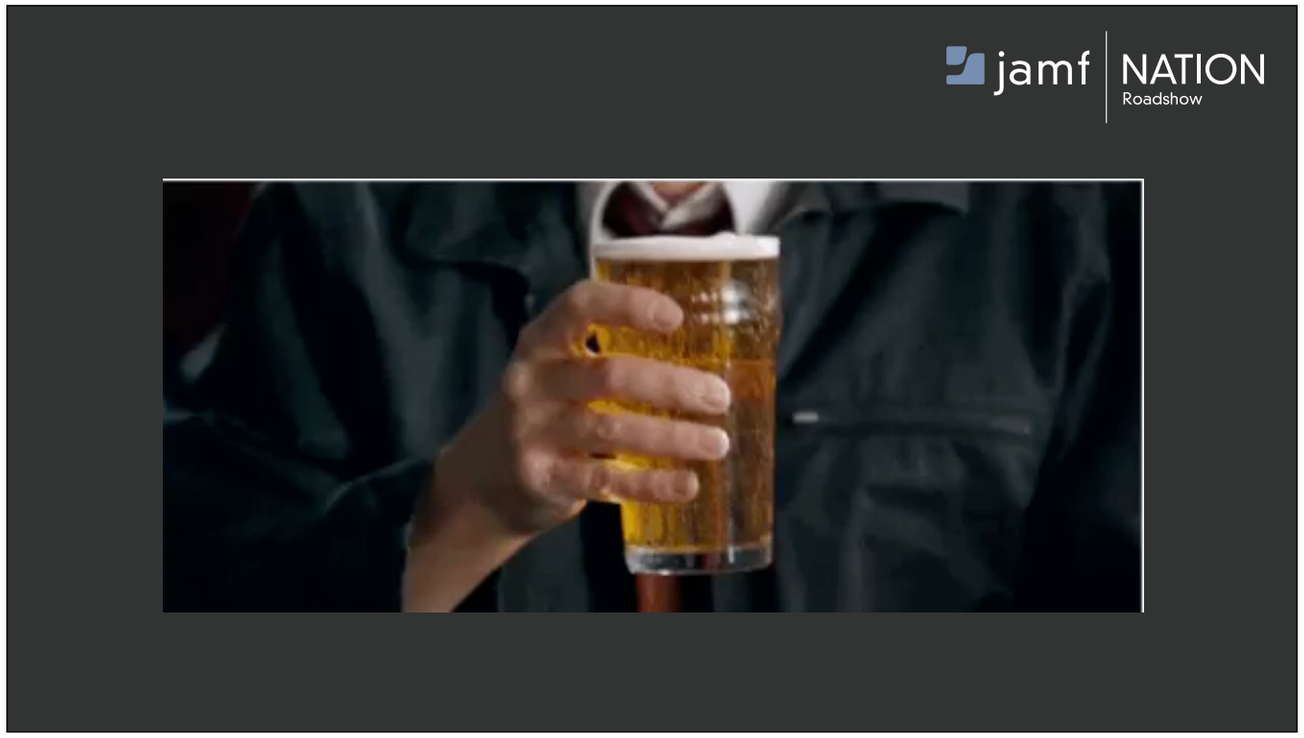
So, I said at the beginning that this was a journey I was TAKING. There are lots of places we could continue to take this to.

<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”…



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

Resources

https://soundmacguy.wordpress.com/2018/05/17/lab-nauseam-dawn-of-the-dep/